Enhanced Formulations for Minimax and Discrete Optimization Problems with Applications to Scheduling and Routing

Ahmed Ghoniem

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

Hanif D. Sherali, Chair Subhash C. Sarin C. Patrick Koelling Ebru Bish

June 29, 2007 Blacksburg, Virginia

Keywords: Operations Research, Minimax Optimization, Reformulation-Linearization Technique, Column Generation, Symmetry-defeating Strategies, Subgradient Optimization, Scheduling, Routing.

Copyright 2007, Ahmed Ghoniem
Enhanced Formulations for Minimax and Discrete Optimization Problems with Applications to Scheduling and Routing
Ahmed Ghoniem

(ABSTRACT)

This dissertation addresses the development of enhanced formulations for minimax and mixed-integer programming models for certain industrial and logistical systems, along with the design and implementation of efficient algorithmic strategies. We first examine the general class of minimax mixed-integer 0-1 problems of the type that frequently arise in decomposition approaches and in a variety of location and scheduling problems. We conduct an extensive polyhedral analysis of this problem in order to tighten its representation using the Reformulation-Linearization/Convexification Technique (RLT), and demonstrate the benefits of the resulting lifted formulations for several classes of problems. Specifically, we investigate RLT-enhanced Lagrangian dual formulations for the class of minimax mixed-integer 0-1 problems in concert with deflected/conjugate subgradient algorithms. In addition, we propose two general purpose lifting mechanisms for tightening the mathematical programming formulations associated with such minimax optimization problems.

Next, we explore novel continuous nonconvex as well as lifted discrete formulations for the notoriously challenging class of job-shop scheduling problems with the objective of minimizing the maximum completion time (i.e., minimizing the makespan). In particular, we develop an RLT-enhanced continuous nonconvex model for the job-shop problem based on a quadratic formulation of the job sequencing constraints on machines. The tight linear programming relaxation that is induced by this formulation is then embedded in a globally convergent branch-and-bound algorithm. Furthermore, we design another novel formulation for the job-shop scheduling problem that possesses a tight continuous relaxation, where the non-overlapping job sequencing constraints on machines are modeled via a lifted asymmetric traveling salesman problem (ATSP) construct, and specific sets of valid inequalities and RLT-based enhancements are incorporated to further tighten the resulting mathematical program. The efficacy of our enhanced models is demonstrated by an extensive computational experiment using classical benchmark problems from the literature. Our results reveal that the LP relaxations produced by the lifted ATSP-based models provide very tight lower bounds, and directly yield a 0% optimality gap for many benchmark problems, thereby substantially dominating other alternative mixed-integer programming models available
for this class of problems. Notably, our lifted ATSP-based formulation produced a 0% optimality gap via the root node LP relaxation for 50% of the classical problem instances due to Lawrence [74].

We also investigate enhanced model formulations and specialized, efficient solution methodologies for applications arising in four particular industrial and sports scheduling settings. The first of these was posed to us by a major trucking company (Volvo Logistics North America), and concerns an integrated assembly and routing problem, which is a unique study of its kind in the literature. In this context, we examine the general class of logistical systems where it is desirable to appropriately ascertain the joint composition of the sequences of vehicles that are to be physically connected along with determining their delivery routes. Such assembly-routing problems occur in the truck manufacturing industry where different models of vehicles designed for a network of customers need to be composed into compatible groups (assemblies) and subsequently dispatched via appropriately optimized delivery routes that are restricted by the particular sequence in which the trucks are connected. A similar structure is exhibited in the business of shipping goods via boat-towed barges along inland waterways, or via trains through railroad networks. We present a novel modeling framework and column generation-based optimization approach for this challenging class of joint vehicle assembly-routing problems. In addition, we suggest several extensions to accommodate particular industrial restrictions where assembly sequence-dependent delivery routes are necessary, as well as those where limited driver- and equipment-related resources are available. Computational experience is provided using large-scale realistic data to demonstrate the applicability of our suggested methodology in practice.

The second application addressed pertains to a production planning problem faced by a major motorcycle manufacturing firm (Harley Davidson Motor Company). We consider the problem of partitioning and sequencing the production of different manufactured items in mixed-model assembly lines, where each model has various specific options and designated destinations. We propose a mixed-integer programming formulation (MPSP1) for this problem that sequences the manufactured goods within production batches in order to balance the motorcycle model and destination outputs as well as the load demands on material and labor resources. An alternative (relaxed) formulation (MPSP2) is also presented to model a closely related case where all production decisions and outputs are monitored within a common sequence of batches, which permits an enhanced tighter representation via an additional set of hierarchical symmetry-defeating constraints that im-
part specific identities amongst batches of products under composition. The latter model inspires a third set partitioning-based formulation in concert with an efficient column generation approach that directly achieves the joint partitioning of jobs into batches along with ascertaining the sequence of jobs within each composed batch. Finally, we investigate a subgradient-based optimization strategy that exploits a non-differentiable optimization formulation, which is prompted by the flexibility in the production process as reflected in the model via several soft-constraints, thereby providing a real-time decision-making tool. Computational experience is presented to demonstrate the relative effectiveness of the different proposed formulations and the associated optimization strategies for solving a set of realistic problem instances.

The third application pertains to the problem of matching or assigning subassembly parts in assembly lines, where we seek to minimize the total deviation of the resulting final assemblies from a vector of nominal and mean quality characteristic values. We introduce three symmetry-defeating enhancements for an existing assignment-based model, and highlight the critical importance of using particular types of symmetry-defeating hierarchical constraints that preserve the model structure. We also develop an alternative set partitioning-based formulation in concert with a column generation approach that efficiently exploits the structure of the problem. A special complementary column generation feature is proposed, and we provide insights into its vital role for the proposed column generation strategy, as well as highlight its benefits in the broader context of set partitioning-based formulations that are characterized by columns having relatively dense non-zero values. In addition, we develop several heuristic procedures. Computational experience is presented to demonstrate the relative effectiveness of the different adopted strategies for solving a set of realistic problem instances.

Finally, we analyze a doubles tennis scheduling problem in the context of a training tournament as prompted by a tennis club in Virginia, and develop two alternative 0-1 mixed-integer programming models, each with three different objective functions that attempt to balance the partnership and the opponentship pairings among the players. Our analysis and computational experience demonstrate the superiority of one of these models over the other, and reflect the importance of model structure in formulating discrete optimization problems. Furthermore, we design effective symmetry-defeating strategies that impose certain decision hierarchies within the models, which serve to significantly enhance algorithmic performance. In particular, our study provides the
insight that the special structure of the mathematical program to which specific tailored symmetry-defeating constraints are appended can greatly influence their pruning effect. We also propose a novel nonpreemptive multi-objective programming strategy in concert with decision hierarchies, and highlight its effectiveness and conceptual value in enhancing problem solvability. Finally, four specialized heuristics are devised and are computationally evaluated along with the exact solution schemes using a set of realistic practical test problems.

Aside from the development of specialized effective models and algorithms for particular interesting and challenging applications arising in different assembly, routing, and scheduling contexts, this dissertation makes several broader contributions that emerge from the foregoing studies, which are generally applicable to solving formidable combinatorial optimization problems. First, we have shown that it is of utmost importance to enforce symmetry-defeating constraints that preserve the structure of mathematical programs to which they are adjoined, so that their pruning effects are most efficiently coupled with the branch-and-bound strategies that are orchestrated within mathematical programming software packages. In addition, our work provides the insight that the concept of symmetry compatible formulations plays a crucial role in the effectiveness of implementing any particular symmetry-defeating constraints. In essence, if the root node LP solution of the original formulation does not conform relatively well with the proposed symmetry-defeating hierarchical constraints, then a significant branching effort might be required to identify a good solution that is compatible with the pattern induced by the selected symmetry-defeating constraints. Therefore, it is advisable to enforce decision hierarchies that conform as much as possible with the problem structure as well as with the initial LP relaxation.

Second, we have introduced an alternative concept for defeating symmetry via augmented objective functions. This concept prompts the incorporation of objective perturbation terms that discriminate amongst subsets of originally indistinguishable solution structures and, in particular, leads to the development of a nonpreemptive multiobjective programming approach based on, and combined with, symmetry-defeating constraints. Interestingly, nonpreemptive multiobjective programming approaches that accommodate symmetry-defeating hierarchical objective terms induce a root node solution that is compatible with the imposed symmetry-defeating constraints, and hence affords an automated alternative to the aforementioned concept of symmetry compatible formulations.
Third, we have proposed a new idea of complementary column generation in the context of column generation approaches that generally provide a versatile framework for analyzing industrial-related, integrated problems that involve the joint optimization of multiple operational decisions, such as assembly and routing, or partitioning and scheduling. In such situations, we have reinforced the insight that assignment-related problems that involve collections of objects (production batches, final assemblies, etc.) whose permutation yields equivalent symmetric solutions may be judiciously formulated as set partitioning models. The latter can then be effectively tackled via column generation approaches, thereby implicitly obviating the foregoing combinatorial symmetric reflections through the dynamic generation of attractive patterns or columns. The complementary column generation feature we have proposed and investigated in this dissertation proves to be particularly valuable for such set partitioning formulations that involve columns having relatively dense non-zero values. The incorporation of this feature guarantees that every LP iteration (involving the solution of a restricted master program and its associated subproblem) systematically produces a consistent set of columns that collectively qualify as a feasible solution to the problem under consideration. Upon solving the problem to optimality as a linear program, the resultant formulation encompasses multiple feasible solutions that generally include optimal or near-optimal solutions to the original integer-restricted set partitioning formulation, thereby yielding a useful representation for designing heuristic methods as well as exact branch-and-price algorithms. In addition, using duality theory and considering set partitioning problems where the number of patterns needed to collectively compose a feasible solution is bounded, we have derived a lower bound on the objective value that is updated at every LP phase iteration. By virtue of this sequence of lower bounds and the availability of upper bounds via the restricted master program at every LP phase iteration, the LP relaxation of the set partitioning problem is efficiently solved as using a pre-specified optimality tolerance. This yields enhanced algorithmic performance due to early termination strategies that successfully mitigate the tailing-off effect that is commonly witnessed for simplex-based column generation approaches.

This work has been partially supported by the National Science Foundation under Grant DMI-0552676.
To my dear parents,

Saad and Hamida.
Acknowledgments

All my gratitude goes to my dear parents for their unconditional and unlimited love. No words can thank them enough for their invaluable care and the sacrifices they have made, and still make, for me and our family. I have an everlasting debt to them, and to them I dedicate my work and love.

I also would like to express my heartfelt gratitude to Dr. Hanif D. Sherali who has been an inspiring role model for excellence in teaching, research, graduate advising, and refined manners. His professionalism, his scholarly standards, his kindness, subtly flavored with a great sense of humor and humility, make Dr. Sherali an excellent advisor and a paternal mentor to me. I also would like to extend my sincere appreciation to Dr. Sherali and Mrs. Sherali for their kindness and great hospitality.

My committee members, Dr. Subhash C. Sarin, Dr. C. Patrick Koelling, and Dr. Ebru Bish, have kindly contributed to my graduate education. May they be thanked for their valuable input and support. I am also beholden to all my teachers and professors who have accompanied me in my educational journey.

I am very much obliged to Prof. Claude Wolff for his kind support, and I greatly value the friendship that relates our families.

I would like to acknowledge my friends and colleagues: Bacel Maddah, Rami Musa, Waleed Nasr, Churlzu Lim, Mik McCrea, Xiaomei Zhu, Jita Desai, Chengbin Zhu, Sherif Abd El-Hamid, Molataf Al-Aromah, and all those who have made my stay at Virginia Tech enriching and enjoyable.
My warmest thoughts go to my siblings Mohammad, Fatimah, and Mahmoud, for what unites our hearts and lives.

I would like to express my kindest feelings to my wife Zeinab for her prayers and love, and for the peace, happiness, and light she brings to my heart.
# Contents

Acknowledgments ix

1 Introduction 1
  1.1 Motivation ................................................. 1
  1.2 Scope of Research ......................................... 2
  1.3 Organization of the Dissertation ......................... 4

2 Literature Review 6
  2.1 Theoretical and Application Areas ......................... 6
    2.1.1 Minimax Optimization Problems ....................... 6
    2.1.2 The Job-shop Scheduling Problem ...................... 7
    2.1.3 Assembly Line Balancing and Scheduling ................ 9
  2.2 Methodologies ............................................. 11
    2.2.1 The RLT Methodology: Enhanced Model Representations .......... 11
    2.2.2 Column Generation ...................................... 13

3 RLT-enhancements for Minimax Optimization Problems 15
  3.1 Introduction ............................................... 15
  3.2 Level-1 RLT Relaxation ...................................... 16
  3.3 Lagrangian Dual Formulations ................................ 19
  3.4 Deflected/Conjugate Subgradient Algorithms .................. 23
  3.5 Lifting Procedures ......................................... 23
    3.5.1 RLT-based Lifting Procedure ............................. 23
    3.5.2 Sequential Lifting Procedure ............................ 26
  3.6 Computational Experience ................................... 28
3.7 Conclusions and Directions for Future Research ........................................ 30

4 RLT-enhanced and Lifted Formulations for the Job-shop Scheduling Problem 31
4.1 Introduction ................................................................................................. 32
4.2 Notation and Some Early Models ................................................................. 33
  4.2.1 Notation .................................................................................................. 33
  4.2.2 Manne’s Model (1960) ......................................................................... 34
  4.2.3 Valid Inequalities in the Literature ....................................................... 35
4.3 RLT-based Continuous Model and Linear Lower Bounding Problem .......... 36
  4.3.1 RLT-based Relaxation .......................................................................... 37
  4.3.2 Lagrangian Dual Formulations .............................................................. 39
4.4 Global Optimization Algorithm .................................................................. 41
  4.4.1 Branch-and-bound Algorithm .............................................................. 41
  4.4.2 Preprocessing and Inference Rules ....................................................... 43
4.5 Lifted ATSP-based Formulations ................................................................ 43
4.6 Computational Experience ......................................................................... 49
  4.6.1 Continuous Relaxations ....................................................................... 49
  4.6.2 One-machine Relaxation ..................................................................... 51
  4.6.3 Nondifferentiable Optimization ............................................................ 56
  4.6.4 Performance of the B&B Algorithm .................................................... 57
4.7 Conclusions and Directions for Future Research ....................................... 57

5 Joint Vehicle Assembly-Routing Problems: An Integrated Modeling and Opti-
mization Approach ...................................................................................... 59
5.1 Introduction and Motivation ...................................................................... 59
5.2 Problem Description and Notation ............................................................. 62
5.3 Mathematical Model and Optimization Algorithm ........................................ 66
  5.3.1 Main Model: Set Partitioning Problem (SPP) .................................... 67
  5.3.2 Pattern Generation and Algorithmic Scheme ....................................... 67
  5.3.3 Solution to the Subproblem SP(̄π, ̄π0) ................................................ 69
5.4 Enhanced Models ....................................................................................... 75
  5.4.1 Assembly Sequence-dependent Routing ............................................. 75
  5.4.2 Driver- and Equipment-based Restrictions ........................................ 77
5.5 Computational Experience ........................................ 80
5.5.1 Basic VARP Model ........................................... 81
5.5.2 Elementary Clustering Strategies .............................. 86
5.5.3 Assembly Sequence-dependent Delivery Routes ............... 87
5.5.4 Equipment-based Restrictions ................................ 89
5.5.5 Rolling-horizon Framework ................................... 89
5.6 Summary, Conclusions, and Future Research ..................... 90

6.1 Introduction and Motivation ...................................... 95
6.2 Problem Description and Notation ................................ 98
   6.2.1 General Notation ........................................... 99
6.3 Mathematical Programming Formulations ......................... 101
   6.3.1 Primary Model Formulation MPSP1 .......................... 101
   6.3.2 Alternative Relaxed Formulation MPSP2 Under Common Monitoring Batches 104
6.4 Integrated Model and Column Generation Approach ............. 106
   6.4.1 Set Partitioning-based Formulation ......................... 106
   6.4.2 Column Generation Approach .............................. 107
6.5 Subgradient Optimization Algorithmic Approach .................. 112
6.6 Computational Experience ........................................ 118
   6.6.1 Mathematical Programming Formulations .................... 118
   6.6.2 Performance of the Column Generation Heuristic CG ........ 121
   6.6.3 Subgradient Optimization Performance ..................... 126
6.7 Summary and Directions for Future Research ..................... 126

7 Models and Algorithms for the Subassembly Parts Assignment Problem 128
7.1 Introduction and Motivation ..................................... 128
7.2 Problem Modeling ................................................. 130
   7.2.1 Notation .................................................. 130
   7.2.2 Assignment-based Models ................................ 131
   7.2.3 Set Partitioning-based Formulation ......................... 133
7.3 Heuristics ....................................................... 133
A.2 Bowman’s Model (1959) ................................................................. 189
A.3 Pritsker et al.’s Model (1969) ....................................................... 190
A.4 Von Lanzenauer and Himes’ Model (1970) ................................. 190
A.5 Wilson’s Model (1989) ................................................................. 192
A.6 Morton and Pentico’s Model (1993) ............................................. 192

Bibliography ............................................................................. 193

Vita ....................................................................................... 204
List of Figures

3.1 Illustrative example of a minimax problem. .............................................. 17
3.2 Flow-chart for the Lagrangian dual optimization algorithm. ........................ 24
3.3 Flow-chart for the RLT-based lifting procedure. ........................................ 26

5.1 Truck loading example ................................................................. 61
5.2 Illustration of subproblem concepts and notation ........................................ 73
5.3 Delivery regions and clusters ............................................................. 81
5.4 Algorithm convergence using the basic model, Instance 5, $\theta_{LF} = 0$, and Strategy 1 . 85

6.1 Flow-chart for the proposed deflected subgradient optimization algorithm .................. 119
6.2 Convergence of Heuristic CG for instance 7, $(n, \omega) = (40,5)$ using $(\tau, \Omega) = (300, 10800)$ 124

7.1 Average CPU time for Model SPAP2 as a function of the data variance using Set 2 . 140
7.2 CPU time growth for Procedure SPH-A as a function of $Q$ using $\varepsilon = 10^{-2}$ .......................... 148
7.3 Convergence of Procedure SPH-A for problem instance $(G, P, Q) = (10,10,1)$ using
$\varepsilon = 10^{-2}$ .................................................................................. 148
7.4 Convergence of Procedure SPH-A for problem instance $(G, P, Q) = (10,10,2)$ using
$\varepsilon = 10^{-2}$ .................................................................................. 149
7.5 Convergence of Procedure SPH-A for problem instance $(G, P, Q) = (10,10,3)$ using
$\varepsilon = 10^{-2}$ .................................................................................. 149
7.6 Convergence of Procedure SPH-A for problem instance $(G, P, Q) = (10,10,4)$ using
$\varepsilon = 10^{-2}$ .................................................................................. 150
8.1 CPU time growth for $P_{2,1}^t(\pi_1, \pi_2)$ as a function of $r$ ................................. 170
List of Tables

3.1 Comparison of LP relaxations using RLT and lifting enhancements ........................................ 29

4.1 LP relaxations of JS-ATSP$i$, $i = 1,\ldots, 4$ .................................................................................. 52
4.2 LP relaxations for the problem instances of Lawrence, LA01-LA18 ............................................. 53
4.3 LP relaxations for the problem instances of Lawrence, LA19-LA36 ............................................. 54
4.4 LP relaxations for the problem instances of Lawrence, LA37-LA40 ............................................ 55
4.5 $\frac{n}{m}$ ratio for the LA instances that yielded a 0% optimality gap ............................................. 55
4.6 LP relaxations for the problem instance of Fisher and Thompson .................................................. 55
4.7 LP relaxations for the problem instances of Adams et al. .............................................................. 55
4.8 LP relaxations for the problem instances of Applegate and Cook ................................................ 56
4.9 Comparison of various relaxations with the strongest cuts by Applegate and Cook ........................ 56
4.10 Comparison of the CPU times (secs) for MIP-NDO and Manne’s model ..................................... 56

5.1 Basic model, $\theta_{LF} = 0$ ................................................................................................................. 83
5.2 Basic model, $\theta_{LF} = 3$ ................................................................................................................. 83
5.3 Effect of $LF$ restrictions on the basic model .................................................................................... 83
5.4 Clustering strategy, basic model, $\theta_{LF} = 0$ .................................................................................. 86
5.5 Clustering benefits, basic model, $\theta_{LF} = 0$ .................................................................................. 86
5.6 Assembly sequence-dependent routing, $\theta_{LF} = 0$ .................................................................... 88
5.7 Assembly sequence-dependent routing, $\theta_{LF} = 3$ .................................................................... 88
5.8 Effect of $LF$-restrictions and assembly sequence-dependent delivery routes ............................... 88
5.9 Summary of $LF$ and assembly sequence-dependent delivery route effects ................................... 88
5.10 Equipment-based restrictions, assembly sequence-dependent routes, and $\theta_{LF} = 0$ ............. 89
5.11 Impact of planning horizon length ................................................................................................. 90

6.1 LP relaxations of the different proposed formulations .................................................................... 122
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.2</td>
<td>MIP performance of Models MPSP1, MPSP2, MPSP(σ), and Heuristic CG</td>
<td>122</td>
</tr>
<tr>
<td>6.3</td>
<td>Objective perturbation strategy for Models MPSP2(ϕ) and MPSP2(σ, ϕ)</td>
<td>123</td>
</tr>
<tr>
<td>6.4</td>
<td>Computational experience using Heuristic CG</td>
<td>125</td>
</tr>
<tr>
<td>6.5</td>
<td>Results for the subgradient optimization algorithmic approach</td>
<td>126</td>
</tr>
<tr>
<td>7.1</td>
<td>Assignment-based models using Set 1</td>
<td>140</td>
</tr>
<tr>
<td>7.2</td>
<td>Comparison of LP relaxations using Set 3</td>
<td>142</td>
</tr>
<tr>
<td>7.3</td>
<td>Comparison of LP relaxations using Set 4</td>
<td>143</td>
</tr>
<tr>
<td>7.4</td>
<td>Comparison of heuristics using Set 3</td>
<td>146</td>
</tr>
<tr>
<td>7.5</td>
<td>Comparison of heuristics using Set 4</td>
<td>147</td>
</tr>
<tr>
<td>8.1</td>
<td>Size comparison of the formulations based on Models 1 and 2</td>
<td>159</td>
</tr>
<tr>
<td>8.2</td>
<td>CPU times in seconds for LP relaxations, Set 2</td>
<td>168</td>
</tr>
<tr>
<td>8.3</td>
<td>Comparison of Models 1 and 2 solved to optimality</td>
<td>171</td>
</tr>
<tr>
<td>8.4</td>
<td>Results for (n, r, p) = (8,8,7)</td>
<td>172</td>
</tr>
<tr>
<td>8.5</td>
<td>Performance of Problem $P_{2,h}^t(\pi_1, \pi_2)$</td>
<td>173</td>
</tr>
<tr>
<td>8.6</td>
<td>Results for $AP^{\mu}(\pi_1, \pi_2)$, Set 1</td>
<td>174</td>
</tr>
<tr>
<td>8.7</td>
<td>Comparison between heuristics, Set 1</td>
<td>176</td>
</tr>
<tr>
<td>8.8</td>
<td>Comparison between heuristics, Set 2</td>
<td>177</td>
</tr>
<tr>
<td>8.9</td>
<td>Percentage of optimal solutions and computational savings</td>
<td>177</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Motivation

From the perspective of solving pure or mixed-integer programs to optimality, three crucial concepts come to prominence. First, it is fundamental to develop strong mathematical programming formulations that are nonetheless computationally tractable. As a consequence, the modeling phase is of utmost importance in operations research. This is particularly critical when the formulation involves discrete variables, or nonconvexity, in general. In such cases, constructing a model that is merely mathematically correct may not be sufficient to solve real-life instances of practical size; it must also possess appropriate structures and characteristics that promote its solvability in practice. The quality of a model is reflected by its structure, compactness, and tightness. The structure imparts appropriate mathematical characteristics that can be exploited in designing suitable solution procedures. The concept of compactness is related to the number of variables and constraints involved, whereas the tightness reflects how closely an underlying relaxation approximates the convex hull of feasible solutions - at least in the vicinity of optimal solutions. An ongoing research challenge consists in developing formulations that tend to achieve a good compromise between tightness and size.

Second, the recognition of inherent symmetries within mathematical programming formulations often proves to be critical when solving combinatorial optimization problems. This simple but powerful concept has, therefore, gained popularity in both mathematical programming (see Sherali and Smith [124]) and in constraint programming (see Kiziltan [67]) communities. Symmetry-defeating constraints impart specific identities to certain indistinguishable subsets of the model defining variables and, therefore, drastically improve algorithmic performance by obviating the wasteful exploration of equivalent symmetric reflections of various classes of solutions. Whenever possi-
ble, incorporating symmetry-breaking constraints within well-structured mathematical programs results in an enhanced pruning effect that can typically reduce the computational effort by several orders of magnitude. This feature is amply demonstrated in various parts of the present dissertation.

Finally, the pursuit of efficient solution methodologies and optimization strategies is a consideration of utmost importance that is intimately coupled with exploiting the structure and the strength of the mathematical programming formulations under investigation. Together, these concepts of relying on tight, compact formulations, symmetry-defeating decision hierarchies, and efficient solution methodologies for solving challenging classes of nonconvex or discrete optimization problems serve as the guiding motivation for the proposed research.

1.2 Scope of Research

This dissertation embraces and develops modeling approaches along with algorithmic strategies for operational and logistical decision problems that arise in various industrial systems. In particular, the focus of the proposed research lies in the development of enhanced formulations and effective solution methodologies for minimax and discrete optimization problems with applications to machine and sports scheduling, joint vehicle loading and routing, and subassembly part assignment problems.

First, we devote our attention to exploring the rich class of minimax mixed-integer 0-1 optimization problems in concert with the unified optimization framework offered by the Reformulation-Linearization Technique (RLT) of Sherali and Adams [108]. This class of problems encompasses a broad spectrum of applications that typically arise in the context of mechanical and design engineering, machine and sports scheduling, and facility location, to name a few. Many such problems are formulated as mathematical programs that are plagued by the weakness of their continuous relaxations, inducing a large optimality gap. Therefore, we explore the possibility of reducing this gap by employing the RLT methodology as an automatic partial convexification and cut generation scheme that strengthens the relationship between the minimax objective function and the model defining variables. Moreover, deflected/conjugate subgradient algorithms present attractive computational benefits that have been demonstrated in recent works for solving nondifferentiable models or reformulations of various optimization problems [76]. In particular, the exploration of RLT-enhanced Lagrangian dual formulations for the general class of minimax optimization problems, and its solving via suitable deflected/conjugate subgradient algorithms, offers promising horizons.
Amongst the plethora of minimax optimization problems that are discussed in the literature, we focus on the job-shop scheduling problem (JSSP) with the objective of minimizing the makespan, which is a notoriously difficult combinatorial optimization problem. The use of integer programming models for machine-sequencing problems emerged in the late nineteen-fifties. From the modeling perspective, these early works provided traditional mathematical programming formulations for the general JSSP. However, little progress was achieved with this trend because of the weakness of the underlying continuous relaxations of the formulated models and the tremendous computational effort required to solve the associated pure or mixed-integer programs. As a consequence, some authors (see French [52], for example) began labeling the application of mathematical programming techniques to the JSSP as a futile endeavor. This skepticism emanated from the intuition that casting the problem into a mathematical program without exploiting its intrinsic structure would only translate the problem formulation into an equivalently difficult one without alleviating any inherent complexity. However, recent developments in solving mixed-integer programs together with modern computer capabilities resurrect some hope in this direction and, with this motivation, we investigate various RLT-enhancements for discrete as well as continuous nonconvex formulations of this problem.

The second part of the proposed research addresses the development of enhanced models and solution methodologies for various industrial and logistical problems. First, we examine a novel class of joint vehicle assembly-routing problems that arises in the truck manufacturing industry as well as in the business of shipping goods via boat-towed barges along inland waterways, or via trains through railroad networks. For such logistical systems, it is desirable to appropriately ascertain the joint composition of the sequences of vehicles that are to be physically connected along with determining their delivery routes. We propose a set partitioning-based formulation for this class of problems that naturally lends itself to the development of effective column generation procedures. We discuss the existing and potential applications of the proposed integrated modeling and optimization approach, and demonstrate its applicability for large-scale problem instances inspired from our recent collaboration with a major trucking company (Volvo Logistics North America).

Following this, we treat, in turn, a scheduling problem and a subassembly part assignment problem that arise in the context of assembly lines. We address the problem of partitioning and sequencing the production of different manufactured items in the assembly lines of a major motorcycle firm (Harley Davidson Motor Company), where the models produced have specific options
and designations. We develop two related mixed-integer programming formulations as well as an integrated set partitioning-based formulation that is effectively solved via a column generation approach. Furthermore, we investigate a subgradient-based optimization strategy that exploits a non-differentiable optimization formulation, which is prompted by the flexibility in the production process as reflected in the model via several soft-constraints, thereby providing a real-time decision-making tool.

We then address a subassembly part assignment problem that poses an interesting and challenging combinatorial task within the production operation of assembly lines. Given the variation that is inherent to manufacturing processes, the aim here is to determine an optimal subassembly part assignment scheme for the different products to be assembled so as to minimize the total deviation from a vector of mean and nominal quality characteristic values. We discuss alternative symmetry-defeating strategies to enhance an existing assignment-based model, and develop a novel set partitioning-based formulation in concert with an efficient column generation algorithm that incorporates a complementary column generation feature.

Finally, we turn our attention to the arena of sports scheduling problem. We address the problem of scheduling a doubles tennis training tournament, which was prompted by a tennis club in Virginia, where it is desirable to build a fair schedule. In the context of this application, fairness is related to the partnership and the opponentship frequency for each pair of players. We develop two alternative mixed-integer programming models with minimax and minisum objective functions that seek to minimize the deviation from the mean partnership and opponentship frequencies. This application exemplifies the critical importance of model structure in formulating discrete optimization problems, and demonstrates the salutary effects of symmetry-defeating constraints on algorithmic performance.

1.3 Organization of the Dissertation

The remainder of this dissertation is organized as follows. Chapter 2 presents a brief overview of the literature pertaining to continuous and discrete minimax optimization, the RLT methodology, the job-shop scheduling problem, and assembly line balancing and scheduling problems. Chapter 3 examines the general class of minimax mixed-integer 0-1 programming problems for which we propose RLT-enhanced formulations and deflected/conjugate subgradient optimization strategies. In Chapter 4, we investigate a continuous nonconvex formulation for the JSSP using the RLT
methodology. Next, we develop tight RLT-enhanced Lagrangian dual formulations that are further improved via semidefinite cuts, and design a globally convergent optimization algorithm. Following this, we examine the development of enhanced continuous relaxations for the deterministic job-shop scheduling problem where the non-overlapping job sequencing constraints on machines are modeled via a lifted asymmetric traveling salesman problem formulation, and various sets of valid inequalities and RLT-based reformulations are proposed to further tighten the resulting mathematical programs. Promising results are presented in our preliminary computational experience. Chapter 5 introduces and discusses the novel class of joint vehicle-loading routing problems along with our integrated modeling and optimization approach. Chapter 6 is devoted to the development of a scheduling model and subgradient-based optimization approach for designing the assembly line production process for a major motorcycle firm. Chapter 7 addresses the subassembly parts assignment problem, and emphasizes the importance of symmetry-defeating decision hierarchies to improve an existing assignment-based model. Moreover, we propose an alternative set partitioning model that is effectively tackled using a column generation solution procedure that exploits the problem structure via a special complementary column generation feature. Chapter 8 analyzes a doubles tennis scheduling problem, and expounds on the fundamental importance of model structure and effective symmetry-defeating strategies.
Chapter 2

Literature Review

We provide in this chapter a brief overview of the literature pertaining to the theoretical and application areas which we investigate in the body of this research, as well as the predominant solution methodologies that are explored and adapted in the present work.

2.1 Theoretical and Application Areas

We discuss in this section the class of continuous and discrete minimax optimization, the state-of-the-art for the deterministic job-shop scheduling problem with the objective of minimizing the maximum completion time, and balancing and scheduling problems in assembly lines.

2.1.1 Minimax Optimization Problems

Ever since the pioneering minimax theorem by von Neumann [141], for what he then called games of strategy, several developments related to minimax theory have been presented in the arena of game theory and beyond. In particular, there has been a growing interest in the area of both continuous and discrete minimax optimization over the last few decades. The broad spectrum of existing results and applications of minimax theory in the field of optimization and game theory is captured in the book edited by Du and Pardalos [42].

In the continuous setting, Bazaraa and Goode [11] and Ahuja [3] have proposed algorithms for linear minimax optimization, whereas Dutta and Vidyasagar [44] and Vincent et al. [139] have investigated algorithms for various classes of nonlinear, continuous minimax optimization problems. In the context of discrete minimax optimization problems, Krarup and Pruzan [70] showed that minimax 0-1 optimization problems are equivalently reducible to minisum 0-1 problems and, as an immediate consequence, algorithms for $p$-median problems can tackle $p$-center problems.
upon transformation of the latter. Several discrete minimax optimization applications such as the minimax spanning tree, the minimax resource allocation, and various minimax production control problems, are discussed in [154].

2.1.2 The Job-shop Scheduling Problem

The job-shop scheduling problem (JSSP) arises in many industrial environments and presents a challenging combinatorial optimization problem. It aims to allocate the limited processing resources (machines) to jobs that have distinctive operations to be performed in a specified order over the machines, in order to optimize a particular objective function under certain constraints. Classical objective functions include minimizing: the total flow-time, the maximum completion time, the number of tardy jobs, or the maximum tardiness. In particular, minimizing the maximum completion time, also called the makespan and denoted $C_{\text{max}}$, has received popular attention. Further information can be specified to capture additional constraints that might be inherent in the job-shop environment under consideration, such as the presence of sequence-dependent set-up times. However, we restrict our attention in the present proposal to the problem where the objective is to minimize the makespan. In the traditional scheduling theory notation suggested by Graham et al. [56], this is designated as $(Jm//C_{\text{max}})$, where $Jm$ denotes a job-shop with $m$ machines; recirculation is not allowed, since it is not explicitly specified; there are no precedence constraints between the operations of different jobs; preemption is not allowed, i.e., operations cannot be interrupted, and each machine can process only one job at a time.

Over the last forty years or so, the deterministic JSSP has motivated a great deal of research, as recorded in the survey by Blazewicz et al. [20] and the state-of-the-art review by Jain and Meeran [63]. The computational intractability of this problem is illustrated by the fact that the 10-job-machine test problem FT10, introduced by Fisher and Thompson [50] in 1963, was provably solved to optimality for the first time by Carlier and Pinson [27] more than two decades later in 1989.

Specially-tailored branch-and-bound (B&B) algorithms are among the most successful approaches that have been proposed to solve the JSSP. Historically, the application of the B&B search technique to machine scheduling problems was first investigated in the sixties by Brooks and White [24], Ignall and Schrage [61], and Lomnicki [77]. The work of Balas [6] in the late sixties was among the first applications of the B&B technique to the JSSP. This was followed by efforts by Florian et al. [51], McMahon and Florian [85], and Lagweg et al. [71]. The culminating work by Carlier and Pinson [27] inspired Applegate and Cook [4] to devise a B&B algorithm based on
an edge-finding strategy. Perregaard and Clausen [98] composed parallel B&B approaches based on the algorithms developed by Carlier and Pinson [27] and Brucker et al. [25]. Alternative B&B techniques that do not rely on disjunctive graphs were developed by Martin [83] and Caseau and Laburthe [28]. Their approaches exploit a time-oriented representation of the job-shop problem and utilize constraint propagation techniques. Recently, Brinkkötter and Brucker [23] used a B&B algorithm coupled with a constraint propagation strategy to solve to optimality some benchmark problems for the first time. While improvements have been reported by some of the more recent B&B methods, these seem to be essentially due to modern computer capabilities rather than the techniques used, which follow traditional concepts and remain very computationally intensive for large problem instances.

The use of integer programming models for machine-sequencing problems emerged in the late fifties with Wagner’s model [142]. This was followed by other formulations due to Bowman [22] and the popular model by Manne [81]. Alternative (mixed-)integer 0-1 formulations were developed by Pritsker et al. [99], Von Lanzenauer and Himes [140], Wilson [151], and Morton and Pentico [91]. In addition, Nepomiastrchy [95] used nonlinear, nonconvex constraints in lieu of the machine disjunctive constraints within Manne’s model, and tackled the problem with penalty function methods that could lead to local, possibly non-global, solutions. In a similar fashion, Rogers [100] formulated these disjunctive restrictions via linear-quadratic constraints. Despite the elegance of these formulations, little progress has been achieved because of the tremendous computational effort required to solve such discrete and continuous nonconvex models, principally due to the weakness of the underlying relaxations. The need for deriving tighter lower bounds motivated research on cutting plane-based approaches. Applegate and Cook [4] offered an interesting analysis of lower bounds based on valid inequalities for both disjunctive and MIP formulations. The authors pointed out through their computational experiment that their cutting plane-based lower bounds dominate the classical preemptive and one-machine lower bounds for various problem instances reported, at the expense of a very significant increase in computational times, which renders this effort of little practical use thus far. They concluded by noting that it remains a research challenge to formulate classes of cutting planes that not only substantially strengthen the lower bound, but also can be integrated within a branch-and-cut algorithm to reduce the overall computational effort.
2.1.3 Assembly Line Balancing and Scheduling

Assembly lines play an important role in manufacturing environments where a commodity can be progressively assembled through a flow-oriented system. In such configurations, the total set of tasks to be performed on a product unit is partitioned into subgroups of tasks, called operations, each of which is then assigned to some workstation. Typically, the allotting of tasks to workstations must be achieved in compliance with the precedence relationships imposed by technological constraints, which are depicted using assembly precedence diagrams/graphs. The station time at any workstation refers to the total amount of processing time required for a product unit to be assembled at this workstation, and should not exceed the cycle time, i.e., the maximum or the average amount of time allowed to complete the operation at any workstation. Further characteristics of assembly lines, such as the existence of parallel lines or parallel workstations at some stages of a single line, the physical layout (serial or U-shaped line), the type of conveyors between workstations, the deterministic versus stochastic nature of task durations, or the possible introduction of buffers between workstations, are discussed in [15].

The global competition and the sophistication of the customer expectations have motivated the development of three main categories of assembly lines, namely, single-model, mixed-model, and multi-model lines [15]. Single-model assembly lines constitute the most primitive assembly systems, and are well-suited for homogeneous mass productions. In this context, the so-called simple assembly line balancing problem (SALBP) consists in optimally determining the number of workstations, the cycle time, and a line balance (a partitioning of tasks among workstations) with respect to some managerial target or objective function such as maximizing the line efficiency. In this perspective, it is often desirable to seek line balances that result in smoothing the loads among workstations (vertical balancing). The SALBP has been widely investigated, and Scholl and Becker [101] provide a recent survey of exact solution methods and heuristic approaches for this class of problems.

Due to the ever-increasing diversity in customer specifications, it has become crucial for manufacturers to offer multiple models, and possibly, various customized versions of any given model. This has led to the development of mixed-model assembly lines, where a heterogeneous sequence of products is manufactured along the same flow line. This is commonly witnessed in the automobile production business ([29], [96]) and similar industries [97]. Two intimately related problems, namely, the mixed-model assembly line balancing problem (MALBP) and the mixed-model sequencing problem (MSP), arise in such environments and have direct impacts on load-balancing issues.
and inefficiencies within an assembly line. The MALBP is a complex problem that arises while designing or reconfiguring a line in the light of a product mix. Given the distinct assembly precedence graph, along with the associated task durations that characterize each product, it is difficult, and highly unlikely, to identify a line balance that simultaneously smoothes station loads for every product, and station times at every workstation. Thus, in addition to vertical balancing, which focuses on station times for a given job, the concept of horizontal balancing has been introduced that deals with smoothing stations times of the different products for each workstation [89]. Secondary objective functions can be formulated to hedge against vertical or horizontal imbalances. Such objectives typically aim to minimize the total absolute deviation of actual station loads for a given job from its overall average station time (given by the ratio of the sum of task durations to the number of workstations), as in the early work by Thomopoulos [136], or to minimize the maximum deviation from the overall average station times over all jobs and workstations, as suggested by Decker [35]. The MSP addresses the problem of sequencing a set of heterogeneous jobs to be manufactured along the assembly line in order to optimize some performance measures. It should be noted that the job sequencing process, in conjunction with the vertical and horizontal balancing considerations, should be carefully addressed to avoid severe inefficiencies that translate in undesirable work overloads and idle times. Ideally, an integrated approach should jointly consider the MALBP and the MSP for a given system, and the line balance and product sequence should be periodically updated if the demand mix significantly changes over time. However, due to the complexity of each of these problems and the accompanying computational effort required, they are addressed separately in the literature, notwithstanding a few exceptions ([89],[136]). Methodologies reported for the MALBP include the transformation of the MALBP into an average SALBP with a combined precedence graph ([48], [137]), or the decomposition of the MALBP into parallel SALBP instances (one for each product), and the development of metaheuristic-based approaches [131], and branch-and-bound-based algorithms ([16], [26], [64]). Various mathematical programming models involving different objective functions have been suggested for the MSP, in concert with appropriate solution methodologies ([9], [35], [102], [153]).

In multi-model assembly lines, the problem is to compose and sequence batches of products that are separated with costly set-ups. In addition to balancing and sequencing considerations, lot sizing issues need to be handled in such assembly systems.
2.2 Methodologies

We briefly present the RLT methodology as a unified framework for developing hierarchies of partial convex hull approximations, and overview the column generation approach as a versatile search paradigm well-suited for large linear and integer-programming formulations that involve a colossal number of variables/columns.

2.2.1 The RLT Methodology: Enhanced Model Representations

The Reformulation-Linearization/Convexification Technique (RLT) methodology of [108] offers a unified approach to generate tight linear (or generally convex) programming representations for both discrete and continuous nonconvex problems. Originally, the motivation behind the development of this technique was to provide an automatic procedure for constructing a hierarchy of increasingly tighter relaxations and to afford strong valid inequalities for binary and zero-one mixed-integer linear programs ([106], [107]). This approach was further extended for the family of continuous, nonconvex polynomial programming problems [127], as well as for more general factorable programs [129].

There are two phases at the heart of the RLT approach. First, an automatic Reformulation Phase transforms the original model via a suitable multiplication of (a subset of) its constraints by certain nonnegative factors derived from the problem variables and constraints. The resulting nonlinear representation is subsequently re-linearized in the Linearization/Convexification Phase, where each distinct product term produced in the reformulation phase is substituted with a new continuous variable, in addition to utilizing logical deductions such as setting $x^2 = x$ for any binary variable $x$. Hence, the original problem formulation is recast into a higher-dimensional space representation, which affords tight linear programming (LP)-based lower bounds, often enhancing the solvability of the original problem.

We shall focus in this discussion on the RLT treatment for the family of mixed-integer 0-1 programs. Consider a mixed-integer 0-1 program that involves $n$ binary variables, $x_j$, and $m$ continuous variables, $y_k$. The RLT methodology generates a hierarchy of increasingly tight relaxations that progressively bridge the gap between the LP relaxation and the convex hull representation of the problem over $n$ levels. That is, the level-$n$ relaxation provides an explicit algebraic characterization of the convex hull of the problem. Although this theoretical result is very powerful, it should be noted that the sophistication of higher-level relaxations is achieved at the expense
of an exponential growth in the size of the problem. In fact, for the level-\( n \) relaxation the reformulation step involves \( 2^n \) nonnegative factors that are used to multiply each constraint. Due to the tremendous computational effort required for high levels of the RLT relaxations, except for special case applications, usually only the first- and the second-level relaxations are utilized to design algorithms for combinatorial optimization problems. This practical restriction should not be discouraging, however, because in many studies spanning a host of applications (see Adams and Johnson [2], Sherali and Alameddine [110], Sherali and Brown [111], and Sherali and Smith [123], for instance), the first-level RLT relaxation or even its partial form provides significantly tightened representations that have enabled the solution of previously unsolved problems. Moreover, this approach has been shown to be effective in tackling a broad spectrum of continuous and discrete nonconvex problems (besides mixed-integer zero-one programs).

Given a mixed-integer zero-one program having \( n \) zero-one variables, \( x_j \), and \( m \) continuous variables, \( y_k \), let \( X \) denote the feasible region of the problem. The level of the relaxation in the RLT hierarchy, \( d \) (\( 1 \leq d \leq n \)), corresponds to the degree of the polynomial factors constructed in the reformulation step. Basic polynomial factors, \( F_d(J_1, J_2) \), also referred to as bound-factor products, consist of the product of some \( d \) binary variables \( x_j \) or their complements \((1 - x_j)\): 

\[
F_d(J_1, J_2) = \left[ \prod_{j \in J_1} x_j \prod_{j \in J_2} (1 - x_j) \right],
\]

where \( J_1 \) and \( J_2 \) are index sets such that \( J_1, J_2 \subseteq \mathbb{N} \equiv \{1, \ldots, n\} \), \( J_1 \cap J_2 = \emptyset \), and \( |J_1 \cup J_2| = d \). The level-\( d \) RLT relaxation is achieved as follows.

- **Reformulation Step:** Construct all the level-\( d \) bound-factors \( F_d(J_1, J_2) \), and multiply each constraint defining \( X \) by these factors. Exploiting the binary nature of the variables \( x_j \), use the relationship \( x_j^2 = x_j \), \( j = 1, \ldots, n \), to simplify the resulting product terms in the nonlinear, polynomial mixed-integer zero-one representation that is obtained. In fact, this process is principally responsible for the tightening phenomenon.

- **Linearization Step:** Re-linearize the nonlinear polynomial mixed-integer zero-one formulation into a higher-dimensional space by replacing each nonlinear product term by a single continuous variable according to: 

\[
w_J = \prod_{j \in J} x_j \quad \text{and} \quad v_{Jk} = y_k \prod_{j \in J} x_j, \ \forall J \subseteq \mathbb{N}, 2 \leq |J| \leq \min\{d + 1, n\}.
\]

Let \( X_d \) be the resulting higher dimensional polyhedral set associated with the re-linearized problem, and let \( X_{Pd} \) represent its projection onto the space of the original variables. The progressive construction of the RLT hierarchy spans the spectrum between the basic LP relaxation (\( X_{P0} \)) and the convex hull (\( \text{conv}(X) \)) yielding:
\[ X_{P_0} \supseteq X_{P_1} \supseteq \ldots \supseteq X_{P_n} \equiv \text{conv}(X). \]

2.2.2 Column Generation

Column generation offers a powerful methodology to solve linear programming formulations that are characterized by a colossal number of columns/variables. This technique has gained wide popularity in recent years due to its ability to address large-scale MIP formulations via efficient heuristics or exact solution methods, as motivated in the insightful surveys by Desrosiers and Lübbecke [39], Lübbecke and Desrosiers [79], and Wilhelm [149]. A broad spectrum of applications prompt formulations where the entire set of valid columns ought not to be simultaneously considered for size considerations and computational efficiency. Whereas classical simplex-like approaches perform an explicit, enumerative pricing of non-basic variables to possibly detect enterable variables, the column generation technique favors an implicit pricing strategy, and exploits a cooperative interaction between a restricted master program (RMP) and a subproblem (SP). Such a progressive search strategy was recognized and orchestrated in the pioneering work by Dantzig and Wolfe [34].

In a classical column generation framework, the RMP is a linear program associated with an original master program (MP) that has an overwhelmingly large number of variables, and comprises a subset of generated columns that is dynamically updated. At any iteration, upon solving the RMP and passing the associated dual values to the SP, the latter triggers a pricing mechanism so as to construct a column that yields a most advantageous reduced cost. If the column constructed by the SP prices out favorably, it is appended to the RMP with the hope that this will induce improvements in the current objective value of the RMP. Otherwise, we conclude that the RMP has been solved to optimality. Here, if the original MP under investigation was itself continuous, the procedure terminates at this point, producing an LP optimal solution. However, in case the MP is an integer program, a (typically tight) lower bound is at hand (for minimization problems), and an additional integer-restricted step may be invoked to produce near-optimal solutions to the MP or, alternatively, a suitable branch-and-price [10] algorithm may be designed if a global optimal solution is desired.

Over the last decade, a substantial amount of research has been devoted to the use of column generation-based approaches for large-scale integer programs, and the discussion of the accompanying computational issues and challenges is maturing. An interesting feature inherent to column generation resides in its flexibility and versatility. Various strategies can be accommodated within the general column generation framework to enhance problem solvability or algorithmic perfor-
mance. For instance, the initialization of the RMP spans from using trivial columns that ignore the structure of the problem, to utilizing advanced starting solutions (generated via tailored heuristics or metaheuristics). In addition, whilst in our previous discussion of the pricing mechanism we suggested that the column under construction ought to be most advantageous from a reduced cost viewpoint, alternative variants may include the generation of a column that simply prices out favorably or even multiple columns. Similarly, depending on the application under investigation, the SP may be relaxed, solved using an $\varepsilon$-optimality tolerance criterion, or computationally bounded using a time-limit. For set partitioning problems, a class of problems that naturally lends itself to column generation approaches, a row aggregation of the RMP is proposed by Elhallaoui et al. [47] to hedge against the burdensome degeneracy generally observed when the columns have relatively dense nonzero values.

A tailing-off effect is commonly witnessed for simplex-based column generation approaches [39]. That is, the fast decrease in the RMP objective value that occurs in the early iterations is followed by persistently slower improvements that are pictorially reminiscent of a long-tail convergence process. It has also been observed that the dual values communicated to the SP tend to significantly oscillate during the iterations with no apparent structure. This erratic behavior is identified as a significant source of computational inefficiency. Stabilization devices, including trust region methods or perturbation strategies, have therefore been suggested to alleviate this problem (see Lübbecke and Desrosiers [79]). In a recent work, Subramanian and Sherali [132] have identified a phenomenon that they call dual noise, which explains the observed stalling of traditional column generation approaches predicated on set partitioning formulations of combinatorial problems. In particular, in the context of airline crew scheduling, they have designed a new deflected subgradient-based scheme that reduces the dual noise and has been demonstrated to significantly accelerate the convergence of the column generation algorithm in comparison with a commercial CPLEX-barrier-based column generation routine implemented at United Airlines.

We now proceed to present the main research component of this dissertation, providing additional discussion of the related literature for each specific topic addressed as relevant at the appropriate juncture.
Chapter 3

RLT-enhancements for Minimax Optimization Problems

This chapter addresses the development of enhanced formulations for the general class of minimax mixed-integer 0-1 optimization problems using the unified optimization framework offered by the Reformulation-Linearization Technique (RLT) of Sherali and Adams [106]. We also propose various Lagrangian dual formulations for the RLT-enhanced formulations. In addition, we investigate two general purpose lifting mechanisms for tightening the mathematical programming formulations associated with such minimax optimization problems.

3.1 Introduction

This chapter addresses the development of enhanced representations for the rich class of minimax mixed-integer 0-1 optimization problems that typically arise in the context of a broad spectrum of applications encompassing mechanical and design engineering, machine and sports scheduling, and facility location, to name a few. Many such problems are formulated as mathematical programs that are plagued by the weakness of their continuous relaxations, inducing a large optimality gap. Therefore, we explore the possibility of reducing this gap by employing the RLT methodology as well as some alternative lifting mechanisms to provide an automatic partial convexification and to generate valid inequalities that strengthen the relationship between the minimax objective function and the model defining constraints.

The remainder of this chapter is organized as follows. Section 3.2 presents the level-1 RLT relaxation for the general class of minimax mixed-integer 0-1 optimization problems. In Section 3.3, various Lagrangian dual formulations are investigated for such cut-enhanced level-1 RLT represen-
tations, and suitable deflected/conjugate subgradient strategies are explored for efficiently solving these bounding problems in Section 3.4. Section 3.5 describes an RLT-based as well as a sequential lifting procedure to tighten a given 0-1 mixed-integer minimax formulation, and Section 3.6 presents computational results.

3.2 Level-1 RLT Relaxation

Consider the following minimax mixed-integer 0-1 program (MMIP):

\[
\text{MMIP: Minimize } \{ \max_{r=1, \ldots, R} \{ \alpha_r x + \alpha r y \} \} \\
\text{subject to } (x, y) \in Z \subseteq \{(x, y) \in \mathbb{R}^n \times \mathbb{R}^m : 0 \leq x \leq e, 0 \leq y \leq u \} \\
x \text{ binary},
\]

where \( Z \) is a polyhedron. It is interesting to observe at the onset here that such a problem can be challenging, even for well-structured sets \( Z \). More specifically, it might be that \( Z \) is a tight approximation for \( Z_c \equiv \text{conv}\{(x, y) \in Z : x \text{ binary}\} \), whereby for linear objective functions, we would obtain tight bounds or near optimal solutions via the LP relaxation of (3.1). However, for the piecewise linear convex objective function (3.1a), the continuous solution no longer occurs at a vertex of \( Z \), and therefore, it can be far from a discrete optimum to this problem. The motivation here is to use RLT in order to construct tight relaxations of an equivalent linearized formulation of MMIP that enhances its solvability.

To illustrate this concept, consider the simple example given below:

\[
\text{Minimize } z \quad (3.2a) \\
\text{subject to } z \geq 2 - 2x \quad (3.2b) \\
z \geq 4x - 1 \quad (3.2c) \\
x \text{ binary}. \quad (3.2d)
\]

Figure 3.1(a) depicts the situation. The LP solution yields \( \bar{x} = 1/2 \) with \( \bar{z} = 1 \), while the integer optimum is \( x^* = 0 \) with \( z^* = 2 \), which portends a 50% optimality gap.
To apply RLT, let us multiply (3.2b) and (3.2c) by \( x \) and \( (1 - x) \), invoke the identity \( x^2 \equiv x \), and then substitute \( \xi \equiv zx \). This produces the set of constraints:

\[
\begin{align*}
[z \geq 2 - 2x] \ast x & \Rightarrow \xi \geq 0 \quad (3.3a) \\
[z \geq 2 - 2x] \ast (1 - x) & \Rightarrow z \geq \xi + 2(1 - x) \quad (3.3b) \\
[z \geq (4x - 1)] \ast x & \Rightarrow \xi \geq 3x \quad (3.3c) \\
[z \geq (4x - 1)] \ast (1 - x) & \Rightarrow z \geq \xi - (1 - x). \quad (3.3d)
\end{align*}
\]

Note that (3.3a) is implied by (3.3c) (along with \( x \geq 0 \)), and that the projection of (3.3b)-(3.3d) onto the \((x, z)\)-space yields upon using (3.3c) respectively in (3.3b) and (3.3d):

\[
z \geq 2 + x \text{ and } z \geq 4x - 1. \quad (3.4)
\]

Figure 3.1(b) depicts the consequent function \( \max\{2 + x, 4x - 1\} \) given via (3.4). Observe that the continuous minimax solution now yields the integer optimum for this problem.

![Figure 3.1](image.png)

(a) System (3.2): LP Relaxation  
(b) System (3.4): RLT Representation

Figure 3.1: Illustrative example of a minimax problem.

In general, to construct the first-order RLT relaxation (RLT1, say), similar to the above example, we perform the following operations as in Sherali and Adams [107]. Consider the following equivalent linearized reformulation of MMIP.
MMIP: Minimize \( z \) \hspace{1cm} (3.5a)
subject to \( z \geq \alpha r_0 + \alpha r_1 x + \alpha r_2 y, \quad \forall r = 1, ..., R \) \hspace{1cm} (3.5b)
\((x,y) \in Z \subseteq \{(x,y) \in R^n \times R^m : 0 \leq x \leq e, 0 \leq y \leq u\}\) \hspace{1cm} (3.5c)
x binary. \hspace{1cm} (3.5d)

In the reformulation phase, we multiply each of the defining constraints in (3.5b,c) with the bound-factors \( x_i \) and \((1 - x_i)\), \( \forall i \), and set \( x_i^2 = x_i, \forall i \) (i.e., \( x_i(1 - x_i) = 0, \forall i \)). Next, in the linearization phase, we substitute

\[ \xi_i = zx_i, \forall i, \ w_{ij} = x_ix_j, \forall i < j, \ v_{ik} = x_iy_k, \forall i, k, \] \hspace{1cm} (3.6)

thereby yielding the following tightened problem representation. (Here, \([\cdot]_L\) denotes the linearization of \([\cdot]\) under (3.6) (or under the RLT variable substitution process in general), after setting \( x_i^2 \) equal to \( x_i, \forall i \).)

RLT1: Minimize \( z \) \hspace{1cm} (3.7a)
subject to \( z \geq \xi_i + [(1 - x_i)(\alpha r_0 + \alpha r_1 x + \alpha r_2 y)]_L, \quad \forall r = 1, ..., R, i = 1, ..., n \) \hspace{1cm} (3.7b)
\( \xi_i \geq [x_i(\alpha r_0 + \alpha r_1 x + \alpha r_2 y)]_L, \quad \forall r = 1, ..., R, i = 1, ..., n \) \hspace{1cm} (3.7c)
\((x, y, w, v) \in Z_{RLT}, \) \hspace{1cm} (3.7d)

where \( Z_{RLT} \) is the set resulting by applying the first-level RLT to the set \( Z \) as in Sherali and Adams [107]. Note that we might choose to apply this RLT process to only the bounding constraints \( 0 \leq x \leq e, 0 \leq y \leq u \) defining (3.1b) in case \( Z \) is a good approximation for the convex hull of the set defined by (3.1b) and (3.1c). Otherwise, RLT products could also be composed for the other constraints defining \( Z \). In any case, \( Z_{RLT} \) includes the restrictions

\( \{w_{ij} \leq x_i, \ w_{ij} \leq x_j, \ w_{ij} \geq x_i + x_j - 1, \text{ and } w_{ij} \geq 0\}, \forall i < j \) \hspace{1cm} (3.8a)
as well as

\( \{v_{ik} \leq y_k, \ v_{ik} \leq u_kx_i, \ v_{ik} \geq y_k + u_kx_i - u_k, \ v_{ik} \geq 0\}, \forall i, k. \) \hspace{1cm} (3.8b)
Remark 3.1. It is straightforward to generalize the application of RLT of any order \(d \in \{2, ..., n\}\), similar to that of order \(d = 1\) used above. For applying RLT of any such order \(d\), we would use factors \(f_d(J_1, J_2)\) of order \(d\) as defined in Sherali and Adams [107] to compose the products, where \(J_1, J_2 \subseteq N \equiv \{1, ..., n\}\), \(J_1 \cap J_2 = \emptyset\), \(|J_1 \cup J_2| = d\). In particular, in lieu of constraints (3.7b) and (3.7c), we would, in general obtain the following restrictions:

\[
[z f_d(J_1, J_2)]_L \geq [f_d(J_1, J_2)(\alpha r_0 + \alpha r_1 x + \alpha r_2 y)]_L, \ \forall \ r, \ \forall \ (J_1, J_2) \text{ of order } d. \tag{3.9}
\]

Note also that in applying RLT to the constraints \(z \geq \alpha r_0 + \alpha r_1 x + \alpha r_2 y, \ \forall \ r\), as well as to those defining \(Z\) in (3.1b), we could use the Special Structures RLT as developed in Sherali et al. [109] wherever applicable, or the reduced RLT strategy described in Sherali et al. [125] for relatively large-scale problems. We will henceforth focus on RLT1 defined by (3.7) above; extensions to these higher-order and special structured cases can be handled in a similar manner. \(\square\)

### 3.3 Lagrangian Dual Formulations

To discuss the application of Lagrangian duality/relaxation for efficiently solving RLT1, let us rewrite this problem as follows, where (3.10d-i) represent an expansion of (3.7d), and where we have used the layering strategy in (3.10e,f) and (3.10g,h) via the introduction of new variable vectors \(w'\) and \(v'\) in order to facilitate the derivation below.

**RLT1: Minimize** \(z\) \hspace{1cm} (3.10a)

subject to \(z \geq \xi_i + [(1 - x_i)(\alpha r_0 + \alpha r_1 x + \alpha r_2 y)]_L, \ \forall \ (r, i)\) \hspace{1cm} (3.10b)

\(\xi_i \geq [x_i(\alpha r_0 + \alpha r_1 x + \alpha r_2 y)]_L, \ \forall \ (r, i)\) \hspace{1cm} (3.10c)

\(Ax + By + Cw + Dv \geq b\) \hspace{1cm} (3.10d)

\(0 \leq w_{ij} \leq x_i, \ x_i + x_j - 1 \leq w'_{ij} \leq x_j, \ \forall \ i < j\) \hspace{1cm} (3.10e)

\(w_{ij} = w'_{ij}, \ \forall \ i < j\) \hspace{1cm} (3.10f)

\(0 \leq v_{ik} \leq u_k x_i, \ y_k + u_k x_i - u_k \leq v'_{ik} \leq y_k, \ \forall \ (i, k)\) \hspace{1cm} (3.10g)

\(v_{ik} = v'_{ik}, \ \forall \ (i, k)\) \hspace{1cm} (3.10h)

\(0 \leq x \leq e, \ 0 \leq y \leq u.\) \hspace{1cm} (3.10i)
Now, upon eliminating $\xi_i, \forall i$, in a Fourier-Motzkin fashion [12], we can equivalently write (3.10b) and (3.10c) as,

\begin{align}
 z &\geq \alpha_r x + \alpha_r y, \quad \forall r = 1, ..., R \\
 z &\geq (x_i(\alpha_{r'} + \alpha_{r'} x + \alpha_{r'} y))_L + [(1 - x_i)(\alpha_0 + \alpha_{r} x + \alpha_{r} y)]_L, \\
 &\forall (r, r'), r \neq r', \forall i. 
\end{align}

(3.11a)

Constraints (3.11b) can be further simplified to the form:

\begin{align}
 z &\geq (\alpha_r + \alpha_{r'} x + \alpha_{r'} y) + [x_i((\alpha_{r'} - \alpha_r) + (\alpha_{r'} x - \alpha r) x + (\alpha_{r'} - \alpha r) y)]_L, \quad \forall (r, r'), r \neq r', \forall i. 
\end{align}

(3.12)

The resulting reformulation is denoted by RLT2 and is stated in (3.13) below.

\textbf{RLT2:} Minimize \{ $z : (3.10a)-(3.10i)$, with (3.11a) and (3.12) replacing (3.10b)-(3.10c) \}.

(3.13)

Denoting Lagrange multipliers $\lambda_r$ associated with (3.11a), $\forall r$; $\lambda_{r'r'i}$ associated with (3.12), $\forall r \neq r', \forall i$; $\pi$ associated with (3.10d); $\mu_{ij}$ associated with (3.10f), $\forall i < j$, and $\gamma_{ik}$ associated with (3.10h), $\forall (i, k)$, we can formulate the following Lagrangian Dual to RLT2.

\textbf{LD1:} Maximize \{ $\theta(\lambda, \pi, \mu, \gamma) : \sum_r \lambda_r + \sum_{r'} \sum_{r \neq r'} \sum_i \lambda_{r'r'i} = 1, \lambda \geq 0, \pi \geq 0, \pi \geq 0, \quad (\mu, \gamma) \text{ unrestricted} \},

(3.14)

where,

\begin{align}
 \theta(\lambda, \pi, \mu, \gamma) &= \text{minimum} \{ \sum_r \lambda_r (\alpha_r x + \alpha_{r'} x + \alpha_{r'} y) \\
 &\quad + \sum_{r'} \sum_{r \neq r'} \sum_i \lambda_{r'r'i} ((\alpha_r x + \alpha_{r'} x + \alpha_{r'} y) \\
 &\quad + [x_i((\alpha_{r'} - \alpha_r) + (\alpha_{r'} x - \alpha r) x + (\alpha_{r'} - \alpha r) y)]_L) \}
\end{align}
\[ + \pi(b - Ax - By - Cw - Dv) \]
\[ + \sum_{i < j} \mu_{ij}(w_{ij} - w'_{ij}) + \sum_{i} \sum_{k} \gamma_{ik}(v_{ik} - v'_{ik}) \]  \hspace{1cm} (3.15a)

subject to
\[ 0 \leq w_{ij} \leq x_{i}, \quad x_{i} + x_{j} - 1 \leq w'_{ij} \leq x_{j}, \quad \forall i < j \]  \hspace{1cm} (3.15b)
\[ 0 \leq v_{ik} \leq u_{k}x_{i}, \quad y_{k} + u_{k}x_{i} - u_{k} \leq v'_{ik} \leq y_{k}, \quad \forall (i, k) \]  \hspace{1cm} (3.15c)
\[ 0 \leq x \leq e, \quad 0 \leq y \leq u. \]  \hspace{1cm} (3.15d)

**Remark 3.2.** Note that \( \theta \) is readily evaluated at any \((\lambda, \pi, \mu, \gamma)\) via (3.15) as follows. First, for each \( i < j \), examining the sign on the coefficient of \( w_{ij} \) in the overall objective expression (3.15a), we put \( w_{ij} = 0 \) if this coefficient is nonnegative and \( w_{ij} = x_{j} \) otherwise. Similarly, \( w'_{ij}, v_{ik}, \) and \( v'_{ik} \) are each equated to their lower or upper (variable) bounds given in (3.15b,c), depending on the signs of their respective coefficients in (3.15a). Having thus reduced the resulting objective function in (3.15a) to one in terms of \( x \) and \( y \), we find an optimal set of values for the \( x \) and \( y \) variables subject to the bounds (3.15d) in a similar fashion. This correspondingly yields the values for \( w, w', v, v' \), which were previously determined as functions of \( x \) and \( y \). Moreover, letting \((\overline{x}, \overline{y}, \overline{w}, \overline{w'}, \overline{v}, \overline{v'})\) be the optimum found for (3.15), a subgradient of \( \theta \) at the given dual solution \((\lambda, \pi, \mu, \gamma)\) has components given by the corresponding coefficients of \( \lambda, \pi, \mu, \gamma \) in (3.15a), evaluated at \((\overline{x}, \overline{y}, \overline{w}, \overline{w'}, \overline{v}, \overline{v'})\). \( \Box \)

**Remark 3.3.** Note that the deflected subgradient optimization scheme suggested for solving LD1 defined in (3.14) will require us to iteratively solve for the projection of a given \((\overline{\lambda}, \overline{\pi}, \overline{\mu}, \overline{\gamma})\) onto the feasible region \( \Lambda \) of (3.14), where
\[ \Lambda \equiv \{(\lambda, \pi, \mu, \gamma) : \sum_{r} \lambda_{r} + \sum_{r \neq r'} \sum_{i} \lambda_{rr'i} = 1, \lambda \geq 0, \pi \geq 0, \]
\[ (\mu, \gamma) \text{ unrestricted}\}. \]  \hspace{1cm} (3.16)

This projection, denoted \( P_{\Lambda}(\overline{\lambda}, \overline{\pi}, \overline{\mu}, \overline{\gamma}) \), is given by the solution to the problem:
\[ \text{Minimize} \ \{|| (\lambda, \pi, \mu, \gamma) - (\overline{\lambda}, \overline{\pi}, \overline{\mu}, \overline{\gamma}) ||^{2} : (\lambda, \pi, \mu, \gamma) \in \Lambda\}. \]  \hspace{1cm} (3.17)
Problem (3.17) can be readily solved following the variable-dimension sequential projection procedure described in Bitran and Hax [19] or Sherali and Shetty [122] (also, see Bazaraa et al. [13]).

Remark 3.4. Note that instead of generating the equivalent set of $O(R^2n)$ constraints (3.11a, 3.12) from the $O(Rn)$ constraints (3.10b,c), we could have left the latter constraints as they were, and accordingly, we could have formulated the following Lagrangian dual problem LD2. Denote $\alpha_{ri}$ and $\beta_{ri}$ as the Lagrange multipliers associated with (3.10b) and (3.10c), respectively, $\forall (r,i)$, and let $\pi, \mu,$ and $\gamma$ be the Lagrange multipliers designated as before with respect to the constraints (3.10d), (3.10f), and (3.10h), respectively. Then, we can formulate LD2 as follows.

LD2: Maximize \[
\theta'(\alpha, \beta, \pi, \mu, \gamma) : \sum_r \sum_i \alpha_{ri} = 1, \quad (3.18a)
\]
\[
\sum_r \beta_{ri} - \sum_r \alpha_{ri} = 0, \forall i, \quad (3.18b)
\]
\[(\alpha, \beta, \pi) \geq 0, (\mu, \gamma) \text{ unrestricted}, \quad (3.18c)\]

where $\theta'(\cdot)$ is evaluated as before by dualizing (3.10b), (3.10c), (3.10d), (3.10f), and (3.10h) using the appropriate Lagrange multipliers. However, note that similar to Remark 3.3 and Equation (3.17), we would now need to solve the projection problem onto the feasible region of (3.18). Although a specialized active set strategy can be devised for this problem (e.g., see Luenberger [80] or Bazaraa et al. [13], this would not be as efficient as solving (3.17). But again, this depends on the relative sizes of these problems.

Yet another alternative strategy is to introduce slack variables in the constraints (3.10b) and (3.10c) along with implied upper bounds on these slacks in the constraints of RLT1 (to be also included in the subproblem constraints). In this manner, the nonnegativity restrictions on $\alpha$ and $\beta$ would be relaxed in (3.18), the effect of the slacks then being to impose suitable penalties for violating these nonnegativity restrictions. Now, the projection problem can be more easily solved in closed-form, although the dual ascent scheme would need to more carefully control the coordination of these unrestricted Lagrange multipliers with the new penalty terms. Whereas this is open to computational investigation, since the Lagrangian dual optimization problem usually requires a fine-tuned process to solve it effectively, we will opt for the formulation LD1.
3.4 Deflected/Conjugate Subgradient Algorithms

In this section, we present an algorithmic framework that is adapted from the variable target value method (VTVM) due to Sherali et al. (1995), in order to optimize the Lagrangian dual formulation(s) described in the foregoing section. Toward this end, consider a Lagrangian dual problem LD in the following generic form:

$$\text{LD: Maximize } \{ \theta(\xi) : \xi \in \Lambda \}.$$  \hfill (3.19)

Figure 3.2 provides a flow-chart for our recommended procedure. Here, given any iterate $\xi_k$, we denote by $g_k$ the subgradient of $\theta$ at $\xi_k$. The direction $d_k$ adopted is a deflection of the subgradient direction using the Average Direction Strategy (ADS) of Sherali and Ulular [128], unless when the procedure is reset. Also, following Sherali and Choi [112], we obtain an estimate for the primal solution (denoted $\hat{\Phi}$ here) over the last 50 (or so) iterations of the procedure. Note that $\theta(\cdot)$ is evaluated via a subproblem defined in the space of the primal variables, denoted by $\Phi$ in this context. If one is interested in maintaining only the $(x,y)$- or $x$-part of the primal solution, $\Phi$ should be treated as this partial primal solution vector in this procedure. Finally, we denote by $v^*$ the primal objective function value for some heuristic solution, which is either separately generated \textit{a priori}, or is composed during the Lagrangian dual/relaxation solution process itself. Also, we shall explore alternative deflected subgradient schemes for optimizing the Lagrangian dual as expounded in Lim and Sherali [76] and Sherali and Lim [119].

3.5 Lifting Procedures

In this section, we describe two lifting mechanisms for tightening the representation of MMIP given by (3.1) in the space of the original $(x,y)$-variables, prior to solving the resulting augmented reformulation via a commercial package such as CPLEX-MIP. The first of these is based on RLT lifting concepts and its optimization via the Lagrangian dual formulation as discussed in Sections 3.2-3.4 above, while the second method utilizes a sequential lifting idea.

3.5.1 RLT-based Lifting Procedure
Suppose that we commence by solving the LP relaxation of Problem MMIP, denoted $\overline{\text{MMIP}}$. Naturally, if the resulting solution $(\bar{z}, \bar{x}, \bar{y})$ yields binary values for $\bar{x}$, then this is also optimal for MMIP. Otherwise, we have that $F \equiv \{ i : \bar{x}_i \text{ is fractional} \} \neq \emptyset$. Now, suppose that we construct
Initialization. Set $T_1 = v^*, \epsilon_0 = 10^{-6}, \epsilon = 0.1, \beta = 1.0, k_{\text{max}} = 2000, k = l = \tau = 1, \gamma = \Delta = 0, r = 0.05, \varphi = 75, \gamma \in [10, 20], \Phi = 0, K = 0, \eta \in [0.75, 0.95], F = 1, \mathcal{F} \in \{4, 5\}$

Pick $\xi_1 \in \Lambda$ (arbitrary, using zeros where possible). Evaluate $\theta(\xi_1)$, and $g_1$. Let $\hat{\xi} = \xi_1, \hat{\gamma} = g_1, \theta_1 = \theta(\xi_1), z_1 = \theta_1$. If $||g_1|| < \epsilon_0$, stop with $\xi_1$ as near optimal.

Put $\epsilon_1 = .15(T_1 - \theta_1)$. Let RESET = 1.

- $k > k_{\text{max}}$ (STOP)

Let $d_k = g_k + \frac{||g_k||}{||g_{k-1}||}d_{k-1}$ if RESET = 0, and $d_k = g_k$ if RESET = 1. If $||d_k|| < \epsilon_0$, put $d_k = g_k$.

Compute the step length $\lambda_k = \frac{\beta_0(T_1 - \theta_k)}{||d_k||^2}$. Set $\xi_{k+1} = P_{\lambda} [\xi_k + \lambda_k d_k]$. Evaluate $\theta_{k+1} \equiv \theta(\xi_{k+1})$ along with $g_{k+1}$. If $||g_{k+1}|| < \epsilon_0$, stop with $\xi_{k+1}$ as near optimal. If $k \geq (k_{\text{max}} - 50)$, replace $\hat{\Phi} \leftarrow \left( \frac{\Phi + \Phi}{k + 1} \right)$, where $\Phi$ is the (partial) primal solution obtained when evaluating $\theta(\xi_{k+1})$. Increment $K$ by 1. Put RESET = 0.

- $\theta_{k+1} > z_k$?

Put $z_{k+1} = \theta_{k+1}, \hat{\xi} = \xi_{k+1}, \hat{\gamma} = g_{k+1}$. Set $\gamma = 0, F = 0, \Delta = \Delta + (\theta_{k+1} - z_k)$.

- $z_{k+1} \geq T_l - \epsilon_l^2$?

Compute $T_{l+1} = \frac{(z_{k+1} + \epsilon_l) + T_l}{2}$

- $\mathcal{F} = \max\{\beta/2, 10^{-6}\}$

- $F \geq \mathcal{F}$?

- $r \leftarrow r/1.08$.

- Set $\epsilon_{l+1} = \max\{\epsilon, 0.15(T_{l+1} - z_{k+1})\}$

- Put $\tau = 0, \Delta = 0, l \leftarrow l + 1$.

Figure 3.2: Flow-chart for the Lagrangian dual optimization algorithm.
the formulation RLT2 given by (3.12), except that we consider only the variables \( x_i, i \in F \), as binary-valued and treat the remaining variables \( x_i, i \notin F \), as continuous, similar to the \( y \)-variables. Let this RLT2 representation be denoted as RLT2\((F)\), and be generically written as follows:

\[
\text{RLT2}(F): \text{Minimize } z
\]

\[
\text{subject to } z \geq \alpha_{r0} + \alpha_{r1}x + \alpha_{r2}y, \quad \forall r = 1, \ldots, R
\]

\[
(e)z \geq H_{11}x + H_{12}y + H_{13}\psi + h_1
\]

\[
G_1x + G_2y \geq g
\]

\[
H_{21}x + H_{22}y + H_{23}\psi \begin{pmatrix} \equiv \\ \geq \end{pmatrix} h_2
\]

\[
H_{31}x + H_{32}y + H_{33}\psi \geq h_3
\]

\[
0 \leq x \leq e, \quad 0 \leq y \leq u,
\]

where \( \psi \equiv (w, v, w', v') \) denotes the new RLT-variables created in this process, and where (3.20b) relates to (3.11a); (3.20c) relates to (3.12) with \((e)\) being a (conformable) vector of ones; (3.20d) relates to those restrictions in (3.10d) that involve only the original \((x, y)\)-variables; (3.20e) relates to the remaining part of (3.10d) as well as to the other constraints (including the equalities (3.10f,h)) that will be dualized in the construction of the associated Lagrangian dual formulation LD1; likewise, (3.20f) relates to the constraints (3.15b,c) that are retained in the subproblem for LD1, and (3.20g) relates to the remaining subproblem constraints (3.15d).

Next, we solve the resulting Lagrangian dual formulation LD1 to (near)-optimality using the deflected subgradient algorithm described in Section 3.4 to obtain the dual solution \((\lambda^*, \pi^*, \mu^*, \gamma^*)\), say. Denote by \( \phi_1 \geq 0, \phi_2, \) and \( \phi_3 \geq 0 \) the corresponding dual variables associated with the constraints (3.20c), (3.20e), and (3.20f), respectively, where \( \phi_1 \) and \( \phi_2 \) are appropriate sub-vectors of \((\lambda^*, \pi^*, \mu^*, \gamma^*)\), and where \( \phi_3 \) is obtained as the optimal dual solution to the subproblem constraints (3.15b,c) (including the nonnegativity restrictions contained therein) when evaluating \( \theta(\lambda^*, \pi^*, \mu^*, \gamma^*) \). Then, by surrogating (3.20c), (3.20e), and (3.20f) using the dual multipliers \( \phi_1, \phi_2, \) and \( \phi_3 \), respectively, and noting by duality that \( \phi_1^TH_{13} - \phi_2^TH_{23} - \phi_3^TH_{33} = 0 \), we obtain a valid inequality in the original \((z, x, y)\)-space as follows:

\[
(e^T\phi_1)z \geq [\phi_1^TH_{11} - \phi_2^TH_{21} - \phi_3^TH_{31}]x + [\phi_1^TH_{12} - \phi_2^TH_{22} - \phi_3^TH_{32}]y + [\phi_1^Th_1 + \phi_2^Th_2 + \phi_3^Th_3]. \quad (3.21)
\]
Solve MMIP to obtain an optimal solution \((\bar{z}, \bar{x}, \bar{y})\). Set \(K = 0\).

Let \(F = \{i: \bar{x}_i \text{ is fractional}\}\), and construct RLT2\((F)\) defined in (3.19) and solve its Lagrangian dual formulation LD1. Using the resulting dual multipliers obtained at termination, construct the valid inequality (3.21) and add this to MMIP. Let \(K \leftarrow K + 1\).

\(K = K_{\max}\?\)

Stop; \((\bar{z}, \bar{x}, \bar{y})\) solves MMIP.

Solve the resulting augmented MMIP using CPLEX-MIP.

Figure 3.3: Flow-chart for the RLT-based lifting procedure.

Note that (3.21) is a strongest surrogate type constraint in that by replacing (3.20c,e,f) by (3.21) within RLT2\((F)\), since the same dual solution (with a dual value of 1 with respect to (3.21)) remains feasible and yields the same objective value as \(\theta^* \equiv \theta(\lambda^*, \pi^*, \mu^*, \gamma^*)\), the resulting objective value would be at least \(\theta^*\), even when LD1 has been solved inexactly. Now, if \(e^T \phi_1 > 0\), we can scale (3.21) by dividing this inequality by \(e^T \phi_1\) and accommodate the new constraint within (3.5b). Else, if \(e^T \phi_1 = 0\), then we can accommodate (3.21) within (3.5c). In any case, this yields an augmented lifted version of MMIP. The foregoing process can now be repeated up to some \(K_{\max} \geq 1\) times, before solving the final reformulation using CPLEX-MIP. Figure 3.3 provides a flow-chart for this procedure, which we denote RLP.

### 3.5.2 Sequential Lifting Procedure

In this section, we describe a sequential lifting procedure (see Nemhauser and Wolsey [94] for this general concept), which we denote SLP, to tighten the formulation for MMIP. Toward this end, consider the \(r^{th}\) inequality in (3.5b) and let us illustrate the process of lifting this inequality using any variable \(x_i, i \in N \equiv \{1, ..., n\}\), to obtain the following augmented inequality, where \(\Delta_i \geq 0\):

\[
z \geq \alpha_{r0} + \alpha_{r1} x + \alpha_{r2} y + \Delta_i x_i.
\]  

\(3.22\)
When \( x_i = 0 \), we have that (3.21) is valid regardless of \( \Delta_i \), and when \( x_i = 1 \), for ensuring validity, we must have that

\[
(\alpha_{r1})_i + \Delta_i \leq z - \alpha_{r0} - \alpha_{r1}^{-i}x^{-i} - \alpha_{r2}y, \text{for any feasible solution to MMIP,}
\]  

(3.23)

where \( \alpha_{r1}^{-i} \) and \( x^{-i} \) denote the respective vectors \( \alpha_{r1} \) and \( x \) with the \( i^{th} \) component deleted. Hence, suppose that we solve the linear program:

\[
\delta_i = \min \{ z - \alpha_{r0} - \alpha_{r1}^{-i}x^{-i} - \alpha_{r2}y : z \geq \alpha_{r0} + \alpha_{r1}x + \alpha_{r2}y, \\
\forall r = 1, \ldots, R, (x, y) \in Z, x_i = 1, 0 \leq x \leq e \}. 
\]  

(3.24)

Note that by virtue of the constraint \( z \geq \alpha_{r0} + (\alpha_{r1})_i + (\alpha_{r1})^{-i}x^{-i} + \alpha_{r2}y \) being present in (3.24), we have that \( \delta_i \geq (\alpha_{r1})_i \). Moreover, by (3.23) it is valid to set \( (\alpha_{r1})_i + \Delta_i = \delta_i \) within (3.22), i.e.,

\[
\Delta_i = \delta_i - (\alpha_{r1})_i \geq 0. 
\]  

(3.25)

**Remark 3.5.** Note that when \( \alpha_{r2} = 0 \) and when \( \alpha_{r0} \) and \( \alpha_{r1}^{-i} \) are integer-valued, we can replace \( \delta_i \) by \( [\delta_i] \) in (3.25) in order to potentially improve (increase) the lifting coefficient \( \Delta_i \). □

By sequentially repeating this for each \( r = 1, \ldots, R \), and each \( x_i, i = 1, \ldots, n \), we thus construct an augmented lifted version of MMIP, which we subsequently solve using CPLEX-MIP. In this process, it might be beneficial to order the \( R \) constraints in (3.1a), using some priority scheme that sequences the more restrictive constraints first. Toward this end, based on the solution of the LP relaxation \( \overline{\text{MMIP}} \) of Problem MMIP, we re-index the \( R \) constraints (3.1) in nonincreasing order of the resulting dual variable values. Likewise, denoting \((\bar{z}, \bar{x}, \bar{y})\) as the optimal solution obtained for Problem \( \text{MMIP} \), we re-index the \( x \)-variables in nonincreasing order of their fractionality given by

\[
\min \{ x_i, (1 - x_i) \}, i = 1, \ldots, n. 
\]

Using the resulting re-indexed formulation of MMIP, we then apply the lifting procedure defined by (3.22) and (3.25) in the order \( r = 1, \ldots, R \) for each \( i = 1, \ldots, n \), replacing the current inequality with the lifted one for subsequent liftings at each step of this process. We illustrate this procedure using the previous example given in (3.2) below.

**Example 3.1.** Consider Problem MMIP given by (3.2), and let us examine (3.2b) in the spirit of (3.22). The lifted constraint is of the type \( z \geq 2 - 2x + \Delta x \), where, by (3.24) and (3.25), we take
\[ \Delta = \min \{ z - 2 : z \geq 2 - 2x, z \geq 4x - 1, x = 1, 0 \leq x \leq 1 \} - (-2) = 3. \]

This yields the lifted inequality \( z \geq 2 + x \), which is the required facet-inducing inequality given in (3.4). It is readily verified that no further lifting of (3.2c) results. Hence, the augmented MMIP formulation is given by (3.4), which yields the convex hull representation for this example. \( \square \)

**Remark 3.6.** Observe that a repeated application of the foregoing sequential lifting process can be quite time-consuming. Instead, we can combine this with the RLT-based lifting procedure by applying the sequential lifting technique only to those original inequalities (3.5b) or the newly generated constraints (3.21) obtained by the RLT-based procedure, which turn out to be active in the resulting final LP relaxation. We provide some computational evidence for this strategy in Section 3.6.

### 3.6 Computational Experience

We now provide some computational experience to study the solvability of Problem MMIP, as well as to examine the relative tightness of its associated RLT-1 enhanced formulation. We also report results pertaining to the proposed Lagrangian dual formulation, which is solved via the deflected/conjugate subgradient optimization scheme described in Section 3.4. Finally, we elaborate on the relative effectiveness of the two lifting procedures delineated in Section 3.5. We consider a test-bed comprised of twelve problem instances, where the size of any instance is characterized by the triplet \((R, n, m)\) as described below. For each problem size \((R, n, m) = (5, 10, 10), (10, 20, 10), (20, 20, 10),\) and \((20, 50, 20)\), three instances have been generated. The values of the coefficients \(\alpha_{r0}, \alpha_{r1},\) and \(\alpha_{r2}\) were generated using a uniform distribution over the interval \([-5, 10]\), and the upper bounds on the \(y\)-variables were generated using a uniform distribution over the interval \([0, 5]\).

We have used AMPL in concert with CPLEX 10.1 for implementation purposes, and all runs have been performed on a Dell Precision 650 workstation having a Xeon(TM) CPU 2.40 GHz processor and 1.50 GB of RAM.

The results reported in Table 3.1 suggest that Problem MMIP and the lifted formulation RLT1 achieve an average optimality gap of 15.7% and 3.9%, respectively, thereby demonstrating the strength of the level-1 RLT relaxation. In addition, Problem RLT1 presents an evident advantage over Problem LD1 and Procedure SLP with respect to both the quality of the solution produced and the accompanying CPU times.
<table>
<thead>
<tr>
<th>Problem, $\nu^*$, $(R, n, m)$</th>
<th>Metrics</th>
<th>MMIP</th>
<th>RLT1</th>
<th>LD1, $k_{max} \epsilon$ (500, 2000)</th>
<th>SLP</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.11</td>
<td>LP</td>
<td>7.94</td>
<td>8.11</td>
<td>(6.74, 6.74)</td>
<td>7.94</td>
</tr>
<tr>
<td>(5, 10, 10)</td>
<td>%Gap</td>
<td>2.08</td>
<td>0</td>
<td>(16.89, 16.89)</td>
<td>2.08</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0</td>
<td>0.01</td>
<td>(18.7, 74.1)</td>
<td>23.78</td>
</tr>
<tr>
<td>4.43</td>
<td>LP</td>
<td>4.25</td>
<td>4.41</td>
<td>(3.51, 3.51)</td>
<td>4.25</td>
</tr>
<tr>
<td>(5, 10, 10)</td>
<td>%Gap</td>
<td>4.10</td>
<td>0.35</td>
<td>(20.76, 20.76)</td>
<td>4.10</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0</td>
<td>0.03</td>
<td>(24.3, 95.5)</td>
<td>27.72</td>
</tr>
<tr>
<td>7.33</td>
<td>LP</td>
<td>6.85</td>
<td>7.33</td>
<td>(6.38, 6.38)</td>
<td>6.85</td>
</tr>
<tr>
<td>(5, 10, 10)</td>
<td>%Gap</td>
<td>6.49</td>
<td>0</td>
<td>(12.96, 12.96)</td>
<td>6.49</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0</td>
<td>0.01</td>
<td>(27.7, 91.8)</td>
<td>26.90</td>
</tr>
<tr>
<td>4.8</td>
<td>LP</td>
<td>2.40</td>
<td>3.64</td>
<td>(0.22, 0.22)</td>
<td>2.40</td>
</tr>
<tr>
<td>(10, 20, 10)</td>
<td>%Gap</td>
<td>49.99</td>
<td>23.97</td>
<td>(95.41, 95.41)</td>
<td>49.99</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0</td>
<td>0.68</td>
<td>(508.6, 1990.7)</td>
<td>126.42</td>
</tr>
<tr>
<td>4.41</td>
<td>LP</td>
<td>3.60</td>
<td>4.36</td>
<td>(2.01, 2.22)</td>
<td>3.60</td>
</tr>
<tr>
<td>(10, 20, 10)</td>
<td>%Gap</td>
<td>18.31</td>
<td>1.13</td>
<td>(54.42, 49.65)</td>
<td>18.31</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0</td>
<td>0.68</td>
<td>(562.0, 2194.9)</td>
<td>126.33</td>
</tr>
<tr>
<td>7.62</td>
<td>LP</td>
<td>5.90</td>
<td>7.33</td>
<td>(3.52, 3.52)</td>
<td>5.90</td>
</tr>
<tr>
<td>(10, 20, 10)</td>
<td>%Gap</td>
<td>22.60</td>
<td>3.82</td>
<td>(53.80, 53.80)</td>
<td>22.60</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0</td>
<td>0.48</td>
<td>(562.0, 2194.9)</td>
<td>120.17</td>
</tr>
<tr>
<td>4.7</td>
<td>LP</td>
<td>6.77</td>
<td>7.69</td>
<td>(5.2, 6.10)</td>
<td>6.77</td>
</tr>
<tr>
<td>(20, 20, 10)</td>
<td>%Gap</td>
<td>15.25</td>
<td>3.77</td>
<td>(35, 23.75)</td>
<td>15.25</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0</td>
<td>2.14</td>
<td>(10004.7, 44374.9)</td>
<td>280.07</td>
</tr>
<tr>
<td>8.23</td>
<td>LP</td>
<td>5.99</td>
<td>7.14</td>
<td>(3.53, -)</td>
<td>5.99</td>
</tr>
<tr>
<td>(20, 20, 10)</td>
<td>%Gap</td>
<td>17.09</td>
<td>1.18</td>
<td>(51.17, -)</td>
<td>17.09</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0</td>
<td>1.11</td>
<td>(695.5, -)</td>
<td>75.17</td>
</tr>
<tr>
<td>9.24</td>
<td>LP</td>
<td>8.44</td>
<td>9</td>
<td>-</td>
<td>8.44</td>
</tr>
<tr>
<td>(20, 20, 10)</td>
<td>%Gap</td>
<td>6.20</td>
<td>0</td>
<td>-</td>
<td>6.20</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0</td>
<td>0.78</td>
<td>-</td>
<td>79.06</td>
</tr>
<tr>
<td>10</td>
<td>LP</td>
<td>6.97</td>
<td>8.04</td>
<td>-</td>
<td>6.97</td>
</tr>
<tr>
<td>(20, 50, 20)</td>
<td>%Gap</td>
<td>13.32</td>
<td>0</td>
<td>-</td>
<td>13.32</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0</td>
<td>140.93</td>
<td>-</td>
<td>346.11</td>
</tr>
<tr>
<td>11</td>
<td>LP</td>
<td>6.94</td>
<td>7.63</td>
<td>-</td>
<td>6.94</td>
</tr>
<tr>
<td>(20, 50, 20)</td>
<td>%Gap</td>
<td>20.86</td>
<td>12.99</td>
<td>-</td>
<td>20.86</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0</td>
<td>229.73</td>
<td>-</td>
<td>382.35</td>
</tr>
<tr>
<td>12</td>
<td>LP</td>
<td>6.48</td>
<td>7.36</td>
<td>-</td>
<td>6.48</td>
</tr>
<tr>
<td>(20, 50, 20)</td>
<td>%Gap</td>
<td>12.11</td>
<td>0.13</td>
<td>-</td>
<td>12.11</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0</td>
<td>232.92</td>
<td>-</td>
<td>517.96</td>
</tr>
</tbody>
</table>

Table 3.1: Comparison of LP relaxations using RLT and lifting enhancements
3.7 Conclusions and Directions for Future Research

We have conducted an extensive polyhedral analysis of the general class of MMIP problems in order to tighten its representation using the Reformulation-Linearization Technique (RLT) of Sherali and Adams [106]. Many such MMIP problems are formulated as mathematical programs that are plagued by the weakness of their continuous relaxations, inducing a large optimality gap. Therefore, we explored the possibility of reducing this gap by employing the RLT methodology as well as some alternative lifting mechanisms to provide an automatic partial convexification and to generate valid inequalities that serve to strengthen the relationship between the minimax objective function and the model defining constraints. We have demonstrated that the level-1 RLT relaxation significantly tightens the MMIP representation, thereby expanding the spectrum of nonconvex problems for which the application of low-level RLT relaxations, even in partial form, has enabled improved, practical solution approaches. In addition, we have proposed RLT-enhanced Lagrangian dual formulations for this class of problems in concert with a suitable deflected/conjugate subgradient algorithm. The latter are further exploited within a novel RLT-based lifting procedure (RLP) that sequentially augments the MMIP formulation with strongest surrogate type constraints. Moreover, we have developed a sequential lifting procedure (SLP) that iteratively strengthens each “minimax constraint” via the incorporation of a nonnegative term that involves a single model-defining binary variable. It is interesting to observe that for a test-bed of randomly generated problem instances the level-1 RLT relaxation consistently outperformed Procedure SLP by yielding tighter lower bounds, while achieving appreciable computational savings.
Chapter 4

RLT-enhanced and Lifted Formulations for the Job-shop Scheduling Problem

In this chapter, we propose novel continuous nonconvex as well as lifted discrete formulations of the notoriously challenging class of job-shop scheduling problems with the objective of minimizing the maximum completion time (i.e., minimizing the makespan). In particular, we develop an RLT-enhanced continuous nonconvex model for the job-shop problem based on a quadratic formulation of the job sequencing constraints on machines. The tight linear programming relaxation that is induced by this formulation is then embedded in a globally convergent branch-and-bound algorithm. Furthermore, we design another novel formulation for the job-shop scheduling problem that possesses a tight continuous relaxation, where the non-overlapping job sequencing constraints on machines are modeled via a lifted asymmetric traveling salesman problem (ATSP) construct, and specific sets of valid inequalities and RLT-based enhancements are incorporated to further tighten the resulting mathematical program. The efficacy of our enhanced models is demonstrated by an extensive computational experiment using classical benchmark problems from the literature. Our results reveal that the LP relaxations produced by the lifted ATSP-based models provide very tight lower bounds, and directly yield a 0% optimality gap for many benchmark problems, thereby substantially dominating other alternative mixed-integer programming models available for this class of problems. Notably, our lifted ATSP-based formulation produced a 0% optimality gap via the root node LP relaxation for 50% of the classical problem instances due to Lawrence [74].
4.1 Introduction

The deterministic job-shop scheduling problem (JSSP) arises in many industrial environments and presents a classical combinatorial optimization problem that has proven to be highly challenging to solve. The extent of research conducted in this field over the last forty years or so has motivated several surveys, such as the survey by Blazewicz et al. [20] and the state-of-the-art review by Jain and Meeran [63]. The computational intractability of this problem is illustrated by the fact that the 10-job-machine test problem FT10, introduced by Fisher and Thompson [50] in 1963, was provably solved to optimality for the first time by Carlier and Pinson [27] more than two decades later in 1989. Although several mathematical programming formulations have been proposed for the JSSP since the late fifties (see Section 4.2 and Appendix A), little progress has been realized with this trend of research, principally because of the weakness of the underlying continuous relaxations of the formulated models and the tremendous consequent computational effort required to solve the associated pure or mixed-integer programs. Reflecting on the difficulty of the JSSP, Conway et al. [31] observed, quite emphatically, that:

“Although it is easy to state, and to visualize what is required, it is extremely difficult to make any progress whatever toward a solution. Many proficient people have considered the problem, and all have come away essentially empty-handed.”

However, recent developments in solving mixed-integer programs together with modern computer capabilities resurrect some hope in this direction and, with this motivation, we investigate in this chapter several new modeling and lifting concepts for the JSSP with the objective of minimizing the maximum completion time.

In Section 4.2, we introduce our notation along with Manne’s model for the deterministic JSSP. In Section 4.3, we propose an enhanced continuous nonconvex mathematical program for this problem using the RLT methodology, and investigate an RLT-based Lagrangian dual formulation that is further enhanced via semidefinite cuts. Section 4.4 delineates and discusses a globally convergent optimization algorithm where RLT formulations play a key role in providing tighter relaxations. In Section 4.5, we propose enhanced LP relaxations for the JSSP based on a novel formulation in which the non-overlapping job sequencing constraints on machines are modeled via a lifted asymmetric traveling salesman problem (ATSP) viewpoint, and various sets of valid inequalities and RLT-lifted constraints are proposed to further tighten the resulting representation. Section 4.6 presents our preliminary findings, and reports promising results for the proposed lifted
4.2 Notation and Some Early Models

Several mathematical programming formulations have been proposed for the JSSP. These early works are reviewed in detail in Appendix A, but we focus here on the most popular and useful model due to Manne [81], as well as certain nonlinear, nonconvex modifications suggested by Nepomiastchy [95] and Rogers [100], which will be exploited using new modeling concepts and RLT-based enhancements discussed later in this chapter.

4.2.1 Notation

Below is a summary of our notation.

- \( M \) = set of \( m \) machines.
- \( J \) = set of \( n \) jobs.
- \( J_j \) = set of ordered operations of job \( j \).
- Dummy operation 0 that marks the start (and the end) of all operation sequences on all machines.
- \( J_0 = J \cup \{0\} \).
- \( F^* \) = set of first operations, that is, the first operation of each job is included in this set.
- \( F^*_i \) = subset of \( F^* \) that is to be performed on machine \( i \), \( \forall i \).
- \( E^* \) = set of last operations, that is, the last operation of each job is included in this set.
- \( E^*_i \) = subset of \( E^* \) that is to be performed on machine \( i \), \( \forall i \).
- \( O_{ij} \) = operation of job \( j \) to be performed on machine \( i \).
- \( p_{ij} \) = processing time of \( O_{ij} \).
- \( A_j = \{(i_1,j,i_2,j) : \text{Operation } O_{i_1 j} \text{ is required to immediately precede operation } O_{i_2 j} \text{ of job } j\} \)
  = set of conjunctive arcs that represent precedence constraints between (ordered) operations belonging to job \( j \).
• $D_i = \{(ij_1, ij_2) : \text{both jobs } j_1 \text{ and } j_2, j_1 < j_2, \text{ require operations } O_{ij_1} \text{ and } O_{ij_2} \text{ to be performed on machine } i \text{ in a disjunctive fashion}\}$.

• $P(O_{ij}) = \text{set of all operations of job } j \text{ that precede operation } O_{ij}$.

• $S(O_{ij}) = \text{set of all operations of job } j \text{ that follow } O_{ij}$.

• $T = \text{upper bound on the makespan}$.

• $[l_{ij}, u_{ij}] = \text{time interval for commencing operation } O_{ij}$. Such lower and upper bounds can be computed by setting $l_{ij} = \sum_{k:O_{kj} \in P(O_{ij})} p_{kj}$ and $u_{ij} = T - (p_{ij} + \sum_{k:O_{kj} \in S(O_{ij})} p_{kj})$.

• $\Gamma_i = \{\text{set of triplets of distinct job indices } (j_1, j_2, j_3) \text{ such that it is possible to perform the respective operations of these jobs in this order on machine } i\}, \forall i \in M.$

4.2.2 Manne’s Model (1960)

For convenience, and because of the popularity of this formulation, we state Manne’s model for the JSSP, and refer the interested reader to Appendix A for a detailed chronological account of alternative existing formulations in the literature.

Decision Variables

• $t_{ij} = \text{starting time of } O_{ij}$.

• $z_{j_1,j_2}^i = \begin{cases} 1 & \text{if operation } j_1 \text{ is performed sometime prior to operation } j_2 \text{ on machine } i \\ 0 & \text{otherwise}, \forall(ij_1, ij_2) \in D_i, i \in M. \end{cases}$

• $C_{max} = \max\{t_{ij} + p_{ij} : O_{ij} \in E^*\}$. $C_{max}$ is the makespan or the maximum completion time of a schedule.

Minimize $C_{max}$ \hspace{1cm} (4.1a)

subject to $C_{max} \geq t_{ij} + p_{ij}, \forall O_{ij} \in E^*$ \hspace{1cm} (4.1b)

$t_{ij} - t_{i_1j} \geq p_{i_1j}, \forall j \in J, (i_1j, i_2j) \in A_j$ \hspace{1cm} (4.1c)

$t_{ij_2} - t_{ij_1} + K(1 - z_{j_1,j_2}^i) \geq p_{ij_1}, \forall i \in M, (ij_1, ij_2) \in D_i$ \hspace{1cm} (4.1d)

$t_{ij_1} - t_{ij_2} + Kz_{j_1,j_2}^i \geq p_{ij_2}, \forall i \in M, (ij_1, ij_2) \in D_i$ \hspace{1cm} (4.1e)

$z \text{ binary, } t \geq 0,$ \hspace{1cm} (4.1f)
where $K$ is a suitably large number. The objective function (4.1a) and Constraint (4.1b) express the objective of minimizing the maximum completion time. Constraint (4.1c) enforces the precedence restrictions between (ordered) operations that belong to job $j$, whereas Constraints (4.1d)-(4.1e) model the non-overlapping job sequencing constraints on machines via disjunctive relationships. Constraint (4.1f) enforces logical binary and nonnegativity restrictions on the problem variables.

This model provides the most compact formulation among early models for the JSSP, and was used in Greenberg’s [57] B&B algorithm for the job-shop problem, as well as in Balas’ [5] application of a specialized version of the filter method to the JSSP. Instead of the discrete non-overlapping job sequencing constraints (4.1d-4.1e) utilized in Manne’s model, Nepomiahtchy [95] suggested the following nonlinear, continuous, nonconvex constraints:

$$(t_{ij} - t_{ij_2} - p_{ij_2})(t_{ij_2} - t_{ij} - p_{ij_1}) \leq 0, \quad \forall i \in M, (ij_1, ij_2) \in D_i.$$ 

The problem was then tackled using a penalty function approach that could terminate at a local, possibly non-global, optimum. In a similar spirit, Rogers [100] adopted the following linear-quadratic constraints to model the foregoing disjunctive relationships:

$$\varphi_{ij_1,ij_2} - t_{ij_1} + t_{ij_2} \geq p_{ij_1}, \quad \forall j_1 \neq j_2 \in J, i \in M \quad (4.2a)$$

$$\varphi_{ij_2,ij_1} - t_{ij_2} + t_{ij_1} \geq p_{ij_2}, \quad \forall j_1 \neq j_2 \in J, i \in M \quad (4.2b)$$

$$\varphi_{ij_1,ij_2} \cdot \varphi_{ij_2,ij_1} = 0, \quad \forall j_1 \neq j_2 \in J, i \in M \quad (4.2c)$$

$$\varphi \geq 0. \quad (4.2d)$$

Here, the role of the binary variables used in Manne’s model is played by the complementarity constraints (4.2d). Again, a local search procedure was proposed to tackle this nonlinear, nonconvex formulation.

### 4.2.3 Valid Inequalities in the Literature

Valid inequalities, or cutting planes, are frequently adopted to strengthen the continuous relaxations of combinatorial optimization problems. The main task here is to formulate classes of valid inequalities that not only tighten the model representation and help significantly improve its continuous relaxation-based lower bound, but also can be generated efficiently within a reasonable amount of time. Ideally, it is desirable to generate valid inequalities that characterize facets of
the convex hull of feasible solutions to the MIP problem, but judiciously generated strong cutting planes or lifted versions of model-defining constraints can also greatly enhance the computational performance.

Applegate and Cook [4] offer an interesting analysis of the effect of valid inequalities on lower bounds for both disjunctive and MIP formulations of the JSSP. Their study includes newly developed valid inequalities as well as those proposed by Balas [7] and Dyer and Wolsey [45]. We identify below certain key valid inequalities that have been proposed in the literature in order to strengthen the underlying LP relaxations of Manne’s model.

- **Basic cuts** (attributed to Dyer and Wolsey [45] in [4]):
  \[ \sum_{j \in C} p_{ij} t_{ij} \geq \min_{j \in C} l_{ij} \sum_{j \in C} p_{ij} + \sum_{j_1, j_2 \in C, j_1 < j_2} p_{ij_1} p_{ij_2}, \quad \forall C \subseteq J, \forall i \in M. \]

- **Half cuts** [4]:
  \[ t_{ij_1} \geq \min_{j \in C} l_{ij} + \sum_{j_2 \in C, j_2 < j_1} z_{ij_2 j_1} p_{ij_2} + \sum_{j_2 \in C, j_1 < j_2} (1 - z_{ij_1 j_2}) p_{ij_2}, \quad \forall j_1 \in J, C \subseteq J, i \in M. \]

- **Basic cuts plus epsilon** [4]:
  \[ \sum_{j \in C} p_{ij} t_{ij} \geq l_{ik} \sum_{j \in C} p_{ij} + \sum_{j_1, j_2 \in C, j_1 < j_2} p_{ij_1} p_{ij_2} - (\sum_{j \in C} z_{ijk} \{l_{ik} - l_{ij}\}^+) + \sum_{j \in C, j < k} (1 - z_{ikj}) \{l_{ik} - l_{ij}\}^+ \sum_{j \in C} p_{ij}, \]
  \[ \forall k \in J, C \subseteq J, i \in M, \text{ where } \{l_{ik} - l_{ij}\}^+ = \max\{0, l_{ik} - l_{ij}\}. \]

- **Triangle cuts** [4]:
  \[ z_{ij_1 j_2} + z_{ij_2 j_3} - z_{ij_1 j_3} \leq 1, \quad \forall j_1 < j_2 < j_3 \in J, i \in M. \]

### 4.3 RLT-based Continuous Model and Linear Lower Bounding Problem

The various mathematical programs reviewed in Section 4.2 and in Appendix A involve either pure or mixed-integer 0-1 formulations that suffer from their size and/or the weakness of their underlying LP relaxations. Therefore, they become computationally prohibitive even for moderately-sized problem instances, and it is crucial to design models that achieve a good balance between size and tightness. This motivates the development of tighter models via the RLT methodology.
4.3.1 RLT-based Relaxation

Adopting the continuous nonconvex disjunctive constraints suggested by Nepomiaitchy [95], we can reformulate Manne’s [81] model as follows, where we have defined a new variable $g^i_{j_1,j_2}$ to represent the difference $t_{ij_1} - t_{ij_2}, \forall i \in M, \forall (ij_1, ij_2) \in D_i$, for the sake of analytical convenience.

Minimize $C_{\text{max}}$ (4.3a)

subject to $C_{\text{max}} \geq t_{ij} + p_{ij}, \forall j \in E_i^*, i \in M$ (4.3b)

$t_{ij_2} - t_{ij_1} \geq p_{ij_2}, \forall j \in J, (i_1j, i_2j) \in A_j$ (4.3c)

$(g^i_{j_1,j_2} - p_{ij_2})(g^i_{j_1,j_2} + p_{ij_1}) \geq 0, \forall i \in M, \forall (ij_1, ij_2) \in D_i$ (4.3d)

$g^i_{j_1,j_2} = t_{ij_1} - t_{ij_2}, \forall i \in M, \forall (ij_1, ij_2) \in D_i$ (4.3e)

$t \geq 0.$ (4.3f)

Based on the order of operations $O_{ij_1}$ and $O_{ij_2}$ in the routing of jobs $j_1$ and $j_2$, we can derive lower and upper bounds on the starting time of any operation $O_{ij}$ of the type $l_{ij} = \sum_{k:O_{kj} \in S(O_{ij})} p_{kj}$ and $u_{ij} = T - \sum_{k:O_{kj} \in S(O_{ij})} p_{kj} - p_{ij}$, where $T$ is some upper bound on the optimal makespan.

Although $T$ may be computed via any adequate heuristic, efficient algorithms (such as the Shifting Bottleneck procedure (SBP) [1]) should be preferred, because the tightness of the bounds on the variables significantly contributes to the strength of the constraints generated by RLT constructs.

Thus, we deduce box-constraints of the type $\alpha^i_{j_1,j_2} \leq g^i_{j_1,j_2} \leq \beta^i_{j_1,j_2}$, where $\alpha^i_{j_1,j_2} = l_{ij_1} - u_{ij_2}$ and $\beta^i_{j_1,j_2} = u_{ij_1} - l_{ij_2}$. Using these bounding constraints, we can augment the job-shop formulation with the following RLT bounding-factor product relationships, denoted $F^i_{j_1,j_2} \geq 0$, of the type:

$(\beta^i_{j_1,j_2} - g^i_{j_1,j_2})(g^i_{j_1,j_2} - \alpha^i_{j_1,j_2}) \geq 0, (\beta^i_{j_1,j_2} - g^i_{j_1,j_2})^2 \geq 0,$ and $(g^i_{j_1,j_2} - \alpha^i_{j_1,j_2})^2 \geq 0, \forall i \in M, \forall (ij_1, ij_2) \in D_i$.

Hence, we derive the following Manne-based RLT-enhanced formulation, which we denote by JQP.

**JQP:** Minimize $C_{\text{max}}$ (4.4a)

subject to $C_{\text{max}} \geq t_{ij} + p_{ij}, \forall j \in E_i^*, i \in M$ (4.4b)

$t_{ij_2} - t_{ij_1} \geq p_{ij_2}, \forall j \in J, (i_1j, i_2j) \in A_j$ (4.4c)

$(g^i_{j_1,j_2} - p_{ij_2})(g^i_{j_1,j_2} + p_{ij_1}) \geq 0, \forall i \in M, (ij_1, ij_2) \in D_i$ (4.4d)

$g^i_{j_1,j_2} = t_{ij_1} - t_{ij_2}, \forall i \in M, (ij_1, ij_2) \in D_i$ (4.4e)

$F^i_{j_1,j_2} \geq 0, \forall i \in M, (ij_1, ij_2) \in D_i$ (4.4f)
As per the RLT methodology, this reformulated problem can be linearized to yield a lower bounding linear program by using the following RLT variable substitution identities:

\[
h_{i,j_2}^i = \left| g_{i,j_2}^i \right|^2, \quad \forall i \in M, (i,j_1, i,j_2) \in D_i.
\]

The RLT-based linear programming relaxation, JLP, is thus obtained as given below.

**JLP:** Minimize \( C_{\text{max}} \) \hfill (4.6a)

subject to \( C_{\text{max}} \geq t_{ij} + p_{ij}, \quad \forall j \in E_i, i \in M \) \hfill (4.6b)

\[
t_{i,j} - t_{i,j} \geq p_{i,j}, \quad \forall j \in J, (i_1,j, i_2,j) \in A_j
\]

\[
h_{i,j_2}^i \geq (p_{ij_2} - p_{ij_1})g_{i,j_2}^i + p_{ij_1}g_{i,j_2}^i, \quad \forall i \in M, (i,j_1, i,j_2) \in D_i
\]

\[
g_{i,j_2}^i = t_{ij_1} - t_{ij_2}, \quad \forall i \in M, (i_j_1, i,j_2) \in D_i
\]

\[
h_{i,j_2}^i \leq g_{i,j_2}^i (\alpha_{i,j_2}^i + \beta_{i,j_2}^i) - \alpha_{i,j_2}^i \beta_{i,j_2}^i, \quad \forall i \in M, (i,j_1, i,j_2) \in D_i
\]

\[
h_{i,j_2}^i \geq 2\beta_{i,j_2}^i g_{i,j_2}^i - (\beta_{i,j_2}^i)^2, \quad \forall i \in M, (i,j_1, i,j_2) \in D_i
\]

\[
h_{i,j_2}^i \geq 2\alpha_{i,j_2}^i g_{i,j_2}^i - (\alpha_{i,j_2}^i)^2, \quad \forall i \in M, (i,j_1, i,j_2) \in D_i
\]

\[
t, h \geq 0.
\]

As obvious from the foregoing derivation, JLP is indeed a lower bounding problem for JQP. Moreover, if the substitution identities (4.5) are satisfied in an optimal solution to JLP, then this solution also yields an optimum for JQP.

**Remark 4.1.** Observe that the bounding constraints of the type \( \alpha_{i,j_2}^i \leq g_{i,j_2}^i \leq \beta_{i,j_2}^i \) are not explicitly enforced in JLP. In fact, the bound-factors \( g_{i,j_2}^i - \alpha_{i,j_2}^i \geq 0 \) and \( \beta_{i,j_2}^i - g_{i,j_2}^i \geq 0 \) are dominated by the higher order bound-factor products inherent within \( [F_{i,j_2}^i]_t \geq 0 \) [127].
Remark 4.2. Note that \( \alpha_{j_1,j_2} \geq 0 \) implies that \( O_{ij_2} \) must precede \( O_{ij_1} \). Similarly, if \( \beta_{j_1,j_2} \leq 0 \), then \( O_{ij_1} \) must precede \( O_{ij_2} \). Therefore, once any disjunction is resolved, i.e. \( \alpha_{j_1,j_2} \geq 0 \) or \( \beta_{j_1,j_2} \leq 0 \) is determined, we replace the associated constraints in (4.6d)-(4.6h) by \( g_{j_1,j_2}^i \geq p_{ij_2} \) or \( g_{j_1,j_2}^i \leq -p_{ij_1} \), respectively.

4.3.2 Lagrangian Dual Formulations

In this section, we investigate a basic Lagrangian dual relaxation that is further enhanced via semidefinite cuts in order to tighten the model formulation (see Sherali and Fraticelli [117] and Sherali and Desai [114]).

Basic Formulation

Denoting Lagrange multipliers \( \lambda_{ij} \) associated with (4.6b), for \( j \in E^*_i, i \in M \), \( \pi_{j,(i_1,j_2)} \) associated with (4.6c) for \( j \in J, (i_1,j, i_2,j) \in A_j \), \( \mu_{ij,j_2} \) associated with (4.6d), \( \phi_{ij_1,j} \) associated with (4.6g), and \( \delta_{ij_1,j_2} \) associated with (4.6h) for \( i \in M, (ij_1,i_2,j) \in D_i \), we can formulate the following Lagrangian Dual to JLP, which we denote by JLD1.

\[
\text{JLD1: Maximize } \{ \theta(\lambda, \pi, \mu, \phi, \delta) : \sum_{i \in M} \sum_{j \in E^*_i} \lambda_{ij} = 1, \lambda \geq 0, \pi \geq 0, \mu \geq 0, \\
\quad \phi \geq 0, \delta \geq 0 \} \tag{4.7}
\]

where

\[
\theta(\lambda, \pi, \mu, \phi, \delta) = \min \{ \sum_{i \in M} \sum_{j \in E^*_i} \lambda_{ij}(t_{ij} + p_{ij}) \\
\quad + \sum_{j=1}^{n} \sum_{(i_1,j_2) \in A_j} \pi_{j,(i_1,j_2)}(p_{i_1j} - t_{i_2j} + t_{i_1j}) \\
\quad + \sum_{i=1, j_1=1, j_2=j_1+1}^{n-1} \sum_{j_1=1, j_2=j_1+1}^{n} \mu_{ij_1,j_2}(p_{ij_1}p_{ij_2} + (p_{ij_2} - p_{ij_1})g_{j_1,j_2}^i - h^i_{j_1,j_2}) \\
\quad + \sum_{i=1, j_1=1, j_2=j_1+1}^{n-1} \sum_{j_1=1, j_2=j_1+1}^{n} \phi_{ij_1,j_2}(2\beta_{j_1,j_2}^i g_{j_1,j_2}^i - (\beta_{j_1,j_2}^i)^2 - h^i_{j_1,j_2}) \\
\quad + \sum_{i=1, j_1=1, j_2=j_1+1}^{n-1} \sum_{j_1=1, j_2=j_1+1}^{n} \delta_{ij_1,j_2}(2\alpha_{j_1,j_2}^i g_{j_1,j_2}^i - (\alpha_{j_1,j_2}^i)^2 - h^i_{j_1,j_2}) \} \tag{4.8a}
\]

subject to

\[
g_{j_1,j_2}^i = t_{ij_1} - t_{ij_2}, \quad \forall i \in M, (ij_1,i_2,j) \in D_i \tag{4.8b}
\]
\[0 \leq h^i_{j1, j2} \leq g^i_{j1, j2} (\alpha^i_{j1, j2} + \beta^i_{j1, j2}) - \alpha^i_{j1, j2} \beta^i_{j1, j2},\]
\[\forall i \in M, (ij_1, ij_2) \in D_i\]  \hspace{1cm} (4.8c)
\[l_{ij} \leq t_{ij} \leq u_{ij}, \hspace{1cm} \forall i \in M, j \in J,\]  \hspace{1cm} (4.8d)

and where we have imposed the implied bounds on the \(t\)-variables in the subproblem constraints (4.8d) in order to ensure a finite optimum for this problem. For convenience, we shall denote the objective function expression in (4.8a) as “Obj(4.8a)”.

**SDP-enhanced Formulation**

JLD1 can be further enhanced by incorporating a class of SDP-based constraints in the spirit of the SDP cuts introduced by Sherali and Fraticelli [117]. To this end, we consider the vector \(g^{(1)} = \begin{bmatrix} 1 \\ g^i_{j1, j2} \end{bmatrix}\), and define the following matrix \(H^i_{j1, j2} \equiv \begin{bmatrix} g^{(1)} T_{(1)} \\ g^i_{j1, j2} & h^i_{j1, j2} \end{bmatrix}\), \(\forall i \in M, \forall (ij_1, ij_2) \in D_i\). Requiring \(H^i_{j1, j2}\) to be positive semidefinite, that is \(H^i_{j1, j2} \succeq 0\), we enforce constraints of the type \(h^i_{j1, j2} \geq (g^i_{j1, j2})^2\), \(\forall i \in M, \forall (ij_1, ij_2) \in D_i\). This leads to the following RLT-based, SDP-enhanced, Lagrangian dual formulation, JLD2, where the Lagrangian multipliers associated with the dualization of (4.6e) are denoted \(\eta_{ij_1, ij_2}, \forall i \in M, (ij_1, ij_2) \in D_i\).

**JLD2**:

Maximize \(\{\theta'(\lambda, \pi, \mu, \phi, \delta, \eta) : \sum_{i \in M} \sum_{j \in E_i} \lambda_{ij} = 1, \lambda \geq 0, \pi \geq 0, \mu \geq 0, \phi \geq 0, \delta \geq 0, \eta \text{ unrestricted} \}\)  \hspace{1cm} (4.9)

where

\[\theta'(\lambda, \pi, \mu, \phi, \delta, \eta) = \text{minimum} \ \\{\text{Obj (4.8a)} + \sum_{i=1}^{m} \sum_{j=1}^{n-1} \sum_{j_2=j_1+1}^{n} \eta_{ij_1, ij_2} (g^i_{j1, j2} - t_{ij_1} + t_{ij_2})\}\]  \hspace{1cm} (4.10a)

subject to

\[(g^i_{j1, j2})^2 \leq h^i_{j1, j2} \leq g^i_{j1, j2} (\alpha^i_{j1, j2} + \beta^i_{j1, j2}) - \alpha^i_{j1, j2} \beta^i_{j1, j2},\]
\[\forall i \in M, (ij_1, ij_2) \in D_i\]  \hspace{1cm} (4.10b)
\[l_{ij} \leq t_{ij} \leq u_{ij}, \hspace{1cm} \forall i \in M, j \in J,\]  \hspace{1cm} (4.10c)
\[\alpha^i_{j1, j2} \leq g^i_{j1, j2} \leq \beta^i_{j1, j2}, \hspace{1cm} \forall i \in M, (ij_1, ij_2) \in D_i.\]  \hspace{1cm} (4.10d)
Deflected subgradient optimization techniques are worthy of exploration in order to solve JLD1 and JLD2. Specialized efficient schemes for evaluating the Lagrangian dual objective functions shall be developed in this research. Observe that the objective coefficients pertaining to the \( h \)-variables in (4.10a) are nonpositive and, therefore, the upper bound on the \( h \)-variables represented in (4.10b) will be binding due to the minimization operation. Hence, we shall also investigate an alternative strategy in which the upper bounding expression in (4.10b) is dualized and accommodated within (4.10a), while requiring \((g_{j1j2}^i)^2 = h_{j1j2}^i\) in lieu of (4.10b). Note that the latter constraint can be equivalently replaced by the convex hull of this restriction and (4.10d). We shall exploit this structure in designing efficient schemes for optimizing such Lagrangian dual formulations.

4.4 Global Optimization Algorithm

In this section, we present a globally convergent B&B algorithm in concert with the RLT-based formulation proposed in Section 4.3.1. We also introduce certain preprocessing rules that are triggered at the root node of the B&B tree to readily fix the disjunction between certain pairs of operations.

4.4.1 Branch-and-bound Algorithm

Let \( \Omega \) denote the hyperrectangle bounding the \( g \)-variables at the root node of the B&B search tree, and accordingly, let us denote the original problem and its corresponding lower bounding problem as JQP(\( \Omega \)) and JLP(\( \Omega \)), respectively. Likewise, for any subnode \( k \), we define the sub-hyperrectangle \( \Omega_k \subseteq \Omega \) and the corresponding problems JQP(\( \Omega_k \)) and JLP(\( \Omega_k \)). Let \( \nu[.] \) be the value at optimality of any given problem [\( . \)]. For convenience, we also denote the vector of \( t_{ij} \)-variables by \( t \), and similarly we introduce the vectors \( g \) and \( h \).

If at any node \( k \) in the B&B tree, the optimal solution \((\tilde{t}, \tilde{g}, \tilde{h})\) obtained for JLP (\( \Omega^k \)) satisfies the variable substitution identities (4.5), then \((\tilde{t}, \tilde{g})\) solves JQP(\( \Omega^k \)). That is, all the RLT variables faithfully reproduce the squared variables they represent, and a feasible solution to the original problem is thereby available that achieves the lower bounding value. As a consequence, the incumbent solution and its value for the original problem, \((t^*, g^*)\) and \( Cmax^* \), can potentially be updated as necessary. Also, if (4.5) holds for JLP(\( \Omega \)) at the root node of the search tree, then the solution obtained to JLP(\( \Omega \)) is indeed optimal to JQP(\( \Omega \)), and the algorithm terminates.

As noted in Remark 4.2, \( g_{j1j2}^i \geq 0 \Rightarrow g_{j1j2}^i \geq p_{j1j2} \), and similarly \( g_{j1j2}^i \leq 0 \Rightarrow g_{j1j2}^i \leq -p_{j1j2} \).
That is, imposing one sign or another to any variable $g_{ij}^1$ is equivalent to a binary decision that fixes the relative order of operations $O_{ij_1}$ and $O_{ij_2}$ on machine $i$. This result is at the heart of the branching rule.

**Branching rule.** Consider some node $k$ in the search tree. The partitioning step is based on the identification of the variable $g_{ij}^1$ that creates the highest discrepancy between an RLT variable and the term it replaces. We select $g_{ij}^1$ such that 

$$(i, j_1, j_2) \in \text{arg max}_{i \in M, (ij_1, ij_2) \in D_i} \left\{ \rho_{ij_1, ij_2} \right\},$$

where $\rho_{ij_1, ij_2} = \left| h_{ij_1, ij_2} - (g_{ij_1, ij_2})^2 \right|$. Upon the selection of $g_{ij_1, ij_2}$, we create two new nodes by partitioning $\Omega^k$ into $\Omega^{k+1} \equiv \Omega^k \cap \{g_{ij_1, ij_2} \geq p_{ij_2}\}$ and $\Omega^{k+2} \equiv \Omega^k \cap \{g_{ij_1, ij_2} \leq -p_{ij_1}\}$.

A formal description of the overall B&B algorithm is given below.

- **Step 0: Initialization Step.** Initialize the incumbent solution $(t^*, g^*)$ and its objective value $Cmax^*$ by computing a heuristic solution. (We used the SBP [1] for this purpose.) Set $k = 1$ and $\Omega^k = \Omega$. Solve JLP($\Omega$) and denote its optimal solution by $(\bar{t}, \bar{g}, \bar{h})$. Determine a branching variable $g_{ij_1, ij_2}$ according to the branching rule. If $\rho_{ij_1, ij_2} = 0$, then $(\bar{t}, \bar{g})$ is optimal to JQP; terminate the algorithm after setting $(t^*, g^*) \leftarrow (\bar{t}, \bar{g})$, and $Cmax^* \leftarrow \nu[JLP(\Omega)]$. Otherwise, if $\rho_{ij_1, ij_2} > 0$, proceed to Step 1, with the selected node $\hat{k} = 1$.

- **Step 1: Branching Step.** Create two new nodes, $(k + 1)$ and $(k + 2)$, by partitioning $\Omega^{\hat{k}}$ into $\Omega^{k+1}$ and $\Omega^{k+2}$ as explained above, and remove the parent node, $\hat{k}$, from the list of active nodes.

- **Step 2: Bounding Step.** Solve JLP($\Omega^{k+1}$) and JLP($\Omega^{k+2}$). Update the incumbent if appropriate. Select and store a branching variable for each of these two nodes. Increment $k \leftarrow k + 2$.

- **Step 3: Fathoming Step.** Fathom any node $k'$ such that $\nu[JLP(\Omega^{k'})] \geq Cmax^*(1 - \epsilon)$ by removing it from the list of active nodes, where $0 \leq \epsilon \leq 1$ is a specified percentage optimality gap (use $\epsilon = 0$ if a global optimal is desired). If the list of active nodes is empty, stop. Otherwise, proceed to Step 4.

- **Step 4: Node Selection Step.** Among the active nodes, select one ($\hat{k}$, say) that has the least lower bound, and go to Step 1.
Proposition 4.1. The foregoing B&B algorithm (run with $\varepsilon = 0$) terminates finitely and produces an optimal solution to JQP at termination.

Proof. The result directly follows from the branching strategy because there are a finite number of ways of resolving the disjunctions. \qed

Note that the deepest level that can be reached in the search tree is $\frac{mn(n-1)}{2}$, which corresponds to fixing the signs of $g_{ij}^i, \forall i \in M, (ij_1, ij_2) \in D_i$.

4.4.2 Preprocessing and Inference Rules

In this section, we present preprocessing strategies to determine the relative order between pairs of operations at the root node. We also overview some inference rules that can be strategically triggered within the B&B algorithm to fix the position of one or several operations during the construction of a schedule. Combining such preprocessing and inference rules can substantially contribute toward the computational efficiency of the B&B algorithm by a priori curtailing the solution search space.

• Preprocessing tests.

1. Bound check. $\alpha_{ij}^i \geq 0 \Rightarrow g_{ij}^i \geq p_{ij}$, and similarly $\beta_{ij}^i \leq 0 \Rightarrow g_{ij}^i \leq -p_{ij}$.

2. Inconsistency check. $p_{ij}^2 \geq \beta_{ij}^i \Rightarrow g_{ij}^i \leq -p_{ij}$, because the reverse decision would result in an inconsistency. Similarly, $-p_{ij}^1 \leq \alpha_{ij}^i \Rightarrow g_{ij}^i \geq p_{ij}$.

• Inference rules. (In progress)

4.5 Lifted ATSP-based Formulations

In a recent paper, Sherali et al. [120] have proposed several lifting concepts and RLT-enhancements for the ATSP with and without precedence constraints, and have demonstrated the tightness of the resulting formulations for various standard benchmark problems. In the context of the JSSP, we shall adopt an insightful modeling approach where the scheduling of jobs to be performed on any machine is viewed as an ATSP problem, and certain sets of valid inequalities and RLT-enhancements are derived, as established in the sequel below.

Decision Variables

• $t_{ij} =$ starting time of $O_{ij}$. 

43
• $x_{j1j2}^i = \begin{cases} 1 & \text{if the operation of job } j_1 \text{ immediately precedes the operation of } j_2 \text{ on machine } i \\ 0 & \text{otherwise, } \forall j_1 \neq j_2 \in J_0, i \in M. \end{cases}$

• $y_{j1j2}^i = \begin{cases} 1 & \text{if the operation of job } j_1 \text{ is performed sometime prior to the operation of } j_2 \text{ on machine } i \\ 0 & \text{otherwise, } \forall j_1 \neq j_2 \in J, i \in M. \end{cases}$

• $C_{max} = \max\{t_{ij} + p_{ij} : O_{ij} \in E^*\}$. $C_{max}$ is the makespan or the maximum completion time of a schedule.

JS-ATSP1: Minimize $C_{max}$

subject to $C_{max} \geq t_{ij1} + p_{ij1} + \sum_{j2 \neq j1} p_{ij2} y_{j1j2}^i, \forall j_1 \in E^*_i, i \in M$ \hspace{1cm} (4.11a)

$\sum_{j2 \in J_0 - \{j\}} x_{j1j2}^i = 1, \forall j_1 \in J_0, i \in M$ \hspace{1cm} (4.11b)

$\sum_{j1 = J_0 - \{j2\}} x_{j1j2}^i = 1, \forall j_2 \in J_0, i \in M$ \hspace{1cm} (4.11c)

$y_{j1j2}^i + y_{j2j1}^i = 1, \forall j_1 < j_2 \in J, i \in M$ \hspace{1cm} (4.11d)

$y_{j1j2}^i \geq x_{j10}^i, \forall j_1 \neq j_2 \in J, i \in M$ \hspace{1cm} (4.11e)

$y_{j2j1}^i \geq x_{j10}^i, \forall j_1 \neq j_2 \in J, i \in M$ \hspace{1cm} (4.11f)

$y_{j1j2}^i \geq x_{j1j2}^i, \forall j_1 \neq j_2 \in J, i \in M$ \hspace{1cm} (4.11g)

$y_{j1j3}^i \geq (y_{j1j2}^i + y_{j2j3}^i - 1) + x_{j2j1}^i, \forall (j_1, j_2, j_3) \in \Gamma_i, i \in M$ \hspace{1cm} (4.11h)

$t_{ij2} \geq t_{ij1} + p_{ij1} - (1 - y_{j1j2}^i)(p_{ij1} + u_{ij1} - l_{ij2})$, 
\hspace{1cm} \forall j_1 \neq j_2 \in J, i \in M \hspace{1cm} (4.11i)

$t_{ij2} \geq t_{ij1} + p_{ij1} + \sum_{j \neq j_1, j \neq j_2} x_{j1j2}^i p_{ij} - (1 - y_{j1j2}^i)(p_{ij1} + u_{ij1} - l_{ij2} + \max\{p_{ij}\})$, 
\hspace{1cm} \forall j_1 \neq j_2 \in J, i \in M \hspace{1cm} (4.11j)

$t_{ij2} - t_{ij1} \geq p_{ij}, \forall j \in J, (i_1j, i_2j) \in A_j$ \hspace{1cm} (4.11k)

$\sum_{i \in M} \sum_{j1 \in F^*_i} x_{0j1}^i \geq 1$ \hspace{1cm} (4.11l)

$t_{ij2} \geq \sum_{j1 \in J - \{j2\}} y_{j1j2}^i p_{ij1}, \forall j_2 \in J, i \in M$ \hspace{1cm} (4.11m)

$t_{ij1} \leq T - \sum_{j2 \in J - \{j1\}} y_{j1j2}^i p_{ij2}, \forall j_1 \in J, i \in M$ \hspace{1cm} (4.11n)
\[ l_{ij} \leq t_{ij} \leq u_{ij}, \quad \forall j \in J, i \in M \quad (4.11p) \]
\[ x \text{ binary, } y \geq 0. \quad (4.11q) \]

The objective function (4.11a), in conjunction with Constraint (4.11b), enforces the definition of the makespan as the maximum completion time of the schedule. Observe that Constraint (4.11b) provides a lifted expression of the makespan constraint formulated in Manne’s model, taking into consideration the completion time of the last operation of every job and augmenting this with the sum of the processing times of the operations scheduled after it. For the remainder of the formulation, in essence, we exploit the analogy between the set of job-operations to be performed on any machine, augmented with a dummy node 0, and the cities to be visited in an ATSP given the base city 0, in order to sequence the operations assigned to this machine via Constraints (4.11c)-(4.11i) and (4.11q) (see [120] for their motivation). Constraint (4.11j) computes the start-times of operations on each machine given the \( y \)-variables and is partially lifted via Constraint (4.11k) as established in Proposition 4.2 below. Constraint (4.11l) enforces the precedence relationships among operations that belong to the same job. Constraint (4.11m) ensures that, for at least one machine, call it \( i \), the first operation to be processed must belong to \( F^{*}_i \). The bounds in Constraints (4.11n)-(4.11p) are determined by examining the relative position of any operation, \( O_{ij} \), in the sequence of operations to be processed on machine \( i \) and in the sequence of operations that belong to job \( j \). Constraint (4.11q) enforces logical binary restrictions on the \( x \)-variables and the nonnegativity of the \( y \)-variables. Observe that the binariness of the \( x \)-variables together with Constraints (4.11e), (4.11h), and (4.11i) induce binary restrictions on the \( y \)-variables.

**Proposition 4.2.** Constraints (4.11k) enforce a set of valid inequalities.

**Proof.**

If \( y_{j_1,j_2} = 1 \), then \( t_{ij_2} \geq t_{ij_1} + p_{ij_1} + \sum_{j \neq j_1, j \neq j_2} x_{ij} p_{ij} \), which is valid since job \( j_1 \) precedes (not necessarily immediately) job \( j_2 \) on machine \( i \). On the other hand, if \( y_{j_1,j_2} = 0 \), then \( t_{ij_2} - l_{ij_2} \geq t_{ij_1} - u_{ij_1} + \sum_{j \neq j_1, j \neq j_2} x_{ij} p_{ij} - \max \{ p_{ij} \} \), which is valid since \( t_{ij_1} - u_{ij_1} \leq 0 \) and \( \sum_{j \neq j_1, j \neq j_2} x_{ij} p_{ij} - \max \{ p_{ij} \} \leq 0 \), while \( t_{ij_2} - l_{ij_2} \geq 0. \) \( \square \)

There are two sets of optional, alternative valid inequalities that can be investigated in the context of JS-ATSP1 based on the formulations developed by Sherali et al. [120]. The first of these
replaces Constraint (4.11i) by the following:

\[ y_{j_1,j_3}^i \geq (x_{j_1,j_2}^i + y_{j_2,j_3}^i - 1) + (x_{j_1,j_3}^i + x_{j_2,j_3}^i), \quad \forall (j_1, j_2, j_3) \in \Gamma_i, i \in M. \]  

(4.12)

Hence, we obtain the following job-shop model:

**JS-ATSP2**: Minimize \( \{C_{max} : (4.11b)-(4.11q), \text{ with (4.12) enforced in lieu of (4.11i)} \} \).

(4.13)

Sherali et al. [120] also describe certain RLT-lifted constraints for the ATSP that are predicated on defining the following product variables in the present context:

\[ f_{j_1,j_3,j_2}^i = x_{j_1,j_3}^i y_{j_3,j_2}^i, \quad \forall (j_1, j_3, j_2) \in \Gamma_i, i \in M. \]  

(4.14)

These variables are then related to the original \( x \)- and \( y \)-variables in JS-ATSP1 via the following valid inequalities:

\[
\sum_{j_3 \in \Gamma_i} f_{j_1,j_3,j_2}^i + x_{j_1,j_2}^i = y_{j_1,j_2}^i, \quad \forall j_1 \neq j_2 \in E_i, i \in M. 
\]  

(4.15a)

\[
\sum_{j_1 \in \Gamma_i} f_{j_1,j_3,j_2}^i + x_{0,j_3}^i = y_{j_3,j_2}^i, \quad \forall j_3 \neq j_2 \in E_i, i \in M. 
\]  

(4.15b)

\[
0 \leq f_{j_1,j_3,j_2}^i \leq x_{j_1,j_3}^i, \quad \forall (j_1, j_3, j_2) \in \Gamma_i, i \in M. 
\]  

(4.15c)

**Proposition 4.3.** (a) Constraints (4.15a)-(4.15c) are valid and (b) Constraints (4.15a)-(4.15c) along with Problem (4.11) guarantee that the RLT-based Constraint (4.14) hold true.

**Proof.**

(a) Observe that Constraint (4.15c) is trivially valid by the binariness of the \( x \)- and \( y \)-variables and the definition of the \( f \)-variables in Constraint (4.14).

The validity of Constraint (4.15a) is established next by distinguishing three cases:
• If \( x_{j_1 j_2}^i = 1 \), then \( y_{j_1 j_2}^i = 1 \) and \( x_{j_1 j_3}^i = 0, \forall j_3 \neq j_1, j_2 \). Hence, \( f_{j_1 j_3 j_2}^i = 0, \forall j_3 \neq j_1, j_3 \) by Constraint (4.14) and, therefore, Constraint (4.15a) holds true.

• If \( (x_{j_1 j_2}^i = 0 \land y_{j_1 j_2}^i = 1) \), then there must exist a unique job \( j \) for which \( \sum_{j_3: \{j_1, j_3, j_2\} \in \Gamma} f_{j_1 j_3 j_2}^i = x_{j_1 j_2}^i y_{j_1 j_2}^i = y_{j_1 j_2}^i = 1 \). Hence, Constraint (4.15a) is valid.

• If \( (x_{j_1 j_2}^i = 0 \land y_{j_1 j_2}^i = 0) \), then job \( j_2 \) precedes \( j_1 \) on machine \( i \), and it must be that \( f_{j_1 j_3 j_2}^i = x_{j_1 j_3}^i y_{j_3 j_2}^i = 0, \forall j_3 \neq j_1, j_2 \).

Likewise, the validity of Constraint (4.15b) is established below by considering three cases:

• If \( x_{0 j_3}^i = 1 \), then \( y_{j_3 j_2}^i = 1 \) and \( x_{j_1 j_3}^i = f_{j_1 j_3 j_2}^i = 0, \forall j_3 \neq j_1, j_2 \) and, hence, Constraint (4.15b) is valid.

• If \( (x_{j_1 j_3}^i = 0 \land y_{j_3 j_2}^i = 1) \), then Constraint (4.15b) equivalently asserts that \( \sum_{j_1 \neq j_3} f_{j_1 j_3 j_2}^i = \sum_{j_1 \neq j_3} x_{j_1 j_3}^i = 1 \), which is valid by Constraint (4.11d).

• If \( (x_{j_1 j_3}^i = 0 \land y_{j_3 j_2}^i = 0) \), then \( f_{j_1 j_3 j_2}^i = 0, \forall j_3 \neq j_1, j_2 \) and, hence, Constraint (4.15b) is valid.

(b) Now, we shall show that Constraints (4.15b)-(4.15c) in concert with Problem (4.11) imply the RLT substitution equations in (4.14).

• If \( [x_{j_1 j_3}^i = 0 \land (y_{j_3 j_2}^i = 0 \lor y_{j_3 j_2}^i = 1)] \), then \( f_{j_1 j_3 j_2}^i = 0 \) by Constraint (4.15c) and, therefore, Constraint (4.14) holds true.

• If \( (x_{j_1 j_3}^i = 1 \land y_{j_3 j_2}^i = 0) \), then \( x_{0 j_3}^i = 0 \) and Constraint (4.15b) implies that \( \sum_{j: \{j, j_3, j_2\} \in \Gamma} f_{j j_3 j_2}^i = 0 \). Therefore, invoking the nonnegativity of the \( f \)-variables, we deduce that \( f_{j_1 j_3 j_2}^i = 0 \), and Constraint (4.14) is valid.

• If \( (x_{j_1 j_3}^i = 1 \land y_{j_3 j_2}^i = 1) \), then \( x_{0 j_3}^i = 0 \) and Constraint (4.15b) implies that \( \sum_{j \neq j_1: \{j, j_3, j_2\} \in \Gamma} f_{j_j_3 j_2}^i = 1 \). However, \( f_{j_1 j_3 j_2}^i \leq 1 \) by (4.15c) and, since \( x_{j_1 j_3}^i = 1 \), then \( x_{j_3 j_2}^i = 0, \forall j \neq j_1 \) such that \( \{j, j_3, j_2\} \in \Gamma \) by (4.11d), and so, \( f_{j_j_3 j_2}^i = 0, \forall j \neq j_1 \) such that \( \{j, j_3, j_2\} \in \Gamma \), by (4.15c). Thus, \( f_{j_1 j_3 j_2}^i = 1 \), and Constraint (4.14) is valid.

Also, under (4.15a)-(4.15c), we can lift Constraint (4.11j) and replace it by the following valid inequality, as proven next in Proposition 4.4.

\[
t_{ij_2} \geq t_{ij_1} + p_{ij_1} + \sum_{j_3: \{j_1, j_3, j_2\} \in \Gamma} f_{j_1 j_3 j_2}^i p_{ij_3} - (1-y_{j_1 j_2}^i) (p_{ij_1} + u_{ij_1} - l_{ij_2}), \quad \forall j_1 \neq j_2 \in J, i \in M. \tag{4.16}
\]
Proposition 4.4. Constraint (4.16) enforces a set of valid inequalities.

Proof. We shall examine three cases to establish the validity of (4.16):

- If \( y_{j_1,j_2}^i = 1 \land x_{j_1,j_2}^i = 1 \), then job \( j_1 \) is the immediate predecessor of job \( j_2 \) on machine \( i \), and \( \sum_{j_3 \neq j_2} x_{j_1,j_3}^i = 0 \) by (4.11c). Hence, \( \sum_{j_3:i=(j_1,j_3,j_2) \in \Gamma} f_{j_1,j_3,j_2}^i p_{ij_3} = 0 \), and (4.16) equivalently asserts that \( t_{ij_2} \geq t_{ij_1} + p_{ij_1} \), which is valid.

- If \( y_{j_1,j_2}^i = 1 \land x_{j_1,j_2}^i = 0 \), then job \( j_1 \) precedes job \( j_2 \), but is not its immediate predecessor on machine \( i \). Therefore, there exists a unique job \( j \) such that \( j \neq j_2 \) and \( x_{j_1,j}^i = 1 \), and \( \sum_{j_3:i=(j_1,j_3,j_2) \in \Gamma} f_{j_1,j_3,j_2}^i p_{ij_3} = x_{j_1,j_2}^i y_{j_1,j_2}^i p_{ij} = p_{ij} \). Hence, (4.16) equivalently asserts that \( t_{ij_2} \geq t_{ij_1} + p_{ij_1} + p_{ij} \), which is valid.

- If \( y_{j_1,j_2}^i = 0 \), then \( \sum_{j_3:i=(j_1,j_3,j_2) \in \Gamma} f_{j_1,j_3,j_2}^i p_{ij_3} = 0 \). Hence, (4.16) equivalently asserts that \( t_{ij_2} - l_{ij_2} \geq t_{ij_1} - u_{ij_1} \), which is true because \( t_{ij_2} - l_{ij_2} \geq 0 \), while \( t_{ij_1} - u_{ij_1} \leq 0 \). \( \square \)

We also introduce the following constraints, which are validated by Proposition 4.5 below:

\[
C_{\text{max}} \geq t_{ij} + p_{ij} + x_{j_1,j_2}^i \sum_{j_2 \neq j,j_1} p_{ij_2} f_{j_1,j_2}^i, \quad \forall j \in E^*_i, i \in M. \tag{4.17}
\]

Proposition 4.5. Constraint (4.17) enforces a set of valid inequalities.

Proof. Observe that Constraint (4.17) is derived from the lifted makespan constraint formulated in (4.11b). Now, if \( x_{j_1,j_1}^i = 0 \), then \( f_{j_1,j_1,j_2}^i = 0 \), \( \forall j_2 \neq j, j_1 \), and Constraint (4.17) reduces to \( C_{\text{max}} \geq t_{ij} + p_{ij} \), which is valid. On the other hand, if \( x_{j_1,j_1}^i = 1 \Rightarrow f_{j_1,j_1,j_2}^i = y_{j_1,j_2}^i \), and Constraint (4.17) reduces to \( C_{\text{max}} \geq t_{ij} + p_{ij} + p_{ij_1} \sum_{j_2 \neq j,j_1} p_{ij_2} y_{j_1,j_2}^i, \) which is again valid. \( \square \)

Noting that (4.15a) and (4.15b) respectively imply (4.11h) and (4.11f) under \( f \geq 0 \), we get the following RLT-lifted alternative formulation of JS-ATSP1.

**JS-ATSP3:** Minimize \( \{ C_{\text{max}} : (4.11b)-(4.11q) \) and (4.17), with (4.11f) and (4.11h) replaced by (4.15a)-(4.15c), and (4.11j) replaced by (4.16)\}. \( \tag{4.18a} \)
**Remark 4.3.** Similar to the variant JS-ATSP2 derived from JS-ATSP1, we could attempt the following alternative to JS-ATSP3.

**JS-ATSP4:** Minimize \( \{C_{\text{max}} : \text{Constraints of JS-ATSP3 where (4.11i) is replaced by (4.12)} \} \).

(4.18b)

### 4.6 Computational Experience

In this section, we present an extensive computational study that reflects the tightness of continuous, as well as certain mixed-integer programming relaxations, pertaining to the different proposed lifted ATSP-based models. Various classes of standard benchmark problems for the JSSP have been tested. These include the test-beds due to Lawrence [74], Fisher and Thompson [50], Applegate and Cook [4], and Adams et al. [1]. All mathematical programs and the proposed B&B algorithm were coded in AMPL and solved using CPLEX 10.1 on a Dell Precision 650 workstation having a Xeon(TM) CPU 2.40 GHz processor and 1.50 GB of RAM. In addition, we present some preliminary computational results for a hybrid implementation that utilizes a nondifferentiable optimization routine (see Sherali and Lim [119] and Lim and Sherali [76]) in concert with the simplex method (or the barrier approach) of CPLEX at the root node, while all the subsequent search tree nodes utilize the default CPLEX MIP Optimizer.

#### 4.6.1 Continuous Relaxations

Each problem instance, reported in Table 4.1 and in the sequel, is denoted by its classical abbreviation, followed by its optimal MIP objective value within parentheses and its size \((n \times m)\). We compare in Tables 4.1-4.8 the relative tightness of the LP relaxations for Manne’s model and for JS-ATSP\(i\), \(i = 1, ..., 4\). For each of these models, the LP objective value is recorded along with the induced % optimality gap and the accompanying CPU time in seconds. Whenever the LP objective value itself achieves a 0% optimality gap, it appears in bold. For Table 4.1, the upper bound utilized on the makespan is based on the Shifting Bottleneck procedure (SBP) [1], whereas for the results reported in Tables 4.2-4.4 and Tables 4.6-4.8, we have considered the best ad hoc upper bound available in the literature (which may be the optimal objective value itself for problem instances that have been optimally solved in the literature).

Observe that for certain problem instances, the LP relaxations of the JS-ATSP\(i\), \(i = 1, ..., 4\), produce a fractional objective value, which could be rounded up to obtain a tighter valid lower
bound. However, we preserved these fractional values in order to provide an indication of the relative strengths of the various different proposed alternative lifting concepts and RLT enhancements. This preliminary study indicates that JS-ATSP1 performs better than JS-ATSP2, and as a consequence, JS-ATSP3 outperforms JS-ATSP4. However, these four lifted models, while strongly dominating Manne’s model, present no significant differences in terms of their relative strengths.

Observe that for 50% of the LA problem instances, the LP relaxation of Model JS-ATSP1 produced a 0% optimality gap (see Tables 4.2-4.4), whereas for these same instances, Manne’s LP relaxations exhibited an optimality gap of 51.86% at an average. Interestingly, as summarized in Table 4.5, all the LA test instances for which the LP optimality gap equals zero are rectangular, and moreover, the greater the $\frac{n}{m}$ ratio, the larger the number of instances that yield a 0% optimality gap. In fact, our results concerning the tightness of the LP relaxations of the lifted formulations echo earlier observations made in the literature on the influence of the number of machines, in particular, on the hardness of test instances. Taillard [134] and Adams et al. [1] highlighted that limiting the number of machines results in easier instances, and Taillard optimally solved problems having up to 1,000,000 operations as long as the number of machines is not greater than 10. Matsuo et al. [84] and Taillard [134] studied the ratio of the number of jobs, $n$, to the number of machines, $m$, versus the hardness of the benchmark problems, and noted that rectangular instances having a ratio greater than 4 tended to be easier than smaller square ($n \cong m$) instances. In the light of these considerations, it can be noted that Taillard’s TD 71-80 benchmark problems having 2000 operations are rectangular instances with a ratio equal to 5 ($n = 5m$). Caseau and Laburthe [28] observed the difficulty to solve square-dimensioned instances such as LA 36-40 (15 jobs and 15 machines), while rectangular benchmark instances such as LA 31-35 (involving 30 jobs and 10 machines) required much less effort to reach optimality. As far as square-dimensioned test instances and our lifted ATSP-based formulations are concerned, Manne’s LP relaxations achieved an optimality gap of 22.74% at an average, whereas Model JS-ATSP1 exhibited an optimality gap of 19.72% at an average for test instances LA16-20 and LA36-40, thereby somewhat improving on Manne’s formulation.

Furthermore, upon comparing the results in Table 4.1 and Tables 4.2-4.4, it is observed that for rectangular problem instances for which a 0% optimality gap was obtained using the ATSP-based formulations along with $T$ equal to the optimal objective value in Constraints (4.11o)-(4.11p), the same optimality gap could be achieved while using a weaker upper bound $T$ based on the Shifting Bottleneck procedure by Adams et al. [1]. This computational trend is also confirmed for square
test instances, for which the use of the tightest known upper bound as the value of $T$, in lieu of the SBP-based bound only yields occasional, minor tightening. For instance, optimality gaps of 16.45% and 17.02% were achieved for ABZ5 using Model JS-ATSP1 in concert with $T$ equal to the optimal and the SBP objective values, respectively, as an upper bound. It is worth noting that, although the use of tight upper bounds is beneficial for the relative strength of the valid inequalities derived and the lifting strategies we have employed, a good quality upper bound may suffice in the context of our ATSP-based formulations. In addition to the $\frac{n}{m}$ ratio, it is our contention that the relative density of solutions in the vicinity of an optimum might be a more general influencing factor.

The LP relaxations of JS-ATSP1 and JS-ATSP2 are more computationally affordable than those of the RLT-enhanced lifted models, JS-ATSP3 and JS-ATSP4. We have also empirically compared the performance of these models from the viewpoint of solving benchmark problems to optimality. Despite the tightness of our enhanced LP relaxations, preliminary computational results, which we are not reporting here, suggest that the difficulty of solving test instances using MIP formulations remains an open challenge. In fact, even for rectangular test instances that presented a 0% optimality gap for the continuous relaxations of the ATSP-based formulations, the LP solutions produced at the root node by CPLEX present significant fractionality in the binary variables, thereby requiring a great deal of branching for producing good upper bounds, which are identified at deeper levels of the search tree. For such instances, equipping the solver (CPLEX 10.1) with heuristics that identify good upper bounds at early stages of the B&B search would permit a better pruning effect and a faster algorithmic convergence.

4.6.2 One-machine Relaxation

In Table 4.9, we provide a preliminary comparison of the relative strength of the one-machine relaxations based on Model JS-ATSP1 against the standard one-machine relaxations pertaining to Manne’s model, as well as the lower-bounds derived by Applegate and Cook [4] upon imposing eight classes of valid inequalities to Manne’s model (this corresponds to the results reported in [4] as “Cuts 3”). Whereas the one-machine relaxations pertaining to Model JS-ATSP1 offer a potential marginal advantage over the aforementioned relaxations, their strength is accompanied with a prohibitive computational effort. For the instance ABZ6, the one-machine relaxations based on Model JS-ATSP1 required an average CPU time of 1754.48 seconds per machine, and yielded an optimality gap of 10.71% against 10.92% and 11.45% for Applegate and Cook’s strongest relaxation and the standard one-machine relaxation, respectively, where the latter were reported [4] to require
<table>
<thead>
<tr>
<th>Instance, n x m</th>
<th>Metrics</th>
<th>Manne</th>
<th>JS-ATSP1</th>
<th>JS-ATSP2</th>
<th>JS-ATSP3</th>
<th>JS-ATSP4</th>
</tr>
</thead>
<tbody>
<tr>
<td>FT10 10 x 10 (930)</td>
<td>LP</td>
<td>655</td>
<td>738.26</td>
<td>738.26</td>
<td>738.50</td>
<td>738.46</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>29.56</td>
<td>20.61</td>
<td>20.61</td>
<td>20.59</td>
<td>20.59</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.03</td>
<td>2.65</td>
<td>7.46</td>
<td>61.179</td>
<td>64.67</td>
</tr>
<tr>
<td>FT20 20 x 5 (1165)</td>
<td>LP</td>
<td>387</td>
<td>1119</td>
<td>1119</td>
<td>1119</td>
<td>1119</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>66.78</td>
<td>3.94</td>
<td>3.94</td>
<td>3.94</td>
<td>3.94</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.01</td>
<td>53.43</td>
<td>28.49</td>
<td>2167.29</td>
<td>312.60</td>
</tr>
<tr>
<td>LA01 10 x 5 (666)</td>
<td>LP</td>
<td>413</td>
<td>666</td>
<td>666</td>
<td>666</td>
<td>666</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>37.98</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.01</td>
<td>0.98</td>
<td>4.29</td>
<td>8.31</td>
<td>8.04</td>
</tr>
<tr>
<td>LA02 10 x 5 (655)</td>
<td>LP</td>
<td>394</td>
<td>635</td>
<td>635</td>
<td>635</td>
<td>635</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>39.84</td>
<td>3.05</td>
<td>3.05</td>
<td>3.05</td>
<td>3.05</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.01</td>
<td>0.45</td>
<td>4.06</td>
<td>9.78</td>
<td>9.98</td>
</tr>
<tr>
<td>LA03 10 x 5 (597)</td>
<td>LP</td>
<td>349</td>
<td>588</td>
<td>588</td>
<td>588</td>
<td>588</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>41.54</td>
<td>1.50</td>
<td>1.50</td>
<td>1.50</td>
<td>1.50</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.01</td>
<td>0.86</td>
<td>4.71</td>
<td>10.15</td>
<td>11.40</td>
</tr>
<tr>
<td>LA04 10 x 5 (590)</td>
<td>LP</td>
<td>369</td>
<td>537</td>
<td>537</td>
<td>537</td>
<td>537</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>37.43</td>
<td>8.98</td>
<td>8.98</td>
<td>8.98</td>
<td>8.98</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.01</td>
<td>0.71</td>
<td>4.06</td>
<td>14.92</td>
<td>19.45</td>
</tr>
<tr>
<td>LA05 10 x 5 (593)</td>
<td>LP</td>
<td>380</td>
<td>593</td>
<td>593</td>
<td>593</td>
<td>593</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>35.91</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.01</td>
<td>0.43</td>
<td>3.87</td>
<td>10.84</td>
<td>10.81</td>
</tr>
<tr>
<td>LA06 15 x 5 (926)</td>
<td>LP</td>
<td>413</td>
<td>926</td>
<td>926</td>
<td>926</td>
<td>926</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>55.39</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.01</td>
<td>12.18</td>
<td>64.69</td>
<td>122.00</td>
<td>412.90</td>
</tr>
<tr>
<td>LA07 15 x 5 (890)</td>
<td>LP</td>
<td>376</td>
<td>800</td>
<td>800</td>
<td>800</td>
<td>800</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>57.75</td>
<td>10.11</td>
<td>10.11</td>
<td>10.11</td>
<td>10.11</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.01</td>
<td>14.71</td>
<td>55.07</td>
<td>390.38</td>
<td>300.92</td>
</tr>
<tr>
<td>LA08 15 x 5 (863)</td>
<td>LP</td>
<td>369</td>
<td>863</td>
<td>863</td>
<td>863</td>
<td>863</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>57.24</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.01</td>
<td>19.35</td>
<td>55.13</td>
<td>245.59</td>
<td>276.23</td>
</tr>
<tr>
<td>LA09 15 x 5 (951)</td>
<td>LP</td>
<td>382</td>
<td>951</td>
<td>951</td>
<td>951</td>
<td>951</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>59.83</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.01</td>
<td>19.02</td>
<td>55.24</td>
<td>230.95</td>
<td>202.28</td>
</tr>
<tr>
<td>LA10 15 x 5 (958)</td>
<td>LP</td>
<td>443</td>
<td>958</td>
<td>958</td>
<td>958</td>
<td>958</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>53.75</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.01</td>
<td>15.18</td>
<td>52.19</td>
<td>291.72</td>
<td>340.64</td>
</tr>
<tr>
<td>LA11 20 x 5 (1222)</td>
<td>LP</td>
<td>413</td>
<td>1222</td>
<td>1222</td>
<td>1222</td>
<td>1222</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>66.20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.03</td>
<td>48.33</td>
<td>12.21</td>
<td>171.42</td>
<td>543.8</td>
</tr>
<tr>
<td>LA12 20 x 5 (1039)</td>
<td>LP</td>
<td>408</td>
<td>1039</td>
<td>1039</td>
<td>1039</td>
<td>1039</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>60.73</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.01</td>
<td>20.95</td>
<td>24.84</td>
<td>174.19</td>
<td>374.48</td>
</tr>
<tr>
<td>LA13 20 x 5 (1150)</td>
<td>LP</td>
<td>382</td>
<td>1150</td>
<td>1150</td>
<td>1150</td>
<td>1150</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>66.78</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.03</td>
<td>36.97</td>
<td>19.52</td>
<td>175.55</td>
<td>1262.79</td>
</tr>
<tr>
<td>LA14 20 x 5 (1292)</td>
<td>LP</td>
<td>443</td>
<td>1292</td>
<td>1292</td>
<td>1292</td>
<td>1292</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>65.71</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.03</td>
<td>27.01</td>
<td>20.28</td>
<td>440.33</td>
<td>1309.68</td>
</tr>
<tr>
<td>LA15 20 x 5 (1207)</td>
<td>LP</td>
<td>378</td>
<td>1207</td>
<td>1207</td>
<td>1207</td>
<td>1207</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>68.68</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.01</td>
<td>18.24</td>
<td>19.07</td>
<td>189.81</td>
<td>145.97</td>
</tr>
<tr>
<td>ABZ5 10 x 10 (1234)</td>
<td>LP</td>
<td>859</td>
<td>1023.93</td>
<td>1023.57</td>
<td>1025.93</td>
<td>1025.79</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>30.38</td>
<td>17.02</td>
<td>17.05</td>
<td>16.86</td>
<td>16.87</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.01</td>
<td>6.24</td>
<td>14.25</td>
<td>95.72</td>
<td>118.26</td>
</tr>
</tbody>
</table>

Table 4.1: LP relaxations of JS-ATSP\textsubscript{i}, i = 1, ..., 4
<table>
<thead>
<tr>
<th>Instance, $n \times m$</th>
<th>Metrics</th>
<th>Manne</th>
<th>JS-ATSP1</th>
<th>JS-ATSP2</th>
<th>JS-ATSP3</th>
<th>JS-ATSP4</th>
</tr>
</thead>
<tbody>
<tr>
<td>LA01 10 x 5 (666)</td>
<td>LP</td>
<td>413</td>
<td>666</td>
<td>666</td>
<td>666</td>
<td>666</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>37.98</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.01</td>
<td>0.90</td>
<td>4.45</td>
<td>10.01</td>
<td>13.01</td>
</tr>
<tr>
<td>LA02 10 x 5 (655)</td>
<td>LP</td>
<td>394</td>
<td>635</td>
<td>635</td>
<td>635</td>
<td>635</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>39.84</td>
<td>3.05</td>
<td>3.05</td>
<td>3.05</td>
<td>3.05</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.01</td>
<td>1.1</td>
<td>3.93</td>
<td>10.78</td>
<td>11.59</td>
</tr>
<tr>
<td>LA03 10 x 5 (597)</td>
<td>LP</td>
<td>349</td>
<td>588</td>
<td>588</td>
<td>588</td>
<td>588</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>41.54</td>
<td>1.50</td>
<td>1.50</td>
<td>1.50</td>
<td>1.50</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.01</td>
<td>1.0</td>
<td>3.81</td>
<td>15.85</td>
<td>13.48</td>
</tr>
<tr>
<td>LA04 10 x 5 (590)</td>
<td>LP</td>
<td>369</td>
<td>537</td>
<td>537</td>
<td>537</td>
<td>537</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>37.45</td>
<td>8.98</td>
<td>8.98</td>
<td>8.98</td>
<td>8.98</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.01</td>
<td>1.82</td>
<td>3.81</td>
<td>12.79</td>
<td>12.76</td>
</tr>
<tr>
<td>LA05 10 x 5 (593)</td>
<td>LP</td>
<td>380</td>
<td>593</td>
<td>593</td>
<td>593</td>
<td>593</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>35.91</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.01</td>
<td>0.61</td>
<td>3.59</td>
<td>13.78</td>
<td>11.01</td>
</tr>
<tr>
<td>LA06 15 x 5 (926)</td>
<td>LP</td>
<td>413</td>
<td>926</td>
<td>926</td>
<td>926</td>
<td>926</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>55.39</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.01</td>
<td>13.32</td>
<td>36.79</td>
<td>297.57</td>
<td>339.40</td>
</tr>
<tr>
<td>LA07 15 x 5 (890)</td>
<td>LP</td>
<td>376</td>
<td>800</td>
<td>800</td>
<td>800</td>
<td>800</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>57.75</td>
<td>10.11</td>
<td>10.11</td>
<td>10.11</td>
<td>10.11</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.01</td>
<td>10.29</td>
<td>30.46</td>
<td>226.04</td>
<td>419.17</td>
</tr>
<tr>
<td>LA08 15 x 5 (863)</td>
<td>LP</td>
<td>369</td>
<td>863</td>
<td>863</td>
<td>863</td>
<td>863</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>57.24</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.01</td>
<td>20.4</td>
<td>38.64</td>
<td>317.79</td>
<td>390.18</td>
</tr>
<tr>
<td>LA09 15 x 5 (951)</td>
<td>LP</td>
<td>382</td>
<td>951</td>
<td>951</td>
<td>951</td>
<td>951</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>59.83</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.01</td>
<td>17.35</td>
<td>38.70</td>
<td>316.96</td>
<td>354.07</td>
</tr>
<tr>
<td>LA10 15 x 5 (958)</td>
<td>LP</td>
<td>441</td>
<td>958</td>
<td>958</td>
<td>958</td>
<td>958</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>53.75</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.01</td>
<td>15.09</td>
<td>36.70</td>
<td>323.01</td>
<td>352.79</td>
</tr>
<tr>
<td>LA11 20 x 5 (1222)</td>
<td>LP</td>
<td>411</td>
<td>1222</td>
<td>1222</td>
<td>1222</td>
<td>1222</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>66.20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.03</td>
<td>22.78</td>
<td>22.71</td>
<td>132.35</td>
<td>323.59</td>
</tr>
<tr>
<td>LA12 20 x 5 (1039)</td>
<td>LP</td>
<td>408</td>
<td>1039</td>
<td>1039</td>
<td>1039</td>
<td>1039</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>60.73</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.03</td>
<td>15.15</td>
<td>19.70</td>
<td>127.70</td>
<td>1662.94</td>
</tr>
<tr>
<td>LA13 20 x 5 (1150)</td>
<td>LP</td>
<td>382</td>
<td>1150</td>
<td>1150</td>
<td>1150</td>
<td>1150</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>66.78</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.03</td>
<td>21.24</td>
<td>21.65</td>
<td>123.68</td>
<td>123.92</td>
</tr>
<tr>
<td>LA14 20 x 5 (1292)</td>
<td>LP</td>
<td>413</td>
<td>1292</td>
<td>1292</td>
<td>1292</td>
<td>1292</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>65.71</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.03</td>
<td>19.56</td>
<td>14.93</td>
<td>209.75</td>
<td>992.82</td>
</tr>
<tr>
<td>LA15 20 x 5 (1207)</td>
<td>LP</td>
<td>378</td>
<td>1207</td>
<td>1207</td>
<td>1207</td>
<td>1207</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>68.68</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.01</td>
<td>33.54</td>
<td>21.56</td>
<td>228.56</td>
<td>1031.5</td>
</tr>
<tr>
<td>LA16 10 x 10 (945)</td>
<td>LP</td>
<td>717</td>
<td>771 (770.74)</td>
<td>771 (770.62)</td>
<td>773 (772.44)</td>
<td>774 (772.32)</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>24.12</td>
<td>18.41</td>
<td>18.41</td>
<td>18.20</td>
<td>18.20</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.01</td>
<td>1.84</td>
<td>3.17</td>
<td>95.72</td>
<td>118.26</td>
</tr>
<tr>
<td>LA17 10 x 10 (784)</td>
<td>LP</td>
<td>646</td>
<td>649 (648.80)</td>
<td>649 (648.79)</td>
<td>649 (648.90)</td>
<td>649 (648.89)</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>17.60</td>
<td>17.21</td>
<td>17.21</td>
<td>17.21</td>
<td>17.21</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.01</td>
<td>1.73</td>
<td>4.2</td>
<td>80.12</td>
<td>83.84</td>
</tr>
<tr>
<td>LA18 10 x 10 (848)</td>
<td>LP</td>
<td>663</td>
<td>674 (673.22)</td>
<td>674 (673.19)</td>
<td>674 (673.62)</td>
<td>674 (673.51)</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>21.81</td>
<td>20.51</td>
<td>20.51</td>
<td>20.51</td>
<td>20.51</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.01</td>
<td>2.07</td>
<td>4.20</td>
<td>73.42</td>
<td>84.23</td>
</tr>
</tbody>
</table>

Table 4.2: LP relaxations for the problem instances of Lawrence, LA01-LA18
<table>
<thead>
<tr>
<th>Instance, n x m</th>
<th>Metrics</th>
<th>Manne</th>
<th>JS-ATSP1</th>
<th>JS-ATSP2</th>
<th>JS-ATSP3</th>
<th>JS-ATSP4</th>
</tr>
</thead>
<tbody>
<tr>
<td>LA19 10 x 10  (842)</td>
<td>LP</td>
<td>617</td>
<td>685</td>
<td>685</td>
<td>685</td>
<td>685</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>21.81</td>
<td>20.51</td>
<td>20.51</td>
<td>20.51</td>
<td>20.51</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.01</td>
<td>3.65</td>
<td>9.70</td>
<td>72.64</td>
<td>153.12</td>
</tr>
<tr>
<td>LA20 10 x 10  (902)</td>
<td>LP</td>
<td>756</td>
<td>756</td>
<td>756</td>
<td>756</td>
<td>756</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>16.18</td>
<td>16.18</td>
<td>16.18</td>
<td>16.18</td>
<td>16.18</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.01</td>
<td>3.70</td>
<td>10.51</td>
<td>62.25</td>
<td>61.78</td>
</tr>
<tr>
<td>LA21 15 x 10  (1046)</td>
<td>LP</td>
<td>717</td>
<td>935</td>
<td>935</td>
<td>935</td>
<td>935</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>31.45</td>
<td>10.61</td>
<td>10.61</td>
<td>10.61</td>
<td>10.61</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.04</td>
<td>43.31</td>
<td>20.39</td>
<td>469.89</td>
<td>594.17</td>
</tr>
<tr>
<td>LA22 15 x 10  (927)</td>
<td>LP</td>
<td>619</td>
<td>801</td>
<td>801</td>
<td>801</td>
<td>801</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>41.54</td>
<td>1.50</td>
<td>1.50</td>
<td>1.50</td>
<td>1.50</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.06</td>
<td>36.68</td>
<td>28.14</td>
<td>636.48</td>
<td>364.81</td>
</tr>
<tr>
<td>LA23 15 x 10  (1032)</td>
<td>LP</td>
<td>640</td>
<td>1032</td>
<td>1032</td>
<td>1032</td>
<td>1032</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>37.98</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.04</td>
<td>17.18</td>
<td>13.98</td>
<td>293.59</td>
<td>566.50</td>
</tr>
<tr>
<td>LA24 15 x 10  (935)</td>
<td>LP</td>
<td>704</td>
<td>857</td>
<td>857</td>
<td>857</td>
<td>857</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>24.70</td>
<td>8.34</td>
<td>8.34</td>
<td>8.34</td>
<td>8.34</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.04</td>
<td>27.57</td>
<td>19.59</td>
<td>792.37</td>
<td>561.86</td>
</tr>
<tr>
<td>LA25 15 x 10  (977)</td>
<td>LP</td>
<td>723</td>
<td>801 (800.69)</td>
<td>801 (800.56)</td>
<td>802 (801.13)</td>
<td>802 (801.06)</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>25.99</td>
<td>18.01</td>
<td>18.01</td>
<td>17.91</td>
<td>17.91</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.04</td>
<td>73.99</td>
<td>40.67</td>
<td>1149.09</td>
<td>1475.7</td>
</tr>
<tr>
<td>LA26 20 x 10  (1218)</td>
<td>LP</td>
<td>717</td>
<td>1218</td>
<td>1218</td>
<td>1218</td>
<td>1218</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>69.87</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.10</td>
<td>251.01</td>
<td>125.51</td>
<td>4011.41</td>
<td>1852.11</td>
</tr>
<tr>
<td>LA27 20 x 10  (1235)</td>
<td>LP</td>
<td>686</td>
<td>1188</td>
<td>1188</td>
<td>1188</td>
<td>1188</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>44.45</td>
<td>3.80</td>
<td>3.80</td>
<td>3.80</td>
<td>3.80</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.11</td>
<td>330.48</td>
<td>126.09</td>
<td>3491.54</td>
<td>2307.0</td>
</tr>
<tr>
<td>LA28 20 x 10  (1216)</td>
<td>LP</td>
<td>756</td>
<td>1216</td>
<td>1216</td>
<td>1216</td>
<td>1216</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>37.82</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.10</td>
<td>330.48</td>
<td>108.14</td>
<td>2565.33</td>
<td>2523.44</td>
</tr>
<tr>
<td>LA29 20 x 10  (1152)</td>
<td>LP</td>
<td>723</td>
<td>1105</td>
<td>1105</td>
<td>1105</td>
<td>1105</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>37.23</td>
<td>4.07</td>
<td>4.07</td>
<td>4.07</td>
<td>4.07</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.10</td>
<td>282.48</td>
<td>116.70</td>
<td>1591.55</td>
<td>5571.30</td>
</tr>
<tr>
<td>LA30 20 x 10  (1355)</td>
<td>LP</td>
<td>726</td>
<td>1355</td>
<td>1355</td>
<td>1355</td>
<td>1355</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>46.42</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.11</td>
<td>153.96</td>
<td>99.98</td>
<td>3449.98</td>
<td>2287.24</td>
</tr>
<tr>
<td>LA31 30 x 10  (1222)</td>
<td>LP</td>
<td>413</td>
<td>1222</td>
<td>1222</td>
<td>1222</td>
<td>1222</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>66.20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.03</td>
<td>21.76</td>
<td>21.53</td>
<td>132.18</td>
<td>323.671</td>
</tr>
<tr>
<td>LA32 30 x 10  (1850)</td>
<td>LP</td>
<td>756</td>
<td>1850</td>
<td>1850</td>
<td>1850</td>
<td>1850</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>59.13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.45</td>
<td>5964.65</td>
<td>2066.22</td>
<td>29360.90</td>
<td>32709.01</td>
</tr>
<tr>
<td>LA33 30 x 10  (1719)</td>
<td>LP</td>
<td>723</td>
<td>1719</td>
<td>1719</td>
<td>1719</td>
<td>1719</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>57.94</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.45</td>
<td>5628.93</td>
<td>922.82</td>
<td>19525.40</td>
<td>40686.7</td>
</tr>
<tr>
<td>LA34 30 x 10  (1721)</td>
<td>LP</td>
<td>656</td>
<td>1721</td>
<td>1721</td>
<td>1721</td>
<td>1721</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>61.88</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.45</td>
<td>4399.48</td>
<td>705.02</td>
<td>21246.70</td>
<td>34888.70</td>
</tr>
<tr>
<td>LA35 30 x 10  (1888)</td>
<td>LP</td>
<td>647</td>
<td>1888</td>
<td>1888</td>
<td>1888</td>
<td>1888</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>65.73</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.01</td>
<td>2768.65</td>
<td>1547.45</td>
<td>11611.9</td>
<td>28147.1</td>
</tr>
<tr>
<td>LA36 15 x 15  (1268)</td>
<td>LP</td>
<td>948</td>
<td>1068 (1067.79)</td>
<td>1068 (1067.79)</td>
<td>1068 (1067.83)</td>
<td>1068 (1067.82)</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>25.23</td>
<td>15.77</td>
<td>15.77</td>
<td>15.77</td>
<td>15.77</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.09</td>
<td>172.92</td>
<td>61.21</td>
<td>4391.34</td>
<td>1746.36</td>
</tr>
</tbody>
</table>

Table 4.3: LP relaxations for the problem instances of Lawrence, LA19-LA36
<table>
<thead>
<tr>
<th>Instance, ( n \times m )</th>
<th>Metrics</th>
<th>Manne</th>
<th>JS-ATSP1</th>
<th>JS-ATSP2</th>
<th>JS-ATSP3</th>
<th>JS-ATSP4</th>
</tr>
</thead>
<tbody>
<tr>
<td>LA37 15 × 15</td>
<td>LP</td>
<td>986</td>
<td>1017 (1016.66)</td>
<td>1017 (1016.66)</td>
<td>1020 (1019.03)</td>
<td>1020 (1019.001)</td>
</tr>
<tr>
<td>(1397)</td>
<td>%Gap</td>
<td>29.42</td>
<td>27.20</td>
<td>27.20</td>
<td>26.98</td>
<td>26.98</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.09</td>
<td>211.31</td>
<td>114.04</td>
<td>6494.45</td>
<td>3377.46</td>
</tr>
<tr>
<td>LA38 15 × 15</td>
<td>LP</td>
<td>943</td>
<td>443</td>
<td>943</td>
<td>943</td>
<td>943</td>
</tr>
<tr>
<td>(1196)</td>
<td>%Gap</td>
<td>21.15</td>
<td>21.15</td>
<td>21.15</td>
<td>21.15</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.09</td>
<td>107.70</td>
<td>65.74</td>
<td>1848.36</td>
<td>1420.18</td>
</tr>
<tr>
<td>LA39 15 × 15</td>
<td>LP</td>
<td>922</td>
<td>990 (989.37)</td>
<td>990 (989.37)</td>
<td>991 (990.74)</td>
<td>991 (990.72)</td>
</tr>
<tr>
<td>(1233)</td>
<td>%Gap</td>
<td>28.22</td>
<td>19.70</td>
<td>19.70</td>
<td>19.62</td>
<td>19.62</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.10</td>
<td>174.43</td>
<td>102.87</td>
<td>2831.23</td>
<td>1887.35</td>
</tr>
<tr>
<td>LA40 15 × 15</td>
<td>LP</td>
<td>955</td>
<td>962 (961.21)</td>
<td>962 (961.20)</td>
<td>962 (961.82)</td>
<td>962 (961.80)</td>
</tr>
<tr>
<td>(1222)</td>
<td>%Gap</td>
<td>21.81</td>
<td>20.51</td>
<td>20.51</td>
<td>20.51</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.09</td>
<td>193.84</td>
<td>121.32</td>
<td>4194.27</td>
<td>3786.11</td>
</tr>
</tbody>
</table>

Table 4.4: LP relaxations for the problem instances of Lawrence, LA37-LA40

<table>
<thead>
<tr>
<th>( \frac{n}{m} ) ratio</th>
<th>Size</th>
<th>Nbr. of instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td>15 × 10</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>10 × 5</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>20 × 10</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>15 × 5</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>30 × 10</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>20 × 5</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 4.5: \( \frac{n}{m} \) ratio for the LA instances that yielded a 0% optimality gap

<table>
<thead>
<tr>
<th>Instance, ( n \times m )</th>
<th>Metrics</th>
<th>Manne</th>
<th>JS-ATSP1</th>
<th>JS-ATSP2</th>
<th>JS-ATSP3</th>
<th>JS-ATSP4</th>
</tr>
</thead>
<tbody>
<tr>
<td>FT06 6 × 6</td>
<td>LP</td>
<td>47</td>
<td>47</td>
<td>47</td>
<td>47</td>
<td></td>
</tr>
<tr>
<td>(55)</td>
<td>%Gap</td>
<td>17.02</td>
<td>17.02</td>
<td>17.02</td>
<td>17.02</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.0</td>
<td>0.10</td>
<td>0.14</td>
<td>0.43</td>
<td>0.40</td>
</tr>
<tr>
<td>FT10 10 × 10</td>
<td>LP</td>
<td>655</td>
<td>739 (738.99)</td>
<td>739 (738.99)</td>
<td>740 (739.41)</td>
<td>740 (739.32)</td>
</tr>
<tr>
<td>(930)</td>
<td>%Gap</td>
<td>20.56</td>
<td>20.53</td>
<td>20.53</td>
<td>20.43</td>
<td>20.43</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.01</td>
<td>2.71</td>
<td>7.14</td>
<td>60.64</td>
<td>69.37</td>
</tr>
<tr>
<td>FT20 20 × 5</td>
<td>LP</td>
<td>387</td>
<td>1119</td>
<td>1119</td>
<td>1119</td>
<td>1119</td>
</tr>
<tr>
<td>(1165)</td>
<td>%Gap</td>
<td>66.78</td>
<td>3.94</td>
<td>3.94</td>
<td>3.94</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.04</td>
<td>66.99</td>
<td>37.60</td>
<td>1133.3</td>
<td>542.07</td>
</tr>
</tbody>
</table>

Table 4.6: LP relaxations for the problem instance of Fisher and Thompson

<table>
<thead>
<tr>
<th>Instance, ( n \times m )</th>
<th>Metrics</th>
<th>Manne</th>
<th>JS-ATSP1</th>
<th>JS-ATSP2</th>
<th>JS-ATSP3</th>
<th>JS-ATSP4</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABZ5 10 × 10</td>
<td>LP</td>
<td>859</td>
<td>1031 (1030.24)</td>
<td>1030 (1029.59)</td>
<td>1035 (1034.13)</td>
<td>1034 (1033.64)</td>
</tr>
<tr>
<td>(1234)</td>
<td>%Gap</td>
<td>30.38</td>
<td>16.45</td>
<td>16.53</td>
<td>16.12</td>
<td>16.20</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.01</td>
<td>2.53</td>
<td>14.0</td>
<td>60.73</td>
<td>118.26</td>
</tr>
<tr>
<td>ABZ6 10 × 10</td>
<td>LP</td>
<td>742</td>
<td>745 (744.68)</td>
<td>745 (744.63)</td>
<td>746 (745.17)</td>
<td>746 (745.12)</td>
</tr>
<tr>
<td>(943)</td>
<td>%Gap</td>
<td>21.31</td>
<td>20.99</td>
<td>20.99</td>
<td>20.89</td>
<td>20.89</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.01</td>
<td>2.51</td>
<td>4.31</td>
<td>110.56</td>
<td>238.75</td>
</tr>
<tr>
<td>ABZ7 20 × 15</td>
<td>LP</td>
<td>410</td>
<td>556</td>
<td>556</td>
<td>556</td>
<td></td>
</tr>
<tr>
<td>(656)</td>
<td>%Gap</td>
<td>37.5</td>
<td>15.24</td>
<td>15.24</td>
<td>15.24</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.25</td>
<td>2506.9</td>
<td>895.26</td>
<td>14110.7</td>
<td>14074.7</td>
</tr>
<tr>
<td>ABZ8 20 × 15</td>
<td>LP</td>
<td>443</td>
<td>566</td>
<td>566</td>
<td>566</td>
<td></td>
</tr>
<tr>
<td>(665)</td>
<td>%Gap</td>
<td>31.87</td>
<td>14.88</td>
<td>14.88</td>
<td>14.88</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.23</td>
<td>1365.47</td>
<td>809.83</td>
<td>10587.8</td>
<td>11682.1</td>
</tr>
<tr>
<td>ABZ9 20 × 15</td>
<td>LP</td>
<td>467</td>
<td>517</td>
<td>517</td>
<td>517</td>
<td></td>
</tr>
<tr>
<td>(679)</td>
<td>%Gap</td>
<td>31.22</td>
<td>23.85</td>
<td>23.85</td>
<td>23.85</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.21</td>
<td>1975.16</td>
<td>1528.82</td>
<td>29254.6</td>
<td>30520.1</td>
</tr>
</tbody>
</table>

Table 4.7: LP relaxations for the problem instances of Adams et al.
<table>
<thead>
<tr>
<th>Instance, $n \times m$</th>
<th>Metrics</th>
<th>Manne</th>
<th>JS-ATSP1</th>
<th>JS-ATSP2</th>
<th>JS-ATSP3</th>
<th>JS-ATSP4</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORB1 10 × 10 (1059)</td>
<td>LP</td>
<td>695</td>
<td>817 (816.56)</td>
<td>817 (816.59)</td>
<td>823 (822.66)</td>
<td>823 (822.61)</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>34.37</td>
<td>22.85</td>
<td>22.85</td>
<td>22.28</td>
<td>22.28</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.04</td>
<td>5.18</td>
<td>5.45</td>
<td>57.04</td>
<td>62.75</td>
</tr>
<tr>
<td>ORB2 10 × 10 (888)</td>
<td>LP</td>
<td>620</td>
<td>699 (698.65)</td>
<td>699 (698.65)</td>
<td>701 (700.40)</td>
<td>701 (700.40)</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>30.18</td>
<td>21.28</td>
<td>21.05</td>
<td>21.05</td>
<td>21.05</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.01</td>
<td>2.01</td>
<td>3.34</td>
<td>51.96</td>
<td>64.48</td>
</tr>
<tr>
<td>ORB3 10 × 10 (1005)</td>
<td>LP</td>
<td>648</td>
<td>766 (765.42)</td>
<td>766 (765.42)</td>
<td>768 (767.37)</td>
<td>768 (767.16)</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>35.52</td>
<td>23.78</td>
<td>23.78</td>
<td>23.58</td>
<td>23.58</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.03</td>
<td>4.09</td>
<td>4.62</td>
<td>44.25</td>
<td>60.59</td>
</tr>
<tr>
<td>ORB4 10 × 10 (1005)</td>
<td>LP</td>
<td>753</td>
<td>787 (786.01)</td>
<td>786 (785.95)</td>
<td>788 (787.35)</td>
<td>788 (787.35)</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.01</td>
<td>1.56</td>
<td>2.20</td>
<td>40.79</td>
<td>47.17</td>
</tr>
<tr>
<td>ORB5 10 × 10 (887)</td>
<td>LP</td>
<td>584</td>
<td>714 (713.24)</td>
<td>714 (713.18)</td>
<td>715 (714.60)</td>
<td>715 (714.60)</td>
</tr>
<tr>
<td></td>
<td>%Gap</td>
<td>34.16</td>
<td>19.50</td>
<td>19.50</td>
<td>19.39</td>
<td>19.39</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>0.01</td>
<td>2.03</td>
<td>5.01</td>
<td>55.09</td>
<td>64.29</td>
</tr>
</tbody>
</table>

Table 4.8: LP relaxations for the problem instances of Applegate and Cook

<table>
<thead>
<tr>
<th>Instance</th>
<th>Manne (1-mac)</th>
<th>JS-ATSP1 (LP)</th>
<th>JS-ATSP1 (1-mac)</th>
<th>Cuts 3 in Apple-</th>
<th>gate &amp; Cook [4]</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABZ6 (943)</td>
<td>835</td>
<td>745</td>
<td>842</td>
<td>840</td>
<td></td>
</tr>
<tr>
<td>ORB1 (1059)</td>
<td>929</td>
<td>817</td>
<td>944</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.9: Comparison of various relaxations with the strongest cuts by Applegate and Cook

5257.35 CPU seconds and 0.12 CPU seconds, respectively.

4.6.3 Nondifferentiable Optimization

In Table 4.10, we provide a preliminary comparison between solving Manne’s model using CPLEX 10.1 versus utilizing the MIP-NDO code [75] to solve the Lagrangian dual JLD1 at the root node via a nondifferentiable optimization routine (see Sherali and Lim [119] and Lim and Sherali [76]), which employs a crossover to a simplex solver using the NDO-based advanced starting solution. For MIP-NDO, upon solving the root node relaxation, the remainder of the search tree nodes are analyzed using the default CPLEX MIP solver. The reported results suggest that using MIP-NDO in lieu of CPLEX 10.1 at just the root node helped solve only one of the four problems tested. It is yet of interest to study different NDO-simplex/barrier hybrid schemes, and to utilize these optimization routines for all the B&B search tree nodes instead of applying it only at the root node.

<table>
<thead>
<tr>
<th>Instance</th>
<th>MIP-NDO</th>
<th>CPLEX 10.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>LA02</td>
<td>2491.00</td>
<td>442.55</td>
</tr>
<tr>
<td>LA03</td>
<td>121.00</td>
<td>98.53</td>
</tr>
<tr>
<td>LA04</td>
<td>17.00</td>
<td>64.51</td>
</tr>
<tr>
<td>LA05</td>
<td>165.00</td>
<td>119.58</td>
</tr>
</tbody>
</table>

Table 4.10: Comparison of the CPU times (secs) for MIP-NDO and Manne’s model
4.6.4 Performance of the B&B Algorithm

The B&B algorithm proposed in Section 4.4 has been coded in AMPL/CPLEX and tested on various small-sized problem instances. In its present form, the implemented B&B appears to be computationally prohibitive for problem instances of size \((n \times m)\) larger than or equal to \(6 \times 6\). Our preliminary computational results suggest that the RLT-based linear programming bounding problem, JLP, induces no improvements on the basic LP relaxations of Manne’s model at the root node as well as at the subsequent deeper search tree nodes. Although algorithmic performance may be improved by triggering certain inference rules upon implementing a branching operation, the contribution of Problem JLP to such enhancements would be minimal, if any. It is likely that the proposed SDP-enhanced Lagrangian dual formulation, JLD2, would produce tighter lower bounds, outperforming JLP, and can be embedded within our B&B algorithmic framework. This investigation is proposed for future research.

4.7 Conclusions and Directions for Future Research

We have proposed novel continuous nonconvex as well as lifted discrete formulations for the challenging class of job-shop scheduling problems with the objective of minimizing the maximum completion time (i.e., minimizing the makespan). Motivated by the encouraging results that are reported in Chapter 3, and more generally in the literature on the benefits of the RLT methodology for minimax and discrete optimization problems, we developed an RLT-enhanced continuous nonconvex model for the job-shop problem based on a quadratic formulation of the job sequencing constraints on machines due to Nepomiaistchy [95]. The lifted linear programming relaxation that is induced by this formulation was then embedded in a globally convergent branch-and-bound algorithm. Furthermore, we designed another novel formulation for the job-shop scheduling problem that possesses a tight continuous relaxation, where the non-overlapping job sequencing constraints on machines are modeled via a lifted asymmetric traveling salesman problem (ATSP) construct, and specific sets of valid inequalities and RLT-based enhancements are incorporated to further tighten the resulting mathematical program.

Our extensive computational experience revealed that the LP relaxations produced by the lifted ATSP-based models provide very tight lower bounds, and substantially reduce the optimality gap that characterizes the LP relaxation of the popular formulation due to Manne [81]. Notably, our lifted ATSP-based formulation produced a 0% optimality gap via the root node LP relaxation for 50% of the classical problem instances due to Lawrence [74], thereby substantially dominating other
alternative mixed-integer programming models available for this class of problems. In addition, our results concerning the tightness of the LP relaxations of the lifted formulations echo earlier observations made in the literature on the influence of the number of machines, in particular, on the hardness of test instances. Explicitly, our computational experience confirmed that the best results are obtained for rectangular instances that are characterized by a relatively large $\frac{n}{m}$ ratio. To take this promising empirical evidence one step further, we suggest that a theoretical investigation of dominance relationships between our ATSP-based formulation and alternative MIP formulations of the JSSP be conducted for future research. We also propose to evaluate the RLT-based Lagrangian dual formulations, and possibly integrate these within our B&B algorithm in lieu of the RLT-based linear programming relaxation to accelerate the computational performance and enhance the B&B pruning effect. Finally, it would be worthwhile to apply the general-purpose lifting procedures introduced in Chapter 3 for strengthening the JSSP formulation, and compare the induced relaxation against our ATSP-based formulations that were lifted using specialized valid inequalities and RLT constructs.
Chapter 5

Joint Vehicle Assembly-Routing Problems: An Integrated Modeling and Optimization Approach

This paper examines logistical systems where it is desirable to appropriately ascertain the joint composition of the sequences of vehicles that are to be physically connected along with determining their delivery routes. Such assembly-routing problems occur in the truck manufacturing industry, for example, where combinations of trucks to be delivered to dealerships are composed and are subsequently dispatched via appropriately optimized delivery routes, which are in turn restricted by the particular sequence in which the trucks are connected. A similar structure is exhibited in the business of shipping goods via boat-towed barges along inland waterways, or via trains through railroad networks. We present a novel unifying model and a column generation-based optimization approach for this challenging class of joint vehicle assembly-routing problems. In addition, we suggest several extensions to accommodate particular industrial settings where assembly sequence-dependent delivery routes are necessary, as well as those where driver- and equipment-based restrictions are imposed. Computational experience is provided using realistic data from a case-study involving a major truck manufacturing company to demonstrate the applicability of the proposed methodology in practice.

5.1 Introduction and Motivation

The ever-increasing competition in the marketplace, along with the geographical extension and spread of customer networks, requires the development of sophisticated mathematical models and
quantitative analyses to optimize logistical operations. Accordingly, we address in this chapter the design of suitable mathematical programming models in concert with efficient, large-scale optimization algorithms and strategies for a class of joint vehicle assembly-routing problems (VARP). This class of problems arises in the context of several applications in various industrial settings such as truck shipment operations, and the shipment of goods via boat-towed barges through inland waterways, or via trains through rail systems. In the first application, given a set of ready-to-deliver trucks, the problem posed is to compose sequences of trucks to be physically connected and then subsequently routed to the customer or dealership locations. The truck models impose restrictions on which ones can be physically connected, and the delivery route is constrained by the composed sequence of trucks. Consequently, the two problems of assembly and routing are intertwined and need to be considered jointly in order to make effective decisions. The essence of this problem also arises in the context of shipping goods through inland waterways, where goods are loaded on barges, and sequences of barges need to be selected and connected to leading boats. It is critical to appropriately compose the combination of barges as well as to simultaneously determine an optimal route for each leading boat in order to deliver the loaded barges to the destinations at least cost. For such an application, Taylor et al. [135] propose a simulation-based approach to assign barge freights to boats having multiple loading and unloading locations within an inland waterway network. Likewise, the shipment of goods via rail systems exhibits a similar structure; goods are loaded on railroad cars or wagons, and there is a need to jointly optimize the composition of trains and the associated delivery routes. We shall generically employ the term vehicle to designate trucks, barges, or rail-cars. Minor appropriate adaptations can be brought to bear on our proposed modeling framework to address specific business contexts.

The problem under investigation can be broadly classified under the umbrella of so-called vehicle routing problems (VRP) [55]. The general VRP is an NP-hard combinatorial optimization problem for which a broad spectrum of exact solution methods [73] and heuristics [32] have been explored. In particular, column generation ([39], [79]) provides a powerful approach for partitioning problems [46], and has been successfully applied to various VRP-related problems ([30], [37], [38]). However, the integration of both assembly and routing operations presents unique characteristics that render the VARP novel and challenging. Despite the substantial literature on the VRP and its classical variants, integrated models where the VRP is coupled with additional operational decisions, such as location, assembly, or scheduling, have received less attention. Some existing research has been devoted to location-routing problems (LRP) [90] where facility location problems are tackled in
conjunction with vehicle routing decisions. Also, several studies have been conducted to investigate joint vehicle routing and (crew) scheduling problems (VRSP) [21] for airlines ([87], [88]) and ferry transportation ([69], [72]).

The class of joint vehicle assembly-routing problems that we introduce in this chapter is an important integrated modeling paradigm in this vein, and arose in our recent work with a major truck manufacturing company. This company has a central facility at which manufactured trucks are assembled and subsequently distributed to a large network of dealers and customers located in North America. While being assembled, each option-customized truck is assigned a chassis number earmarked for a specific dealership. The problem is complicated by several features such as the high volume of trucks to be shipped over a planning horizon of a few days, the variety of truck models that can be ordered by any customer, and specified compatibility restrictions between the truck models that govern the validity of different possible assembly configurations (i.e., the manner in which sequences of trucks are physically connected – see Figure 5.1 for an illustration). In essence, the decision-making problem seeks to optimally partition the set of available vehicles by creating assembly combinations and delivery routes over a planning horizon of several days, while satisfying the feasibility constraints governing both these operations. Naturally, casting the problem into a mathematical program where all possible assembly combinations are explicitly considered to generate the best vehicle partitioning and routing scheme would produce a colossal model of little practical use. Rather, we exploit the structure of the problem and design an efficient column generation-based heuristic strategy that identifies promising assembly combinations and their associated routes in an iterative fashion to derive near-optimal solutions.

The remainder of this chapter is organized as follows. In Section 5.2, we introduce our notation along with a formal description of the problem. Thereafter, in Section 5.3, we present our proposed integrated model and optimization approach to the VARP. Several extensions and specializations to this general modeling framework are discussed in Section 5.4 for accommodating different industrial requirements where assembly sequence-dependent delivery routes are necessary, as well as those
where driver- and equipment-based restrictions are imposed. Computational results for practical problem instances are reported in Section 5.5 in order to demonstrate the applicability of the proposed methodology in an industrial setting. Finally, Section 5.6 concludes the chapter with a summary and directions for future research.

5.2 Problem Description and Notation

The shipment operations in the VARP involve two interrelated decisions, namely, vehicle assembly and vehicle routing, in order to satisfy the demand emanating from a network of customers or delivery destinations. In the former, various types or models of vehicles need to be connected together for shipment according to one of several compatible configurations. The set of valid vehicle model combinations are specified in a so-called assembly matrix. In addition to single vehicles, such valid combinations may include pairs of vehicles or several vehicles placed between a lead-vehicle and a rear-vehicle. If, however, all pairs of vehicle models are compatible for connection in the application under investigation, the need for specifying an assembly matrix does not arise. However, restrictions pertaining to the assembly sequence size need to be respected, i.e., the sequence- or tow-size, and hence the number of destinations in the routing of a given vehicle combination, should not exceed some maximal permitted value, $s_{\text{max}}$. Furthermore, as discussed in the extensions addressed in Section 5.4.1, some industrial settings (such as truck delivery operations) require the delivery route to be coordinated with the specific vehicle-assembly composition utilized. In such contexts, the next location to be visited in the routing sequence must be that of the current lead-and/or rear-vehicle. Once the current combination of vehicles reaches a certain delivery location, the relevant vehicles are dropped off, and the remaining set of vehicles, if any, heads to the next similarly constrained destination.

In a nutshell, the VARP consists in determining the best vehicle configuration strategy and the associated routes so as to minimize the total operational costs over some planning horizon. These costs include that for fuel, assembly and disassembly, returning the driver from the final destination back to the source, carrying inventory of unshipped vehicles at the source, and lateness penalties for delivering vehicles beyond the designated due-dates.

In this section, we provide a formal description of the VARP, along with the inputs, parameters and decision variables that are involved in our proposed modeling and optimization approach. Several extensions of this basic framework to consider assembly sequence-dependent routing and further
driver and equipment resource-based restrictions are addressed in Sections 4.1 and 4.2, respectively.

**Input**

- **Single source**: Vehicles are assembled at and shipped from this facility (multiple sources can be handled similarly in a parallel fashion).
- **Destinations** $\ell = 1, ..., L$ representing delivery or customer locations.
- **Planning horizon of** $D$ days, $d = 0, 1, ..., D$, where $d = 0$ is the current day for a model run. (Typically, $D = 2$ in our trucking company case-study, where problems are solved in a rolling horizon framework.)
- $s_{max} \equiv$ maximal assembly sequence- or tow-size. (For instance, $s_{max} = 4$ in the truck shipment case-study we consider in this chapter.)
- **Vehicle models or types** $r = 1, ..., R$ that are involved. (Due to compatibility constraints between models, this identification is critical for truck shipment operations, whereas it might be superfluous in the context of barges or rail-cars.)
- Let $t = 1, ..., T$ index the vehicles (of different types) that are included in the model run for which specific data on availabilities, destinations, and due-dates are known.
- $r_t \in \{1, ..., R\} \equiv$ type of vehicle $t$.
- $\alpha_t \equiv$ day $\in \{0, 1, ..., D\}$ on which vehicle $t$ is made available for possible shipment.
- $\beta_t \equiv$ due-date for vehicle $t$.
- $\ell_t \equiv$ destination $\in \{1, ..., L\}$ earmarked for the particular vehicle $t$.
- Graph $G(N, A)$ represents the vehicle delivery network, where $N$ and $A$ are respectively the sets of nodes and arcs, and where Node 0 represents the source location, Node $t$ represents vehicle $t$ (hence, distinct nodes might represent the same location), and $(i, j) \in A$ represents a directed arc between the pair of nodes $i$ and $j$. (The network may or may not be a complete graph on the nodes $1, ..., T$; also arcs $(i, 0)$ do not exist, $\forall i = 1, ..., T$.)
- $d_{ij} = $ shipment distance associated with arc $(i, j)$.
- $p_{ij} = $ time to traverse arc $(i, j)$ (in days). (Note: If $\ell_i = \ell_j$, then $d_{ij} = 0$ and $p_{ij} = 0$.)
**Cost Coefficients**

- \( c_1 \equiv \) fuel cost per mile.
- \( c_{2h} \equiv \) assembly and disassembly cost that is associated with configuration \( h \) (where \( h = 1, \ldots, H \), indexes the different configurations composed according to the assembly matrix, as defined in the sequel).
- \( c_{3\ell} \equiv \) cost of transporting a driver back to the source from destination \( \ell \), for \( \ell = 1, \ldots, L \).
- \( c_4 \equiv \) inventory carrying cost per day per vehicle parked unshipped at the source.
- \( c_{5\ell} \equiv \) per unit, per day, penalty cost for being late at delivery location \( \ell \).

**Assembly Matrix**

This matrix delineates all admissible combinations of vehicle models, including single vehicles, pairs of vehicles, and configurations of more than two vehicles that compose specified vehicle types between a particular lead-vehicle and rear-vehicle. For any vehicle combination, or *pattern*, to be valid, the underlying composition must comply with this assembly matrix.

**Modeling Constructs**

Consider the following definition of *patterns* \( p^d_j, j = 1, \ldots, J \):

\[
\begin{pmatrix}
    t = 1 \\
    t = 2 \\
    \vdots \\
    t \\
    \vdots \\
    t = T
\end{pmatrix}
\]

where \( p^d_{jt} = \begin{cases} 
1 & \text{if vehicle } t \text{ is included in pattern } j, \text{ which is designated for shipment on day } d \\
0, & \text{otherwise.}
\end{cases} \)

Note that the day \( d \) appended to characterize a particular pattern is the earliest possible day for which all the vehicles configured in the pattern are made available. Hence, the index \( d \) is directly related to the principal index \( j \) that (implicitly) enumerates the different possible patterns.
**Pattern Cost**

Given a certain pattern, \( p^d_j \), the destinations involved and the day that the trip begins are known. Some destinations might receive more than one vehicle, and each pattern includes at most \( s_{max} \) destinations. In addition to the assembly-disassembly costs, any such assembly and routing sequence involves the following costs:

(a) Mileage cost, based on \( c_1 \).
(b) Driver return cost, based on \( c_3 \ell \) and the final destination visited in the delivery route.
(c) Penalty for lateness costs, based on \( c_5 \ell \) and the (integer-valued) days on which deliveries are made.

Since the number of destinations in any pattern is at most \( s_{max} \), which is typically small (of the order of four in our trucking case-study), the best routing sequence that minimizes the total cost of factors (a), (b), and (c) above can be found by enumeration. (Alternatively, this could be accomplished by formulating and solving an optimization subproblem.) This complexity is easily handled by the proposed pricing procedure where a promising pattern is generated along with its best routing sequence. The accompanying cost comprising the items (a), (b), and (c) that corresponds to this optimal routing sequence is denoted \( c^{d,\text{seq}}_{0j} \). In addition, \( c^d_{0j} \), the overall pattern cost factor, incorporates the costs for assembling and disassembling the vehicles, as well as the inventory carrying costs. Hence, we get

\[
c^d_{0j} = c^{d,\text{seq}}_{0j} + c_{2h} + \sum_{t : p^d_{jt} = 1} [d - \alpha_t]c_4
\]

where, as discussed before, \( c_{2h} \) depends on the number of vehicles involved in the particular configuration \( h \) associated with the pattern \( p^d_j \).

**Average Load-Factor**

Associated with each pattern \( j = 1, ..., J \), is a load-factor, \( LF_j \), defined as the number of trucks in this pattern. Management may desire the average load-factor to be at least a certain threshold amount \( \theta_{LF} \) (say, \( \theta_{LF} \in [2.25, 3.25] \)). Alternatively, we could simply focus the optimization effort on the total cost exclusively, i.e., use \( \theta_{LF} = 0 \) in the model. In our approach, we explore both these options, treating the average load-factor restrictions as a soft-constraint whenever \( \theta_{LF} > 0 \). In addition, we associate a weighted load-factor, \( WLF_j \), with each pattern \( j \) as defined below:
\[
WLF_j = \left[ \sum_{\text{legs of trip}} \frac{\text{(distance on leg) (# of vehicles on leg)}}{\text{(total distance of trip)}} \right].
\] (5.2)

The \( WLF \)-values reflect the distribution of the load-factors along the delivery routes, where an average weighted load-factor that ranges between 1.75 and 2.25 is typically desirable in the trucking case-study investigated in this chapter. Observe, however, that an exact representation of \( WLF \) within the proposed model is beyond the scope of this chapter, and would, in fact, introduce significant nonlinearities, rendering such an approach intractable. Our optimization framework directly incorporates \( LF \)-based restrictions, and only separately monitors the \( WLF \)-values for accounting purposes. Section 5.6 provides some additional comments on actually integrating \( WLF \)-based restrictions within the proposed model.

### 5.3 Mathematical Model and Optimization Algorithm

As alluded in Section 5.1, the proposed column generation-based approach relies on a dynamic model generation and solution strategy that is coordinated using a defined (restricted) master program and a subproblem. The restricted master program seeks to optimize a vehicle set partitioning problem with a limited subset of patterns, and its linear programming (LP) relaxation produces dual variables for the subproblem. The latter triggers a pricing mechanism that investigates the existence of any “promising” patterns to be introduced into the restricted master program, along with their best associated routing sequences and costs. If any such improving candidate is obtained via the subproblem, the set of patterns in the restricted master program is augmented accordingly. If, however, no candidate pattern is generated by the subproblem, we conclude that no further improvement to the continuous (LP) relaxation is possible, and we next determine an optimal integer solution to the current discrete restricted master program and terminate the optimization procedure (without any further branch-and-price operations).

We provide below a mathematical formulation of the restricted master program, and discuss its initialization, as well as the pricing mechanism embedded in the subproblem in order to generate further attractive patterns.
5.3.1 Main Model: Set Partitioning Problem (SPP)

Let
\[ x_j = \begin{cases} 
1 & \text{if pattern } j \text{ is selected,} \\
0 & \text{otherwise}, \forall j = 1, ..., J, 
\end{cases} \]

and denote \([1]\) as a vector of \(T\)-ones. Then the principal set partitioning problem [94] to be solved may be stated as follows.

**SPP:**

\[
\text{Minimize } \sum_{j=1}^{J} c_{0j}^d x_j + \mu s \\
\text{subject to } \sum_{j=1}^{J} \left[ P_j^d \right] x_j = [1] \\
\sum_{j=1}^{J} (LF_j) x_j + s \geq \left( \sum_{j=1}^{J} x_j \right) \theta_{LF} \\
x \text{ binary, } s \geq 0.
\]

Constraint (5.3b) requires that each vehicle be delivered via some particular pattern and, therefore, achieves a partitioning scheme for the set of vehicles. Constraint (5.3c) is a soft-constraint that encourages the average load-factor attained to be at least \(\theta_{LF}\), failing which, the net total load-factor deficit \(s \equiv \max \{0, \sum_j (\theta_{LF} - LF_j) x_j\}\) is penalized in the objective function by a commensurate cost parameter \(\mu\). A suitable value of \(\mu\) can be prescribed based on a sensitivity analysis using a variety of real data scenarios. The objective function (5.3a) minimizes the total cost incurred, including the load-factor penalty term. As suggested above, model runs could be performed using either \(\theta_{LF} > 0\) (e.g., \(\theta_{LF} \in [2.25, 3.25]\) is recommended by our trucking case-study), or with \(\theta_{LF} = 0\), whence Constraint (5.3c) and the associated objective term would be omitted.

5.3.2 Pattern Generation and Algorithmic Scheme

The key feature that dictates the utility of this model is the mix of patterns generated. The structure of Problem SPP can readily handle thousands of columns, which can be generated in two phases.

**Phase I:** To initialize the set of patterns in Problem SPP, all single load columns are generated, which ensures the feasibility of the starting solution. In addition, patterns that are feasible to
the assembly matrix and availability restrictions, and are intuitively appealing with respect to the visited relative destination locations, can be pre-generated. Accordingly, judicious configurations based on experience, historical practice, and expert opinion can be advantageously incorporated into the decision-making process. In addition, certain proposed patterns can be designated to be forcibly selected (by setting the corresponding $x_j$-value to one), if so desired.

**Phase II:** This is an automatic column generation phase that analytically constructs attractive columns as desired. At any iterative step in the overall process, we first solve the LP relaxation of (5.3), denoted SPP, where the binary restriction in Constraint (5.3d) is replaced by $x \geq 0$ (noting that (5.3b) enforces $x_j \leq 1, \forall j$). Let $\pi \in R^T$ and $\pi_0$ be the dual variables associated with Constraints (5.3b) and (5.3c), respectively. The column for the single variable $s$ is explicitly priced if $s$ is nonbasic; else, $s$ is part of the current basis. Next, we price the $x_j$-variables. Note that the reduced cost for any variable $x_j$ is given by

$$RC_j = c_{0j}^d - \pi^T p_j^d - \pi_0(LF_j - \theta_{LF}).$$  \hspace{1cm} (5.4)

Let

$$y_t = \begin{cases} 
 1 & \text{if vehicle } t \text{ is part of the particular pattern } j \text{ to be generated} \\
 0 & \text{otherwise.} 
\end{cases}$$

Then, we have

$$RC_j = \pi_0 \theta_{LF} + c_{0j}^d - \sum_{t=1}^{T} \pi_t y_t - \pi_0 LF_j.$$  \hspace{1cm} (5.5)

We seek a desirable pattern to add to Problem SPP, that is, a pattern $p_j^d$ for which $RC_j < 0$. Having accomplished this, we would then incorporate the corresponding column, say $p_{j+1}^d$, and its associated variable, $x_{j+1}$, into Problem SPP, increment $J$ by one, and reiterate. If, however, no such column exists, we then solve the current model SPP to optimality as a 0-1 program, and terminate the entire procedure with the resulting (heuristic) solution. This overall algorithmic scheme is described below.
Algorithm A

Initialization. Generate an initial model SPP via Phase I. Let $J$ be the number of columns at hand.

**Step 1.** Solve SPP. Let $(\bar{\pi}, \bar{\pi}_0)$ be the dual variables obtained at optimality.

**Step 2.** Solve a subproblem, $\text{SP}(\bar{\pi}, \bar{\pi}_0)$, which generates a feasible pattern, $J+1$, having $RC_{J+1} < 0$, if possible. (This subproblem is formulated and discussed in detail in Section 5.3.3 below.) In case such a pattern is obtained, generate the corresponding column, $p^J_{J+1}$, increment $J$ by 1, and return to Step 1. Otherwise, proceed to Step 3.

**Step 3.** Solve the current model SPP as a 0-1 program (using a standard commercial package such as CPLEX) and terminate with the resulting heuristic solution. (Note that due to the problem complexity, we do not perform any further branch-and-price operations to determine an optimal solution.)

5.3.3 Solution to the Subproblem $\text{SP}(\bar{\pi}, \bar{\pi}_0)$

At Step 2 of Algorithm A, the principal task is to effectively minimize $RC_j$ given by (5.5), subject to the resulting pattern configuration being feasible with respect to the assembly matrix and conforming with the accompanying optimal routing scheme. Toward this end, we can solve the following pricing problem over the graph $G(N, A)$ introduced in Section 5.2, where some additional notation and decision variables required to define the formulated subproblem are specified below.

**Notation**

- $S_{\ell} \equiv \{j \in \{1, \ldots, T\} : \ell_j = \ell\}, \forall \ell = 1, \ldots, L$ (set of vehicles associated with location $\ell$).

- $h = 1, \ldots, H =$ possible loading configurations based on the assembly matrix.

- $q_{hr} =$ number of vehicles of type $r$ in configuration $h$.

- $\bar{q}_h =$ total number of vehicles in configuration $h$; $\bar{q}_h = \sum_{r=1}^{R} q_{hr}$.

- $\tau_{\text{max}} =$ upper bound on the possible delivery of any vehicle.
Decision Variables

- \( y_t = \begin{cases} 1 & \text{if vehicle } t \text{ is selected for delivery in the generated pattern,} \\ 0 & \text{otherwise.} \end{cases} \)

- \( \xi_t = \begin{cases} 1 & \text{if node } t \text{ is the last node visited in the generated pattern,} \\ 0 & \text{otherwise.} \end{cases} \)

- \( g_h = \begin{cases} 1 & \text{if configuration } h \text{ from the assembly matrix is selected for the generated pattern,} \\ 0 & \text{otherwise.} \end{cases} \)

- \( w_{ij} = \begin{cases} 1 & \text{if arc } (i, j) \text{ is traversed in the route corresponding to the generated pattern,} \\ 0 & \text{otherwise.} \end{cases} \)

- \( \tau_0 \in \{0, 1, \ldots, D\} = \text{day of departure for the generated pattern (beginning of day).} \)

- \( \tau_j = \text{time of visitation of node } j \) (\( \tau_j \equiv 0 \) if node \( j \) is not part of the generated pattern).

- \( \Delta_t = \text{days late (integer) for delivering vehicle } t, \text{ given that this is part of the generated pattern} \) (\( \Delta_t \equiv 0 \) otherwise). For accounting purposes, we will take \( \Delta_t = \max \{0, [\tau_t - (\beta_t + 0.9)]\} \) whenever \( y_t = 1 \). Hence, if \( \beta_t = 2 \), for example, then \( \Delta_t = 0 \) if \( \tau_t \leq 2.9 \), \( \Delta_t = 1 \) if \( 2.9 < \tau_t \leq 3.9 \), etc.

- \( \eta_t = \begin{cases} \tau_0 & \text{if } y_t = 1, \\ 0 & \text{otherwise.} \end{cases} \)

- \( \{u_j, \forall j \in S_\ell\} = \text{continuous variables to represent the relative order in which the visited nodes in } S_\ell \text{ are traversed (being zero for unvisited nodes), for the purpose of prohibiting subtours among nodes in } S_\ell, \forall \ell = 1, \ldots, L. \)

The initial prescribed model for generating patterns is then given as follows.

\[
\text{SP}(\overline{\pi}, \overline{\pi}_0): \text{Minimize } \overline{\pi}_0 \theta_{LF} + \left[ \sum_{h=1}^{H} c_{2h} g_h + \sum_t c_4 [\eta_t - \alpha_t y_t] + c_1 \sum_{(i,j)} d_{ij} w_{ij} + \sum_t c_3 \xi_t \right] - \sum_t \overline{\pi}_t y_t - \overline{\pi}_0 (LF) \\
\text{subject to } \sum_{j=1}^T w_{0j} = 1
\]  

(5.6a)
\[
\sum_{j \neq \{0, i\}} w_{ij} = y_i - \xi_i, \quad \forall i = 1, \ldots, T \tag{5.6c}
\]

\[
\sum_{j \neq i} w_{ji} = y_i, \quad \forall i = 1, \ldots, T \tag{5.6d}
\]

\[
\sum_{t=1}^{T} \xi_t = 1 \tag{5.6e}
\]

\[
\xi_t \leq y_t, \quad \forall t = 1, \ldots, T \tag{5.6f}
\]

\[
\sum_{h=1}^{H} g_h = 1 \tag{5.6g}
\]

\[
\sum_{t: r_t = r} y_t = \sum_{h} q_{hr} g_h, \quad \forall r = 1, \ldots, R \tag{5.6h}
\]

\[
\tau_j \geq \tau_i + p_{ij} - (1 - w_{ij})(p_{ij} + \tau_{\text{max}}) + w_{ji}(\tau_{\text{max}} - p_{ji}), \quad \forall i = 1, \ldots, T, \; j = 1, \ldots, T, \; j \neq i \tag{5.6i}
\]

\[
\tau_j \geq \tau_0 + p_{0j} - (1 - w_{0j})(p_{0j} + \tau_{\text{max}}), \quad \forall j = 1, \ldots, T \tag{5.6j}
\]

\[
0 \leq \tau_j \leq \tau_{\text{max}} y_j, \quad \forall j = 1, \ldots, T \tag{5.6k}
\]

\[
\tau_0 \geq \alpha_t y_t, \quad \forall t = 1, \ldots, T \tag{5.6l}
\]

\[
\tau_0 \leq D \tag{5.6m}
\]

\[
w_{ij} + w_{ji} \leq 1, \quad \forall i, j = 1, \ldots, T, \; i \neq j \tag{5.6n}
\]

\[
0 \leq u_j \leq (|S_\ell| - 1)y_j, \quad \forall j \in S_\ell, \forall \ell \in \{1, \ldots, L : |S_\ell| \geq 2\} \tag{5.6o}
\]

\[
u_j \geq u_i + 1 - (1 - w_{ij})|S_\ell| + w_{ji}(|S_\ell| - 2), \quad \forall i, j \in S_\ell, i \neq j, \forall \ell \in \{1, \ldots, L : |S_\ell| \geq 2\} \tag{5.6p}
\]

\[
\Delta_t \geq \tau_t - \beta_t - 0.9, \quad \forall t = 1, \ldots, T \tag{5.6q}
\]

\[
\Delta_t \geq 0, \quad \forall t = 1, \ldots, T \tag{5.6r}
\]

\[
\eta_t \geq Dy_t + \tau_0 - D, \quad \forall t = 1, \ldots, T \tag{5.6s}
\]

\[
\eta_t \geq 0, \quad \forall t = 1, \ldots, T \tag{5.6t}
\]

\[
LF = \sum_{h} \bar{q}_h g_h \tag{5.6u}
\]

\[(w, y, \xi, g) \text{ binary, } \Delta_t \text{ integer, } \forall t. \tag{5.6v}
\]

The objective function (5.6a) minimizes the reduced cost given by (5.5), where \(c^d_{0j}\) is expanded
in the term appearing within \([\cdot]\), as defined by (5.1). In this expression \([\cdot]\), the first term (noting (5.6g) and (5.6v)) is the total assembly and disassembly cost associated with the generated pattern, the second term is the inventory carrying cost for the vehicles included in this pattern, the third term is the accompanying mileage cost, the fourth term is the related driver return cost from the final location, and the fifth term is the delay penalty cost for the delivered vehicles in the generated pattern.

Constraint (5.6b) asserts that exactly one node (vehicle delivery to its corresponding location) should follow node 0. Equation (5.6c) ensures that if \(y_i = 1 \text{ and } \xi_i = 0\), then exactly one node \((\neq 0, i)\) follows node \(i\); else, if \(y_i = \xi_i = 0\) or \(y_i = \xi_i = 1\), then \(w_{ij} = 0, \forall j \neq 0, i\). Note that the case \(y_i = 0 \text{ and } \xi_i = 1\) does not arise since (5.6f) (or even (5.6v)) precludes this.

Constraint (5.6d) asserts that each node \(i\) should be approached from exactly one other node if \(y_i = 1\), and \(w_{ji} = 0, \forall j \neq i, y_i = 0\). Constraints (5.6e)-(5.6f) delineate exactly one node as the last node to be visited from among the selected ones (having \(y_t = 1\)), where (5.6c) assures that the node having \(\xi_i = 1\) (so \(y_i = 1\) from (5.6f)) will be the last node in the sequence. Constraint (5.6g) selects exactly one model configuration from the assembly matrix, and Constraint (5.6h) ensures that the total number of vehicles of each type delivered conforms with the selected configuration.

Constraints (5.6i)-(5.6n) account for the visitation times of all the nodes, including the start time \(\tau_0\) at node 0, and also serve as the lifted Miller-Tucker-Zemlin (MTZ) subtour elimination constraints (SECs), in addition to the two-city Dantzig-Fulkerson-Johnson (DFJ) SECs (5.6n) (see Sherali and Driscoll [116] for a general discussion on these restrictions). Observe that if \(w_{ij} = 1\), then \(w_{ji} = 0\) from (5.6n) and then (5.6i) requires that \(\tau_j \geq \tau_i + p_{ij}\). On the other hand, if \(w_{ij} = w_{ji} = 0\), then (5.6i) is redundant (implied by \(\tau_j \geq 0\)), and if \(w_{ji} = 1\), so that \(w_{ij} = 0\) by (5.6n), then (5.6i) asserts that we should have \(\tau_i \leq \tau_j + p_{ji}\), which is valid since the equality here is justified in this case. Constraint (5.6j) accounts for the visitation time of the first node \(j\) when \(w_{0j} = 1\), and is redundant when \(w_{0j} = 0\). Constraint (5.6k) sets \(\tau_j = 0, \forall j\) such that \(y_j = 0\), and (5.6l)-(5.6m) ensure that the shipment begins before day \(D\), but no earlier than the latest time of availability of the included vehicles in the generated pattern.

Observe that the foregoing lifted MTZ SECs exclude only the positive-length subtours. However, additional constraints are required in order to eliminate zero-length subtours for nodes that correspond to the same location. To obviate such subtours, lifted MTZ SECs (similar to Constraint
(5.6i)) are enforced in Constraints (5.6o)-(5.6p). To this end, while it is also possible to introduce tighter path-flow-based SECs as in Sherali et al. [120] in lieu of (5.6o)-(5.6p), we found the latter restrictions to suffice in our context. Some alternative strategies to eliminate zero-length subtours are presented below following the model description.

Constraints (5.6q)-(5.6r) (with (5.6v)) enforce the definition of $\Delta_t \equiv \max \{0, \lceil \tau_t - \beta_t - 0.9 \rceil \}, \forall t$, while (5.6s)-(5.6t), in concert with $c_4 > 0$ in the objective function, enforce the definition of $\eta_t = \tau_0$ if $y_t = 1$, and $\eta_t = 0$ if $y_t = 0, \forall t$. Equation (5.6u) enforces the definition of $LF$ as $\bar{q}_h$ for the selected configuration ($g_h = 1$), and (5.6v) enforces the required logical binary and integrality restrictions.

Example 5.1. Figure 5.2 depicts an illustrative example of an assembly composed within Problem SP along with its associated optimal routing decisions. The vehicles selected for the assembly composition must be delivered to their corresponding destination nodes in the prescribed order (in this example, nodes 3, 2, and 5). All selected delivery nodes have exactly one incoming and one outgoing directed arc, except the last node in the delivery sequence (here, node 5).

It is worth noting that for $s_{\text{max}} \leq 4$ zero-length subtours can be eliminated by imposing Constraint (5.6w) stated below in lieu of Constraints (5.6o)-(5.6p) as established by Proposition 5.1.

$$ \sum_{t=1}^{T} y_t \leq (\xi_j + w_{0j})(1 - T) + 2T - 1, \quad \forall j = 1, \ldots, T. \quad (5.6w) $$
Constraint (5.6w) is valid because it asserts that if $\xi_j = w_{0j} = 1$, then $\sum_{t=1}^{T} y_t \leq 1$, which, in conjunction with (5.6v), implies that a node can simultaneously be the first and the last one to be visited only if the generated pattern involves a single vehicle. If $\xi_j + w_{0j} \leq 1$, then (5.6w) is redundant.

**Proposition 5.1.** For $s_{\text{max}} \leq 4$, the delivery routes generated via SP with (6w) replacing (6o)-(6p) cannot include zero-length subtours.

**Proof.** On the contrary, suppose that some zero-length subtour is produced by the stated subproblem and let $V$ be the number of nodes or vehicles in this subtour. Without loss of generality, assume that $s_{\text{max}} = 4$. The cases $V = 2$ and $V = 4$ are impossible as they violate (5.6n) and (5.6b)-(5.6h), respectively. Hence, suppose that $V = 3$, and let $n_1, n_2, n_3, n_4$ be the nodes that represent the four vehicles to be delivered (with $y_{n_i} = 1$, $i = 1, 2, 3, 4$), such that $n_2, n_3, n_4$ belong to the zero-length subtour. Consequently, $w_{0n_1} = 1$ by (6b) and (6d), and since the subtour has a cost of zero, we have $p_{ij} = 0, \forall i, j \in \{2, 3, 4\}$. Note that it must be that $\xi_{n_1} = 1$, because if not, then $\xi_{n_i} = 1$ for some node $n_i, i \in \{2, 3, 4\}$, in the subtour, and therefore, $n_i$ violates (5.6c). Hence, $\xi_{n_1} = 1$ and $w_{0n_1} = 1$, which contradicts Constraint (5.6w).

While clearly more compact, (5.6w) might compromise on the relative tightness of the underlying relaxation in comparison with (5.6o)-(5.6p). However, our computational experience reported in Section 5.5 exhibits an average overall savings in computational effort of 15.33% by using (6w) in lieu of (5.6o)-(5.6p). Furthermore, note that an alternative general and realistic way to preclude zero-length subtours (for any value of $s_{\text{max}}$) is to assume that $p_{ij} = \varepsilon, \forall i, j = 1, \ldots, T, \ell_i = \ell_j, i \neq j$, where $\varepsilon > 0$ is an estimate of the time required to disassemble and drop-off a vehicle at any location. In this case, all subtours are directly eliminated by Constraints (5.6i)-(5.6n), and Constraints (5.6o)-(5.6p) become unnecessary.

**Remark 5.1.** Classical perturbation-based (theoretical or practical) cycling prevention rules (see Bazaraa et al. [12]) can be implemented within our optimization framework. It may also be useful to store a collection of some $N_{\text{max}}$ most recently basic columns that are currently non-basic, pricing them explicitly for re-entry at each Step 1-Step 2 iteration of Algorithm A, and also, finally incorporating them within the SPP solution process in Step 3 of Algorithm A. Alternatively, we can simply store all the columns that have been generated to solve SPP to optimality, and subsequently, include these within SPP in Step 3 of Algorithm A (see our computational experience in
Section 5.5). The column generation strategy that is utilized within Problem SP exploits one of the following three strategies:

(a) **Strategy 1:** We solve Problem SP once at each iteration without any additional restrictions.

(b) **Strategy 2:** While solving the subproblem during each iteration, we impose the additional requirement that the total number of vehicles selected to construct a pattern is bounded above by a parameter $s_{lim}$, where $s_{lim}$ is initialized at 2; when no pattern prices out favorably with the current $s_{lim}$ value, the latter is incremented by one until no pattern prices out favorably with $s_{lim} = s_{max}$.

(c) **Strategy 3:** We generate multiple columns within every Step 1-Step 2 iteration of Algorithm A by solving Problem SP iteratively while requiring the number of vehicles selected for assembly to equal 2, ..., $s_{max}$, in turn (see Section 5.5.1).

### 5.4 Enhanced Models

In this section, we propose enhanced models that consider several specific extensions to the basic VARP model presented in Section 5.3 in order to accommodate the need for assembly sequence-dependent delivery routes and/or driver- and equipment-based restrictions.

#### 5.4.1 Assembly Sequence-dependent Routing

The model described in Section 5.3 tentatively ignores the fact that the selected configuration might partially restrict the sequence in which the nodes are visited. Note that in the truck shipment business (as in our case-study), each specified pattern not only conforms with a valid configuration from the assembly matrix, but also, it accordingly governs the set of admissible sequences for making the deliveries. In such applications, starting from the source node, the next location along a feasible route must involve dropping off one or several trucks from either (or both) ends of the current vehicle combination. In other words, the equipment used for physically constructing the initial configuration restricts the sequence in which trucks can be transported along the route. Imposing that only lead- and/or rear-trucks can be separated from the sequence of trucks at any visited dealership location ensures that we will not require a re-connection operation during the shipment process for which the necessary equipment might not be available. Thus, operational activities at any delivery location are limited to unloading the appropriate trucks. More importantly, this constraint guarantees that, after each delivery in the routing sequence, the reduced combination of trucks remains technologically valid, i.e., compatible with respect to the assembly matrix. Throughout this section, we assume that $s_{max} = 4$ as in the relevant trucking industry case-study, so that
the column generation subproblem (5.6) refers to the model SP where (5.6w) is enforced in lieu of (5.6o)-(5.6p) by virtue of Proposition 5.1 (and in the light of the results reported in Table 5.1 of Section 5.5).

To accommodate the consideration of such assembly sequence-dependent routing, we run the column generation routine embodied by Problem (5.6) for different fixed values of \( \bar{q}_h \). Note that when \( \bar{q}_h \) equals 1 or 2, no sequencing issue arises. Hence, we can first run Problem (5.6) with configurations restricted (and hence, the number of visited locations restricted) to \( \bar{q}_h \in \{1, 2\} \). Next, consider the case of \( \bar{q}_h = 3 \), and observe that we only need to ensure that the first location visited conforms with the lead or rear vehicle. Hence, for each vehicle type \( r \), let

\[
E_r = \{ h : \text{configuration } h \text{ permits vehicle type } r \text{ to be positioned as the lead- or rear-vehicle} \}, \quad \forall r = 1, ..., R. \tag{5.7a}
\]

In this context, we need to ensure in the model that if \( w_{0j} = 1 \) for any node, we have that the selected configuration belongs to \( E_{rj} \). This can be accomplished by incorporating the following set of constraints within Problem (5.6), under the restriction \( \bar{q}_h = 3 \).

\[
\sum_{h \in E_{rj}} g_h \geq w_{0j}, \quad \forall j = 1, ..., T. \tag{5.7b}
\]

Likewise, consider the case of \( \bar{q}_h = 4 \). In this case, we need to restrict configurations according to the first two locations visited. Accordingly, let us define

\[
E_{rr'} = \{ h : \text{configuration } h \text{ permits vehicle type } r \text{ to be delivered first and then vehicle type } r' \text{ second} \}, \quad \forall r, r' = 1, ..., R. \tag{5.8a}
\]

We then impose the following restrictions within Model (5.6), which effectively enforce in concert with (5.6g) that \( \sum_{h \in E_{rr'_{j_k}}} g_h = 1 \) whenever \( w_{0j} = w_{jk} = 1 \), and are redundant otherwise:

\[
\sum_{h \in E_{rr'_{jk}}} g_h \geq w_{0j} + w_{jk} - 1, \quad \forall j, k = 1, ..., T, \ j \neq k. \tag{5.8b}
\]
Note that for this case, we could also include constraints (5.7b) in order to tighten the model representation. Furthermore, observe that when vehicles (nodes) $j$ and $k$ correspond to the same type, we have that $E_{r_{j}r_{k}} \equiv E_{r_{k}r_{j}}$. In addition, in order to avoid generating an excessive number of constraints of the type (5.8b), we could adopt a relaxation strategy in which not all constraints of the type (5.8b) are incorporated a priori. Rather, relaxed constraints are resurrected as necessary in a sequential solution process.

**Remark 5.2.** In order to contain the solution effort, it would be worthwhile to examine the spatial distribution of delivery locations and, hence, determine an aggregation of these locations into consolidated “super-locations” for the purpose of solving the overall distribution problem. The clusters can be generated using the method of Sherali and Desai [115], for example, or be determined via an analysis of the particular geographical scatter-plots. Alternatively, the set partitioning structure can be subdivided into smaller non-overlapping subsets (based on a geographical clustering), each of which could then be optimized independently. This strategy is illustrated in Section 5.5.2.

**Remark 5.3.** The overall strategy can adopt a rolling-horizon framework on a daily basis, in that only the imminent day’s decisions prescribed by the model need be implemented. The shipments that are not as yet committed can be accommodated within the following day’s run (using advance-start pattern columns), and could be revised in light of the most recent available data information. Section 5.5.5 provides some computational evidence on the benefits of the look-ahead feature in such a rolling-horizon framework.

### 5.4.2 Driver- and Equipment-based Restrictions

Thus far, we have assumed an unlimited supply of drivers as well as that of the assembly equipment required to physically construct the different vehicle configurations or patterns. As such, the foregoing model could be used to strategically ascertain desirable levels of such resources. On the other hand, whenever these resource limitations need to be explicitly considered, we present in this section enhancements to the foregoing modeling and algorithmic strategies to accomplish this.

First, let us consider limitations imposed by drivers. Note that an integration of the foregoing model with a crew scheduling model that considers individual driver assignments along with constraints that govern the number of consecutive work-hours and on-off-days is an arduous endeavor that is beyond the scope of the present effort. Instead, we assume that for each run of the model, we are given data pertaining to some $k = 1, \ldots, K$ categories of drivers, along with a partitioning of
the geographical map into \(k = 1, ..., K\) corresponding regions (possibly overlapping) that depend on the distance from the source, such that any pattern can involve visiting a subset of locations within a particular region alone, and such that a driver in category \(k\) can only serve a pattern associated with region \(k\), \(\forall k = 1, ..., K\). For example, we might have \(K = 3\), with \(k = 1, 2, 3\) respectively representing short-, medium-, and long-haul trips. Accordingly, define the following, for each \(k = 1, ..., K\).

- \(\delta_k\) = number of available drivers of category \(k\).
- \(J_k\) = subset of patterns \(\{1, ..., J\}\) that pertain to region \(k\).

Then, in Model SPP given by (5.3), we add the following set of constraints:

\[
\sum_{j \in J_k} x_j \leq \delta_k, \quad \forall k = 1, ..., K.
\] (5.9)

Next, let us address the equipment-based restrictions. In order to compose different vehicle configurations, we need to use specific equipment that depend not only on the models of the trucks being connected, but on their particular specifications as well. (For example, the type of fifth wheel on a truck governs the type of saddle used in connecting this truck to another admissible truck in our truck shipment case-study.) Hence, let \(e = 1, ..., E\) index the different types of equipment, and define the following parameters and equipment consumption entities.

- \(\lambda_e\) = available units of equipment type \(e\) at the beginning of the model run, \(e = 1, ..., E\).
- \(\lambda = [\lambda_1, ..., \lambda_E]^T\).
- \(P_j\) = vector having components \(P_{je}, e = 1, ..., E\), where \(P_{je}\) is the number of units of equipment type \(e\) that is required for composing pattern \(j\), \(\forall j = 1, ..., J\). (Note that each generated pattern \(j\) inherently entails a particular configuration sequence of the specific trucks contained in this pattern.)
- \(\psi_{et_1t_2}\) = number of units of equipment \(e\) used to connect trucks \(t_1\) and \(t_2\), \(\forall e = 1, ..., E, t_1, t_2 = 1, ..., T, t_1 \neq t_2\).

Incorporating both the driver and equipment considerations, the enhanced revision of the original Model SPP given by (5.3), denoted ESPP, can be stated as follows.
ESPP: Minimize \[ \sum_{j=1}^{J} c_{0j}^d x_j + \mu s \] (5.10a)

subject to \[ \sum_{j=1}^{J} \left[ p_j^d \right] x_j = \left[ 1 \right] \] (5.10b)
\[ \sum_{j=1}^{J} P_j x_j \leq \lambda \] (5.10c)
\[ \sum_{j \in J_k} x_j \leq \delta_k, \quad \forall k = 1, \ldots, K \] (5.10d)
\[ \sum_{j=1}^{J} (LF_j - \theta_{LF}) x_j + s \geq 0 \] (5.10e)
\[ x \text{ binary}, s \geq 0. \] (5.10f)

We can identically employ Algorithm A as in Section 5.3.2 to solve Problem ESPP, where the pricing subproblem \( \text{SP}(\cdot) \) for generating columns is revised as follows. Let \( \pi_1, -\pi^2, -\pi^3, \pi_0 \) be the dual variables associated with (5.10b), (5.10c), (5.10d), and (5.10e), respectively, in the LP solution to ESPP at any stage in the process, where \( (\pi^2, \pi^3, \pi_0) \geq 0 \). Because of the nature of Constraint (5.10d), given any dual solution \( (\pi_1, \pi_2, \pi_3, \pi_0) = (\bar{\pi}_1, \bar{\pi}_2, \bar{\pi}_3, \bar{\pi}_0) \) to the LP relaxation of the current master program, we would need to solve a separate subproblem, \( \text{SP}_k(\bar{\pi}_1, \bar{\pi}_2, \bar{\pi}_3, \bar{\pi}_0) \), for each \( k = 1, \ldots, K \) that pertains to category/region \( k \). Similar to the derivation of (5.6a), the minimization of the reduced cost would lead to the following objective function, in lieu of (5.6a), where \( T_k \) is the subset of vehicles \( \{1, \ldots, T\} \) that pertain to locations within region \( k \).

\[ \text{SP}_k(\bar{\pi}_1, \bar{\pi}_2, \bar{\pi}_3, \bar{\pi}_0): \text{Minimize} \left[ \sum_{h=1}^{H} c_{2h} g_h + \sum_{t \in T_k} c_4 \left[ \eta_t - \alpha_t y_t \right] + c_1 \sum_{(i,j)} d_{ij} w_{ij} + \sum_{t \in T_k} c_3 \xi_t + \sum_{t \in T_k} \Delta_t c_5 \xi_t \right] + \bar{\pi}_3 + \bar{\pi}_0 \theta_{LF} - \sum_{t \in T_k} \bar{\pi}_1^t y_t - \bar{\pi}_0 (LF) + \sum_{e=1}^{E} \sum_{t_1 \in T_k} \sum_{t_2 \in T_k, t_2 \neq t_1} \pi_e^2 \psi_{e t_1 t_2} w_{t_1 t_2}. \] (5.11)

The constraints for this subproblem are identical to those in Problem (5.6), except that we restrict the vehicles to \( t \in T_k \) (in lieu of considering all the vehicles \( t = 1, \ldots, T \)), and, when using assembly sequence-dependent routing restrictions, we also accommodate the appropriate constraints from (5.8b).
Remark 5.4. Regarding the last term in (5.11), which pertains to the equipment usage, in case the consumption of equipment depends solely on the configuration, we can define a parameter $\psi_{eh}$ to denote the number of units of equipment $e$ consumed by configuration $h$, $\forall e = 1, ..., E$, $h = 1, ..., H$, and replace this objective term by the following simpler term:

$$\sum_{e=1}^{E} \sum_{h=1}^{H} \bar{\pi}^2 \psi_{eh} g_h.$$  \hspace{1cm} (5.12a)

Likewise, if we can partition the set of equipments $\{1, ..., E\}$ into subsets $E_1$ and $E_2$ such that the consumption of $e \in E_1$ depends on the particular vehicle $t$ and that of $e \in E_2$ depends on the particular configuration $h$, with respective consumption values $\psi_{1,eh}$ and $\psi_{2,eh}$, then we could model this by replacing the last term in (5.11) by the following:

$$\sum_{e \in E_1} \sum_{t \in T_k} \bar{\pi}^2 \psi_{1,et} y_t + \sum_{e \in E_2} \sum_{h=1}^{H} \bar{\pi}^2 \psi_{2,eh} g_h.$$  \hspace{1cm} (5.12b)

In the next section, we will experiment with both the basic and enhanced modeling constructs discussed respectively in Sections 3 and 4.

5.5 Computational Experience

In this section, we provide some computational experience using realistic, but generic and non-proprietary, data pertaining to our case-study involving a major truck manufacturing company operating in the USA. The basic formulation of Section 5.3 as well as the two enhanced models described in Sections 5.4.1 and 5.4.2 are investigated under various managerial requirements or operational scenarios. The truck configurations are restricted here to a maximum size of $s_{max} = 4$ as for our aforementioned application. Below are summarized the key characteristics of the problem instances studied.

- Instance 1: $(T, R, L, H, D) = (10, 4, 7, 11, 2)$.
- Instance 2: $(T, R, L, H, D) = (15, 4, 7, 11, 2)$.
- Instance 3: $(T, R, L, H, D) = (20, 4, 7, 12, 2)$.
- Instance 4: $(T, R, L, H, D) = (50, 4, 50, 12, 2)$.
- Instance 5: $(T, R, L, H, D) = (100, 4, 50, 12, 2)$.
The distribution network is composed of a single source node located in VA and 50 delivery destinations selected from 14 U.S. states: VA, NC, KY, WV, MD, TN, OH, NJ, PA, DE, NY, CT, SC, GA (see Figure 5.3). Actual geographical distances and estimated travel times between the 50 selected cities have been used in our computations. The data parameters pertaining to each vehicle (ready-date, due-date, model type, etc.) have been randomly generated within practical ranges. We have used AMPL in concert with CPLEX 9 for implementation purposes, and all runs have been performed on an HP workstation xw8200 having a Xeon(TM) CPU 2.40 GHz processor and 1.50 GB of RAM. Also, in practice, note that the proposed algorithm could be initialized based on a heuristic solution or using intuitively appealing or experience-driven columns that are recommended by the decision-maker. However, for illustrative purposes, all runs below are initialized using the simple single-truck patterns. To summarize the solutions generated via Algorithm A, we use the quadruplet \( (v_1, v_2, v_3, v_4) \), where \( v_i \) represents the number of patterns generated that are comprised of \( i \) trucks. Therefore, the total number of patterns in the final solution \( x^* \) equals \( \sum_{i=1}^{4} v_i = \sum_{j} x_j^* \), and \( \sum_{i=1}^{4} iv_i \) corresponds to the total number of trucks in the instance under investigation.

5.5.1 Basic VARP Model

Tables 5.1-5.3 present the results for solving the joint assembly-routing problem VARP using our proposed approach for the test problem instances 1-5, and with \( \theta_{LF} = 0 \) and 3, respectively. The results exhibit that Algorithm A produces tight lower and upper bounding solutions, achieving a 0% optimality gap for several test instances. Considering \( \theta_{LF} = 0 \), Instances 1, 2, 4, and 5 were readily
solved to optimality, whereas an optimality gap within 1.3% was obtained for Instance 3. Similarly, for $\theta_{LF} = 3$, optimal solutions (for three cases) or near-optimal solutions (within a 0.1-2.3% optimality gap) were generated using Algorithm A. This computational experience suggests that a basic implementation of our methodology is efficient for problem instances that involve up to 50 trucks. For larger problem instances, such as Instance 5 (100 vehicles), the computational effort becomes prohibitive for the basic model introduced in Section 5.3, and clustering-based enhancements can be advantageously exploited to obtain near-optimal solutions in manageable computational times (see Section 5.5.2).

The computational effort involved in Algorithm A is entirely concentrated in the Step 1-Step 2 iterations, whereas the final restricted SSP (denoted R-SPP) is instantaneously solved to optimality all throughout our computational experience using CPLEX 9. Also, the computational times pertaining to the Step 1-Step 2 iterations are reported for the three proposed column generation strategies (see Remark 5.1), and it is worth noting that the master programs were trivially solved (within less than 0.1 sec) and the bulk of the computational effort was consumed while solving Problem SP. For the relatively smaller-sized instances 1-3, there was no significant difference between these three strategies. However, for Instances 4 and 5 and using $\theta_{LF} = 0$, Strategy 3 achieved a savings in computational effort of 26.62% and 29.26% over Strategy 1 and Strategy 2, respectively. The total CPU times reported in Table 5.1 and in the sequel are based on using Constraint (5.6w) in lieu of (5.6o)-(5.6p) and, in fact, our preliminary computational experience suggested that enforcing Constraint (5.6w) achieved an average savings in computational effort of 15.33% (23.73% for Instances 1-3 and 2.74% for Instances 4 and 5).

The results obtained for the basic model with $\theta_{LF} = 3$ are reported in Table 5.2. The costs of the starting and final solutions are decomposed into the operational cost and the penalty $\mu s$ that is incurred if Constraint (5.3c) is violated. To render such violations undesirable, a large penalty coefficient $\mu = 3000$ has been selected, noting that the operational cost associated with typical patterns empirically ranges between 200 and 2500 for the tested problem instances. Again, the results presented in Table 5.2 indicate that for Instance 4, Strategy 3 achieves savings in computational effort of 55.04% and 44.91% over Strategy 1 and Strategy 2, respectively, thereby significantly outperforming the latter.

Table 5.3 summarizes the structure of the initial and final solutions for $\theta_{LF} = 0$ and 3, and
### Table 5.1: Basic model, $\theta_{LF} = 0$

<table>
<thead>
<tr>
<th>Problem</th>
<th>Initial cost</th>
<th>Objective</th>
<th>Total CPU time (s)</th>
<th>Objective</th>
<th>CPU</th>
<th>% Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance 1</td>
<td>5050</td>
<td>4015</td>
<td>7.1</td>
<td>4190</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Instance 2</td>
<td>7630</td>
<td>4945</td>
<td>16.0</td>
<td>5155</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Instance 3</td>
<td>10210</td>
<td>5678.33</td>
<td>48.9</td>
<td>5748.33</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Instance 4</td>
<td>23372</td>
<td>12444</td>
<td>12444</td>
<td>12444</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Instance 5</td>
<td>46744</td>
<td>18403</td>
<td>191652.0</td>
<td>18403</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### Table 5.2: Basic model, $\theta_{LF} = 3$

<table>
<thead>
<tr>
<th>Problem</th>
<th>Initial cost</th>
<th>Objective</th>
<th>Total CPU time (s)</th>
<th>Objective</th>
<th>CPU</th>
<th>% Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance 1</td>
<td>5050</td>
<td>60000</td>
<td>4190</td>
<td>7</td>
<td>1.42</td>
<td>1.52</td>
</tr>
<tr>
<td>Instance 2</td>
<td>7630</td>
<td>90000</td>
<td>5155</td>
<td>6</td>
<td>2.5</td>
<td>1.88</td>
</tr>
<tr>
<td>Instance 3</td>
<td>10210</td>
<td>120000</td>
<td>5748.33</td>
<td>5</td>
<td>3.33</td>
<td>2.46</td>
</tr>
<tr>
<td>Instance 4</td>
<td>23372</td>
<td>300000</td>
<td>12446.5</td>
<td>6</td>
<td>3.33</td>
<td>2.46</td>
</tr>
<tr>
<td>Instance 5</td>
<td>46744</td>
<td>600000</td>
<td>18403</td>
<td>6</td>
<td>3.125</td>
<td>2.71</td>
</tr>
</tbody>
</table>

### Table 5.3: Effect of LF restrictions on the basic model

<table>
<thead>
<tr>
<th>Problem</th>
<th>$\theta_{LF}$</th>
<th>Starting solution</th>
<th>Final solution</th>
<th>Penalty</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance 1</td>
<td>0</td>
<td>(10, 0, 0, 0)</td>
<td>(11, 0, 13)</td>
<td>4015</td>
<td></td>
</tr>
<tr>
<td>Instance 2</td>
<td>0</td>
<td>(15, 0, 0, 0)</td>
<td>(17, 24, 32)</td>
<td>4945</td>
<td></td>
</tr>
<tr>
<td>Instance 3</td>
<td>0</td>
<td>(20, 0, 0, 0)</td>
<td>(28, 50, 29)</td>
<td>5745</td>
<td></td>
</tr>
<tr>
<td>Instance 4</td>
<td>0</td>
<td>(50, 0, 0, 0)</td>
<td>(101, 181, 125)</td>
<td>12444</td>
<td></td>
</tr>
<tr>
<td>Instance 5</td>
<td>0</td>
<td>(100, 0, 0, 0)</td>
<td>(234, 357, 320)</td>
<td>18403</td>
<td></td>
</tr>
</tbody>
</table>

---

83
provides insights into the effect of enforcing restrictions pertaining to the LF-factors. As larger patterns or delivery sequences empirically tend to have a load-factor close to or greater than 3, the final solution generated for $\theta_{LF} = 3$ typically partitions the original set of trucks into fewer patterns, having a larger load of trucks in average. Hence, single-truck patterns are less likely to be generated in the final solution under more significant restrictions (with high penalties) pertaining to LF considerations. Furthermore, as the total number of trucks to be delivered increases, the least cost solution with $\theta_{LF} = 0$ itself tends to have a relatively higher average load-factor and average weighted load-factor (see Instances 4 and 5 in Table 5.3). In addition, Table 5.3 reports the number of patterns $J - T$ that have been generated during the Step 1-Step 2 iterations (i.e., the $T$ patterns that are introduced in the initial step are not counted), for each of the three strategies proposed in Remark 5.1 (which are abbreviated as S1, S2, and S3). This suggests that for the basic VARP model with Strategies 1 and 2, load-factor considerations introduce an additional burden of balancing the operational cost and the LF-induced penalty, thereby causing a slower algorithmic convergence that is reflected by an increase in the number of columns generated, $J - T$, as well as in the total CPU times (see Tables 5.1 and 5.2). In this context, the column generation algorithm attempts to eliminate the LF-induced penalty during the initial Step 1-Step 2 iterations and, in fact, eventually succeeds in producing optimal or near-optimal solutions that comply with the load-factor requirement. Observe that for Strategy 2, the progressive increase in the upper bound imposed on the number of vehicles selected within any pattern under composition causes a slower elimination of the LF-induced penalty. Also, note that Strategy 3 is computationally robust with respect to load-factor considerations. In fact, whereas Strategy 2 attempts to guide the computational effort in the subproblem by favoring smaller loads at early stages of the algorithm, Strategy 3 consistently tends to introduce a balanced mix of patterns of sizes 2, 3, and 4, at every Step 1-Step 2 iteration, thereby yielding a faster algorithmic convergence. The last column of Table 5.3 presents the objective values for $\theta_{LF} = 0$ and 3, and provides within parentheses the ratio of the latter over the former.

As far as the proportion of patterns selected in the final solution is concerned, this represents 18.75% and 9.32% of the total number of patterns generated at an average for the basic VARP model using Strategy 3 with $\theta_{LF} = 0$ and 3, respectively. As noted above, for $\theta_{LF} = 3$, patterns characterized by higher loads tend to be preferred, thereby inducing fewer patterns in the final solution, and also, the additional burden of balancing the operational cost and the LF-induced penalty yields an overall greater number of generated patterns.
Figure 5.4a displays the decrease in the objective value as a function of the number of Step 1-Step 2 iterations of Algorithm A for Strategy 1. The rate of improvement decreases with the iterations, and the objective value stalls over several iteration subsequences due to degeneracy. It is also worth noting that the computational effort associated with solving SP significantly grows as the iterations progress as evident from Figure 5.4b, that is, it becomes increasingly computationally challenging to generate patterns that induce possible marginal improvements in the master program. Therefore, for large-scale test problems such as Instance 5, the total computational effort (see the last three columns in Table 5.1) is predominantly predicated on the drastic increase in the computational effort associated with Problem SP as the LP iterations progress, irrespective of the specific constraints enforced to defeat zero-length subtours, and notwithstanding the computational benefits that result from generating multiple columns at every LP phase iteration (as in Strategy 3).

Finally, it is interesting to note that the LP solution obtained from the Step 1-Step 2 iterations of Algorithm A often satisfies the required binary restrictions, and hence yields an optimal MIP solution, rendering this optimization framework well-suited for this class of problems. Given that the present application has $s_{max} = 4$, the patterns generated are relatively sparse and do not significantly overlap, thereby inhibiting the tendency for the continuous solution to SPP to fractionate. However, in general, in order to solve VARP to optimality, one would need to explore branch-and-price strategies [10]. We relegate this, along with other more intricate issues as identified in
Section 5.6 for future research.

5.5.2 Elementary Clustering Strategies

Observe that for large-scale problem instances, the feasible patterns, and hence, the set partitioning structure embedded in the master problem, are highly sparse. To exploit this sparsity in controlling computational effort, we can consider geographical clusters or balanced sub-partitions to be optimized in parallel. In fact, the set partitioning structure of the VARP model naturally lends itself to clustering and permits the subdivision of the original set of vehicles into smaller non-overlapping subsets, which can be handled in parallel to synthesize a complete solution to the original problem.

To illustrate this concept, we applied the following clustering strategy to Instance 5 for the case of $\theta_{LF} = 0$. The states involved in the delivery effort are clustered into three geographical areas: Cluster 1 includes TN, NC, SC, and GA; Cluster 2 is composed of VA, WV, OH, KY, MD, and DE, and Cluster 3 includes PA, CT, NJ, and NY (see Figure 5.3). Table 5.4 summarizes the results obtained, where the CPU time reported in the final column corresponds to the maximum CPU time consumed over the three specified cluster-based problems, since these are solved in parallel on different equivalent machines. Table 5.5 compares the performance of the suggested geographical clustering strategy against the results reported in Table 5.1, whereby the relatively more restricted clustered problem sacrifices 2.92% of optimality while saving 97.44% of the required computational effort in comparison with the unrestricted approach without any clustering. This provides supporting evidence for the potential of employing carefully designed clustering procedures.

<table>
<thead>
<tr>
<th>Instance 5</th>
<th># of vehicles</th>
<th>Initial objective</th>
<th>MIP objective</th>
<th>CPU time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster 1</td>
<td>32</td>
<td>11266</td>
<td>4614</td>
<td>339.6</td>
</tr>
<tr>
<td>Cluster 2</td>
<td>28</td>
<td>11170</td>
<td>4386</td>
<td>182.7</td>
</tr>
<tr>
<td>Cluster 3</td>
<td>40</td>
<td>24308</td>
<td>9957</td>
<td>4897.5</td>
</tr>
<tr>
<td>Complete Solution</td>
<td>100</td>
<td>46744</td>
<td>18957</td>
<td>4897.5</td>
</tr>
</tbody>
</table>

Table 5.4: Clustering strategy, basic model, $\theta_{LF} = 0$

<table>
<thead>
<tr>
<th>Instance 5</th>
<th>Objective value</th>
<th>Optimality gap %</th>
<th>CPU time (s)</th>
<th>CPU time reduction %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without Clustering</td>
<td>18403</td>
<td>191652</td>
<td>191652.0</td>
<td></td>
</tr>
<tr>
<td>With Clustering</td>
<td>18957</td>
<td>2.92</td>
<td>4897.5</td>
<td>97.44</td>
</tr>
</tbody>
</table>

Table 5.5: Clustering benefits, basic model, $\theta_{LF} = 0$
5.5.3 Assembly Sequence-dependent Delivery Routes

Tables 5.6 and 5.7 present some computational tests pertaining to the enhanced model discussed in Section 5.4.1. Table 5.8 summarizes the effect of integrating the $LF$ considerations into the generation of assembly sequence-dependent delivery routes. The results reported in Tables 5.6-5.8 indicate that incorporating $LF$-requirements results in a cost increase of 11.39% as well as an increase in the number of patterns generated (of 22.7% and 28.3% at an average), which induces an increase of 18.9%, 5.5% in the average computational effort for Strategies 1 and 2, respectively. In contrast, Strategy 3 exhibits an average increase of 18.3% in the number of patterns generated due to $LF$-requirements, while being computationally robust with respect to this parameter. The effect of the $LF$ requirement is revealed by comparing Tables 5.1 and 5.2, and Tables 5.6 and 5.7, whereas the impact of the sequence-dependent routing considerations is evident by comparing Tables 5.1 and 5.6, and Tables 5.2 and 5.7. These comparisons are summarized in Table 5.9 from an objective value point of view, where $\nu$ represents the total operational cost under the specified $LF$ and assembly sequence-dependent delivery route assumptions. Note that the three column generation strategies can possibly produce different near-optimal solutions as reported in Table 5.7 for Instances 3 and 4. In such cases, we have reported the relevant objective values along with their corresponding optimality gaps. In addition, for $\theta_{LF} = 0$, Strategy 3 achieved an average savings in computational effort of 54.05% and 52.71% over Strategies 1 and 2, respectively, for Instances 4 and 5. Also, for Instances 4 and 5 and using $\theta_{LF} = 3$, Strategy 3 yields an average savings in computational effort of 54.11% and 52.71% over Strategies 1 and 2, respectively, thereby confirming the superiority of Strategy 3 over the two alternative algorithmic strategies that we have investigated.

The results reported in Table 5.3 indicate that it is possible to satisfy the $LF$ requirement ($\theta_{LF} = 3$) for the instances investigated without incurring any penalty in the final solution. It is therefore anticipated, and verified in Table 5.8, that the incorporation of assembly sequence-dependent delivery routes in conjunction with the $LF$-requirement would entail a marginal cost increase while avoiding the very costly $LF$-induced penalty. Naturally, the additional consideration pertaining to attaining at least some specified $LF$ value comes at a (computational and operational) price, which decision-makers need to examine against its benefits with respect to trips, driver preferences and convenience, resource utilization, and costs.
### Table 5.6: Assembly sequence-dependent routing, $\theta_{LF} = 0$

<table>
<thead>
<tr>
<th>Problem</th>
<th>Initial cost</th>
<th>Objective Total CPU time (s)</th>
<th>R-SPP optimum (Step 3)</th>
<th>% Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Strategy 1</td>
<td>Strategy 2</td>
<td>Strategy 3</td>
</tr>
<tr>
<td>Instance 1</td>
<td>5050</td>
<td>4015</td>
<td>4.3</td>
<td>3.5</td>
</tr>
<tr>
<td>Instance 2</td>
<td>7630</td>
<td>5495</td>
<td>37.0</td>
<td>36.2</td>
</tr>
<tr>
<td>Instance 3</td>
<td>10210</td>
<td>6617.5</td>
<td>119.6</td>
<td>90.7</td>
</tr>
<tr>
<td>Instance 4</td>
<td>23372</td>
<td>13101.5</td>
<td>119.6</td>
<td>90.7</td>
</tr>
<tr>
<td>Instance 5</td>
<td>46744</td>
<td>18436</td>
<td>22713.1</td>
<td>253160.1</td>
</tr>
</tbody>
</table>

### Table 5.7: Assembly sequence-dependent routing, $\theta_{LF} = 3$

<table>
<thead>
<tr>
<th>Problem</th>
<th>Initial cost</th>
<th>Objective Total CPU time (s)</th>
<th>R-SPP optimum (Step 3)</th>
<th>% Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Strategy 1</td>
<td>Strategy 2</td>
<td>Strategy 3</td>
</tr>
<tr>
<td>Instance 1</td>
<td>5050</td>
<td>60000</td>
<td>4618.33</td>
<td>12.3</td>
</tr>
<tr>
<td>Instance 2</td>
<td>7630</td>
<td>90000</td>
<td>6510</td>
<td>66.3</td>
</tr>
<tr>
<td>Instance 3</td>
<td>10210</td>
<td>120000</td>
<td>7171.67</td>
<td>234.3</td>
</tr>
<tr>
<td>Instance 4</td>
<td>23372</td>
<td>300000</td>
<td>13223</td>
<td>12541.2</td>
</tr>
<tr>
<td>Instance 5</td>
<td>46744</td>
<td>600000</td>
<td>18436</td>
<td>281956.0</td>
</tr>
</tbody>
</table>

### Table 5.8: Effect of $LF$-restrictions and assembly sequence-dependent delivery routes

<table>
<thead>
<tr>
<th>Problem</th>
<th>$\theta_{LF}$</th>
<th>Starting solution</th>
<th>(v_1, v_2, v_3, v_4)</th>
<th>Penalty</th>
<th>(v_1, v_2, v_3, v_4)</th>
<th>$\sum_{i \in T} v_i$</th>
<th>Final solution</th>
<th>Penalty</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance 1</td>
<td>0</td>
<td>(10, 0, 0, 0)</td>
<td>0</td>
<td>(5, 1, 1, 0)</td>
<td>7</td>
<td>(8, 11)</td>
<td>4.42</td>
<td>1.36</td>
<td>4015</td>
</tr>
<tr>
<td>Instance 2</td>
<td>3</td>
<td>(10, 0, 0, 0)</td>
<td>60000</td>
<td>(0,1,0,2)</td>
<td>3</td>
<td>(18, 32, 20)</td>
<td>3.33</td>
<td>2.09</td>
<td>4865</td>
</tr>
<tr>
<td>Instance 3</td>
<td>0</td>
<td>(15, 0, 0, 0)</td>
<td>0</td>
<td>(3.4, 0.1)</td>
<td>8</td>
<td>(26, 32, 25)</td>
<td>1.87</td>
<td>1.56</td>
<td>5495</td>
</tr>
<tr>
<td>Instance 4</td>
<td>3</td>
<td>(15, 0, 0, 0)</td>
<td>90000</td>
<td>(0,2,1,2)</td>
<td>5</td>
<td>(31, 60, 42)</td>
<td>1.91</td>
<td>0</td>
<td>6510</td>
</tr>
<tr>
<td>Instance 5</td>
<td>0</td>
<td>(20, 0, 0, 0)</td>
<td>0</td>
<td>(4.3, 2.1)</td>
<td>10</td>
<td>(24, 44, 33)</td>
<td>1.35</td>
<td>0</td>
<td>6675</td>
</tr>
<tr>
<td>Instance 6</td>
<td>3</td>
<td>(20, 0, 0, 0)</td>
<td>120000</td>
<td>(0,2,0,4)</td>
<td>6</td>
<td>(44, 78, 66)</td>
<td>3.33</td>
<td>2.31</td>
<td>7670</td>
</tr>
<tr>
<td>Instance 7</td>
<td>0</td>
<td>(50, 0, 0, 0)</td>
<td>0</td>
<td>(3,5,3,7)</td>
<td>18</td>
<td>(98, 152, 112)</td>
<td>2.77</td>
<td>2.12</td>
<td>0</td>
</tr>
<tr>
<td>Instance 8</td>
<td>3</td>
<td>(50, 0, 0, 0)</td>
<td>300000</td>
<td>(1,5,1,9)</td>
<td>16</td>
<td>(102, 219, 162)</td>
<td>3.125</td>
<td>2.18</td>
<td>0</td>
</tr>
<tr>
<td>Instance 9</td>
<td>0</td>
<td>(100, 0, 0, 0)</td>
<td>0</td>
<td>(1,12,1,18)</td>
<td>32</td>
<td>(221, 341, 319)</td>
<td>3.125</td>
<td>2.72</td>
<td>0</td>
</tr>
<tr>
<td>Instance 10</td>
<td>3</td>
<td>(100, 0, 0, 0)</td>
<td>600000</td>
<td>(1,12,1,18)</td>
<td>32</td>
<td>(285, 408, 322)</td>
<td>3.125</td>
<td>2.72</td>
<td>18436</td>
</tr>
</tbody>
</table>

### Table 5.9: Summary of $LF$ and assembly sequence-dependent delivery route effects

<table>
<thead>
<tr>
<th>Problem</th>
<th>Basic model</th>
<th>Assembly sequence-dependent model</th>
<th>Final solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\theta_{LF} = 0$, $v_1$</td>
<td>$\theta_{LF} = 3$, $v_2$</td>
<td>$\theta_{LF} = 0$, $v_3$</td>
<td>$\theta_{LF} = 3$, $v_4$</td>
</tr>
<tr>
<td>Instance 1</td>
<td>4015</td>
<td>4190</td>
<td>4015</td>
</tr>
<tr>
<td>Instance 2</td>
<td>4945</td>
<td>5155</td>
<td>5495</td>
</tr>
<tr>
<td>Instance 3</td>
<td>5755</td>
<td>5885</td>
<td>6675</td>
</tr>
<tr>
<td>Instance 4</td>
<td>12444</td>
<td>12459</td>
<td>13140</td>
</tr>
<tr>
<td>Instance 5</td>
<td>18403</td>
<td>18403</td>
<td>18436</td>
</tr>
</tbody>
</table>

88
5.5.4 Equipment-based Restrictions

We provide some sample computations in this section pertaining to equipment-based restrictions where the objective term (5.12a) is incorporated within the objective function of Problem SP (given in (5.11)). For illustrative purposes, we have assumed that there are no restrictions on the number of drivers available and have focused our attention on equipment-based constraints. We have also enforced assembly sequence-dependent restrictions on the delivery routes. Table 5.10 presents the results for Instance 5 for two equipment resource-restricted scenarios having $E = 2$ and $\lambda_1 = \lambda_2 = 32$, and $\lambda_1 = \lambda_2 = 40$, respectively (see Constraint (5.10c)). Observe that more restrictive values of $\lambda$ tend to favor single-vehicle patterns which consume no equipment resources and, as a consequence, yield relatively higher operational costs. Hence, our optimization framework can be utilized by decision-makers to ascertain investments in equipment resources that are necessary to achieve desired target costs and performance measures.

5.5.5 Rolling-horizon Framework

In this section, we discuss the effect of using an extended rolling-horizon framework and the importance of striking a tradeoff between the quality of the solution produced via a look-ahead feature and the ensuing computational effort. Thus far, we have considered a planning horizon that comprises three days ($d = 0, 1, 2$). Instead, we could adopt a shorter planning horizon, and solve the problem under investigation over two planning periods, the first of which considers vehicles that are available on Days 0 and 1, whereas the second deals with Days 1 and 2. Specifically, upon solving the restricted problem over Period 1, any assembly that contains vehicles that are available on Day 0, and perhaps other additional ones that are available on Day 1, is operationally committed for implementation. However, assemblies that exclusively involve vehicles that are made available on Day 1 are carried over to the next period and are re-optimized along with vehicles that are made available on Day 2.

For an illustration, consider the results reported for Instance 4 in Table 5.11. In the first period, a restricted problem of 39 vehicles has been solved using Algorithm A. This yielded a solution that
Problem | Period | Starting solution | Final solution | Total cost | CPU time (s) | (% Gap) |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance 4</td>
<td>1</td>
<td>(39,0,0,0)</td>
<td>17662</td>
<td>(1,2,2,7) → (1,1,1,6)</td>
<td>12 → 9</td>
<td>9379 → 7082</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>(20,0,0,0)</td>
<td>9768</td>
<td>(11,1,1,1)</td>
<td>144</td>
<td>8007</td>
</tr>
<tr>
<td>Instance 5</td>
<td>1</td>
<td>(78,0,0,0)</td>
<td>35324</td>
<td>(0,9,0,15) → (0,5,0,9)</td>
<td>24 → 14</td>
<td>13231 → 7671</td>
</tr>
</tbody>
</table>
| | 2 | (54,0,0,0) | 25867 | (22,4,0,6) | 32 | 16980 | 341.10 | 90

Table 5.11: Impact of planning horizon length

consists of 12 vehicle assemblies and a corresponding objective value that equals 9379. Out of these 12 assemblies, only 9 were implemented during Period 1, thereby incurring an actual cost of 7082. The other three (discarded) assemblies involved vehicles that were made available on Day 1 and, therefore, these vehicles were considered for re-optimization in Period 2. The total operational cost incurred over the two periods is reported in the last column of Table 5.11, followed by a percentage optimality gap that compares this total cost against the results obtained in Table 5.1 where a longer planning horizon was utilized (with $D = 2$).

The results reported in Table 5.11 suggest that using a more restricted look-ahead feature within a rolling-horizon framework yields an optimality gap within 17.52% and 25.34%, and achieves a savings in computational effort of 80% and 76.6% for Instances 4 and 5, respectively. Ultimately, decision-makers would need to select the magnitude of the planning horizon in the light of the operational cost of the solutions generated and the accompanying computational effort.

5.6 Summary, Conclusions, and Future Research

An integrated modeling and optimization approach is proposed in this chapter to address joint vehicle loading-routing problems, possibly involving assembly sequence-dependent delivery routes, as well as equipment- and driver-based restrictions. This approach automates decisions with respect to assembly, inventory, delivery, driver-return, and lateness penalty costs, thereby promoting more consistent cost-effective decisions and eliminating the labor-intensive, manual ad hoc decision-making process. The model can be strategically implemented under the assumption of unlimited connection equipment and driver resources, and its output can be used to ascertain the ideal accompanying equipment and driver needs.

Our computational results demonstrate the efficiency of the proposed methodology for the basic model, as well as its extensions and specializations. In particular, problem instances involving up to 50 trucks over a planning horizon of three days are efficiently solved by all models. For larger
problem instances, our results suggest that it is beneficial to employ clustering strategies. In addition, our model restricts (as a soft-constraint) the average load-factor over the delivery routes to be at least a certain specified threshold amount. The relative importance of this feature, and the fine-tuning of the penalty associated with its violation, can be determined by decision-makers in the light of the total cost incurred, resource utilization, and personnel policies. In fact, this optimization framework enables managers to investigate various what-if scenarios related to variations in cost parameters, effect of enforcing different load-factors, value of improved advance-information from the plants in regard to required shipments to dealerships (i.e., considering extended look-ahead horizons), and the introduction of new valid combinations in the assembly matrix, among others.

The efficiency of our column generation approach is essentially due to the sparsity of the columns generated as well as the flexibility in the number of vehicles that can be selected to compose any pattern within the column generation subproblem, which varies between one and four in the trucking industry. These features tend to mitigate the computational effort associated with the subproblem, thereby allowing the problem to be solved in manageable times. In addition, these characteristics of the problem tend to produce continuous solutions to the set partitioning problem where the selected patterns are relatively sparse and do not significantly overlap and, hence, the associated binary-restricted solutions are generally optimal or near-optimal (with an optimality gap within 1.3% when load-factor considerations are relaxed and within 6.4% when load-factor requirements are imposed). We have proposed three alternative strategies pertaining to the column generation process in the subproblem, and have demonstrated for larger problem instances the computational superiority of Strategy 3, which aims at generating iteratively patterns of sizes 2, 3, and 4 (whenever possible) within every LP phase iteration. The incorporation of load-factor requirements introduces an increase in the objective value as well as a computational burden for Strategies 1 and 2 that results from the need to balance the operational cost and the potential violation-induced penalty pertaining to the load-factor.

Various operational scenarios and computational strategies can be investigated in future research. Some of these are briefly identified below.

1. This model could be integrated with a separate model for retrieving and replenishing equipment, as well as for scheduling individual drivers subject to work-hours and on-off-days restrictions. This feature can also be used to assess marginal values accruing from the provision of additional equipment and driver resources at different times of the year.
2. Future research can investigate an extension of this problem to involve multiple source facilities. That is, some pattern can be generated at one source, and at any time over its delivery route, it may visit another source facility to complement its assembled load, subject to tow-size and assembly-compatibility restrictions.

3. It may be interesting to investigate possible connections between the problem addressed in this chapter and integrated lot-sizing problems in production systems, using the analogy between the vehicle assembly phase and the batch-sizing operation.

4. The design of sophisticated clustering techniques and their integration with our modeling framework in an automated preprocessing step is an attractive topic for future research, especially for industrial-sized problem instances. Parallel computing may assist in solving large-scale problem instances. Typically, each processor may be dedicated to the generation of promising patterns of a certain size only, and the joint effort of the multiple processors could then be coordinated through the master problem.

5. The proposed column generation algorithm for the underlying set partitioning problem SPP can be further improved with respect to numerical computational performance by incorporating certain stabilization augmentations in the spirit of [43].

6. It is advantageous to employ a deflected subgradient optimization strategy along with a primal solution recovery scheme as discussed in Lim and Sherali [76] (see also Sherali and Choi [112]) for solving the Lagrangian dual of Problem SPP or ESPP, in lieu of using LP solvers, because of the following reasons. First, this can generate dual solutions for use in the subproblems relatively much faster. Second, as discussed in Subramanian and Sherali [132], this approach can reduce the dual-noise and stalling phenomena that are common with LP solvers when solving set partitioning problems using column generation techniques (even using interior point algorithms), and can promote the generation of an improved set of columns. Third, the hard-constraints (5.10b), in concert with the driver- and equipment-based restrictions (5.10c) and (5.10d), can possibly result in an infeasible problem ESPP. Because the Lagrangian dual approach would dualize these constraints, it would have an automatic “softening” effect on these restrictions.

7. Finally, it might be of interest to incorporate average weighted load-factor restrictions (see Equation (5.2)) within the model, by replacing $LF_j$ with $WLF_j$ in Constraint (5.3c). A
heuristic for accommodating such a consideration might be to solve the subproblem (5.6) as proposed herein, but then to compute the actual $WLF$-value obtained for the generated column and substitute that in place of $LF$ in the reduced cost function (5.6a). If the resulting reduced cost value is negative, then this generated column can be passed to the master program. Some preliminary results obtained using this strategy support the efficacy of this approach.
Chapter 6

Balancing Production and Resource Consumption in Mixed-model Assembly Lines: Integrated Partitioning-Sequencing Models

We address the problem of partitioning and sequencing the production of different manufactured items in assembly lines at a major motorcycle manufacturing plant, where each model has various specific options and designated destinations. We propose a mixed-integer programming formulation (MPSP1) for this problem that sequences the manufactured goods within production batches in order to balance the motorcycle model-option and destination outputs as well as the load demands on material and labor resources. An alternative (relaxed) formulation (MPSP2) is also presented to model a closely related case where all production decisions and outputs are monitored within a common sequence of batches, which permits an enhanced tighter representation via an additional set of hierarchical symmetry-defeating constraints that impart specific identities amongst batches of products under composition. The latter model inspires a third set partitioning-based formulation in concert with an efficient column generation approach that directly achieves the joint partitioning of jobs into batches along with ascertaining the sequence of jobs within each composed batch. Finally, we investigate a subgradient-based optimization strategy for Models MPSP1 and MPSP2 that exploits an equivalent non-differentiable optimization formulation. Computational experience is presented to demonstrate the relative effectiveness of the different proposed formulations and their associated optimization strategies for solving a set of realistic problem instances.
6.1 Introduction and Motivation

Assembly lines play an important role in manufacturing environments where a commodity can be progressively assembled through a flow-oriented system. Due to the ever-increasing diversity in customer specifications, it has become crucial for manufacturers to offer multiple models, and possibly, various customized versions of any given model. This has led to the development of mixed-model assembly lines, where a heterogeneous sequence of product variants is manufactured along the same flow line. This is commonly witnessed in the automobile production business ([29]) and similar industries. Two intimately related problems, namely, the mixed-model assembly line balancing problem (MALBP) and the mixed-model sequencing problem (MSP), arise in such environments and have direct impacts on load-balancing issues and inefficiencies within an assembly line [15].

The MALBP is a complex problem that arises while designing or reconfiguring a line in the light of a product-mix. Given an associated assembly precedence graph along with the task durations for each product, it is difficult, and highly unlikely, to identify a line balance that simultaneously smoothes station loads for every product, and station times at every workstation. Thus, in addition to vertical balancing, which focuses on station times for a given job, the concept of horizontal balancing has been introduced that deals with smoothing station times of the different products for each workstation [89]. The MSP addresses the problem of sequencing a set of heterogeneous jobs to be manufactured along the assembly line in order to optimize some performance measures. Ideally, an integrated approach should jointly consider the MALBP and the MSP for a given system, and the line balance and product sequence should be periodically updated if the demand mix significantly changes over time. However, due to the complexity of each of these problems and the accompanying computational effort required, they are addressed separately in the literature, notwithstanding a few exceptions ([89],[136]).

The MSP has gained wide popularity in conjunction with the development of just-in-time (JIT) production systems in the automotive industry. Previous research work has investigated the level-scheduling problem that aims at stabilizing the subassembly parts usage (for instance, see Ding and Cheng [40] and Inman and Bulfin [62]). Various studies have also tackled bi-criteria optimization problems, addressing the level-scheduling problem in concert with the so-called car-sequencing problem that seeks to smooth workstation loads in assembly lines (see Drexl and Kimms [41], Korkmazel and Meral [68], and Zeramdini et al. [155]). Other research efforts have considered different
bicriteria optimization problems where smoothing the subassembly parts usage is conjugated with a conflicting objective, such as minimizing job lateness or the number of setups. Heuristics ([78], [155]) and metaheuristics ([86], [152]) are commonly adopted to solve variants of the MSP due to their complexity. However, Zhang et al. [156] investigate the level-scheduling problem in a mixed-model JIT assembly line, and propose a Lagrangian relaxation approach that incorporates perturbation strategies to overcome slow algorithmic convergence. Also, Drexl and Kimms [41] derive a set partitioning formulation to jointly address the smoothing of parts usage and workstation loads, and propose a column generation-based approach that produces tight linear relaxations of the underlying formulation.

Although there exists a substantial amount of research on multi-criteria, and in particular, bicriteria, formulations for the MSP, integrated models have received less attention in the literature. A few exceptions include integrated balancing-sequencing formulations ([89], [136]) and joint planning-sequencing models [152]. Various mathematical programming models involving different objective functions have been suggested for the MSP along with appropriate solution methodologies ([9], [102], [153]).

We consider the mixed-model partitioning and sequencing problem (MPSP) that was posed to us by a major motorcycle manufacturing company (Harley-Davidson Motor Company). Given a set of motorcycles that are being assembled, we seek to partition these items into production batches along with sequencing the motorcycles within any composed production batch, so as to balance the model and destination outputs as well as the load demands on material and labor resources at various workstations. This problem arises due to the customization of products with several options and characterizations based on customer destinations, and the subsequent need to meet multiple objectives that pertain to balancing motorcycle model and destination outputs as well as resource consumptions. The originality of this problem resides in the challenge posed by the integrated partitioning and sequencing operations as well as the associated composite objective function.

The assembly line is comprised of several (over 50) stations and manufactures some nine principal models of motorcycles (or bikes). Furthermore, each model can be equipped with either a carbureted or an electronic fuel-injection (EFI) option, which yields 18 possible motorcycle model-option combinations. Note that there exist several other options regarding the type of frame and suspension with which each motorcycle is assembled. However, the foregoing carbureted or EFI
options are the principal distinguishing features that govern other accompanying option specifications. Hence, we will focus on the aforementioned 18 motorcycle model-option compositions. In addition, there are three particular destinations delineated for each model-option. For convenience in our discussion, we shall refer to each of the resultant 54 model-option-destination combinations as a product manufactured by the plant.

There are two types of resources involved in the production operation – material and labor. Each product entails a different work-content and material usage at each station in the assembly line. Naturally, if the products are sequenced so that high work-content vehicles appear consecutively, then the operators can get overwhelmed by a relatively high persistent workload, thereby making them fall behind schedule, and there is also a danger of running out of material because of rapid consumption rates, thereby causing the assembly line to stall. Hence, there is a need to carefully sequence the production operations so as to smoothen the demands on the material and labor resources (in a formal sense as described in the sequel). Furthermore, the model-options produced, as well as those delivered to each of the three different destinations, need to be evenly distributed throughout the sequence. Future extensions to this work may investigate the design of an integrated hierarchical planning system that coordinates and interfaces the sequencing model and optimization strategy suggested herein with a line balancing module.

The remainder of this chapter is organized as follows. In Section 6.2, we introduce our notation along with a formal description of the problem. Thereafter, in Section 6.3, we construct a primary mathematical programming formulation for the MPSP, along an alternative (relaxed) formulation to model the case where all production decisions and outputs are monitored over a common set of batches. The structure of the latter model permits an enhanced representation through the addition of a set of hierarchical symmetry-defeating valid inequalities. This modeling approach also prompts a third integrated set partitioning-based formulation that is presented in Section 6.4 in concert with an efficient column generation approach, which directly achieves the joint partitioning of jobs into batches along with ascertaining the sequence of jobs within each composed batch. A subgradient-based algorithmic approach for solving either of the primary or relaxed models is described in Section 6.5. Section 6.6 empirically investigates the relative effectiveness of the different proposed formulations and their associated optimization strategies for solving a set of realistic problem instances, and Section 6.7 concludes this chapter with a summary and directions for future research.
6.2 Problem Description and Notation

In this section, we present our notation along with a more formal description of the aforementioned mixed-model partitioning and sequencing problem (MPSP) faced at the Harley-Davidson Motor Company. This company is concerned with sequencing the production of bikes from the viewpoint of balancing the usage of material and labor resources as well as smoothing the output with respect to the different model-options manufactured and the different shipment destinations. More specifically, this load balancing needs to accomplish the following, to the extent possible.

First, consider the material and labor usage. Each product consumes a specified amount of material and labor at each station in terms of the percentage capacity at that station. This percentage is referred to as the material or labor consumption efficiency at the station. It is desired that these efficiency values at each station, when averaged over two consecutive bike productions, as well as over three consecutive bike productions, remain less than or equal to 95%. Management would like to eliminate the “three-bike violations” in average consumption efficiency values, while reducing the “two-bike violations” as much as possible. Furthermore, although there are over 50 stations in the assembly line, it is sufficient to simultaneously control these consumption efficiency values at some 10 key/bottleneck stations.

As far as smoothing the output is concerned, this needs to be achieved with respect to certain forecast data that specifies what percentage of the total mix should be comprised of each model-option type, as well as what percentage of the total mix should pertain to each destination type. The production sequence should be such that over every composed motorcycle production batch, the actual percentage of model-option types and destination types in this mix should preferably not exceed the average forecasted value by some specified factor. Note that the overall given set of products to be sequenced is assumed to meet the total anticipated demand that is forecasted for each of these model-option and destination types.

Remark 6.1. The foregoing discussion focuses on material and labor usage and the output rates for different bike types. However, three of the nine models are relatively more time consuming to manufacture than the others, and it is desirable to also reduce the frequency with which these three models occur relatively close together in the sequence. Presently, it is envisaged that this goal will be achieved by swapping the production of these three motorcycle models with the others as desired in a second pass after determining an optimal sequence based on the foregoing considerations,
without disrupting the distribution characteristics established in the first pass. However, in our model formulation, we will also recommend how this restriction can be accommodated in the first pass itself via a soft-constraint.

We provide below the following glossary of notation, as pertaining to the problem under investigation.

6.2.1 General Notation

Indices

- \( j = 1,...,n \): Index for the jobs to be sequenced, where each job is a particular product (model-option-destination combination). (We shall use jobs and products interchangeably.)
- \( k = 1,...,n \): Position-slots for sequencing the jobs.
- \( m = 1,...,M \): Index for model-options \((M = 18)\).
- \( d = 1,...,D \): Index for destinations \((D = 3)\).
- \( r = 1,...,R \): Index for batches of position-slots over which the balancing of output is conducted.
- \( r' = 1,...,R' \): Index for batches of position-slots over which the frequency of time-consuming model production is controlled (see Remark 6.1).
- \( s = 1,...,S \): Index for the (key/bottleneck) stations for monitoring two-bike and three-bike efficiency violations \((S = 10)\).

Sets

- \( J \equiv \{1,...,n\} \): Set of jobs.
- \( J^1_m = \{j \in J: \text{job } j \text{ corresponds to model-option } m\}, \forall m. \)
- \( J^2_d = \{j \in J: \text{job } j \text{ corresponds to destination } d\}, \forall d. \)
- \( J^+ = \{j \in J: \text{job } j \text{ corresponds to one of the (three) relatively more time-consuming model types}\}. \)
- \( N_r = \{\text{set of consecutive position-slots } k \text{ in the batch } r \text{ for monitoring outputs}\}, \forall r. \) (Note that \( N_i \cap N_j = \emptyset, \forall i \neq j, \) and \( \bigcup_{r=1}^{R} N_r = \{1,...,n\}. \))
\(N^+_{r'} = \{\text{set of consecutive position-slots } k \text{ in batch } r' \text{ for monitoring the production of the relatively more time-consuming model types}\}, \forall r'.\) (Note that we could have \(N^+_{i'} \cap N^+_{j'} \neq \emptyset\), for \(i \neq j\), but \(\bigcup_{r'=1}^{R'} N^+_{r'} = \{1,\ldots,n\}\).)

### Parameters

- \(\alpha_{js}\) = Material usage for job \(j\) at station \(s\) in terms of the related percentage of this station’s capacity, \(\forall j, s\).
- \(\beta_{js}\) = Labor usage for job \(j\) at station \(s\) in terms of the related percentage of this station’s capacity, \(\forall j, s\).
- \(F^1_m\) = Percentage of forecasted demand/production that pertains to model-option \(m\), \(\forall m\). (Note that \(\sum_{m=1}^{M} F^1_m = 100\%\).)
- \(F^2_d\) = Percentage of forecasted demand/production that pertains to destination \(d\), \(\forall d\). (Note that \(\sum_{d=1}^{D} F^2_d = 100\%\).)
- \(\gamma^1_m \in [1, 1.5]\) = Maximum tolerance factor for the ratio of actual to forecasted production levels for model-option \(m\) over each monitored set of position-slots \(N_1,\ldots,N_R\), \(\forall m\).
- \(\gamma^2_d \in [1, 1.5]\) = Maximum tolerance factor for the ratio of actual to forecasted production levels for destination \(d\) over each monitored set of position-slots \(N_1,\ldots,N_R\), \(\forall d\).
- \(\theta^+\) = Maximum total number of time-consuming model related jobs that can be sequenced within in any of the monitored sets of position-slots \(N^+_{1},\ldots,N^+_{R'}\).
- \(\lambda_1,\ldots,\lambda_7\) = Relative priority weights ascribed to the seven objective function terms (see Equation (6.1a) below), where \(\lambda_i > 0, \forall i = 1,\ldots,7\), and where \(\sum_{i=1}^{7} \lambda_i = 1\).

### Decision Variables

Our modeling approach involves a principal set of binary decision variables that pertain to the sequence-position of jobs. In addition, we introduce various sets of continuous variables related to specific slacks and deviations that contribute toward measuring the balancing of the output and resource consumptions.
Principal Decision Variables

\[ x_{jk} = \begin{cases} 
1 & \text{if job } j \text{ is sequenced in position } k \\
0 & \text{otherwise, } \forall j, k.
\end{cases} \]

Auxiliary Variables

- \( \psi_{1+}, \psi_{1-} \): Positive or negative deviations, respectively, between the inflated forecast \( \gamma_{1}^{1} F_{1}^{1} \) and the actual production level of model-option \( m \) over position-slots in \( N_{r}, \forall r, m \).
- \( \psi_{2+}, \psi_{2-} \): Positive or negative deviations, respectively, between the inflated forecast \( \gamma_{2}^{2} F_{2}^{2} \) and the actual production level for destination \( d \) over position-slots in \( N_{r}, \forall r, d \).
- \( \delta^{2+}_{ks}, \delta^{2-}_{ks} \): Positive or negative deviations, respectively, as occur between 95\% and the average material consumption efficiency values achieved at station \( s \) over position-slots \( k \) and \( k + 1 \), for \( k = 1, ..., n - 1, s = 1, ..., S \).
- \( \mu^{2+}_{ks}, \mu^{2-}_{ks} \): Positive or negative deviations, respectively, as occur between 95\% and the average labor consumption efficiency values achieved at station \( s \) over position-slots \( k \) and \( k + 1 \), for \( k = 1, ..., n - 1, s = 1, ..., S \).
- \((\delta^{3+}_{ks}, \delta^{3-}_{ks})\) and \((\mu^{3+}_{ks}, \mu^{3-}_{ks})\): Similar to above, except for monitoring the three-bike efficiency violations, for \( k = 1, ..., n - 2, s = 1, ..., S \).
- \( \varphi_{+}, \varphi_{-} \): Positive or negative deviations, respectively, as occur between the maximum targeted production \( \theta^{+} \) and the actual production of the relatively more time-consuming bikes during each of the monitored position-slots \( N_{r}^{+}, \forall r' \).

6.3 Mathematical Programming Formulations

In this section, we first provide a mathematical programming formulation (MPSP1) for the mixed-model partitioning and sequencing problem that is described in Section 6.2. Thereafter, we present an alternative formulation (MPSP2) for modeling the situation where the production decisions and all outputs are monitored over a common set of position-slot batches.

6.3.1 Primary Model Formulation MPSP1

A mathematical programming formulation for the mixed-model partitioning and sequencing problem described in Section 6.2, denoted Model MPSP1, is stated below.
MPSP1: Minimize \[ \lambda_1 \left[ \sum_{k=1}^{n-1} S_{k,s}^2 \right] + \lambda_2 \left[ \sum_{k=1}^{n-1} S_{k,s}^2 \right] + \lambda_3 \left[ \sum_{k=1}^{n-2} S_{k,s}^3 \right] + \lambda_4 \left[ \sum_{k=1}^{n-2} S_{k,s}^3 \right] \]
\[ + \lambda_5 \left[ \sum_{r=1}^{R} \sum_{m=1}^{M} \psi_{r,m}^1 \right] + \lambda_6 \left[ \sum_{r=1}^{R} \sum_{d=1}^{D} \psi_{r,d}^2 \right] + \lambda_7 \left[ \sum_{r'=1}^{R'} \varphi_{r'} \right] \] (6.1a)

subject to \[ \sum_{k=1}^{n} x_{j,k} = 1, \quad \forall j \] (6.1b)
\[ \sum_{j=1}^{n} x_{j,k} = 1, \quad \forall k \] (6.1c)
\[ 100 * \left\{ \frac{\sum_{j \in J^1_m} \sum_{k \in N_r} x_{j,k}}{|N_r|} \right\} + \psi_{r,m}^1 - \psi_{r,m}^1 = \gamma_{m} F_{m}^1, \quad \forall r, m \] (6.1d)
\[ 100 * \left\{ \frac{\sum_{j \in J^2_d} \sum_{k \in N_r} x_{j,k}}{|N_r|} \right\} + \psi_{r,d}^2 - \psi_{r,d}^2 = \gamma_{d} F_{d}^2, \quad \forall r, d \] (6.1e)
\[ \frac{1}{2} \left( \sum_{j=1}^{n} \alpha_{j,s} x_{j,k} + \sum_{j=1}^{n} \alpha_{j,s} x_{j,k+1} \right) + \delta_{k,s}^2 - \delta_{k,s}^2 = 95, \quad \forall k = 1, ..., n-1, s = 1, ..., S \] (6.1f)
\[ \frac{1}{2} \left( \sum_{j=1}^{n} \beta_{j,s} x_{j,k} + \sum_{j=1}^{n} \beta_{j,s} x_{j,k+1} \right) + \mu_{k,s}^2 - \mu_{k,s}^2 = 95, \quad \forall k = 1, ..., n-1, s = 1, ..., S \] (6.1g)
\[ \frac{1}{3} \left( \sum_{j=1}^{n} \alpha_{j,s} x_{j,k} + \sum_{j=1}^{n} \alpha_{j,s} x_{j,k+1} + \sum_{j=1}^{n} \alpha_{j,s} x_{j,k+2} \right) + \delta_{k,s}^3 - \delta_{k,s}^3 = 95, \quad \forall k = 1, ..., n-2, s = 1, ..., S \] (6.1h)
\[ \frac{1}{3} \left( \sum_{j=1}^{n} \beta_{j,s} x_{j,k} + \sum_{j=1}^{n} \beta_{j,s} x_{j,k+1} + \sum_{j=1}^{n} \beta_{j,s} x_{j,k+2} \right) + \mu_{k,s}^3 - \mu_{k,s}^3 = 95, \quad \forall k = 1, ..., n-2, s = 1, ..., S \] (6.1i)
\[ \sum_{j \in J^+ k \in N^+_{\mathcal{r'}}} x_{j,k} + \varphi_{r'}^+ - \varphi_{r'}^- = \theta^+, \quad \forall r' \] (6.1j)
\[ x \text{ binary, } (\psi^{1\pm}, \psi^{2\pm}, \delta^{2\pm}, \mu^{2\pm}, \delta^{3\pm}, \mu^{3\pm}, \varphi^\pm) \geq 0. \] (6.1k)

Model MPSP1 is a 0-1 mixed-integer program. Its objective function accommodates seven terms.
having suitable weights to stress their relative importances. These terms respectively represent the total violations in the two-bike efficiencies for material and labor usages, the total violations in the three-bike efficiencies for material and labor usage, the total surplus amounts in the model-option and destination output percentage mix constraints, and the total violation in the frequency restrictions for the production of the relatively more time-consuming models. Constraints (6.1b)-(6.1c) are the assignment constraints that sequence the jobs by matching them with the position-slots. Constraints (6.1d) and (6.1e) are soft-constraints that measure the deviations from the specified forecasts of the percentage of each model-option and each destination type, respectively, in a mix that is assigned to the position-slots in \( N_r \). Constraints (6.1f) and (6.1g) are soft-constraints that respectively record the positive or negative deviations in the two-bike average material and labor resource usages from the targeted 95% value. The first two corresponding objective terms therefore aim to reduce the total excess over 95% of the two-bike average material and labor resource usages. The constraints (6.1h) and (6.1i) are similar soft-constraints for the three-bike average material and labor resource usages, respectively. Constraint (6.1j) records the positive and negative deviations in the production frequency of the relatively more time-consuming models from the targeted smoothened value \( \theta^+ \) over each set \( N_r' \) of position-slots, with the excess over this target value being penalized via the last term in the objective function. Finally, (6.1k) enforces logical conditions on the variables.

**Remark 6.2.** Note that the only set of “hard-constraints” enforced are the assignment constraints (6.1b) and (6.1c); the remainder of the constraints are “soft-constraints” that aim at achieving certain target values, where the surplus variables are penalized in the objective function. Also, Constraint (6.1j) and the corresponding last term in the objective function (6.1a) are optional as discussed in Remark 6.1. Finally, note that for any pair of jobs \( j_1 \) and \( j_2 \), the expression

\[
\left| \sum_{k=1}^{n} kx_{j_1k} - \sum_{k=1}^{n} kx_{j_2k} \right|
\]

(6.2)

gives the absolute gap in terms of the number of position-slots that separate jobs \( j_1 \) and \( j_2 \). By maximizing the minimum such separation over pairs of jobs in any given category (including model-option types, destination types, or time-consuming types), we can alternatively address the spread of jobs belonging to such a common category. This could be explored in future research.
6.3.2 Alternative Relaxed Formulation MPSP2 Under Common Monitoring Batches

We propose in this section a relaxed formulation of Model MPSP1 under an additional assumption that all batches are of equal size, \( \omega \) (and, therefore, \( n = \omega R \)), and that \( N_r = N_r^+ \), \( \forall r = r' \), i.e., all outputs, including those for the relatively time-consuming bike types, are monitored over common batches. We shall refer to this assumption as Assumption A. Observe that Model MPSP1 requires the \( R \) batches of jobs under composition to each comply closely, within a limited interval, to the overall specified model-option and destination-based forecasts, and accounts for the two- and three-bike violations within consecutive pairs and triplets of jobs across all batches. However, noting that the ratio \( \frac{n}{R} \geq 5 \) for any problem instance of practical relevance for our bike manufacturing application, the relative contribution of two- and three-bike violations between two consecutive batches is largely dominated by such violations that occur within batches. In this perspective, the relaxed model MPSP2 delineated below under Assumption A, considers the \( R \) batches that we are composing to be independent. That is, Model MPSP2 is obtained upon relaxing constraints related to violations by pairs and triplets of consecutive jobs that belong to two adjacent batches, and the objective function (6.3a) below achieves a lower-approximation of (6.1a) by omitting the associated penalty terms. As a consequence, under Assumption A, the optimal objective value of Model MPSP2 provides a lower bound on the optimal objective value of Model MPSP1, and any feasible solution to Model MPSP2 can be post-evaluated via Model MPSP1 to obtain an upper bound on the latter (see Section 6.6).

**MPSP2: Minimize**

\[
\sum_{r=1}^{R} \left\{ \lambda_1 \left[ \sum_{\ell=(r-1)\omega+1}^{r\omega-1} \sum_{s=1}^{S} \delta_{\ell s}^2 \right] + \lambda_2 \left[ \sum_{\ell=(r-1)\omega+1}^{r\omega-1} \sum_{s=1}^{S} \mu_{\ell s}^2 \right] + \lambda_3 \left[ \sum_{\ell=(r-1)\omega+1}^{r\omega-2} \sum_{s=1}^{S} \delta_{\ell s}^3 \right] + \lambda_4 \left[ \sum_{\ell=(r-1)\omega+1}^{r\omega-2} \sum_{s=1}^{S} \mu_{\ell s}^3 \right] + \lambda_5 \left[ \sum_{m=1}^{M} \psi_{1r}^1 \right] + \lambda_6 \left[ \sum_{d=1}^{D} \psi_{2r}^2 \right] + \lambda_7 \left[ \varphi_{-r}^1 \right] \right\}
\]

subject to

\( \sum_{k=1}^{n} x_{jk} = 1, \quad \forall j \)  

\( \sum_{j=1}^{n} x_{jk} = 1, \quad \forall k \)
\[
100 \left\{ \frac{\sum_{j \in J_m} \sum_{k \in N_r} x_{jk}}{\omega} \right\} + \psi_{rm}^1 - \psi_{rm}^- = \gamma_m F_m^1, \quad \forall r, m \tag{6.3d}
\]

\[
100 \left\{ \frac{\sum_{j \in J_m^2} \sum_{k \in N_r} x_{jk}}{\omega} \right\} + \psi_{rd}^2 - \psi_{rd}^- = \gamma_d F_d^2, \quad \forall r, d \tag{6.3e}
\]

\[
\frac{1}{2} \left( \sum_{j=1}^{n} \alpha_{js} x_{j\ell} + \sum_{j=1}^{n} \alpha_{js} x_{j,\ell+1} \right) + \delta_{\ell s}^2 - \delta_{\ell s}^- = 95,
\forall r, \ell = (r-1)\omega + 1, \ldots, r\omega - 1, \forall s \tag{6.3f}
\]

\[
\frac{1}{2} \left( \sum_{j=1}^{n} \beta_{js} x_{j\ell} + \sum_{j=1}^{n} \beta_{js} x_{j,\ell+1} \right) + \mu_{\ell s}^2 - \mu_{\ell s}^- = 95,
\forall r, \ell = (r-1)\omega + 1, \ldots, r\omega - 1, \forall s \tag{6.3g}
\]

\[
\frac{1}{3} \left( \sum_{j=1}^{n} \alpha_{js} x_{j\ell} + \sum_{j=1}^{n} \alpha_{js} x_{j,\ell+1} + \sum_{j=1}^{n} \alpha_{js} x_{j,\ell+2} \right) + \delta_{\ell s}^3 - \delta_{\ell s}^- = 95,
\forall r, \ell = (r-1)\omega + 1, \ldots, r\omega - 2, \forall s \tag{6.3h}
\]

\[
\frac{1}{3} \left( \sum_{j=1}^{n} \beta_{js} x_{j\ell} + \sum_{j=1}^{n} \beta_{js} x_{j,\ell+1} + \sum_{j=1}^{n} \beta_{js} x_{j,\ell+2} \right) + \mu_{\ell s}^3 - \mu_{\ell s}^- = 95,
\forall r, \ell = (r-1)\omega + 1, \ldots, r\omega - 2, \forall s \tag{6.3i}
\]

\[
\sum_{j \in J^+} \sum_{k \in N_r} x_{jk} + \varphi_r^+ - \varphi_r^- = \theta^+, \quad \forall r
\tag{6.3j}
\]

\[
x \text{ binary, } (\psi_{1\pm}, \psi_{2\pm}, \delta_{2\pm}, \mu_{2\pm}, \delta_{3\pm}, \mu_{3\pm}, \varphi_{1\pm}) \geq 0. \tag{6.3k}
\]

Observe that Model MPSP2 exhibits an inherent symmetry with respect to batches and, therefore, any permutation of the \( R \) batches of bikes would, in fact, produce an equivalent solution. Sherali and Smith [124] have expounded on the conceptual and computational benefits of incorporating symmetry-defeating constraints to enhance algorithmic performance. To combat such undesirable symmetric reflections, we propose to incorporate within Model MPSP2 the following set of hierarchical symmetry-defeating valid inequalities, denoted by \((\sigma)\), which enforce specific identities amongst batches.

\[
\sum_{j=1}^{n} \sum_{k \in N_r} j x_{jk} \leq \sum_{j=1}^{n} \sum_{k \in N_{r+1}} j x_{jk}, \quad r = 1, \ldots, R - 1. \tag{6.4}
\]
We shall refer to the foregoing symmetry-defeating enhanced model as $\text{MPSP2}(\sigma)$, and shall evaluate the computational effect of incorporating this hierarchy of symmetry-defeating constraints (6.4) in Section 6.6.1.

**Remark 6.3.** We shall also investigate the potential benefits of defeating symmetry via objective perturbation strategies as discussed in Section 6.6.2. Formally, we incorporate the perturbation term $\varepsilon \sum_{j=1}^{n} \sum_{k=1}^{n} k j x_{jk}$ in the objective function (6.3a), which imitates the effect of the symmetry-defeating constraints by discriminating among symmetric reflections through their perturbation-based contributions to the objective function. This strategy is denoted $\text{MPSP2}(\phi)$ in the sequel. Finally, if the hierarchical constraints ($\sigma$) and the objective perturbation term are jointly enforced, we obtain an augmented formulation denoted $\text{MPSP2}(\sigma, \phi)$.

### 6.4 Integrated Model and Column Generation Approach

Adopting Assumption A, we propose in this section a set partitioning-based formulation in concert with a suitable column generation approach that appropriately ascertains the joint batch composition along with determining the job sequence within each composed batch.

#### 6.4.1 Set Partitioning-based Formulation

Observe that the problem under investigation captures two intertwined operations that need to be jointly addressed for effective operational decisions to be made. The first operation aims at partitioning the manufactured jobs into $R$ batches, whereas the second is concerned with the sequencing of jobs within any composed batch. The penalties incurred for model-option and destination forecast violations as well as the violations pertaining to the total number of time-consuming jobs are only determined by the batch composition, and the actual job sequence within a composed batch is responsible for the two- and three-bike resource violations incurred for this batch. The problem, then, is to partition the set of $n$ jobs being manufactured into $R$ complementary batches and determine the associated job sequencing decisions so that the total resulting penalty or cost is minimized. Note that any batch can be represented using the following modeling construct, which we refer to as a pattern or batch composition:
\[
P^i = \begin{bmatrix}
  j = 1 \\
  j = 2 \\
  \vdots \\
  j = n \\
\end{bmatrix}
\]

and where \( i = 1, \ldots, I \) indexes all such possible patterns. Define \( z_i \) to be a binary variable that assumes a value of 1 if and only if batch composition \( P^i \) is selected for execution in the manufacturing process. Also, let \( c_i \) be the cost associated with pattern \( P^i \), which comprises the total penalties incurred with respect to model and destination forecasts, two- and three-bike violations, and the number of more time-consuming jobs selected within any given batch under composition, given an optimal accompanying sequence of the jobs present within \( P^i \) (Section 6.4.2 provides a formal expression of this cost). Denoting \([e]\) as a vector of \(n\)-ones, the problem under investigation can be stated as the following set partitioning problem \( \text{SPP} \):

\[
\text{SPP}: \text{Minimize } \sum_{i=1}^{I} c_i z_i \\
\text{subject to } \sum_{i=1}^{I} P^i z_i = [e] \\
z \text{ binary.}
\]

Although the sequencing decisions associated with any pattern \( P^i \) are not explicitly expressed within Problem SPP, these are nonetheless captured by the cost \( c_i \) associated with this batch. Naturally, we solve Problem SPP using a column generation procedure that progressively constructs attractive patterns via an interaction between a restricted master program (RMP) based on Problem SPP and a suitable subproblem (SP), as detailed in Section 6.4.2.

### 6.4.2 Column Generation Approach

This strategy iteratively composes a set of promising patterns that achieve a partitioning of the entire set of \(n\) jobs into \(R\) sequenced batches, while minimizing the total cost of violation-induced penalties. In this context, the RMP examines some \( \hat{I} \) possible patterns, \( R \leq \hat{I} \leq I \), for selecting a set of \( R \) batches that comply with the set partitioning restrictions (6.5b). To initialize the RMP,
a consistent set of \( \hat{I} = R \) patterns is constructed. For the sake of simplicity and for independently evaluating the proposed procedure, we shall simply compose \( R \) initial batches where the \( i^{th} \) such batch consists of the jobs indexed by the set \( \{ (i - 1)\omega + 1, ..., i\omega \}, i = 1, ..., R \).

Upon solving the LP relaxation of Problem RMP, the corresponding values of the dual variables \( \pi = \bar{\pi} \) associated with the constraints (6.5b) are passed to the subproblem SP(\( \bar{\pi} \)), which conducts a pricing to generate a most promising pattern for appending to Problem RMP. We provide below some notation pertaining to Problem SP that is formulated and explained subsequently.

- \( \omega = \frac{n}{R} \): Batch size, i.e., the number of jobs to be selected to compose a batch.
- \( \ell = 1, ..., \omega \): Relative position-slots for sequencing the jobs selected for composing the batch.
- \( y_j = \begin{cases} 1 & \text{if job } j \text{ is selected in the batch under construction} \\ 0 & \text{otherwise, } \forall j. \end{cases} \)
- \( \eta_{j\ell} = \begin{cases} 1 & \text{if job } j \text{ is sequenced in position } \ell \\ 0 & \text{otherwise, } \forall j, \ell. \end{cases} \)
- \( \hat{\psi}_m^{1\pm} \): Positive or negative deviations of the actual production of model-option \( m \) over the \( \omega \) position-slots from the inflated forecast \( \gamma_m^1 F_m^1 \) in the batch under composition, \( \forall m = 1, ..., M \).
- \( \hat{\psi}_d^{2\pm} \): Positive or negative deviations of the actual production for destination \( d \) over the \( \omega \) position-slots from the inflated forecast \( \gamma_d^2 F_d^2 \) in the batch under composition, \( \forall d = 1, ..., D \).
- \( \hat{\delta}_{\ell s}^{2\pm} \): Positive or negative deviations that occur between 95% and the average material consumption efficiency values achieved at station \( s \) over position-slots \( \ell \) and \( \ell + 1 \), for \( \ell = 1, ..., \omega - 1, s = 1, ..., S \).
- \( \hat{\mu}_{\ell s}^{2\pm} \): Positive or negative deviations that occur between 95% and the average labor consumption efficiency values achieved at station \( s \) over position-slots \( \ell \) and \( \ell + 1 \), for \( \ell = 1, ..., \omega - 1, s = 1, ..., S \).
- \( \hat{\delta}_{\ell s}^{3\pm} \) and \( \hat{\mu}_{\ell s}^{3\pm} \): Similar to above, except for monitoring the three-bike efficiency violations, for \( \ell = 1, ..., \omega - 2, s = 1, ..., S \).
- \( \varphi^{+}, \varphi^{-} \): Positive or negative deviations, respectively, as occur between the maximum targeted production \( \theta^{+} \) and the actual production of the relatively more time-consuming bikes over the batch under composition.
\( \text{SP}(\bar{\pi}): \) Minimize \( \lambda_1 \left[ \sum_{\ell=1}^{\omega-1} \sum_{s=1}^{S} \hat{\delta}_{\ell s}^{2-} \right] + \lambda_2 \left[ \sum_{\ell=1}^{\omega-1} \sum_{s=1}^{S} \hat{\mu}_{\ell s}^{2-} \right] + \lambda_3 \left[ \sum_{\ell=1}^{\omega-2} \sum_{s=1}^{S} \hat{\delta}_{\ell s}^{3-} \right] + \lambda_4 \left[ \sum_{\ell=1}^{\omega-2} \sum_{s=1}^{S} \hat{\mu}_{\ell s}^{3-} \right] \\
+ \lambda_5 \left[ \sum_{m=1}^{M} \hat{\psi}_m^{1-} \right] + \lambda_6 \left[ \sum_{d=1}^{D} \hat{\psi}_d^{2-} \right] + \lambda_7 \left[ \hat{\varphi}^- \right] - \sum_{j=1}^{n} \bar{\pi}_j y_j \\ 
\text{subject to} \sum_{j=1}^{n} y_j = \omega \\
\sum_{j=1}^{n} \eta_{j\ell} = 1, \ \forall \ell \\
\sum_{\ell=1}^{\omega} \eta_{j\ell} = y_j, \ \forall j \\
100 * \left\{ \frac{\sum_{j \in J_m^1} y_j}{\omega} \right\} + \hat{\psi}_m^{1+} - \hat{\psi}_m^{1-} = \gamma_m F_m, \ \forall m \\
100 * \left\{ \frac{\sum_{j \in J_d^1} y_j}{\omega} \right\} + \hat{\psi}_d^{2+} - \hat{\psi}_d^{2-} = \gamma_d F_d, \ \forall d \\
\frac{1}{2} \left( \sum_{j=1}^{n} \alpha_j \eta_{j\ell} + \sum_{j=1}^{n} \alpha_j \eta_{j,\ell+1} \right) + \hat{\delta}_{\ell s}^{2-} - \hat{\delta}_{\ell s}^{2-} = 95, \\
\forall \ell = 1, ..., \omega - 1, \forall s \\
\frac{1}{2} \left( \sum_{j=1}^{n} \beta_j \eta_{j\ell} + \sum_{j=1}^{n} \beta_j \eta_{j,\ell+1} \right) + \hat{\mu}_{\ell s}^{2+} - \hat{\mu}_{\ell s}^{2-} = 95, \\
\forall \ell = 1, ..., \omega - 1, \forall s \\
\frac{1}{3} \left( \sum_{j=1}^{n} \alpha_j \eta_{j\ell} + \sum_{j=1}^{n} \alpha_j \eta_{j,\ell+1} + \sum_{j=1}^{n} \alpha_j \eta_{j,\ell+2} \right) + \hat{\delta}_{\ell s}^{3+} - \hat{\delta}_{\ell s}^{3-} = 95, \\
\forall \ell = 1, ..., \omega - 2, \forall s \\
\frac{1}{3} \left( \sum_{j=1}^{n} \beta_j \eta_{j\ell} + \sum_{j=1}^{n} \beta_j \eta_{j,\ell+1} + \sum_{j=1}^{n} \beta_j \eta_{j,\ell+2} \right) + \hat{\mu}_{\ell s}^{3+} - \hat{\mu}_{\ell s}^{3-} = 95, \\
\forall \ell = 1, ..., \omega - 2, \forall s \\
\sum_{j \in J^+} y_j + \varphi^+ - \varphi^- = \theta^+ \\
y, \eta \text{ binary}, (\hat{\psi}_m^{1\pm}, \hat{\psi}_d^{2\pm}, \hat{\delta}_{\ell s}^{2\pm}, \hat{\mu}_{\ell s}^{2\pm}, \hat{\delta}_{\ell s}^{3\pm}, \hat{\mu}_{\ell s}^{3\pm}, \hat{\varphi}^+) \geq 0. 

The objective function in (6.6a) minimizes the reduced cost of the variable associated with the potential batch that is generated via $\text{SP}(\bar{\pi})$ based on the achieved values of the $y$-variables. Constraint (6.6b) asserts that $\omega$ jobs are selected to compose the pattern under construction. Constraint (6.6c) ensures that any position-slot $\ell$ is assigned to exactly one job, and Constraint (6.6d) guarantees that a job is not sequenced within the position-slots pertaining to the batch under composition unless it is selected to belong to it and, in the latter case, it is assigned exactly one position-slot. Constraints (6.6e)-(6.6k) express similar restrictions as in Model MPSP2 that are related to model-option and destination forecasts, resource consumption, and the total number of time-consuming jobs selected within the batch under composition. Finally, Constraint (6.6l) imposes logical restrictions on the variables.

Our proposed column generation approach, which we shall denote by **Heuristic CG**, also integrates the *complementary column generation* (CCG) feature recommended by Ghoniem and Sherali [53]. That is, if the pattern generated by $\text{SP}(\bar{\pi})$ has a negative reduced cost, it is simultaneously complemented by constructing $R - 1$ accompanying patterns that form a full set partitioning solution in conjunction with the batch that has priced out favorably. Otherwise, if no such negative reduced-cost column exists, we can conclude that the current solution to Problem RMP solves Problem SPP. Formally, let $\bar{y}$ represent the solution that prescribes the selected set of jobs for composing the batch that has priced out favorably via Problem $\text{SP}(\bar{\pi})$, and consider the set $Z \equiv \{p : \bar{y}_p = 1\}$. We then solve Problem $\text{SP}(\bar{\pi})$ with the additional requirement that $y_p = 0, \forall p \in Z$, and generate a corresponding solution, denoted by $\bar{y}_\text{new}$. Next, the set $Z$ of prohibited job indices is augmented by setting $Z \leftarrow Z \cup \{p : \bar{y}_p^{\text{new}} = 1\}$, and the foregoing step is repeated. This is continued until the desired $R - 1$ complementary batch compositions are generated, and Problem RMP is then augmented with the entire resulting block of $R$ patterns that constitute a feasible solution to Problem SPP, so that we update $\hat{I} \leftarrow \hat{I} + R$. Once the LP phase terminates, the current RMP is solved to optimality as a 0-1 integer program, and the patterns selected via this final set partitioning problem are prescribed as the final output.

**Remark 6.4.** We also consider an alternative *sequential rounding scheme* (SRS) in lieu of the aforementioned final binary-restricted step of Heuristic CG. Upon solving Problem SPP, we consider the associated solution, $\bar{z}$, and define the set $I_1 \equiv \{i \in \hat{I} : \bar{z}_i = 1\}$. If $|I_1| \neq R$, we fix the selection of patterns for all indices in $I_1$, as well as an additional pattern that corresponds to $\max\{\bar{z}_i : i \notin I_1\}$, and then Heuristic CG is applied to the remaining jobs in concert with the SRS.
The proposed column generation approach can be judiciously adapted to provide (near-) optimal solutions to various classes of combinatorial optimization problems that can be formulated via set partitioning programs. Furthermore, observe that for applications where the patterns constructed tend to be sparse (for instance, see Sherali and Ghoniem [118]), the CCG feature may be omitted without necessarily compromising the quality of the binary-restricted solution produced at the final step of this procedure. In contrast, Ghoniem and Sherali [53] expound on the critical role the CCG feature plays for relatively dense patterns. In fact, omitting the CCG feature may produce a pool of patterns that collectively fail to qualify as a complete set partitioning solution to Problem SPP during the final binary-restricted step, notwithstanding the individual quality of many of these generated patterns as well as the tightness of the underlying LP relaxation bound. Note also that the sequential rounding scheme adopted in Heuristic CG(SRS) attempts to accomplish the generation of a similar complementary set of columns, and is thereby accelerated through the \textit{a priori} generation of sets of complementary columns.

\textbf{Remark 6.5.} We also consider another variant of Heuristic CG, denoted \textbf{Heuristic CG}. In the latter, Problem SP is partially relaxed by enforcing binary restrictions only on the $y$-variables while letting the $\eta_{j,\ell}$-variables be continuous, with $0 \leq \eta_{j,\ell} \leq 1, \forall j, \ell$. That is, the motorcycle selection variables, $y$, are binary-restricted to produce binary-valued columns, whereas the corresponding, lower-approximating resulting pattern costs, denoted $\hat{c}$, are incorporated within an adapted Problem $\hat{SPP}$. Hence, the LP relaxation of $\hat{SPP}$ can be solved using the foregoing column generation scheme to more easily produce a lower bound on SPP, and the resulting solution from Heuristic CG can be post-evaluated to yield an upper bound on SPP. Note that we could consider a weaker lower bound by relaxing all the binary variables in the subproblem. However, generating fractional-valued patterns would strongly compromise the quality of this relaxation and, in fact, preliminary computational tests suggested that the resulting lower bounds would be as weak as those produced via Models MPSP2 and MPSP2($\sigma$) (see Section 6.6.2).

\textbf{Proposition 6.1.} Consider SPP: $\min\{c^T x : Ax = e, x \geq 0\}$ (assumed feasible), and let $R$ be a known upper bound on $\sum_j x_j$ for SPP. Given any $\bar{\pi}$, suppose that the subproblem SP satisfies

$$\hat{c} \leq \{c_j - \bar{\pi} a_j : (c_j, a_j) \in S\} \quad (6.7a)$$
where \( \bar{c} \leq 0 \), \( A \equiv [(a_j)] \), and \( S \) represents the set of columns \((c_j, a_j)\) for SPP. Then

\[
LB \equiv e^T \bar{\pi} + R\bar{c} \tag{6.7b}
\]
is a valid lower bound for SPP.

Proof. Let \( \nu^* \) be the optimal value for SPP. Thus, we have by the redundancy of \( \sum_j x_j \leq R \) and duality that

\[
\nu^* = \min \{ e^T x : Ax = e, \sum_j x_j \leq R, x \geq 0 \}
\]

\[
= \max \{ e^T \pi + \pi_0 R : \pi^T a_j + \pi_0 \leq c_j, \forall j, \pi_0 \leq 0 \} \tag{6.7c}
\]

But from (6.7a), we have that \( c_j - \bar{\pi}^T a_j \geq \bar{c}, \forall j \), or that \((\pi, \pi_0) \equiv (\bar{\pi}, \bar{c})\) is feasible to the (maximization) dual in (6.7c), which establishes (6.7b).

Proposition 1 can be used in the context of two algorithmic strategies. First, for some RMP, given a complementary dual solution \( \bar{\pi} \), we know that the minimum value in (6.7a), denoted \( \nu(\text{SP}) \), is nonpositive. Hence, if SP is solved exactly, we can use \( \bar{c} = \nu(\text{SP}) \) and compute a lower bound on SPP via (6.7b), terminating the column generation algorithm when this lower bound and the best available upper bound computed for SPP are within certain pre-specified tolerance. Second, this schema can be implemented in case SP is optimized only to some tolerance or solved as a relaxation to compute a lower bound \( \bar{c} \leq \nu(\text{SP}) \). Naturally, in such a context, any column passed to SPP must belong to the set \( S \) described in Proposition 6.1. Also, note that in the context of Problem SPP that \( \sum_j x_j = R \). We shall denote \textbf{Heuristic CG}(LB) and \textbf{Heuristic CG}(LB) as variants of the respective column generation procedures that result from incorporating the aforementioned LB-based strategy.

### 6.5 Subgradient Optimization Algorithmic Approach

The subgradient-based optimization algorithmic scheme proposed in this section is largely motivated by the need of the motorcycle manufacturing company that presented this problem to have an effective stand-alone solution strategy that does not resort to commercial MIP solvers such as
Cplex (due to cost and software-maintenance considerations), other than using an available routine for solving linear assignment problems. To describe such an algorithmic approach for solving either of the models MPSP1 or MPSP2, let us re-write this problem in a more compact convenient form as follows, denoted generically as MPSP.

**MPSP:** Minimize $\sum_{q=1}^{Q} c_q g_q$ \hspace{1cm} (6.8a)

subject to $h_q^T x + g_q^+ - g_q^- = h_{0q}, \forall q = 1, ..., Q$ \hspace{1cm} (6.8b)

$x \in X_a \equiv \{ x \text{ binary: (6.1b) and (6.1c)} \}$ \hspace{1cm} (6.8c)

$(g_q^+, \forall q) \geq 0.$ \hspace{1cm} (6.8d)

Here,

$$x \equiv (x_{11}, x_{12}, ..., x_{1n}, x_{21}, x_{22}, ..., x_{2n}, ..., x_{n1}, x_{n2}, ..., x_{nn})^T,$$ \hspace{1cm} (6.8e)

and in the context of Model MPSP1 (and likewise for MPSP2), (6.8b) represents the set of constraints (6.1d)-(6.1j), (6.8c) represents the standard assignment constraints (6.1b)-(6.1c) along with $x$ binary, and (6.8a) accordingly represents the objective function in generic form. Furthermore, we can directly accommodate each restriction (6.8b) within the objective function (6.8a) itself equivalently via the corresponding term:

$$c_q \max\{0, h_q^T x - h_{0q}\}, \forall q = 1, ..., Q.$$ \hspace{1cm} (6.9)

Using (6.8) and (6.9), we can view Problem MPSP2 as follows.

**MPSP-ND:** Minimize $\sum_{q=1}^{Q} c_q \max\{0, h_q^T x - h_{0q}\}$ \hspace{1cm} (6.10a)

subject to $x \in X_a.$ \hspace{1cm} (6.10b)

To further re-cast Model MPSP-ND into a form amenable for solution, let us retain (6.1b) from $X_a$ within the constraint set and accommodate (6.1c) within the objective function (6.10a) via a penalty term of the form:
\[
\sum_{k=1}^{n} \sigma_k \max \{ \sum_{j=1}^{n} x_{jk} - 1, 1 - \sum_{j=1}^{n} x_{jk} \},
\]  
(6.11)

where \(\sigma_k > 0, \forall k\), are suitable penalty multipliers. Let \(e_k \in R^n\) be the \(k^{th}\) unit (column) vector, \(\forall k\), and let

\[
E_k \equiv [e_k^T, \ldots, e_k^T]^T.
\]  
(6.12)

Then, (6.11) can be stated in a convenient vector form as

\[
\sum_{k=1}^{n} \sigma_k \max \{ E_k^T x - 1, 1 - E_k^T x \}.
\]  
(6.13)

Furthermore, denoting

\[
X = \{ x : \sum_{k=1}^{n} x_{jk} = 1, \forall j = 1, \ldots, n, x \geq 0 \},
\]  
(6.14)

we can use (6.13) and (6.14) to re-write Problem (6.10) as follows.

\textbf{MPSP-ND:} Minimize \(f(x) : x \in X, x \) binary,

\[f(x) \equiv \sum_{q=1}^{Q} c_q \max \{0, h_q^T x - h_{0q}\} + \sum_{k=1}^{n} \sigma_k \max \{E_k^T x - 1, 1 - E_k^T x\}.
\]  
(6.15b)

The continuous relaxation of Problem MPSP-ND, denoted \textbf{MPSP-ND} (obtained by omitting the binary restrictions in (6.15a)), is a conveniently formulated non-differentiable optimization problem (see Shor [130]) that we will solve via a suitable convergent conjugate subgradient method proposed by Sherali and Ulular [128], along with several specialized modifications. (See also Barahona and Anbil [8], Sherali et al. [113], Sherali and Lim [119], and Lim and Sherali [76].) Below, we describe some key elements of this algorithmic approach and then present a flow-chart for the basic procedure.
**Initial Solution** $x^1 \in X_a$. To initialize the method, we will utilize the starting solution $x^1$ as an optimal solution to the following linear assignment problem, fashioned based on (6.10a)-(6.10b).

\[
\text{Minimize } \{ \sum_{q=1}^{Q} c_q h_q^T x : x \in X_a \}. \tag{6.16}
\]

Note that 0 is a lower bound for both Models MPSP-ND and MPSP-ND and, actually, a targeted optimal value. Hence, if $f(x^1) = 0$ (or it is sufficiently small, say, $f(x^1) \leq \varepsilon_0$, for some tolerance $\varepsilon_0 > 0$ ), then we can terminate with $x^1$ as an optimal (or a near-optimal) solution to Problem MPSP-ND.

**Subgradient $\xi^t$ of $f(x)$ at $x = x^t$ (Denoted $\xi^t \in \partial f(x^t)$) at Iteration $t$**

From Bazaraa et al. [13], given any iterate $x^t \in X$, we can compute a subgradient $\xi^t$ of $f(x)$ at $x = x^t$ as follows:

\[
\xi^t = \sum_{q=1}^{Q} \xi^t_{1q} + \sum_{k=1}^{n} \xi^t_{2k} \tag{6.17a}
\]

where

\[
\xi^t_{1q} = \begin{cases} 
0 & \text{if } h_q^T x^t \leq h_0q \\
c_q h_q & \text{otherwise}, \quad \forall q = 1,...,Q
\end{cases} \tag{6.17b}
\]

\[
\xi^t_{2k} = \begin{cases} 
0 & \text{if } \sum_{j=1}^{n} x^t_{jk} = 1 \\
\sigma_k E_k & \text{if } \sum_{j=1}^{n} x^t_{jk} > 1 \\
-\sigma_k E_k & \text{if } \sum_{j=1}^{n} x^t_{jk} < 1, \quad \forall k = 1,...,n.
\end{cases} \tag{6.17c}
\]

Whenever $\xi^t = 0$ (or $\|\xi^t\| < \varepsilon$, for some tolerance $\varepsilon > 0$), we can terminate with $x^t$ being an optimal or a near-optimal solution to Problem MPSP-ND.

**Projection Routine $P_X[\hat{x}]$:**

In the process of the algorithm, given a tentative solution $\hat{x}$ at any iteration, we will need to determine its projection (closest point) in $X$ given by
\[ x = P_X[\hat{x}], \text{ where } x \text{ solves the problem:} \]  
\[ \text{Minimize } \{ \| x - \hat{x} \|^2 : x \in X \}. \]  
\( \text{(6.18a)} \)

This projection problem (6.18) is readily solved in polynomial time via the variable-dimension routine described in Sherali and Shetty [121] (see also Bitran and Hax [19]).

**Purification of \( x^* \in X \) to an Extreme Point \( x^{**} \in X_a \).**

Given a (near-) optimal solution \( x^* \in X \) to Problem MPSP-ND, we now describe a strategy for judiciously *rounding* (or *purifying*) this to a desired solution \( x^{**} \in X_a \) for Problem MPSP. Ideally, we would like such a “purified” solution \( x^{**} \in X_a \) to be closest to \( x^* \) in the sense that \( x^{**} = P_{X_a}[x^*]. \) However, in lieu of solving this relatively more difficult projection problem, we shall adopt the following simpler routine.

Observing that \( x^* \in X \) is a 0-1 vector, we could attempt to minimize the number of switches of 0 and 1 values in deriving \( x^{**} \) from \( x^* \). Hence, we can solve

\[ \text{Minimize } \left\{ \sum_{(j,k):x_{jk}^* = 0} x_{jk} - \sum_{(j,k):x_{jk}^* = 1} x_{jk} : x \in X_a \right\}. \]  
\( \text{(6.19a)} \)

To address the many possible alternative optimal solutions to Problem (6.19a), and also to preserve the quality of the solution \( x^* \) with respect to the function \( f(x) \), we can also minimize the first-order approximation of \( f(x) \) at \( x = x^* \) as given by \( f(x^*) + (x - x^*)^T \xi^* \), where \( \xi^* \) is a subgradient of \( f \) at \( x = x^* \). That is, effectively, we would alternatively like to solve:

\[ \text{Minimize } \{ \xi^T x : x \in X_a \}. \]  
\( \text{(6.19b)} \)

Accommodating both the objective functions (6.19a) and (6.19b) by using a weighting parameter \( \rho, 0 \leq \rho \leq 1 \), we will solve the following problem to determine an optimal solution \( x^{**} \) based on \( x^* \):

\[ \text{Minimize } \left\{ \rho \left[ \sum_{(j,k):x_{jk}^* = 0} x_{jk} - \sum_{(j,k):x_{jk}^* = 1} x_{jk} \right] + (1 - \rho)[\xi^T x] : x \in X_a \right\}. \]  
\( \text{(6.20)} \)
We shall experiment with different values of $\rho \in [0, 1]$ in our computations.

**Problem Reset**

Based on monitoring the progress of the proposed algorithm for solving Problem $\text{MPSP-ND}$, we shall periodically reset this problem by adjusting the penalty parameters $(\sigma_k, \forall k)$. This would effectively change the objective function $f(\cdot)$, and we will continue optimizing this revised objective function starting from the current incumbent solution. The precise criteria for triggering such a *problem reset* strategy will be investigated computationally.

**Algorithmic Parameters**

The following notation and parameters pertain to the specific details of the proposed deflected subgradient-based algorithm.

- $t$ = iteration counter.
- $t_{\max}$ = maximum number of iterations (typically, $t_{\max} = 2000 - 3000$).
- $\varepsilon$ = tolerance for the norm of subgradients and directions ($\varepsilon = 0.01$).
- $\varepsilon_0$ = optimality tolerance with respect to (6.16) ($\varepsilon_0 = 0.01$ or some data-based $\%$ value).
- $\Delta$ = counter for consecutive failures to improve the objective value.
- $\Delta_{\max}$ = maximum value of $\Delta$ before resetting to the incumbent solution ($\Delta_{\max} = 10$).
- $\beta_0$ = step-length parameter (initialized at $\beta_0 = 2$).
- $x^*$ = incumbent solution.
- $v^*$ = $f(x^*)$.
- $x^t$ = current solution (iterate) at iteration $t$.
- $\xi^t$ = subgradient of $f$ at $x = x^t$.
- $d^t$ = direction of motion at iteration $t$ (adopted from the solution $x^t$).
- $\text{RESET} = 1$ if we have reset the iterates to the incumbent solution and 0, otherwise. (Note that this is an *algorithmic* reset parameter, and not related to the *problem reset* issue discussed in the foregoing section.)
Figure 6.1 presents a flow-chart for the proposed algorithm to solve Problem MPSP-ND, based on the conjugate subgradient method of Sherali and Ulular [128]. In our computations, we shall also experiment with other subgradient deflection strategies as in the Volume Algorithm of Barahona and Anbil [8] and those studied in Lim and Sherali [76] and Subramanian and Sherali [132]. These developed solution procedures will be compared against the direct solution of Problems MPSP1 and MPSP2 using CPLEX 10.1.

6.6 Computational Experience

We empirically evaluate in this section the relative effectiveness of the different proposed formulations and the associated optimization strategies for solving a set of realistic problem instances. The proposed mathematical programs (MPSP1, MPSP2, and MPSP2(σ)) as well as Heuristic CG were coded in AMPL and solved using CPLEX 10.1 on a Dell Precision 650 workstation having a Xeon(TM) CPU 2.40 GHz processor and 1.50 GB of RAM. The proposed subgradient-based algorithmic approach was coded in AMPL. For the latter, any open-source linear assignment or network-flow code can be used to initialize the algorithm by solving Problem (6.16) and to solve the purification problem at termination. Alternatively, these assignment problems can be solved using the RELAX and RELAX-IV codes developed by Bertsekas and Tseng ([17], [18]).

The test-bed is composed of 10 randomly generated realistic problem instances, each being characterized by the number $n$ of jobs involved. In our test-bed, $n$ varies between 10 and 50 with an incremental step of 10 jobs. For each test problem, the batch size $\omega$ equals 5, and every job/bike was assigned a model-option from the set \{1,...,18\} and a destination from the set \{1, 2, 3\} using uniform distributions. The $\alpha$- and $\beta$-values were generated over the interval [93,97] using a uniform distribution. The forecast percentages, $F^1$ and $F^2$, were randomly generated so that the sum of each of the model-option and destination forecasts equals 100%. Finally, the target value $\theta^+$ was generated over the interval [2,5] using a uniform distribution.

Throughout this section, for any discrete Model “M”, we shall denote its continuous relaxation by $\overline{M}$.

6.6.1 Mathematical Programming Formulations

Table 6.1 reports the LP relaxation values of the different proposed models and the associated
Initialize with \( x^1 \in X_a \). If \( f(x^1) \leq \epsilon_0 \), stop and prescribe the solution \( x^1 \). Else, set \( x^* = x^t, \nu^* = f(x^t), t = 1, \) RESET = 1, \( \beta_0 = 2 \), and \( \Delta = 0 \).

1. Find \( \xi^t \in \partial f(x^t) \)
2. If \( \|\xi^t\| < \epsilon^2 \)
   - Stop; set \( x^* = x^t \) and purify \( x^* \) to \( x^{**} \in X_a \)
3. Let \( d^t = \begin{cases} -\xi^t, & \text{if} \ \text{RESET} = 1 \\ -\xi^t + \frac{\|\xi^t\|}{\|d^{t-1}\|}d^{t-1}, & \text{if} \ \text{RESET} = 0 \end{cases} \)
4. If \( \|d^t\| < \epsilon^2 \)
   - \( \text{RESET} = 1 \)
   - Put \( d^t = -\xi^t \)
   - \( \text{RESET} = 1 \)
5. Let \( x^{t+1} = P_X[x^t + \frac{\beta_0 f(x^t)}{\|d^t\|^2}d^t] \), and compute \( f(x^{t+1}) \)
6. Increment \( t \leftarrow t + 1, \Delta \leftarrow \Delta + 1 \)
7. If \( f(x^t) < \nu^* \)
   - Stop; purify \( x^* \) to \( x^{**} \in X_a \)
   - \( \nu^* < \epsilon_0 \)
8. If \( \Delta \geq \Delta_{\text{max}} \)
   - \( \beta_0 \leftarrow \max\{\beta_0, 10^{-3}\} \)
9. If \( t \geq t_{\text{max}} \)
   - Reset Problem
10. Reset parameters, put \( \text{RESET} = 1, x^t = x^*, \) recompute \( \nu^* \equiv f(x^t) \) with respect to revised \( f \).

**Figure 6.1:** Flow-chart for the proposed deflected subgradient optimization algorithm
% optimality gaps with respect to the best available upper bound that is obtained via Heuristic CG (see Table 6.4) for each test instance. Although the continuous relaxations of Models MPSP2 and MPSP2(σ) exhibited an evident computational advantage with respect to CPU times, these formulations are plagued by the weakness of their LP bounds, which lie within an optimality gap of 88.56% at an average. Also, observe that the LP bounds of Model MPSP2(σ) equal those of MPSP2, because symmetry-defeating constraints typically eliminate certain subsets of symmetric feasible solutions, but do not necessarily strengthen the LP relaxation of the formulation under investigation.

Heuristic CG produced very tight lower bounds via SPP within an optimality gap of 0.14% for Instances 1-8, and solved four of these instances to exact optimality. Observing the substantial increase in the computational effort incurred for larger problem instances, we have imposed a computational time limit of τ = 240 seconds on Problem SP, and have also required the LP phase to terminate when its cumulative CPU effort exceeds an overall time limit of Ω = 10800 seconds (i.e., 3 hours), as measured at the end of any LP phase iteration of Heuristic CG. The instances for which these computational time limits were triggered are signaled by the symbol † in Table 6.1. Also, note that because of these computational time limits, the objective value produced by the LP phase of Heuristic CG may not qualify as a valid lower bound on Problem SPP, and the reported % optimality gap is derived for this tentative lower bound and the best known upper bound.

Next, we considered Heuristic CG( ˆSP) as described in Remark 6.5. Problem ˆSPP exhibited a roughly consistent optimality gap of 15.64% at an average for Instances 1-12, and achieved an average savings in computational effort of 81.67% over Problem SPP for Instances 1-8. As discussed in Section 6.6.2, Heuristic CG( ˆSP) offers an attractive tradeoff between the quality of its LP relaxation (via the generated columns) and the ensuing computational effort and can, therefore, be embedded in different variants of Heuristic CG that recover binary values for the η-variables upon solving the partially relaxed Problem ˆSP.

Table 6.2 presents the computational results pertaining to solving the models to optimality for several small-sized instances. The optimal objective values of Models MPSP2 and MPSP2(σ) provide lower bounds on the objective value of Model MPSP1 because Assumption A holds true for our test cases. In addition, we post-evaluate the solution produced by Models MPSP2 and MPSP2(σ) via Model MPSP1, and deduce an upper bound on the latter. Hence, Models MPSP2 and MPSP2(σ) can be utilized to bracket Model MPSP1 between tight lower and upper bounds.
The results reported in Table 6.2 reveal the limitations of the proposed mixed-integer programming formulations MPSP1, MPSP2, and MPSP2(\(\sigma\)), and reflect the computational burden that is characteristically induced by composite objective functions that aim to balance violation-incurred penalties. The simple assignment structure that lies at the heart of these formulations is, in fact, obscured because of this balancing effort, and the popular mathematical programming software package CPLEX 10.1 fails to identify a feasible solution to these formulations within three hours even for certain small-sized test instances (which is symbolized by “N/A” in this section). In contrast, the constructive and dynamic nature of Heuristic CG offers an attractive alternative to these formulations, and ensures that a feasible solution to the problem is generated at each LP phase iteration.

Table 6.3 presents some preliminary computational results pertaining to the objective perturbation strategy described in Remark 6.3. Considering Models MPSP2(\(\phi\)) and MPSP2(\(\sigma, \phi\)), we experimented with various values of \(\varepsilon\) that vary between 0 and 1, and the objective value reported in Table 6.3 is that of (6.3a) (after eliminating the contribution of the perturbation term). The objective perturbation-based formulation MPSP2(\(\phi\)) offers a more attractive alternative to Models MPSP2(\(\sigma\)) and MPSP2(\(\sigma, \phi\)). We also report in Table 6.3 the % deviation of the objective value (when available) produced by MPSP2(\(\phi\)) and MPSP(\(\sigma, \phi\)) from the best upper bound reported in Table 6.4 using Heuristic CG and its variants. These results demonstrate the relative superiority of the column generation-based procedures.

6.6.2 Performance of the Column Generation Heuristic CG

Table 6.4 reports the results obtained for Heuristic CG and its variants, and reflects the computational burden associated with an increase in the size of the problem. For Heuristic CG(\(LB\)) and Heuristic \(\hat{CG}(LB)\), the LP phase was terminated when the RMP was solved with a 1% optimality tolerance using the lower bound, \(LB\), introduced in Proposition 6.1. As mentioned in Section 6.6.1, we have resorted to imposing computational time limits \((\tau, \Omega) = (240, 10800)\) seconds on Problem SP and the LP phase of Heuristic CG for larger problem instances. Runs where no such time limits were imposed are characterized by time bounds that equal \((\infty, \infty)\). Heuristic CG produced optimal solutions for several small and moderately-sized instances, and near-optimal solutions within an optimality gap of 0.14% for Instances 1-8 (using \(\omega = 5\)), and an optimality gap within 4.09% for all test instances reported in Table 6.4. For Instances 1-8, the % optimality gap reported for the different column generation variants relate the MIP solution produced to the lower bound obtained.
### Table 6.1: LP relaxations of the different proposed formulations

<table>
<thead>
<tr>
<th>Instance, ((n, \omega))</th>
<th>Metrics</th>
<th>MPSP1</th>
<th>MPSP2</th>
<th>MPSP2((\sigma))</th>
<th>SPP</th>
<th>SPP†</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1, (10,5))</td>
<td>LP</td>
<td>2.20</td>
<td>1.65</td>
<td>1.65</td>
<td>17.90</td>
<td>20.21</td>
</tr>
<tr>
<td></td>
<td>CPU (s)</td>
<td>0.14</td>
<td>0.10</td>
<td>0.07</td>
<td>12.15</td>
<td>42.19</td>
</tr>
<tr>
<td></td>
<td>% gap</td>
<td>-</td>
<td>91.84</td>
<td>91.84</td>
<td>11.43</td>
<td>0</td>
</tr>
<tr>
<td>(2, (10,5))</td>
<td>LP</td>
<td>7.30</td>
<td>5.87</td>
<td>5.87</td>
<td>15.77</td>
<td>18.84</td>
</tr>
<tr>
<td></td>
<td>CPU (s)</td>
<td>0.15</td>
<td>0.07</td>
<td>0.09</td>
<td>7.04</td>
<td>28.18</td>
</tr>
<tr>
<td></td>
<td>% gap</td>
<td>-</td>
<td>68.84</td>
<td>68.84</td>
<td>16.3</td>
<td>0</td>
</tr>
<tr>
<td>(3, (20,5))</td>
<td>LP</td>
<td>12.61</td>
<td>8.04</td>
<td>8.04</td>
<td>39.44</td>
<td>45.94</td>
</tr>
<tr>
<td></td>
<td>CPU (s)</td>
<td>5.27</td>
<td>2.51</td>
<td>2.55</td>
<td>183.43</td>
<td>1562.21</td>
</tr>
<tr>
<td></td>
<td>% gap</td>
<td>-</td>
<td>82.5</td>
<td>82.5</td>
<td>14.15</td>
<td>0</td>
</tr>
<tr>
<td>(4, (20,5))</td>
<td>LP</td>
<td>5.48</td>
<td>2.97</td>
<td>2.97</td>
<td>21.15</td>
<td>25.31</td>
</tr>
<tr>
<td></td>
<td>CPU (s)</td>
<td>6.15</td>
<td>2.13</td>
<td>2.02</td>
<td>369.67</td>
<td>1071.92</td>
</tr>
<tr>
<td></td>
<td>% gap</td>
<td>-</td>
<td>88.28</td>
<td>88.28</td>
<td>16.54</td>
<td>0.12</td>
</tr>
<tr>
<td>(5, (30,5))</td>
<td>LP</td>
<td>14.75</td>
<td>9.03</td>
<td>9.03</td>
<td>70.15</td>
<td>81.43</td>
</tr>
<tr>
<td></td>
<td>CPU (s)</td>
<td>37.18</td>
<td>26.89</td>
<td>31.86</td>
<td>4175.88</td>
<td>20209.30</td>
</tr>
<tr>
<td></td>
<td>% gap</td>
<td>-</td>
<td>88.91</td>
<td>88.91</td>
<td>13.85</td>
<td>0</td>
</tr>
<tr>
<td>(6, (30,5))</td>
<td>LP</td>
<td>11.42</td>
<td>6.35</td>
<td>6.35</td>
<td>54.32</td>
<td>67.59</td>
</tr>
<tr>
<td></td>
<td>CPU (s)</td>
<td>34.05</td>
<td>13.72</td>
<td>16.17</td>
<td>3443.48</td>
<td>20994.10</td>
</tr>
<tr>
<td></td>
<td>% gap</td>
<td>-</td>
<td>90.62</td>
<td>90.62</td>
<td>19.74</td>
<td>0.13</td>
</tr>
<tr>
<td>(7, (40,5))</td>
<td>LP</td>
<td>14.52</td>
<td>8.39</td>
<td>8.39</td>
<td>67.55</td>
<td>80.76</td>
</tr>
<tr>
<td></td>
<td>CPU (s)</td>
<td>47.79</td>
<td>62.02</td>
<td>91.72</td>
<td>11167.70</td>
<td>62639.40</td>
</tr>
<tr>
<td></td>
<td>% gap</td>
<td>-</td>
<td>89.63</td>
<td>89.63</td>
<td>16.47</td>
<td>0.14</td>
</tr>
<tr>
<td>(8, (40,5))</td>
<td>LP</td>
<td>19.42</td>
<td>10.60</td>
<td>10.60</td>
<td>79.10</td>
<td>92.99</td>
</tr>
<tr>
<td></td>
<td>CPU (s)</td>
<td>223.44</td>
<td>73.53</td>
<td>56.09</td>
<td>10921.80</td>
<td>59594.01</td>
</tr>
<tr>
<td></td>
<td>% gap</td>
<td>-</td>
<td>88.61</td>
<td>88.61</td>
<td>15.08</td>
<td>0.08</td>
</tr>
<tr>
<td>(9, (50,5))</td>
<td>LP</td>
<td>9.59</td>
<td>4.10</td>
<td>4.10</td>
<td>96.25</td>
<td>112.33†</td>
</tr>
<tr>
<td></td>
<td>CPU (s)</td>
<td>215.7</td>
<td>117.89</td>
<td>143.07</td>
<td>12192.10</td>
<td>12137.50†</td>
</tr>
<tr>
<td></td>
<td>% gap</td>
<td>-</td>
<td>96.44</td>
<td>96.44</td>
<td>16.37</td>
<td>2.4</td>
</tr>
<tr>
<td>(10, (50,5))</td>
<td>LP</td>
<td>23.86</td>
<td>10.92</td>
<td>10.92</td>
<td>116.95</td>
<td>129.93†</td>
</tr>
<tr>
<td></td>
<td>CPU (s)</td>
<td>223.44</td>
<td>205.79</td>
<td>274.38</td>
<td>12813.20</td>
<td>11561.70†</td>
</tr>
<tr>
<td></td>
<td>% gap</td>
<td>-</td>
<td>91.75</td>
<td>91.75</td>
<td>11.64</td>
<td>1.83</td>
</tr>
</tbody>
</table>

### Table 6.2: MIP performance of Models MPSP1, MPSP2, MPSP(\(\sigma\)), and Heuristic CG

<table>
<thead>
<tr>
<th>Instance, ((n, \omega))</th>
<th>Metrics</th>
<th>MPSP1</th>
<th>MPSP2</th>
<th>MPSP2((\sigma))</th>
<th>Heuristic CG</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1, (10,5))</td>
<td>MIP</td>
<td>21.46</td>
<td>20.21</td>
<td>20.21</td>
<td>20.21</td>
</tr>
<tr>
<td></td>
<td>CPU (s)</td>
<td>326.66</td>
<td>78.08</td>
<td>81.51</td>
<td>42.19</td>
</tr>
<tr>
<td></td>
<td>Post-evaluation</td>
<td>-</td>
<td>21.46</td>
<td>21.46</td>
<td>-</td>
</tr>
<tr>
<td>(2, (10,5))</td>
<td>MIP</td>
<td>20.87</td>
<td>18.84</td>
<td>18.84</td>
<td>18.84</td>
</tr>
<tr>
<td></td>
<td>CPU (s)</td>
<td>111.66</td>
<td>11.53</td>
<td>17.51</td>
<td>28.18</td>
</tr>
<tr>
<td></td>
<td>Post-evaluation</td>
<td>-</td>
<td>22.27</td>
<td>22.22</td>
<td>-</td>
</tr>
<tr>
<td>(3, (20,5))</td>
<td>MIP</td>
<td>54.57</td>
<td>47.21</td>
<td>47.21</td>
<td>45.94</td>
</tr>
<tr>
<td></td>
<td>CPU (s)</td>
<td>10800†</td>
<td>10800†</td>
<td>10800†</td>
<td>1562.21</td>
</tr>
<tr>
<td></td>
<td>Post-evaluation</td>
<td>-</td>
<td>55.58</td>
<td>55.58</td>
<td>-</td>
</tr>
<tr>
<td>(4, (20,5))</td>
<td>MIP</td>
<td>32.31</td>
<td>26.45</td>
<td>26.45</td>
<td>25.34</td>
</tr>
<tr>
<td></td>
<td>CPU (s)</td>
<td>10727.3</td>
<td>10800†</td>
<td>10800†</td>
<td>1071.97</td>
</tr>
<tr>
<td></td>
<td>Post-evaluation</td>
<td>-</td>
<td>33.69</td>
<td>33.69</td>
<td>-</td>
</tr>
<tr>
<td>(5, (30,5))</td>
<td>MIP</td>
<td>113.24</td>
<td>93.65</td>
<td>93.65</td>
<td>81.43</td>
</tr>
<tr>
<td></td>
<td>CPU (s)</td>
<td>10800.00†</td>
<td>10800.00†</td>
<td>10800.00†</td>
<td>10666.60†</td>
</tr>
<tr>
<td></td>
<td>Post-evaluation</td>
<td>-</td>
<td>109.45</td>
<td>109.45</td>
<td>-</td>
</tr>
<tr>
<td>(6, (30,5))</td>
<td>MIP</td>
<td>105.04</td>
<td>77.77</td>
<td>N/A</td>
<td>68.97</td>
</tr>
<tr>
<td></td>
<td>CPU (s)</td>
<td>10800.00†</td>
<td>10800.00†</td>
<td>10800.00†</td>
<td>11369.47†</td>
</tr>
<tr>
<td></td>
<td>Post-evaluation</td>
<td>-</td>
<td>96.18</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Instance, ((n, \omega))</td>
<td>Metrics</td>
<td>(\varepsilon = 1)</td>
<td>(\varepsilon = 0.5)</td>
<td>(\varepsilon = 0.1)</td>
<td>(\varepsilon = 0)</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>1, ((10,5))</td>
<td>Objective</td>
<td>23.06, 21.60</td>
<td>23.06, 21.60</td>
<td>20.89, 20.77</td>
<td>20.21, 20.21</td>
</tr>
<tr>
<td></td>
<td>CPU (s)</td>
<td>0.07, 0.50</td>
<td>0.12, 0.78</td>
<td>3.65, 12.84</td>
<td>98.24, 81.51</td>
</tr>
<tr>
<td></td>
<td>% deviation</td>
<td>12.35, 6.44</td>
<td>12.35, 6.44</td>
<td>3.25, 2.70</td>
<td>0, 0</td>
</tr>
<tr>
<td>2, ((10,5))</td>
<td>Objective</td>
<td>22.87, 22.13</td>
<td>22.87, 20.88</td>
<td>19.23, 20.37</td>
<td>18.84, 18.84</td>
</tr>
<tr>
<td></td>
<td>CPU (s)</td>
<td>0.06, 0.71</td>
<td>0.17, 1.56</td>
<td>5.15, 19.54</td>
<td>11.06, 16.61</td>
</tr>
<tr>
<td></td>
<td>% deviation</td>
<td>17.62, 14.66</td>
<td>17.62, 9.77</td>
<td>2.02, 7.51</td>
<td>0, 0</td>
</tr>
<tr>
<td>3, ((20,5))</td>
<td>Objective</td>
<td>58.43, 51.07</td>
<td>57.73, 51.07</td>
<td>50.46, 51.07</td>
<td>47.24, 47.28</td>
</tr>
<tr>
<td></td>
<td>CPU (s)</td>
<td>0.31, 10800</td>
<td>1.39, 10800</td>
<td>9580.74, 10800</td>
<td>10800, 10800</td>
</tr>
<tr>
<td></td>
<td>% deviation</td>
<td>27.19, 11.17</td>
<td>25.66, 11.17</td>
<td>9.84, 11.17</td>
<td>2.83, 2.92</td>
</tr>
<tr>
<td>4, ((20,5))</td>
<td>Objective</td>
<td>38.58, 32.05</td>
<td>36.62, 32.05</td>
<td>28.81, 30.36</td>
<td>25.95, 26.66</td>
</tr>
<tr>
<td></td>
<td>CPU (s)</td>
<td>0.36, 10800</td>
<td>0.78, 10800</td>
<td>176.96, 10800</td>
<td>10800, 10800</td>
</tr>
<tr>
<td></td>
<td>% deviation</td>
<td>52.25, 26.48</td>
<td>44.51, 26.48</td>
<td>13.69, 19.81</td>
<td>2.41, 5.21</td>
</tr>
<tr>
<td>5, ((30,5))</td>
<td>Objective</td>
<td>100.22, 104.09</td>
<td>97.92, 100.89</td>
<td>88.74, 96.4</td>
<td>88.74, 96.4</td>
</tr>
<tr>
<td></td>
<td>CPU (s)</td>
<td>2.7, 10800</td>
<td>18.2, 10800</td>
<td>10800, 10800</td>
<td>10800, 10800</td>
</tr>
<tr>
<td></td>
<td>% deviation</td>
<td>23.08, 27.83</td>
<td>20.25, 23.9</td>
<td>8.98, 18.38</td>
<td>8.98, 18.38</td>
</tr>
<tr>
<td>6, ((30,5))</td>
<td>Objective</td>
<td>91.4, 89.28</td>
<td>86.21, 89.28</td>
<td>75.85, 89.28</td>
<td>75.85, 89.28</td>
</tr>
<tr>
<td></td>
<td>CPU (s)</td>
<td>2.7, 10800</td>
<td>18.2, 10800</td>
<td>10800, 10800</td>
<td>10800, 10800</td>
</tr>
<tr>
<td></td>
<td>% deviation</td>
<td>35.05, 31.91</td>
<td>31.78, 31.91</td>
<td>12.07, 31.91</td>
<td>12.07, 31.91</td>
</tr>
<tr>
<td>7, ((40,5))</td>
<td>Objective</td>
<td>107.43, N/A</td>
<td>102.30, N/A</td>
<td>102.30, N/A</td>
<td>102.30, N/A</td>
</tr>
<tr>
<td></td>
<td>CPU (s)</td>
<td>3.6, 10800</td>
<td>36.78, 10800</td>
<td>10800, 10800</td>
<td>10800, 10800</td>
</tr>
<tr>
<td></td>
<td>% deviation</td>
<td>32.84, N/A</td>
<td>26.5, N/A</td>
<td>26.5, N/A</td>
<td>26.5, N/A</td>
</tr>
<tr>
<td>8, ((40,5))</td>
<td>Objective</td>
<td>125.46, N/A</td>
<td>121.63, N/A</td>
<td>110.08, N/A</td>
<td>110.08, N/A</td>
</tr>
<tr>
<td></td>
<td>CPU (s)</td>
<td>4.2, 10800</td>
<td>23.7, 10800</td>
<td>10800, 10800</td>
<td>10800, 10800</td>
</tr>
<tr>
<td></td>
<td>% deviation</td>
<td>34.82, N/A</td>
<td>30.7, N/A</td>
<td>18.29, N/A</td>
<td>18.29, N/A</td>
</tr>
<tr>
<td>9, ((50,5))</td>
<td>Objective</td>
<td>140.12, N/A</td>
<td>133.20, N/A</td>
<td>130.54, N/A</td>
<td>130.54, N/A</td>
</tr>
<tr>
<td></td>
<td>CPU (s)</td>
<td>7.2, 10800</td>
<td>343.1, 10800</td>
<td>10800, 10800</td>
<td>10800, 10800</td>
</tr>
<tr>
<td></td>
<td>% deviation</td>
<td>24.2, N/A</td>
<td>18.06, N/A</td>
<td>15.71, N/A</td>
<td>15.45, N/A</td>
</tr>
</tbody>
</table>

Table 6.3: Objective perturbation strategy for Models MPSP2(\(\phi\)) and MPSP2(\(\sigma, \phi\))
in the LP phase of Heuristic CG (using \((\tau, \Omega) = (\infty, \infty)\)). For Instances 9 and 10, we have used the lower bound \(LB\) introduced in Proposition 6.1 to compute the \% optimality gap. Table 6.4 suggests that all the proposed column generation-based strategies exhibit only minor differences from a performance perspective, and tend to offer a good tradeoff between the quality of the solution produced and the accompanying computational effort. Unless Heuristic CG is solved without imposing time limits, Heuristic CG\((LB)\) offers the second best strategy, and has the advantage of targeting a specified \% optimality tolerance within the LP phase. Figure 6.2 displays the decrease in the objective value of Problem RMP and the growth in the computational effort as a function of the LP phase iterations, and reveals that it becomes computationally challenging to generate patterns that induce possible marginal improvements in Problem RMP as the LP phase iterations progress. Recalling that each LP phase iteration comprises solving Problem SP \(R\) times based on the CCG feature discussed in Section 6.4.2, Figure 6.2 provides the insight that the number of batches (out of \(R\)) where the computational effort attains the time limit \(\tau\) increases as the LP phase iterations progress, and reflects the problematic characteristic of slow convergence experienced by the simplex-based column generation approaches.

Figure 6.2: Convergence of Heuristic CG for instance 7, \((n, \omega) = (40, 5)\) using \((\tau, \Omega) = (300, 10800)\)
<table>
<thead>
<tr>
<th>Instance, ((n, \omega))</th>
<th>Method</th>
<th>Time bound ((\tau, \Omega))</th>
<th>Initial objective</th>
<th>LP Phase</th>
<th>MIP Step</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1, (10,5)</td>
<td>Heur. CG</td>
<td>23.07</td>
<td>20.21</td>
<td>0.03</td>
</tr>
<tr>
<td></td>
<td>1, (10,5)</td>
<td>H. CG(SRS)</td>
<td>23.07</td>
<td>20.21</td>
<td>0.03</td>
</tr>
<tr>
<td></td>
<td>1, (10,5)</td>
<td>Heur. CG</td>
<td>23.07</td>
<td>17.90</td>
<td>24.7</td>
</tr>
<tr>
<td></td>
<td>1, (10,5)</td>
<td>Heur. CG(LB)</td>
<td>23.07</td>
<td>20.21</td>
<td>27.4</td>
</tr>
<tr>
<td></td>
<td>2, (10,5)</td>
<td>Heur. CG</td>
<td>30.41</td>
<td>18.84</td>
<td>28.0</td>
</tr>
<tr>
<td></td>
<td>2, (10,5)</td>
<td>H. CG(SRS)</td>
<td>30.41</td>
<td>18.84</td>
<td>30.0</td>
</tr>
<tr>
<td></td>
<td>2, (10,5)</td>
<td>Heur. CG</td>
<td>30.41</td>
<td>15.77</td>
<td>15.2</td>
</tr>
<tr>
<td></td>
<td>2, (10,5)</td>
<td>Heur. CG(LB)</td>
<td>30.41</td>
<td>18.84</td>
<td>24.9</td>
</tr>
<tr>
<td></td>
<td>2, (10,5)</td>
<td>Heur. CG(LB)</td>
<td>30.41</td>
<td>15.77</td>
<td>15.6</td>
</tr>
<tr>
<td></td>
<td>3, (20,5)</td>
<td>Heur. CG</td>
<td>58.43</td>
<td>45.94</td>
<td>1562.1</td>
</tr>
<tr>
<td></td>
<td>3, (20,5)</td>
<td>H. CG(SRS)</td>
<td>58.43</td>
<td>45.94</td>
<td>1551.8</td>
</tr>
<tr>
<td></td>
<td>3, (20,5)</td>
<td>Heur. CG</td>
<td>58.43</td>
<td>39.44</td>
<td>213.6</td>
</tr>
<tr>
<td></td>
<td>3, (20,5)</td>
<td>Heur. CG(LB)</td>
<td>58.43</td>
<td>45.94</td>
<td>951.4</td>
</tr>
<tr>
<td></td>
<td>3, (20,5)</td>
<td>Heur. CG(LB)</td>
<td>58.43</td>
<td>39.44</td>
<td>182.9</td>
</tr>
<tr>
<td></td>
<td>4, (20,5)</td>
<td>Heur. CG</td>
<td>48.33</td>
<td>25.31</td>
<td>1071.9</td>
</tr>
<tr>
<td></td>
<td>4, (20,5)</td>
<td>H. CG(SRS)</td>
<td>48.33</td>
<td>25.31</td>
<td>1315.5</td>
</tr>
<tr>
<td></td>
<td>4, (20,5)</td>
<td>Heur. CG</td>
<td>48.33</td>
<td>25.31</td>
<td>272.2</td>
</tr>
<tr>
<td></td>
<td>4, (20,5)</td>
<td>Heur. CG(LB)</td>
<td>48.33</td>
<td>25.31</td>
<td>795.7</td>
</tr>
<tr>
<td></td>
<td>4, (20,5)</td>
<td>Heur. CG(LB)</td>
<td>48.33</td>
<td>21.15</td>
<td>253.5</td>
</tr>
<tr>
<td></td>
<td>5, (20,5)</td>
<td>Heur. CG</td>
<td>114.12</td>
<td>81.43</td>
<td>20209.3</td>
</tr>
<tr>
<td></td>
<td>5, (20,5)</td>
<td>Heur. CG</td>
<td>114.12</td>
<td>81.49</td>
<td>10999.6</td>
</tr>
<tr>
<td></td>
<td>5, (20,5)</td>
<td>Heur. CG(SRS)</td>
<td>114.12</td>
<td>81.43</td>
<td>11193.8</td>
</tr>
<tr>
<td></td>
<td>5, (20,5)</td>
<td>Heur. CG</td>
<td>114.12</td>
<td>70.15</td>
<td>4227.7</td>
</tr>
<tr>
<td></td>
<td>5, (20,5)</td>
<td>Heur. CG(LB)</td>
<td>114.12</td>
<td>81.43</td>
<td>9528.3</td>
</tr>
<tr>
<td></td>
<td>5, (20,5)</td>
<td>Heur. CG(LB)</td>
<td>114.12</td>
<td>70.16</td>
<td>3023.5</td>
</tr>
<tr>
<td></td>
<td>6, (30,5)</td>
<td>Heur. CG</td>
<td>97.39</td>
<td>67.59</td>
<td>20094.1</td>
</tr>
<tr>
<td></td>
<td>6, (30,5)</td>
<td>H. CG(SRS)</td>
<td>97.39</td>
<td>67.59</td>
<td>9894.1</td>
</tr>
<tr>
<td></td>
<td>6, (30,5)</td>
<td>Heur. CG</td>
<td>97.39</td>
<td>67.59</td>
<td>16051.1</td>
</tr>
<tr>
<td></td>
<td>6, (30,5)</td>
<td>Heur. CG</td>
<td>97.39</td>
<td>54.32</td>
<td>3453.0</td>
</tr>
<tr>
<td></td>
<td>6, (30,5)</td>
<td>Heur. CG</td>
<td>97.39</td>
<td>67.61</td>
<td>14395.3</td>
</tr>
<tr>
<td></td>
<td>6, (30,5)</td>
<td>Heur. CG</td>
<td>97.39</td>
<td>54.32</td>
<td>3013.2</td>
</tr>
<tr>
<td></td>
<td>7, (40,5)</td>
<td>Heur. CG</td>
<td>114.30</td>
<td>80.76</td>
<td>62839.4</td>
</tr>
<tr>
<td></td>
<td>7, (40,5)</td>
<td>Heur. CG</td>
<td>114.30</td>
<td>81.43</td>
<td>10934.7</td>
</tr>
<tr>
<td></td>
<td>7, (40,5)</td>
<td>Heur. CG</td>
<td>114.30</td>
<td>81.07</td>
<td>11206.1</td>
</tr>
<tr>
<td></td>
<td>7, (40,5)</td>
<td>Heur. CG</td>
<td>114.30</td>
<td>67.55</td>
<td>11067.4</td>
</tr>
<tr>
<td></td>
<td>7, (40,5)</td>
<td>Heur. CG</td>
<td>114.30</td>
<td>80.92</td>
<td>13783.2</td>
</tr>
<tr>
<td></td>
<td>7, (40,5)</td>
<td>Heur. CG</td>
<td>114.30</td>
<td>67.55</td>
<td>11008.3</td>
</tr>
<tr>
<td></td>
<td>8, (40,5)</td>
<td>Heur. CG</td>
<td>140.05</td>
<td>92.99</td>
<td>59954.0</td>
</tr>
<tr>
<td></td>
<td>8, (40,5)</td>
<td>Heur. CG</td>
<td>140.05</td>
<td>93.30</td>
<td>11209.3</td>
</tr>
<tr>
<td></td>
<td>8, (40,5)</td>
<td>H. CG(SRS)</td>
<td>140.05</td>
<td>93.33</td>
<td>10976.3</td>
</tr>
<tr>
<td></td>
<td>8, (40,5)</td>
<td>Heur. CG</td>
<td>140.05</td>
<td>79.10</td>
<td>10817.4</td>
</tr>
<tr>
<td></td>
<td>8, (40,5)</td>
<td>Heur. CG</td>
<td>140.05</td>
<td>93.02</td>
<td>15934.3</td>
</tr>
<tr>
<td></td>
<td>8, (40,5)</td>
<td>Heur. CG</td>
<td>140.05</td>
<td>79.10</td>
<td>9060.4</td>
</tr>
<tr>
<td></td>
<td>9, (50,5)</td>
<td>Heur. CG</td>
<td>144.61</td>
<td>112.15</td>
<td>11237.5</td>
</tr>
<tr>
<td></td>
<td>9, (50,5)</td>
<td>H. CG(SRS)</td>
<td>144.61</td>
<td>111.72</td>
<td>11081.2</td>
</tr>
<tr>
<td></td>
<td>9, (50,5)</td>
<td>Heur. CG</td>
<td>144.61</td>
<td>93.19</td>
<td>12139.8</td>
</tr>
<tr>
<td></td>
<td>9, (50,5)</td>
<td>Heur. CG</td>
<td>144.61</td>
<td>110.95</td>
<td>14118.1</td>
</tr>
<tr>
<td></td>
<td>9, (50,5)</td>
<td>Heur. CG</td>
<td>144.61</td>
<td>92.90</td>
<td>12011.1</td>
</tr>
<tr>
<td></td>
<td>10, (50,5)</td>
<td>Heur. CG</td>
<td>174.82</td>
<td>129.91</td>
<td>11561.7</td>
</tr>
<tr>
<td></td>
<td>10, (50,5)</td>
<td>H. CG(SRS)</td>
<td>174.82</td>
<td>130.38</td>
<td>11779.9</td>
</tr>
<tr>
<td></td>
<td>10, (50,5)</td>
<td>Heur. CG</td>
<td>174.82</td>
<td>113.72</td>
<td>12707.9</td>
</tr>
<tr>
<td></td>
<td>10, (50,5)</td>
<td>Heur. CG</td>
<td>174.82</td>
<td>129.09</td>
<td>16935.6</td>
</tr>
<tr>
<td></td>
<td>10, (50,5)</td>
<td>Heur. CG</td>
<td>174.82</td>
<td>113.83</td>
<td>11164.3</td>
</tr>
</tbody>
</table>

Table 6.4: Computational experience using Heuristic CG
6.6.3 Subgradient Optimization Performance

Table 6.5 presents preliminary computational results for the subgradient-based optimization scheme displayed in Figure 6.1. We initialized the $c_q$-coefficients, $\forall q$, with the corresponding $\lambda$-penalties, and set $[\sigma_k = 1, \forall k]$. We have monitored a separate sequence of feasible solutions deduced from the fractional projected iterates via a rounding procedure. Despite the significant decrease in the LP objective value as the iterations progress, the same 0-1 feasible solution was persistently produced from the projected iterates. Thus, the initial feasible solution was not improved upon termination of the algorithm.

<table>
<thead>
<tr>
<th>Instance, $n \times m$</th>
<th>Initial objective</th>
<th>Final objective, $\nu$</th>
<th>% Gap, $\nu - LB^*$</th>
<th>Number of iterations</th>
<th>CPU time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, (10,5)</td>
<td>22.70</td>
<td>22.70</td>
<td>10.97</td>
<td>200</td>
<td>242.22</td>
</tr>
<tr>
<td>2, (10,5)</td>
<td>24.46</td>
<td>24.46</td>
<td>22.98</td>
<td>200</td>
<td>251.12</td>
</tr>
</tbody>
</table>

Table 6.5: Results for the subgradient optimization algorithmic approach

6.7 Summary and Directions for Future Research

This chapter has examined a production planning and scheduling problem faced by a major motorcycle manufacturing firm (Harley Davidson Motor Company). In this context, we have addressed the problem of partitioning and sequencing the production of different manufactured items in mixed-model assembly lines, where each model has various specific options and designated destinations. We proposed two mixed-integer programming formulations for this problem that seek to sequence the manufactured goods within production batches in order to balance the motorcycle model and destination outputs as well as the load demands on material and labor resources. Recognizing the symmetry inherent to the second MIP formulation, we enforced an additional set of hierarchical symmetry-defeating constraints that impart specific identities amongst batches of products under composition. The latter model inspired a third set partitioning-based formulation in concert with an efficient column generation approach that directly achieves the joint partitioning of jobs into batches along with ascertaining the sequence of jobs within each composed batch. Finally, we explored a subgradient-based optimization strategy that exploits a non-differentiable optimization formulation, which is prompted by the flexibility in the production process as reflected in the model via several soft-constraints, thereby providing a real-time decision-making tool. Our computational experience demonstrated the relative effectiveness of the different proposed formulations and the associated optimization strategies for solving a set of realistic problem instances. We
also exhibited the insightful benefits of a *complementary column generation* (CCG) feature that is further discussed in Chapter 7. The versatility of our integrated modeling and optimization framework renders it attractive to capture additional specifications that arise in mixed-model assembly lines such as the incorporation of due-dates and set-up cost considerations. Future research can address the interesting problem of integrating our partitioning-sequencing model with the problem of *line balancing* in assembly lines.
Chapter 7

Models and Algorithms for the Subassembly Parts Assignment Problem

This chapter addresses the problem of matching or assigning subassembly parts in assembly lines where we seek to minimize the total deviation of the resulting final assemblies from a vector of nominal and mean quality characteristic values. We introduce various symmetry-defeating enhancements to an existing assignment-based model, and highlight the critical importance of using particular types of symmetry-defeating hierarchical constraints that preserve the model structure. We also propose a set partitioning-based formulation in concert with a column generation approach that efficiently exploits the structure of the problem. In addition, we develop several heuristic procedures. Computational experience is presented to demonstrate the relative effectiveness of the different proposed strategies for solving a set of realistic problem instances.

7.1 Introduction and Motivation

Variation is inherent to manufacturing processes that are concerned with producing subassembly components or items within specified tolerance requirements, and has posed significant challenges for practitioners and quality experts alike. Formally, we consider assembly lines where each of some $P$ final products is composed by selecting a single component from each of $G$ groups (or bins) that contain $P$ subassembly parts. Each of the $P$ subassembly parts in a given group fulfill the same functional role while exhibiting distinctive quality characteristic values that pertain to physical and operational aspects (such as length, height, weight, and production cost). We seek
to compose the $P$ final parts so as to minimize the total deviation with respect to a vector of nominal and mean values for the set of quality characteristics under investigation, and refer to this problem as the subassembly parts assignment problem (SPAP). In an interesting variant to this problem, box-constraints are imposed on the final parts to be assembled in order to comply with specified tolerance requirements, and the objective is to maximize the total number of valid parts that can possibly be assembled. However, the latter problem is beyond the scope of this work, and is considered for future investigation.

In the context of variation reduction, selective assembly is considered to be a cost-effective approach that has gained wide popularity. This method partitions the subassembly parts into several subgroups, and the mating is performed among subgroups (not at the individual component level), so long as the resulting assembled parts satisfy the required quality characteristics. The interchangeability amongst subassembly parts that belong to the same subgroup permits some relaxation of the quality requirements for individual subassembly parts, and focuses attention on the overall quality characteristics of the final product. To implement this method, it is necessary to measure the quality characteristics pertaining to each subassembly part in order to ascertain the dispatching of functionally identical subassembly parts into appropriate subgroups and to select mating subgroups. A major difficulty in applying this method arises from the possible mismatches between the cardinalities of the mated subgroups, as noted by Mansoor [82]. In this perspective, Kannan and Jayabal [65] proposed a grouping strategy for minimizing the surplus of subassembly components in a selective assembly, and Kannan et al. [66] devised a genetic algorithm-based method that aims at dispatching the functionally identical subassembly parts into six subgroups in order to minimize the overall variation in a selective assembly.

In a recent paper, Musa et al. [92] considered the SPAP where the matching is performed at the individual subassembly part level. This problem is of practical interest in the context of examining and evaluating the performance of newly installed production and assembly lines. The authors proposed mathematical programming formulations for the variation reduction and the throughput maximization problems. However, these models are computationally intractable even for small-sized problem instances and, therefore, merely serve the purpose of formally stating the problem. The authors have also developed a simple, constructive heuristic that is adequate only for problems that involve a single quality characteristic. One of the issues that we discuss in the present chapter is the symmetry that is inherent in the formulation proposed in [92], and we devise various modeling
and optimization strategies in order to impart and exploit particular mathematical structures in the problem formulation so as to enhance its solvability.

The remainder of this chapter is organized as follows. Section 7.2 introduces various symmetry-defeating enhanced mathematical programming formulations and a set partitioning model for the SPAP. Thereafter, we delineate in Section 7.3 a column generation optimization approach based on a set partitioning formulation of the problem, and also describe two additional heuristics that adopt different strategies to directly tackle the SPAP. Section 7.4 reports our computational experience using a set of realistic test problems, and highlights the relative effectiveness of the foregoing solution approaches and the critical importance of the complementary column generation (CCG) feature that is incorporated within our proposed column generation algorithm. We close the chapter in Section 7.5 with a summary and conclusions.

7.2 Problem Modeling

In this section, we introduce our notation, and discuss thereafter several symmetry-defeating-based enhancements to an existing mathematical programming formulation of Musa et al. [92].

7.2.1 Notation

- $G$: Number of subassembly groups.
- $g$: Group index; $g \in \{1, ..., G\}$.
- $P$: Batch size, that is, $P$ equals the number of components in each subassembly group, and also equals the total number of final products to be assembled.
- $i$: Item index within its group; $i \in \{1, ..., P\}$.
- $p$: Part or product index; $p \in \{1, ..., P\}$.
- $r$: Overall item index; $r \in \{1, ..., GP\}$, where the items are numbered consecutively over groups $g = 1, ..., G$, in that order.
- $Q$: Number of quality characteristics.
- $q$: Quality characteristic index; $q \in \{1, ..., Q\}$.
- $d_{giq}$: Contribution of component $i$ in group $g$ to quality characteristic $q$. Likewise, we define the contribution of item $r$ to quality characteristic $q$ as $d_{rq}$.
• \( nom_q \): User-specified nominal value desired for quality characteristic \( q \).

• \( mean_q \): Mean value for the overall quality characteristic \( q \) over the \( P \) products; \( mean_q = \sum_{g=1}^{G} \sum_{i=1}^{P} d_{gip} \).

• \( x_{gip} \in \{0, 1\}; \ x_{gip} = 1 \) if and only if component \( i \) in group \( g \) is assigned to product \( p \).

• \( V_{pq} \): The contribution of the various subassembly parts in product \( p \) to the value of quality characteristic \( q \); \( V_{pq} \equiv \sum_{g=1}^{G} \sum_{i=1}^{P} x_{gip}d_{giq} \).

• \( V_p \): \( Q \times 1 \) vector that records the \( V_{pq} \)-values for product \( p \).

• \( u_{max} \equiv \sum_{g=1}^{G} \sum_{i=1}^{P} d_{giq} \). Note that \( V_{pq} \leq u_{max}, \forall p,q \).

• The linear programming (LP) relaxation of any mathematical program \( \Phi \) will be referred to as \( \bar{\Phi} \), and its objective value will be denoted by \( \nu(\bar{\Phi}) \).

### 7.2.2 Assignment-based Models

For clarity in exposition, the formulation proposed by Musa et al. [92] is denoted by \textbf{SPAP1} and is stated below. Several enhancements are proposed to this model in the sequel.

\textbf{SPAP1}: Minimize \( \sum_{p=1}^{P} \sum_{q=1}^{Q} \left( \sum_{g=1}^{G} \sum_{i=1}^{P} x_{gip}d_{giq} - nom_q \right) + \left( \sum_{g=1}^{G} \sum_{i=1}^{P} x_{gip}d_{giq} - mean_q \right) \) \hspace{1cm} (7.1a)

subject to \( \sum_{i=1}^{P} x_{gip} = 1, \ \forall g,p \) \hspace{1cm} (7.1b)

\( \sum_{p=1}^{P} x_{gip} = 1, \ \forall g,i \) \hspace{1cm} (7.1c)

\( x \) binary. \hspace{1cm} (7.1d)

The objective function stated in (7.1a) seeks to minimize the total deviation from both the nominal and mean values of the different quality characteristics under scrutiny, while Constraints (7.1b)-(7.1d) jointly enforce classical restrictions for \( G \) assignment problems. Note that for implementing \textbf{SPAP1}, the objective function would be linearized by introducing suitable continuous
variables and appropriate relational constraints.

Observe that there exists an inherent symmetry between the products to be assembled according to this model, and therefore, any permutation of the product indices in any feasible solution (and in particular, any optimal solution) produces an equivalent solution that has the same objective value. Sherali and Smith [124] have expounded on the critical importance of recognizing the presence of such symmetries, and then imposing so-called symmetry-defeating or hierarchical constraints in order to enhance algorithmic performance and problem solvability. To hedge against such symmetric reflections, we introduce below several classes of symmetry-defeating constraints that can be appended to Model SPAP1 so as to impart specific identities to the different products.

- **Decision hierarchy (σ):**

\[
\sum_{q=1}^{Q} V_{pq} \leq \sum_{q=1}^{Q} V_{p+1,q}, \quad p = 1, \ldots, P - 1.
\]  

(7.2)

Note that the decision hierarchy (σ) specifies an order among the \(P \) products based on the cumulative contribution of the subassembly parts to their quality characteristics.

- **Lexicographic decision hierarchy (φ):**

Denoting a lexicographic ordering of vectors in \(R^Q \) in terms of their component values \(q = 1, \ldots, Q \) by \(\leq\), we impose that \(V_p \leq V_{p+1}, p = 1, \ldots, P - 1\). This symmetry-defeating hierarchy is denoted by (φ), and for practical purposes, is implemented as the following linear constraint:

\[
\sum_{q=1}^{Q} M^{q} V_{pq} \leq \sum_{q=1}^{Q} M^{q} V_{p+1,q}, \quad p = 1, \ldots, P - 1,
\]  

(7.3)

where \(M = 1 + \lceil u_{\text{max}} \rceil\). We shall refer to the foregoing two symmetry-defeating enhanced models as SPAP1(σ) and SPAP1(φ), respectively.

- An alternative strategy to combat the aforementioned symmetric reflections consists in assigning item \(i\) in the first group to product \(i\), by setting \(x_{1ii} = 1, \forall i\), which also results in a problem reduction. We shall denote this model by SPAP2. Furthermore, if some of the resultant partial products \(p\) retain the same \([d_{ipq}, \forall q]-\)values, then we impose either (σ) or (φ) within such subsets of indistinguishable partial products.
7.2.3 Set Partitioning-based Formulation

Recall that a final assembled product consists of $G$ subassembly parts, each being selected from a group or bin that contains $P$ distinct subassembly parts. Any final product can therefore be represented using the following modeling construct, which we refer to as a pattern or product composition:

\[
P^j = \begin{bmatrix}
  r = 1 \\
  r = 2 \\
  \vdots \\
  r = GP
\end{bmatrix}
\]

where $P^j_r = \begin{cases} 
1 & \text{if subassembly } r \text{ is included in pattern } j \\
0 & \text{otherwise.}
\end{cases}$

Let $j = 1, ..., J$ index all such possible patterns, and define $z_j$ as a binary variable that assumes a value of 1 if and only if product composition $P^j$ is selected for assembly, $j = 1, ..., J$. Also, let $c_j$ be the cost associated with pattern $P^j$; $c_j = \sum_{q=1}^{Q} \left( \left| \sum_{r=1}^{GP} P^j_r d_{rq} \right| - nom_q \right) + \left| \sum_{r=1}^{GP} P^j_r d_{rq} - mean_q \right|$. Denoting $[e]$ as a vector of $GP$-ones, the SPAP can then be modeled as the following set partitioning problem SPP:

\[
\text{SPP: Minimize } \sum_{j=1}^{J} c_j z_j 
\]

subject to \[
\sum_{j=1}^{J} P^j z_j = [e] 
\]

\[z \text{ binary.}\]

Observe that conducting the optimization effort over the set of all possible product compositions would be a formidable endeavor, rendering Problem SPP of little practical use. Thus, we shall investigate Problem SPP in concert with a suitable column generation procedure that is proposed in Section 7.3.1.

7.3 Heuristics

We propose in this section three heuristics for the SPAP problem. The first heuristic tackles Problem SPP introduced in Section 7.2.3 via a column generation algorithm that incorporates a complementary column generation (CCG) feature. The second heuristic is based on iteratively
constructing the best possible product out of the subassembly parts that have not been assigned as yet, until all the $P$ products are assembled. The third heuristic involves a sequential group-based scheme that assigns, in turn, the subassembly parts in each bin to those in the first bin while composing products.

7.3.1 Set Partitioning-based Column Generation Heuristic (SPH)

The proposed heuristic SPH tackles Problem SPP via a column generation scheme that dynamically generates improved final products through an interaction between a restricted master program (RMP) and a subproblem (SP). This strategy iteratively composes a set of promising final products that achieve a partitioning of the entire set of $GP$ subassembly parts, while minimizing the total deviation from the vectors of nominal and mean values. In this perspective, the RMP examines some $\hat{J}$ possible patterns, $P \leq \hat{J} \leq J$, for selecting a set of $P$ products that satisfy the set partitioning restrictions (7.4b). To initialize the RMP, a judicious set of $\hat{J} = P$ patterns is composed in order to prescribe a valid set partitioning solution for all the subassembly parts in the problem. Typically, Heuristics BPH and SGH proposed in Sections 7.3.2 and 7.3.3, respectively, can be utilized for this purpose. However, for the sake of simplicity and for independently evaluating Heuristic SPH, we shall generate $P$ initial patterns where the $j^{th}$ such pattern is composed by including the $j^{th}$ item from each of the $G$ groups.

Upon solving the LP relaxation of Problem RMP, the corresponding values of the dual variables $\pi = \bar{\pi}$ associated with Constraints (7.4b) are passed to the subproblem $\text{SP}(\bar{\pi})$, which conducts a pricing to generate a most promising pattern for appending to Problem RMP. This subproblem is stated below, where the binary variables $y_r, r = 1, ..., GP$, define the elements of the pattern under construction.

$$\text{SP}(\bar{\pi}): \text{Minimize} \quad \sum_{q=1}^{Q} \left( \sum_{r=1}^{GP} y_r d_{rq} - \text{nom}_q \right) + \sum_{r=1}^{GP} y_r d_{rq} - \text{mean}_q \right) - \sum_{r=1}^{GP} \bar{\pi}_r y_r \quad (7.5a)$$

subject to

$$\sum_{r=(g-1)P+1}^{gP} y_r = 1, \quad \forall g = 1, \ldots, G \quad (7.5b)$$

$y$ binary. \quad (7.5c)

The objective function in (7.5a) minimizes the reduced cost of the variable associated with the potential pattern that is generated via $\text{SP}(\bar{\pi})$ based on the achieved values of the $y$-variables.
straint (7.5b) ensures that exactly one subassembly part is selected from each of the $G$ groups in order to compose the pattern under construction.

If the pattern generated by $\text{SP}(\pi)$ has a negative reduced cost, i.e., it prices out favorably, we introduce this into Problem RMP, and increment the set $\hat{J}$ by one. Otherwise, we can conclude that the current solution to Problem $\overline{\text{RMP}}$ solves Problem $\overline{\text{SPP}}$.

Heuristic SPH also integrates the following complementary column generation (CCG) feature that simultaneously constructs $P - 1$ accompanying patterns that form a full set partitioning solution in conjunction with any pattern generated via $\text{SP}(\pi)$ for introduction into Problem RMP. Let $\bar{y}$ be the solution obtained for Problem $\text{SP}(\pi)$ such that the resulting pattern has a negative reduced cost, and consider the set $Z \equiv \{r : \bar{y}_r = 1\}$. We then solve Problem $\text{SP}(\pi)$ with the additional requirement that $y_r = 0, \forall r \in Z$, and generate a corresponding solution, denoted by $\bar{y}^{\text{new}}$. Next, the set $Z$ of prohibited item indices is augmented by setting $Z \leftarrow Z \cup \{r : \bar{y}_r^{\text{new}} = 1\}$, and the foregoing step is repeated. This is continued until the desired $P - 1$ complementary product compositions are generated, and the RMP is then augmented with the entire resulting block of $P$ patterns that constitute a feasible solution to Problem SPP, so that we update $\hat{J} \leftarrow \hat{J} + P$.

Once the LP phase terminates, the current RMP is solved to optimality as a 0-1 integer program, and the patterns selected via this final set partitioning problem are prescribed as the final output. To appreciate the crucial role played by the aforementioned CCG feature, we shall consider two implementation variants of Heuristic SPH, denoted by Procedures SPH-A and SPH-B, where the CCG feature is incorporated only in Procedure SPH-A. We shall use “Heuristic SPH” to indifferently refer to either of these two procedures.

Observe that Heuristic SPH can be adapted and judiciously applied to a broad spectrum of combinatorial optimization problems that involve a set partitioning structure. Although the LP relaxations of such set partitioning models typically tend to produce very tight lower bounds on the optimal value of the underlying 0-1 integer programming formulation, there is no guarantee that the final step in which the binary restriction is enforced in the set partitioning model will produce a (near-) optimal solution to the original 0-1 program. Furthermore, such a strategy might not even produce a feasible solution to the original problem, unless an initial feasible solution is introduced and retained within the RMP. Interestingly, a column generation scheme similar to Procedure SPH-B has been reported to be particularly promising for the class of joint vehicle loading-routing
problems investigated by Sherali and Ghoniem [118]. In the latter application, the patterns generated by the LP phase are relatively sparse and do not significantly overlap, which often tends to produce binary-valued variables in the LP solution to the RMP, leading to (near-) optimal solutions in the final step of the column generation approach. In contrast, for Problem SPAP, we shall see in Section 7.4 that Procedure SPH-B empirically produces excellent results for small- to moderate-sized problem instances, but performs poorly on larger problem instances. In fact, while the pool of patterns that is produced by the LP phase contains some very promising final assemblies, these may not qualify as a complete set partitioning solution to the entire problem and, therefore, the final binary-restricted optimization step cannot avail of these desirable assemblies and ends up selecting a significantly suboptimal set of product compositions for larger problem instances, which might be the arbitrary initial solution itself that was introduced into the RMP. In contrast, Heuristic SPH-A obviates this difficulty by generating and incorporating an accompanying set of patterns that complement any product composition that prices out favorably. As demonstrated in Section 7.5, SPH-A consistently produced (near-) optimal solutions for the different problem instances in our test-bed.

A final note is in order concerning the termination criterion pertaining to Heuristic SPH. It is empirically observed that the last iteration in the LP Phase of Heuristic SPH tends to be excessively time-consuming for larger problem instances (see Section 7.4). That is, the computational effort to solve the associated SP(\overline{\pi}) typically exceeds by several orders of magnitude (more than 3000-fold for the instance \((G,P,Q) = (10,10,1)\) in Section 7.4) the cumulative computational time for all the previous iterations in the LP Phase of Heuristic SPH. In fact, it becomes progressively more difficult to generate patterns via Problem SP(\overline{\pi}) that achieve marginal improvements in the LP objective value of the RMP as the iterations proceed, and the conclusion that no patterns price out favorably eventually comes at the expense of a high computational effort. Therefore, we investigate in Sections 7.4.2 and 7.4.3 the computational benefits of solving Problem SP(\overline{\pi}) using some absolute \(\varepsilon\)-optimality tolerance criterion, i.e., we seek a solution whose value is guaranteed to be within an \(\varepsilon\)-neighborhood of the true optimal objective value. The latter strategy would, in fact, be particularly effective if the computational burden that is witnessed for large problem instances is predominantly predicated by a slow tail-end convergence due to a sequence of near-zero-reduced cost columns.
7.3.2 Sequential Constructive Heuristic (SCH)

This heuristic aims at constructing each of the \( P \) final products in an iterative fashion. At each iteration \( 1 \leq k \leq P - 1 \), we solve the following problem \( \text{SCH}(k) \) to optimality for composing the \( k^{th} \) product, given the composition of the previous \( k - 1 \) products (for \( k \geq 2 \)). Here, letting \( X_0 \equiv \emptyset \) for \( k = 1 \) and denoting the solution obtained for Problem \( \text{SCH}(k-1) \) for \( k \geq 2 \) as \( [\bar{x}_{gi,k-1}, \forall g, i] \), we define the set \( X_{k-1} \equiv \{(g, i) : \bar{x}_{gi,k-1} = 1\} \cup X_{k-2} \) in the formulation given below.

\[
\text{SCH}(k): \ \text{Minimize} \ \sum_{q=1}^{Q} (|V_{kq} - nom_q| + |V_{kq} - mean_q|)
\]

\[
\text{subject to} \ \sum_{i=1}^{P} x_{gik} = 1, \ \forall g
\]

\[
x_{gik} = 0, \ \forall (g, i) \in X_{k-1}
\]

\[
x_{gik} \text{ binary, } \forall g, i.
\]

Observe that this constructive scheme terminates in \( P - 1 \) iterations, the \( P^{th} \) final product being composed of the single subassembly part remaining in each group.

7.3.3 Sequential Group-based Heuristic (SGH)

The third proposed heuristic begins by matching the subassembly parts in the first and second groups, and subsequently, each of the resulting \( P \) partially composed products are assigned a single component from the third group, and so forth. This stage-wise assembly procedure solves the problem over \( G - 1 \) iterations, and terminates when the \( P \) products are fully assembled. Various strategies may be adopted to tackle the two-group subassembly part assignment subproblem that arises at each iteration of this heuristic. These are delineated below.

- **SGH-A:** Observe that the two-group subassembly part assignment subproblem that characterizes each iteration of this heuristic can be viewed as a particular two-group instance of Problem SPP introduced in Section 7.2.3, and can, therefore, be solved using Heuristic SPH (we used Procedure SPH-A for this purpose).

- **SGH-B:** Let the vector of quality characteristics for each product \( p = 1, ..., P \), under assembly at the end of iteration \( k \in \{2, ..., G\} \) be denoted by \( [V_{pq}^k, \forall q = 1, ..., Q] \). This vector is initialized by setting \( [V_{pq}^1 = d_{1pq}, \forall q], \forall p \), and is subsequently updated after each iteration.
\[ k \geq 2 \] by setting \( V_{pq}^k \leftarrow V_{pq}^{k-1} + \sum_{i=1}^{P} \bar{x}_{kip}d_{kiq}, \forall q, \forall p \), where \( \bar{x}_{kip}, \forall i, p \) represents the optimal solution to Problem SPAP2\((k)\), \( k = 2, \ldots, G \), stated below.

**SPAP2\((k)\):** Minimize \[
\sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{i=1}^{P} x_{kip}(V_{pq}^{k-1} + d_{kiq}) - \frac{k}{G} \text{nom}_q + \\
\left| \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{i=1}^{P} x_{kip}(V_{pq}^{k-1} + d_{kiq}) - \sum_{g=1}^{P} \frac{\sum_{i=1}^{P} d_{giq}}{P} \right| 
\]

subject to \[
\sum_{i=1}^{P} x_{kip} = 1, \quad \forall p \quad (7.7b) \\
\sum_{p=1}^{P} x_{kip} = 1, \quad \forall i \quad (7.7c) \\
x_{kip} \text{ binary}, \forall i, p. \quad (7.7d)
\]

### 7.4 Computational Experience

We now provide some computational experience to study the solvability of the various assignment-based symmetry-defeating-enhanced formulations, as well as to examine the relative effectiveness of the proposed exact and heuristic solution approaches. We consider a test-bed composed of twenty problem instances that belong to the following four sets, where the size of any instance is characterized by the triplet \((G, P, Q)\), and where the values of the \(d_{giq}\)-parameters are generated to four decimal places as described below.

- **Set 1** is composed of six small-sized problem instances having \((G, P, Q) = (2,7,3), (2,9,2), (3,8,2), (3,9,2), (3,9,3), \) and \((3,12,3)\). (The particular foregoing values of \(G\), \(P\), and \(Q\) in these instances were generated using uniform distributions over the intervals \([2,3]\), \([1,10]\), and \([1,3]\).) In addition, the \(\text{nom}_q\)-values were generated using a uniform distribution over the interval \([G, 2G]\), and the corresponding \(d_{giq}\)-parameters were generated using a Normal distribution having a mean of \(\frac{\text{nom}_q}{G}\) and a variance of 0.05. (The \(\text{nom}_q\)- and \(d_{giq}\)-values pertaining to Sets 3 and 4 below are derived following this same data generation process.) Set 1 serves the purpose of empirically evaluating the proposed symmetry-defeating constraints as well as the assignment-based formulations.
• **Set 2** is composed of eighty moderately-sized problem instances, each having \((G, P, Q) = (3,10,3)\). For these instances, the \(nom_q\)-values were generated using a uniform distribution over the interval \([G, 2G]\), and the corresponding \(d_{qiq}\)-parameters were generated using a Normal distribution having a mean of \( \frac{nom_q}{G} \) and a variance that varies over the interval \([0.05,0.75]\) with an incremental step of 0.10. For each of these eight variance values, ten problem instances were generated. **Set 2** is designed to evaluate the potential influence of the data variance on the computational performance.

• **Set 3** is composed of six moderately-sized problem instances having \((G, P, Q) = (2,12,2), (2,15,1), (3,10,1), (3,15,1), (5,15,1), and (5,20,1)\).

• **Set 4** is composed of six relatively larger realistic test problems having \((G, P, Q) = (10,10,1), (10,10,2), (10,10,3), (10,10,4), (12,12,2), and (15,15,3)\).

The mathematical programs implemented were coded in AMPL and solved using CPLEX 10.1 on a Dell Precision 650 workstation having a Xeon(TM) CPU 2.40 GHz processor and 1.50 GB of RAM.

7.4.1 Assignment-based Models

We report in this section the computational effort associated with solving SPAP1 as well as the new proposed assignment-based models SPAP1\((\sigma)\), SPAP1\((\phi)\), and SPAP2 to optimality. The results reported in Table 7.1 indicate that Model SPAP2 achieved an average savings in computational effort of 94.26% over Model SPAP1. Observe that only the symmetry-enhanced model SPAP2 was able to solve the problems in Set 1 in manageable computational times, although it too becomes intractable for any larger problem instances. Also, note that Model SPAP1 substantially outperforms its symmetry-defeating augmented variants SPAP1\((\sigma)\) and SPAP1\((\phi)\), and respectively achieves an average computational savings of 99.94% and 99.81% over the latter for the first three problems in Set 1. This outcome provides an interesting perspective that, as generally observed in Sherali and Smith [124], whereas addressing symmetry issues is important to enhance problem solvability, different symmetry-defeating constraints might have a different effect on the overall effort required, and some such devices might even hamper the solution process. The latter occurs in the present context because the constraint sets \((\sigma)\) and \((\phi)\) inhibit the predominant assignment-based structure of the original model, unlike the model formulation SPAP2. Also, in spite of the fact that these assignment-based models are difficult to solve to optimality, their underlying LP relaxations
<table>
<thead>
<tr>
<th>Instance</th>
<th>Optimal objective value</th>
<th>CPU time in secs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SPAP2</td>
<td>SPAP1</td>
</tr>
<tr>
<td>(G, P, Q) = (2, 7, 3)</td>
<td>1.5032</td>
<td>0.25</td>
</tr>
<tr>
<td>(G, P, Q) = (2, 9, 2)</td>
<td>0.6983</td>
<td>0.01</td>
</tr>
<tr>
<td>(G, P, Q) = (3, 8, 2)</td>
<td>0.4684</td>
<td>0.51</td>
</tr>
<tr>
<td>(G, P, Q) = (3, 9, 2)</td>
<td>1.1140</td>
<td>99.74</td>
</tr>
<tr>
<td>(G, P, Q) = (3, 9, 3)</td>
<td>1.1575</td>
<td>3.98</td>
</tr>
<tr>
<td>(G, P, Q) = (3, 12, 3)</td>
<td>1.6133</td>
<td>305.99</td>
</tr>
</tbody>
</table>

Table 7.1: Assignment-based models using Set 1

Figure 7.1: Average CPU time for Model SPAP2 as a function of the data variance using Set 2

do provide tight lower bounds that can be useful for assessing the quality of heuristics for larger
problem instances (see Sections 7.4.2 and 7.4.3).

As far as the effect of variance over the test instances in Set 2 is concerned, Figure 7.1 displays
the average computational effort associated with each variance value considered in Set 2. Although
different instances having the same size and data variance might yield variations in the ensuing
computational effort, Figure 7.1 suggests that the data variance bears no statistically significant
effects on the average computational effort. The problem instances in Set 2 were solved in 65.83
CPU time seconds at an average, and Figure 7.1 provides the insight that the difficulty experienced
in solving the SPAP problem is primarily due to its intrinsic combinatorial nature and the size of
the problem instance under scrutiny, (G, P, Q).
7.4.2 LP Relaxations

We provide in this section a comparison between the relative strengths of SPAP1, SPAP2, and SPP for the test problem instances in Sets 3 and 4. Table 7.2 compares the lower bounds $\nu(\text{SPAP1})$, $\nu(\text{SPAP2})$, and $\nu(\text{SPP})$ against the best incumbent solution value, $UB^*$ (which is produced by SPH-A - see Section 7.4.3), in terms of the resulting percentage optimality gaps. These results reveal that the LP phase of Heuristic SPH provides a very tight lower bound on the optimal objective value, and, moreover, provides an optimal solution for two out of four problem instances in Set 3, and near-optimal solutions within less than an optimality gap of 0.68\% for the other instances in Set 3. Note that the column generation-based relaxation values are reported pertaining to the general Heuristic SPH, as these are not affected by the incorporation of the CCG feature (which is nonetheless critical for solving Problem SPP in the discrete sense as seen in Section 7.4.3 below). Here, the % optimality gap column values for these runs should be interpreted as $100 \frac{UB^* - \nu(\text{SPP})}{UB^*}$, where $UB^*$ is the best incumbent for the problem instance under investigation. Also, observe that $\nu(\text{SPAP1})$ equals $\nu(\text{SPAP2})$ for all problems instances, except for $(G, P, Q) = (2, 12, 2)$ and $(2, 15, 1)$. However, the assignment-based models are intractable even for moderate-sized problem instances, notwithstanding the tightness of their underlying LP relaxations, as noted in Section 7.4.1.

Table 7.3 provides the corresponding results using the Set 4 instances, where we have displayed the results for Procedure SPH-A using $\varepsilon = 10^{-2}$. In the column corresponding to $UB^*$, we have displayed two values that are derived from Table 7.5 and may be the same for certain test instances. The first value corresponds to the best incumbent obtained for the problem instance under investigation via Procedure SPH-A using $\varepsilon \leq 10^{-2}$, whereas the second value pertains to the solution obtained via Procedure SPH-A using $\varepsilon = 10^{-2}$. Also, note that because of this $\varepsilon$-optimality tolerance criterion, the optimal objective value $\nu(\text{SPP})$ produced by the LP phase of Heuristic SPH may not qualify as a valid lower bound on Problem SPAP and, therefore, the % optimality gap reported for Procedure SPH-A in Table 7.3 should be interpreted as $100 \frac{\nu(\text{SPP}) - \nu(\text{SPP})}{\nu(\text{SPP})}$ (using $\varepsilon = 10^{-2}$).
<table>
<thead>
<tr>
<th>Instance $(G, P, Q)$</th>
<th>LP objective value, $LB$</th>
<th>% optimality gap, $\frac{UB^* - LB}{UB^*}$</th>
<th>CPU time in secs</th>
<th>Optimal MIP objective value, $UB^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(2,12,2)$</td>
<td>0.9038</td>
<td>63.22</td>
<td>0</td>
<td>2.4576</td>
</tr>
<tr>
<td>SPAP1</td>
<td>1.2234</td>
<td>50.21</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>SPAP2</td>
<td>2.4576</td>
<td>0</td>
<td>2.5</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instance $(G, P, Q)$ = (2,15,1)</th>
<th>LP objective value, $LB$</th>
<th>% optimality gap, $\frac{UB^* - LB}{UB^*}$</th>
<th>CPU time in secs</th>
<th>Optimal MIP objective value, $UB^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPAP1</td>
<td>0.1997</td>
<td>77.01</td>
<td>0</td>
<td>0.8688</td>
</tr>
<tr>
<td>SPAP2</td>
<td>0.4179</td>
<td>51.89</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>SPH</td>
<td>0.8688</td>
<td>0</td>
<td>3.1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instance $(G, P, Q)$ = (3,10,1)</th>
<th>LP objective value, $LB$</th>
<th>% optimality gap, $\frac{UB^* - LB}{UB^*}$</th>
<th>CPU time in secs</th>
<th>Optimal MIP objective value, $UB^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPAP1</td>
<td>0.6997</td>
<td>7.61</td>
<td>0</td>
<td>0.7574</td>
</tr>
<tr>
<td>SPAP2</td>
<td>0.6997</td>
<td>7.61</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>SPH</td>
<td>0.7546</td>
<td>0.37</td>
<td>4.6</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instance $(G, P, Q)$ = (3,15,1)</th>
<th>LP objective value, $LB$</th>
<th>% optimality gap, $\frac{UB^* - LB}{UB^*}$</th>
<th>CPU time in secs</th>
<th>Optimal MIP objective value, $UB^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPAP1</td>
<td>1.0865</td>
<td>3.22</td>
<td>0</td>
<td>1.1227</td>
</tr>
<tr>
<td>SPAP2</td>
<td>1.0865</td>
<td>3.22</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>SPH</td>
<td>1.1207</td>
<td>0.17</td>
<td>8.9</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instance $(G, P, Q)$ = (5,15,1)</th>
<th>LP objective value, $LB$</th>
<th>% optimality gap, $\frac{UB^* - LB}{UB^*}$</th>
<th>CPU time in secs</th>
<th>Best upper bound, $UB^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPAP1</td>
<td>1.2961</td>
<td>0.48</td>
<td>0.01</td>
<td>1.3024</td>
</tr>
<tr>
<td>SPAP2</td>
<td>1.2961</td>
<td>0.48</td>
<td>0.01</td>
<td></td>
</tr>
<tr>
<td>SPH</td>
<td>1.2962</td>
<td>0.47</td>
<td>84.54</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instance $(G, P, Q)$ = (5,20,1)</th>
<th>LP objective value, $LB$</th>
<th>% optimality gap, $\frac{UB^* - LB}{UB^*}$</th>
<th>CPU time in secs</th>
<th>Best upper bound, $UB^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPAP1</td>
<td>1.4314</td>
<td>0.74</td>
<td>0</td>
<td>1.4421</td>
</tr>
<tr>
<td>SPAP2</td>
<td>1.4314</td>
<td>0.74</td>
<td>0.04</td>
<td></td>
</tr>
<tr>
<td>SPH</td>
<td>1.4322</td>
<td>0.68</td>
<td>228.31</td>
<td></td>
</tr>
</tbody>
</table>

Table 7.2: Comparison of LP relaxations using Set 3
<table>
<thead>
<tr>
<th>Instance</th>
<th>LP objective value, $LB$</th>
<th>% optimality gap, $100\frac{UB^<em>-LB}{UB^</em>}$</th>
<th>CPU time in secs</th>
<th>Best upper bound, $UB^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(G, P, Q) = (10, 10, 1)$</td>
<td>SPAP1 1.4314</td>
<td>0.02</td>
<td>0</td>
<td>1.4318, 1.4620</td>
</tr>
<tr>
<td></td>
<td>SPAP2 1.4314</td>
<td>0.02</td>
<td>0.04</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPH-A, $\varepsilon = 10^{-2}$ 1.4424</td>
<td>1.34</td>
<td>30.28</td>
<td></td>
</tr>
<tr>
<td>$(G, P, Q) = (10, 10, 2)$</td>
<td>SPAP1 0.3538</td>
<td>1.77</td>
<td>0.01</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPAP2 0.3538</td>
<td>1.77</td>
<td>0.03</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPH-A, $\varepsilon = 10^{-2}$ 0.3752</td>
<td>6.92</td>
<td>40.39</td>
<td>0.3602, 0.4031</td>
</tr>
<tr>
<td>$(G, P, Q) = (10, 10, 3)$</td>
<td>SPAP1 1.0259</td>
<td>7.42</td>
<td>0.01</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPAP2 1.0259</td>
<td>7.42</td>
<td>0.06</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPH-A, $\varepsilon = 10^{-2}$ 1.0602</td>
<td>4.32</td>
<td>162.87</td>
<td>1.1081, 1.1081</td>
</tr>
<tr>
<td>$(G, P, Q) = (10, 10, 4)$</td>
<td>SPAP1 0.5647</td>
<td>22.47</td>
<td>0.03</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPAP2 0.5647</td>
<td>22.47</td>
<td>0.09</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPH-A, $\varepsilon = 10^{-2}$ 0.6091</td>
<td>16.37</td>
<td>4113.63</td>
<td>0.7284, 0.7284</td>
</tr>
<tr>
<td>$(G, P, Q) = (12, 12, 2)$</td>
<td>SPAP1 1.0642</td>
<td>0.29</td>
<td>0.03</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPAP2 1.0642</td>
<td>0.29</td>
<td>0.07</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPH-A, $\varepsilon = 10^{-2}$ 1.0881</td>
<td>2.56</td>
<td>97.02</td>
<td>1.0674, 1.1168</td>
</tr>
<tr>
<td>$(G, P, Q) = (15, 15, 3)$</td>
<td>SPAP1 3.8587</td>
<td>2.08</td>
<td>0.34</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPAP2 3.8587</td>
<td>2.08</td>
<td>0.09</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPH-A, $\varepsilon = 10^{-2}$ 3.8955</td>
<td>1.14</td>
<td>15882.25</td>
<td>1.9407, 3.9407</td>
</tr>
</tbody>
</table>

Table 7.3: Comparison of LP relaxations using Set 4
7.4.3 Comparison of Heuristics

The results reported in Tables 7.4 and 7.5 provide a comparison between the performance of the proposed heuristics SPH-A, SPH-B, SCH, SGH-A, and SGH-B, using the problem instances in Sets 3 and 4. For computing the assured % optimality gap achieved for any test instance by each heuristic in these tables, we have utilized the best lower bound, \( LB^* \), based on Heuristic SPH and Models SPAP1/2 for Sets 3 and 4, respectively (see Tables 7.2 and 7.3 and the discussion in Section 7.4.2 on the LP phase objective value produced by Heuristic SPH using an \( \varepsilon \)-optimality criterion). Procedure SPH-A produced (near-) optimal solutions for all the test problems in Sets 3 and 4 as reported in Tables 7.4 and 7.5. In contrast, Procedure SPH-B performed poorly for the test instances \((G, P, Q) = (5, 15, 1)\) and \((G, P, Q) = (5, 20, 1)\) from Set 3 and, therefore, is not utilized for the more challenging Set 4 test-bed. In fact, the solution produced by this implementation variant turned out to be simply the solution introduced to initialize the column generation algorithm. This shortcoming is overcome by integrating the CCG feature (see Section 7.3.1) within Procedure SPH-A, which consistently produced near-optimal solutions for larger problem instances as shown in Table 7.5. For Set 4, Procedure SPH-A (using \( \varepsilon = 10^{-2} \)) achieved an average improvement in the quality of the solution of 42.16%, 40.7%, and 39.97%, over Procedures SCH, SGH-A, and SGH-B, at the expense of an average increase in the computational effort of 76.63%, 99.49%, and 97.45%, respectively. Using Sets 3 and 4, Procedure SPH-A produced near-optimal solutions within an optimality gap of 0.20% and 8.49% at an average, using \( \varepsilon = 0 \) and \( \varepsilon = 10^{-2} \), respectively.

Our computational experience also reveals the strong computational benefits of Procedure SGH-A over Procedure SGH-B for problem instances having \( P \geq 15 \), whereas Procedure SGH-B computationally outperformed Procedure SGH-A for problem instances having \( P \leq 10 \). In fact, the number of final products, \( P \), is the critical parameter that affects the computational effort required for solving the two-group subassembly parts assignment subproblem that is embedded within any iteration of Procedure SGH.

The results reported in Table 7.5 for the problem instance \((G, P, Q) = (10,10,1)\) demonstrate the effectiveness of Procedure SPH-A in concert with the \( \varepsilon \)-optimality tolerance criterion discussed in Section 7.3.1. When we tried to solve the problem instance \((G, P, Q) = (10,10,1)\) via Heuristic SPH, the procedure was prematurely terminated when the total computational time exceeded 300,000 seconds, out of which approximately 299,930 seconds were consumed in solving Problem SP for the final LP phase iteration. On the other hand, a value of \( \varepsilon = 5 \times 10^{-5} \) achieved an optimality...
gap of 0.02% for this problem while consuming only 76.23 CPU seconds, in comparison with the > 300,000 CPU seconds required for \( \varepsilon = 0 \). (Using \( \varepsilon = 10^{-4} \) and \( \varepsilon = 10^{-3} \) for this test case respectively yielded optimality gaps of 0.04% and 0.19% in 46.21 and 49.05 seconds.) Furthermore, Table 7.5 demonstrates the effect of the \( \varepsilon \)-optimality criterion on the optimality gap achieved and the accompanying computational effort for several other test cases as well. Figure 7.2 depicts the effect of increasing the number of quality characteristics under investigation using the test cases \((10,10,Q)\) for \( Q = 1,2,3, \) and 4, along with a value of \( \varepsilon = 10^{-2} \). Figures 7.3-7.6 display the decrease in the objective value as a function of the number of LP phase iterations of Procedure SPH-A for problem instances \((G,P,Q) = (10,10,1), (10,10,2), (10,10,3), \) and \((10,10,4)\), where each iteration encompasses the generation of \( P \) complementary patterns as prescribed by the CCG feature. The rate of improvement decreases with the iterations performed, and the objective value persistently stalls over several iteration subsequences due to degeneracy. In addition, Figures 7.3-7.6 provide the insight that the computational effort associated with the LP phase iterations presents small-to-moderate variations for the test cases \((G,P,Q) = (10,10,1), (10,10,2), \) and \((10,10,3)\), whereas for the problem instance \((G,P,Q) = (10,10,4)\) the computational effort significantly grows as the iterations progress as depicted in Figure 6(b). That is, for larger values of \( Q \), the average CPU time required for an LP phase iteration grows substantially, and it becomes increasingly computationally burdensome to generate patterns that induce possible marginal improvements in the restricted master program. The computational trends exhibited in Figures 7.2-7.6 were also confirmed by running additional quadruplets of test cases having \((G,P,Q) = (10,10,1), (10,10,2), (10,10,3), \) and \((10,10,4)\), which we do not report here for the sake of brevity.
<table>
<thead>
<tr>
<th>Instance</th>
<th>Objective value</th>
<th>% optimality gap, (100 \frac{\nu - \bar{L}}{\nu} )</th>
<th>CPU time in secs</th>
<th>Optimal objective value</th>
</tr>
</thead>
<tbody>
<tr>
<td>((G, P, Q) = (2, 12, 2))</td>
<td>2.8202</td>
<td>12.85</td>
<td>0.45</td>
<td>2.4576</td>
</tr>
<tr>
<td>SCH</td>
<td>2.4576</td>
<td>0</td>
<td>4.36</td>
<td></td>
</tr>
<tr>
<td>SGH-A</td>
<td>2.4576</td>
<td>0</td>
<td>4.01</td>
<td></td>
</tr>
<tr>
<td>SGH-B</td>
<td>2.4576</td>
<td>0</td>
<td>2.51</td>
<td></td>
</tr>
<tr>
<td>SPH-A</td>
<td>2.4576</td>
<td>0</td>
<td>2.51</td>
<td></td>
</tr>
<tr>
<td>SPH-B</td>
<td>2.4576</td>
<td>0</td>
<td>2.51</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instance</th>
<th>Objective value</th>
<th>% optimality gap, (100 \frac{\nu - \bar{L}}{\nu} )</th>
<th>CPU time in secs</th>
<th>Optimal objective value</th>
</tr>
</thead>
<tbody>
<tr>
<td>((G, P, Q) = (2, 15, 1))</td>
<td>1.1166</td>
<td>22.19</td>
<td>0.58</td>
<td>0.8688</td>
</tr>
<tr>
<td>SCH</td>
<td>0.8688</td>
<td>0</td>
<td>5.55</td>
<td></td>
</tr>
<tr>
<td>SGH-A</td>
<td>0.8688</td>
<td>0</td>
<td>0.34</td>
<td></td>
</tr>
<tr>
<td>SGH-B</td>
<td>0.8688</td>
<td>0</td>
<td>5.00</td>
<td></td>
</tr>
<tr>
<td>SPH-A</td>
<td>0.8688</td>
<td>0</td>
<td>3.11</td>
<td></td>
</tr>
<tr>
<td>SPH-B</td>
<td>0.8688</td>
<td>0</td>
<td>3.11</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instance</th>
<th>Objective value</th>
<th>% optimality gap, (100 \frac{\nu - \bar{L}}{\nu} )</th>
<th>CPU time in secs</th>
<th>Optimal objective value</th>
</tr>
</thead>
<tbody>
<tr>
<td>((G, P, Q) = (3, 10, 1))</td>
<td>1.3368</td>
<td>76.49</td>
<td>0.37</td>
<td>0.7574</td>
</tr>
<tr>
<td>SCH</td>
<td>0.9120</td>
<td>20.41</td>
<td>4.51</td>
<td></td>
</tr>
<tr>
<td>SGH-A</td>
<td>0.9120</td>
<td>20.41</td>
<td>0.14</td>
<td></td>
</tr>
<tr>
<td>SGH-B</td>
<td>0.9120</td>
<td>20.41</td>
<td>0.14</td>
<td></td>
</tr>
<tr>
<td>SPH-A</td>
<td>0.9120</td>
<td>20.41</td>
<td>0.14</td>
<td></td>
</tr>
<tr>
<td>SPH-B</td>
<td>0.9120</td>
<td>20.41</td>
<td>0.14</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instance</th>
<th>Objective value</th>
<th>% optimality gap, (100 \frac{\nu - \bar{L}}{\nu} )</th>
<th>CPU time in secs</th>
<th>Optimal objective value</th>
</tr>
</thead>
<tbody>
<tr>
<td>((G, P, Q) = (3, 15, 1))</td>
<td>2.0831</td>
<td>85.54</td>
<td>0.54</td>
<td></td>
</tr>
<tr>
<td>SCH</td>
<td>1.5016</td>
<td>33.67</td>
<td>6.65</td>
<td></td>
</tr>
<tr>
<td>SGH-A</td>
<td>1.5016</td>
<td>33.67</td>
<td>0.24</td>
<td></td>
</tr>
<tr>
<td>SGH-B</td>
<td>1.5016</td>
<td>33.67</td>
<td>0.24</td>
<td></td>
</tr>
<tr>
<td>SPH-A</td>
<td>1.5016</td>
<td>33.67</td>
<td>0.24</td>
<td></td>
</tr>
<tr>
<td>SPH-B</td>
<td>1.5016</td>
<td>33.67</td>
<td>0.24</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instance</th>
<th>Objective value</th>
<th>% optimality gap, (100 \frac{\nu - \bar{L}}{\nu} )</th>
<th>CPU time in secs</th>
<th>Ratio (\frac{\nu^<em>}{UB^</em>})</th>
</tr>
</thead>
<tbody>
<tr>
<td>((G, P, Q) = (5, 15, 1))</td>
<td>2.4518</td>
<td>47.13</td>
<td>0.70</td>
<td>1.882</td>
</tr>
<tr>
<td>SCH</td>
<td>2.1124</td>
<td>38.63</td>
<td>9.92</td>
<td>1.621</td>
</tr>
<tr>
<td>SGH-A</td>
<td>2.1111</td>
<td>38.60</td>
<td>27.74</td>
<td>1.620</td>
</tr>
<tr>
<td>SGH-B</td>
<td>2.1111</td>
<td>38.60</td>
<td>27.74</td>
<td>1.620</td>
</tr>
<tr>
<td>SPH-A</td>
<td>1.3024</td>
<td>0.47</td>
<td>472.87</td>
<td>1</td>
</tr>
<tr>
<td>SPH-B</td>
<td>5.5838</td>
<td>76.78</td>
<td>472.84</td>
<td>4.287</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instance</th>
<th>Objective value</th>
<th>% optimality gap, (100 \frac{\nu - \bar{L}}{\nu} )</th>
<th>CPU time in secs</th>
<th>Ratio (\frac{\nu^<em>}{UB^</em>})</th>
</tr>
</thead>
<tbody>
<tr>
<td>((G, P, Q) = (5, 20, 1))</td>
<td>2.8505</td>
<td>49.75</td>
<td>0.81</td>
<td>1.976</td>
</tr>
<tr>
<td>SCH</td>
<td>2.0807</td>
<td>31.16</td>
<td>17.72</td>
<td>1.442</td>
</tr>
<tr>
<td>SGH-A</td>
<td>2.1096</td>
<td>32.11</td>
<td>843.14</td>
<td>1.462</td>
</tr>
<tr>
<td>SGH-B</td>
<td>2.1096</td>
<td>32.11</td>
<td>843.14</td>
<td>1.462</td>
</tr>
<tr>
<td>SPH-A</td>
<td>1.4421</td>
<td>0.68</td>
<td>1131.39</td>
<td>1</td>
</tr>
<tr>
<td>SPH-B</td>
<td>5.791</td>
<td>75.26</td>
<td>4733.62</td>
<td>4.015</td>
</tr>
</tbody>
</table>

Table 7.4: Comparison of heuristics using Set 3
<table>
<thead>
<tr>
<th>Instance</th>
<th>Objective value, $\nu$</th>
<th>% optimality gap = $(100 \frac{UB - LB}{UB})$</th>
<th>CPU time in (s)</th>
<th>Ratio $\frac{UB}{LB}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCH</td>
<td>2.8715</td>
<td>50.13</td>
<td>0.50</td>
<td>2.005</td>
</tr>
<tr>
<td>SGL-A</td>
<td>1.9717</td>
<td>27.45</td>
<td>10.07</td>
<td>1.352</td>
</tr>
<tr>
<td>SGL-B</td>
<td>1.9717</td>
<td>27.45</td>
<td>3.07</td>
<td>1.352</td>
</tr>
<tr>
<td>SPH-A ($e = 10^{-2}$)</td>
<td>1.4620</td>
<td>2.09</td>
<td>30.38</td>
<td>1.02</td>
</tr>
<tr>
<td>SPH-A ($e = 10^{-3}$)</td>
<td>1.4342</td>
<td>0.19</td>
<td>49.05</td>
<td>1.001</td>
</tr>
<tr>
<td>SPH-A ($e = 10^{-4}$)</td>
<td>1.4321</td>
<td>0.04</td>
<td>46.21</td>
<td>$\approx 1$</td>
</tr>
<tr>
<td>SPH-A ($e = 5 \times 10^{-5}$)</td>
<td>1.4318</td>
<td>0.02</td>
<td>76.23</td>
<td>1</td>
</tr>
<tr>
<td>SCH</td>
<td>0.5978</td>
<td>40.82</td>
<td>0.89</td>
<td>1.65</td>
</tr>
<tr>
<td>SGL-A</td>
<td>1.0919</td>
<td>67.6</td>
<td>11.67</td>
<td>1.03</td>
</tr>
<tr>
<td>SGL-B</td>
<td>0.9122</td>
<td>61.21</td>
<td>2.09</td>
<td>2.53</td>
</tr>
<tr>
<td>SPH-A ($e = 10^{-2}$)</td>
<td>0.4031</td>
<td>12.23</td>
<td>49.36</td>
<td>1.11</td>
</tr>
<tr>
<td>SPH-A ($e = 10^{-3}$)</td>
<td>0.3602</td>
<td>1.78</td>
<td>470.89</td>
<td>1</td>
</tr>
<tr>
<td>SCH</td>
<td>0.5799</td>
<td>45.60</td>
<td>0.70</td>
<td>1.83</td>
</tr>
<tr>
<td>SGL-A</td>
<td>1.5799</td>
<td>35.07</td>
<td>16.38</td>
<td>1.42</td>
</tr>
<tr>
<td>SGL-B</td>
<td>1.5799</td>
<td>35.07</td>
<td>2.79</td>
<td>1.42</td>
</tr>
<tr>
<td>SPH-A ($e = 10^{-2}$)</td>
<td>1.1081</td>
<td>7.42</td>
<td>180.48</td>
<td>1</td>
</tr>
<tr>
<td>SCH</td>
<td>1.1051</td>
<td>48.90</td>
<td>79.95</td>
<td>1.51</td>
</tr>
<tr>
<td>SGL-A</td>
<td>2.7712</td>
<td>79.62</td>
<td>14.29</td>
<td>3.8</td>
</tr>
<tr>
<td>SGL-B</td>
<td>2.7712</td>
<td>79.62</td>
<td>4.07</td>
<td>3.8</td>
</tr>
<tr>
<td>SPH-A ($e = 10^{-2}$)</td>
<td>0.7284</td>
<td>22.47</td>
<td>4345.8</td>
<td>1</td>
</tr>
<tr>
<td>SCH</td>
<td>1.1168</td>
<td>48.03</td>
<td>79.95</td>
<td>1.50</td>
</tr>
<tr>
<td>SGL-A</td>
<td>1.0704</td>
<td>30.58</td>
<td>1291.26</td>
<td>$\approx 1$</td>
</tr>
<tr>
<td>SGL-B</td>
<td>1.0704</td>
<td>30.58</td>
<td>8.23</td>
<td>1.50</td>
</tr>
<tr>
<td>SPH-A ($e = 10^{-2}$)</td>
<td>1.0674</td>
<td>4.71</td>
<td>157.68</td>
<td>1.04</td>
</tr>
<tr>
<td>SPH-A ($e = 10^{-3}$)</td>
<td>0.5894</td>
<td>0.58</td>
<td>1291.26</td>
<td>$\approx 1$</td>
</tr>
<tr>
<td>SCH</td>
<td>1.1168</td>
<td>48.03</td>
<td>79.95</td>
<td>1.50</td>
</tr>
<tr>
<td>SGL-A</td>
<td>1.0704</td>
<td>30.58</td>
<td>1291.26</td>
<td>$\approx 1$</td>
</tr>
<tr>
<td>SGL-B</td>
<td>1.0704</td>
<td>30.58</td>
<td>8.23</td>
<td>1.50</td>
</tr>
<tr>
<td>SPH-A ($e = 10^{-2}$)</td>
<td>1.0674</td>
<td>4.71</td>
<td>157.68</td>
<td>1.04</td>
</tr>
<tr>
<td>SPH-A ($e = 10^{-3}$)</td>
<td>0.5894</td>
<td>0.58</td>
<td>1291.26</td>
<td>$\approx 1$</td>
</tr>
</tbody>
</table>

Table 7.5: Comparison of heuristics using Set 4
Figure 7.2: CPU time growth for Procedure SPH-A as a function of $Q$ using $\varepsilon = 10^{-2}$

Figure 7.3: Convergence of Procedure SPH-A for problem instance $(G, P, Q) = (10, 10, 1)$ using $\varepsilon = 10^{-2}$
Figure 7.4: Convergence of Procedure SPH-A for problem instance \((G, P, Q) = (10,10,2)\) using \(\varepsilon = 10^{-2}\)

Figure 7.5: Convergence of Procedure SPH-A for problem instance \((G, P, Q) = (10,10,3)\) using \(\varepsilon = 10^{-2}\)
Figure 7.6: Convergence of Procedure SPH-A for problem instance \((G, P, Q) = (10, 10, 4)\) using \(\varepsilon = 10^{-2}\)

### 7.5 Summary and Conclusions

We have investigated in this chapter the subassembly parts assignment problem (SPAP) that aims at composing final assemblies in a fashion that minimizes the total deviation from a vector of nominal and mean values of certain quality characteristics that are of interest to the manufacturer in the assembled products. Upon recognizing the symmetry inherent within this problem, we have revisited and enhanced an earlier assignment-based formulation (SPAP1) [92] via symmetry-defeating strategies. The latter either incorporate tailored hierarchical constraints (Models SPAP1(\(\sigma\)) and SPAP1(\(\phi\))) that impart specific identities to sets of originally indistinguishable variables, or a priori assign (without loss of optimality) the subassembly parts that belong to a single group to the final products (Model SPAP2). Although Sherali and Smith [124] have discussed the conceptual and computational benefits of appending such symmetry-defeating constraints, they have also called for caution against the negative effects that might result from altering certain specially structured formulations by enforcing general hierarchical constraints. Indeed our study reveals that whereas our proposed symmetry-defeating enhanced formulation SPAP2 significantly outperforms Model SPAP1, both the sets of hierarchical constraints (\(\sigma\)) and (\(\phi\)) consistently deteriorate the algorithmic performance. Despite their imposing unique identities to the products under assembly, such (\((\sigma)\) and (\(\phi\))) hierarchical constraints do not preserve the structure underlying the assignment-based
formulations, thereby rendering the popular mathematical programming software (CPLEX 10.1) inefficient. In contrast, Model SPAP2 offers an appealing alternative that combines the benefits of combating the inherent problem symmetry and preserving the problem structure.

In addition, we have introduced a set partitioning-based model formulation along with a column generation heuristic (SPH) that integrates a novel concept of complementary column generation (CCG). Our computational experience shows that this proposed procedure (SPH-A) consistently produces near-optimal solutions to relatively large problem instances, while the most promising assignment-based formulation, SPAP2, becomes intractable even for certain moderately-sized test problems. In order to hedge against the slow tail-end convergence that is witnessed at the final iteration in the LP phase of Heuristic SPH, we have proposed a practical termination criterion that reveals a substantial savings in computational effort for large-scale problems without compromising the quality of the solution produced.

We have also developed and tested two additional heuristic procedures, namely, Procedure SCH, which iteratively constructs products one at a time, and Procedure SGH, which adopts a sequential group-based strategy in simultaneously composing the final assemblies. The proposed column generation approach SPH-A strongly dominated both these procedures. For large-sized problem instances (in Set 4), Procedure SPH-A (using $\varepsilon = 10^{-2}$) achieved an average improvement in the quality of the solution of 42.16%, 40.7%, and 39.97%, over Procedures SCH, SGH-A, and SGH-B, at the expense of an average increase in the computational effort of 76.63%, 99.49%, and 97.45%, respectively. It is worth noting that for moderately-sized test instances (Set 3), Procedure SPH-A produced near-optimal solutions within 0.20% optimality gap in average (using $\varepsilon = 0$), whereas this heuristic approach achieved an optimality gap that is less than 8.49% in average (using $\varepsilon = 10^{-2}$) for large-sized problem instances (Set 4).
Chapter 8

Models and Algorithms for the Scheduling of a Doubles Tennis Training Tournament

In this chapter, we address a doubles tennis scheduling problem in the context of a training tournament, and develop two alternative 0-1 mixed-integer programming models with various objective functions that attempt to balance the partnership and the opponentship pairings among the players. Our analysis and computational experience demonstrate the superiority of one of these models over the other, and reflect the importance of model structure in formulating discrete optimization problems. Furthermore, we propose effective symmetry-defeating strategies that impose certain decision hierarchies within the models, which serve to significantly enhance algorithmic performance. In particular, our study provides the insight that the special structure of the mathematical program to which specific tailored symmetry-defeating constraints are appended can greatly influence their pruning effect. We also propose a nonpreemptive multi-objective programming strategy in concert with decision hierarchies, and highlight its effectiveness and conceptual value. Finally, various specialized heuristics are devised and are computationally evaluated along with the exact solution schemes using a set of realistic practical test problems.

8.1 Introduction

Sports management is an active arena of investigation that prompts a broad range of operations research applications [60]. These include tournament scheduling, capacity planning, and network optimization problems for cities hosting sports events, to name a few. In general, sports scheduling problems aim at finding feasible solutions in compliance with complex time and/or capacity require-
ments. The related problems involve some form of optimization such as maximizing fairness, or minimizing the total distance traveled or the fatigue experienced by players. Many sports leagues need to consider the problem of round-robin tournaments with additional constraints pertaining to the so-called home-away patterns.

In this chapter, we investigate a doubles tennis scheduling problem in the context of a training tournament, as prompted by a tennis club in Virginia, USA. The tournament involves $n$ players and $r$ rounds (or training days), where some $p$ ($= 0 \mod 4$) out of the $n$ players (with possibly $p = n$) can participate in each round due to restrictions on available resources (courts and/or coaches). It is desirable to build a fair schedule, where fairness is reflected by the partnership and the opponent-ship frequency for each pair of players, and the number of games played by each player. That is, the number of games where a certain player has another one as his/her partner or opponent should be balanced for all players, and all players should have the opportunity to play the same number of games, if possible. All courts are assumed to be identical in this process, but could be subsequently rotated among the players via a separate problem once the composition and schedule of games are determined.

To the best of our knowledge, the problem under investigation has not been studied in the literature. However, we provide here a brief overview of the existing literature on sports scheduling problems in general, since some underlying concepts and results are also beneficial in our context. Such models and approaches have been developed for various sports such as soccer [105], basketball ([14], [59], [93]), ice-hockey [49], table-tennis ([103],[104]), and tennis [36]. The classical case of round-robin tournaments and some of its variants have been extensively studied in the light of graph-theoretic approaches ([143]-[148]). In many applications, the problem is stated as a complex constraint satisfaction problem ([59], [93]). In other cases, challenging optimization problems emerge with performance measures or cost functions based on minimizing the number of breaks in play at the same location, whether at home or away, or minimizing the total distance traveled ([14], [49]).

Due to the size and complexity of sports scheduling problems, exact solution methods are believed to be of limited use in practice. Thus, most algorithmic approaches for tournament scheduling tend to rely on efficient heuristics. Various two-phase methods have been designed to generate a tournament pattern using virtual teams in the first phase, and subsequently, invoking actual constraints for individual teams in the second phase ([49], [93]). Metaheuristics have been implemented
to solve various temporally relaxed sports scheduling problems. These efforts include the use of tabu search [58], hybrid tabu search and genetic algorithms ([33], [103]), and simulated annealing [150]. Suzuka et al. [133] show that the break minimization problem can be formulated as a variation of the minimum cut problem, and subsequently tackle it with a semidefinite programming-based approximation algorithm due to Goemans and Williamson [54]. Approaches based on constraint programming are gaining popularity for temporally constrained problems [59]. Exploiting the complementary strengths of integer and constraint programming is a promising and active trend in research dealing with the solution of sports scheduling problems, among other combinatorial optimization problems [138].

The remainder of this chapter is organized as follows. Section 8.2 introduces our notation, and presents two alternative mathematical programming models of interest. In Section 8.3, we derive various types of effective symmetry-defeating constraints that drastically enhance the solvability of the proposed models, and present nonpreemptive multi-objective programming formulations based on, and conjugated with, such aggregate decision hierarchies. A greedy heuristic, two sequential round-based heuristics, and a two-phase heuristic are presented in Section 8.4. Next, we provide some computational experience in Section 8.5 to assess the relative merits of the proposed models and to exhibit the benefits of the different classes of symmetry-defeating constraints developed for both exact solutions methods and heuristics. Finally, Section 8.6 concludes the chapter with a summary and directions for future research.

8.2 Alternative Models

In this section, we formulate two alternative models for the doubles tennis scheduling problem. To this end, consider the following notation that is common to both models.

- \( n \): total number of players.
- \( p \): number of players in any round; \( p = 0 \mod 4 \), and \( p \leq n \).
- \( r \): number of rounds in the tournament.
- \( j = 1, ..., n \): index set of players.
- \( i = 1, ..., \bar{n} \equiv \frac{n(n-1)}{2} \): index set of pairs of players.
- \( k = 1, ..., r \): index set of rounds.
8.2.1 Model 1

Our first 0-1 integer programming model is formulated as follows.

Additional notation for Model 1

- $x_{j_1j_2k} \in \{0, 1\}; x_{j_1j_2k} = 1$ if and only if players $j_1$ and $j_2$, $j_1 < j_2$, are partners in round $k$.
- $y_{j_1j_2k} \in \{0, 1\}; y_{j_1j_2k} = 1$ if and only if players $j_1$ and $j_2$, $j_1 < j_2$, are opponents in round $k$.
- For convenience, let $z_{j_1j_2} = \sum_{k=1}^{r} x_{j_1j_2k}$ and $w_{j_1j_2} = \sum_{k=1}^{r} y_{j_1j_2k}, \forall j_1 < j_2$.
- Let $\bar{z} = \frac{2}{n(n-1)} \sum_{j_1=1, j_2=j_1+1}^{n-1} z_{j_1j_2}$ be the average partnership frequency. Note that $\bar{z}$ is a constant, since $\bar{z} = \frac{2}{n(n-1)} \sum_{j_1=1, j_2=j_1+1}^{n-1} \sum_{k=1}^{r} x_{j_1j_2k} = \frac{rp}{n(n-1)}$.
- Similarly, let $\bar{w} = \frac{2}{n(n-1)} \sum_{j_1=1, j_2=j_1+1}^{n-1} \sum_{k=1}^{r} w_{j_1j_2} = \frac{2rp}{n(n-1)}$ be the average opponentship frequency.
- $z_{\max} = \max_{j_1<j_2} z_{j_1j_2}$ and $z_{\min} = \min_{j_1<j_2} z_{j_1j_2}$.
- $[jj']$: represents $jj'$ if $j < j'$, and represents $j'j$ if $j' < j$, for any pair of players $j \neq j'$.
- $v_{j_1j_2} = j_1 \times j_2$ (product of player indices; this will be useful later in Section 8.3), $\forall j_1 < j_2$.

Ideally, it is desirable to have $z_{j_1j_2} = \bar{z}$ and $w_{j_1j_2} = \bar{w}$, $\forall j_1 < j_2$. However, specific values of the parameters $(n, p, r)$ may not permit such a perfect balance. Thus, the objective of the underlying optimization problem is essentially two-fold: we would like to minimize the deviation from both the average partnership value, $\bar{z}$, and the average opponentship value, $\bar{w}$. Appropriate objective functions (to be minimized) include $f_1 = \max_{j_1<j_2} \{ \max_{j_1 < j_2} |z_{j_1j_2} - \bar{z}|, \max_{j_1 < j_2} |w_{j_1j_2} - \bar{w}| \}$, $f_2 = \max_{j_1<j_2} |z_{j_1j_2} - \bar{z}| + \max_{j_1<j_2} |w_{j_1j_2} - \bar{w}|$, and $f_3 = \sum_{j_1=1, j_2=j_1+1}^{n-1} \sum_{j_1=1, j_2=j_1+1}^{n-1} (|z_{j_1j_2} - \bar{z}| + |w_{j_1j_2} - \bar{w}|)$. It can be argued that minimizing such objective functions also implicitly tends to ensure that the players will participate in roughly the same number of games in the tournament. We formulate the following 0-1 programs, denoted $P_{1,h}$ based on the choice of the foregoing objective function $f_h, h = 1, 2, 3$. (We assume throughout that while implementing any model having such objective functions, the latter are linearized by introducing appropriate continuous variables and accompanying relational constraints.)

\[ P_{1,h}: \text{Minimize } f_h \]
subject to  
\[
\sum_{j_1=1}^{n-1} \sum_{j_2=j_1+1}^{n} x_{j_1,j_2k} = \frac{p}{2}, \quad \forall k \tag{8.1b}
\]
\[
\sum_{j_2=1, j_2 \neq j_1}^{n} x_{j_1,j_2k} \leq 1, \quad \forall j_1, \forall k \tag{8.1c}
\]
\[
x_{j_1,j_2k} + y_{j_1,j_2k} \leq 1, \quad \forall j_1 < j_2, \forall k \tag{8.1d}
\]
\[
x_{j_1,j_2k} + y_{j_1,j_3k} - y_{j_2,j_3k} \leq 1, \quad \forall j_1 \neq j_2, \forall j_3 \notin \{j_1,j_2\}, \forall k \tag{8.1e}
\]
\[
\sum_{j_2=1, j_2 \neq j_1}^{n} y_{j_1,j_2k} = 2 \sum_{j_2=1, j_2 \neq j_1}^{n} x_{j_1,j_2k}, \quad \forall j_1, \forall k \tag{8.1f}
\]
\[
x_{j_1,j_2k}, y_{j_1,j_2k} \in \{0,1\}, \quad \forall j_1 < j_2, \forall k. \tag{8.1g}
\]

Constraints (8.1b) enforce the requirement that exactly \( p \) players participate in every round, while Constraints (8.1c) ensure that any player can be involved in at most one team, in each round. Constraints (8.1d)-(8.1f) introduce additional logical specifications that are implicitly required for any feasible schedule. Consistency Constraints (8.1d) guarantee that no two players are paired as both partners and opponents in any round. Transitivity Constraints (8.1e) assert that for any pair of players \( j_1 \neq j_2 \), and for any \( j_3 \) different from \( j_1 \) and \( j_2 \), if \( j_1 \) and \( j_2 \) are partners and \( j_3 \) is an opponent for \( j_1 \) in any round \( k \), then necessarily in this round, \( j_2 \) and \( j_3 \) must be opponents. Constraints (8.1f) enforce that if player \( j_1 \) does not participate in some round (zero right-hand side), then the number of opponents for this player in that round must equal 0; otherwise it should equal 2. Finally, Constraints (8.1g) record the required logical binary restrictions.

**Remark 8.1.** Observe that the following Constraints (8.1h) enforce a minimal set of sufficient transitivity relationships, and are subsumed by Constraints (8.1e) that introduce certain additional logical transitivity restrictions in order to tighten the continuous relaxation. We shall denote the reduced formulation where (8.1h) are substituted in lieu of (8.1e) by Problem \( P_{1,h}^{(R)} \), and we shall use Problem \( P_{1,h}^{(R)} \) to refer to either of the problems \( P_{1,h} \) or \( P_{1,h}^{(R)} \).

\[
x_{j_1,j_2k} + y_{j_1,j_3k} - y_{j_2,j_3k} \leq 1, \quad \forall j_1 < j_2, \forall j_3 \notin \{j_1,j_2\}, \forall k \tag{8.1h}
\]
8.2.2 Model 2

In this section, we provide an alternative formulation for the doubles tennis scheduling problem. While the focus in Model 1 is on pairing players, Model 2 views the problem at an aggregate level, and attempts to pair teams in a balanced fashion.

Additional notation for Model 2

- \( S_j = \{i: \text{player } j \text{ is part of pair } i\}, \forall j = 1,...,n \).
- \( \bar{p} \equiv \frac{p}{4} \): number of games in each round.
- \( C = \{(i_1, i_2): \text{pairs } i_1 < i_2 \text{ are compatible with respect to each other in that (players in } i_1 \text{) } \cap \text{(players in } i_2 \text{) } = \emptyset \} \). Thus, \( C \) is the set of all feasible games.
- \( O_{j_1j_2} = \{(i_1, i_2) \in C: \text{player } j_1 \text{ is in pair } i_1 \text{ and player } j_2 \text{ is in pair } i_2, \text{ or vice versa}\}, \forall j_1 < j_2 \). Therefore, \( O_{j_1j_2} \) provides the set of feasible games where players \( j_1 \) and \( j_2 \) are opponents.
- \( t_{i_1i_2k} \in \{0,1\}, \forall (i_1, i_2) \in C, \forall k; t_{i_1i_2k} = 1 \text{ if and only if } (i_1, i_2) \in C \text{ play a match in round } k \).
- \( s_{ik} \in \{0,1\}, \forall i = 1,...,\bar{n}, \forall k; s_{ik} = 1 \text{ if and only if pair } i \text{ is selected to play in round } k \).
- \( y_{j_1j_2k}, \forall j_1 < j_2, \forall k, \) is defined as for Model 1.
- \([ii']\): represents \( ii' \) if \( i < i' \), and represents \( i'i \) if \( i' < i \), for any pair of teams \( i \neq i' \). Similarly, we define \([(i,i')]\) as \( (i,i') \) if \( i < i' \), and \((i',i)\) otherwise.
- \( v_i \): product of player indices for players in pair \( i \) (this will be used later in Section 8.3).

For Model 2, we derive various zero-one programming formulations, \( P_{2,h}, h = 1,2,3, \) that share a common set of Constraints (8.2b)-(8.2f), while being characterized by specific objective functions, \( g_h, h = 1,2,3, \) which are respectively equivalent to \( f_h, h = 1,2,3, \) and are defined as follows: \( g_1 = \max \{ \max_{i} \left| \sum_{k=1}^{r} s_{ik} - \bar{z} \right|, \max_{j_1<j_2} \left| \sum_{k=1}^{r} y_{j_1j_2k} - \bar{w} \right| \}, g_2 = \max_{i} \left| \sum_{k=1}^{r} s_{ik} - \bar{z} \right| + \max_{j_1<j_2} \left| \sum_{k=1}^{r} y_{j_1j_2k} - \bar{w} \right|, g_3 = \sum_{i=1}^{\bar{n}} \left| \sum_{k=1}^{r} s_{ik} - \bar{z} \right| + \sum_{j_1=1}^{n-1} \sum_{j_2=j_1+1}^{n} \left| \sum_{k=1}^{r} y_{j_1j_2k} - \bar{w} \right| \). (Again, these objective functions/models are linearized in implementations.)

\[
P_{2,h}: \text{Minimize } g_h \quad \text{subject to } \sum_{(i_1,i_2) \in C} t_{i_1i_2k} = \bar{p}, \forall k \quad (8.2a)
\]
\[ s_{ik} = \sum_{i':[(i,i')]} C t_{i'i'k}, \quad \forall i, \forall k \tag{8.2c} \]

\[ \sum_{i \in S_j} s_{ik} \leq 1, \quad \forall j, \forall k \tag{8.2d} \]

\[ y_{j_1 j_2 k} = \sum_{(i_1,i_2) \in O_{j_1 j_2}} t_{i_1 i_2 k}, \quad \forall j_1 < j_2, \forall k \tag{8.2e} \]

\[ t_{i_1 i_2 k} \in \{0,1\}, \quad \forall (i_1,i_2) \in C, \forall k. \tag{8.2f} \]

In Model 2, Constraints (8.2b) enforce that the total number of games played in any round equals \( \bar{p} \equiv \frac{p}{4} \). Constraints (8.2c) relate the \( s \)- and \( t \)-variables, while Constraints (8.2d) ensure that any player is involved in at most one team in each round. Note that Constraints (8.2c), (8.2d), and (8.2f) implicitly enforce the binariness of variables \( s_{ik} \), and hence assure that each team plays at most one game in any round. Constraints (8.2e) provide an explicit definition of the \( y \)-variables in terms of the \( t \)-variables, and Constraints (8.2f) impose binary logical restrictions on the principal \( t \)-variables.

**Remark 8.2.** Constraints (8.2c) and (8.2e) may be retained for structure or clarity in the model. However, it is evident that the \( s \)- and \( y \)-variables can be eliminated from the formulation, and Model 2 can be entirely defined in terms of the \( t \)-variables. We refer to such a compact reformulation in terms of the \( t \)-variables alone as \( P_{t2,h}^f \), for each \( h = 1, 2, 3 \), and we shall study the effect of omitting these constraints and variables on the solver performance in Section 8.5. (Again, we shall use the notation \( P_{t2,h}^{(t)} \) when we wish to jointly refer to either of the problems \( P_{2,h} \) or \( P_{2,h}^f \).)

**Remark 8.3.** Note that the models \( P_{1,h}^{(R)} \) and \( P_{2,h}^{(t)} \), \( h = 1, 2, 3 \), are equivalent in the sense of 0-1 feasible solutions, but possess different structures. While the elementary component that motivates the structure of Model 1 is at the player’s level, the focus in Model 2 is on teams and feasible games. This aggregate view of the problem that Model 2 offers implicitly eliminates (via a suitable definition of the set \( C \)) logical requirements specified through Constraints (8.1d)-(8.1f) in Model 1.

The relative sizes of the foregoing model formulations are summarized in Table 8.1 in terms of the number of binary variables and constraints.
8.3 Enhancements Based on Symmetry Considerations

8.3.1 Symmetry-defeating Constraints

From the perspective of solving the proposed 0-1 programming formulations to optimality, it is important to recognize the symmetries inherent in the structure of the problem. In our context, there exists a symmetry among players, pairs of players, and rounds. Sherali and Smith [124] expound on the concept of how a branch-and-bound solver can be encumbered by the presence of such symmetries, and recommend the use of decision hierarchies within the model in order to enhance solvability. In this spirit, we propose specialized aggregate hierarchical constraints to characterize players and/or rounds, and hence, significantly reduce the set of alternative optimal solutions. For instance, any permutation of the rounds and/or a re-indexing of the players in any feasible schedule for the problem would produce an alternative equivalent feasible solution that has the same objective value. To counter such symmetric reflections, we can impose the hierarchical constraints denoted \( (\delta_1) \) below that tend to impart specific identities to rounds in Model 1, along with hierarchical constraints \( (\delta_2) \) or \( (\delta_3) \) that alternatively attempt to distinguish among the different players in Model 1.

- Decision hierarchy \( (\delta_1) \):
  \[
  \sum_{j_1=1}^{n-1} \sum_{j_2=j_1+1}^{n} v_{j_1,j_2} x_{j_1,j_2,k} \leq \sum_{j_1=1}^{n-1} \sum_{j_2=j_1+1}^{n} v_{j_1,j_2} x_{j_1,j_2,k+1}, \quad \forall k = 1, \ldots, r - 1.
  \]

- Decision hierarchy \( (\delta_2) \):
  \[
  \sum_{k=1}^{r} \sum_{j_2=1}^{n} x_{[j_1,j_2]k} \leq \sum_{k=1}^{r} \sum_{j_2=1}^{n} x_{[j_1+1,j_2]k}, \quad \forall j_1 = 1, \ldots, n - 1.
  \]

- Decision hierarchy \( (\delta_3) \):
  \[
  \sum_{k=1}^{r} \sum_{j_2=1}^{n} k v_{[j_1,j_2]} x_{[j_1,j_2]k} \leq \sum_{k=1}^{r} \sum_{j_2=1}^{n} k v_{[j_1+1,j_2]} x_{[j_1+1,j_2]k}, \quad \forall j_1 = 1, \ldots, n - 1.
  \]

Note that we can compose \( (\delta_1) \) with either \( (\delta_2) \) or \( (\delta_3) \) for Model 1. Similarly, we define the following aggregate decision hierarchies for Model 2, where we can conjugate \( (\pi_1) \) with either \( (\pi_2) \) or \( (\pi_3) \).

<table>
<thead>
<tr>
<th></th>
<th>( P_{1,h} )</th>
<th>( P_{1,R}^h )</th>
<th>( P_{2,h} )</th>
<th>( P_{2,R}^h )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of 0-1 variables</td>
<td>( rn(n-1) )</td>
<td>( rn(n-1) )</td>
<td>( r[3C_4^n + n(n-1)] )</td>
<td>( 3rC_4^n )</td>
</tr>
<tr>
<td>Number of constraints</td>
<td>( r(n^3 - \frac{5n^2}{2} + \frac{7n}{2} + 1) )</td>
<td>( r(n^3 - n^2 + \frac{7n}{2} + 1) )</td>
<td>( r(1+n^2) )</td>
<td>( r(1+n) )</td>
</tr>
</tbody>
</table>

Table 8.1: Size comparison of the formulations based on Models 1 and 2.
- Decision hierarchy ($\pi_1$): \[ \sum_{(i_1,i_2) \in C} v_{i_1} v_{i_2} r_{i_1 i_2 k} \leq \sum_{(i_1,i_2) \in C} v_{i_1} v_{i_2} r_{i_1 i_2 (k+1)}, \quad \forall k = 1,\ldots,r - 1. \]

- Decision hierarchy ($\pi_2$): \[ r \sum_{k=1}^{r} \sum_{i \in S_j} s_{i k} \leq \sum_{k=1}^{r} \sum_{i \in S_{j+1}} s_{i k}, \quad \forall j = 1,\ldots,n - 1. \]

- Decision hierarchy ($\pi_3$): \[ \sum_{k=1}^{r} \sum_{i \in S_j} k v_i s_{i k} \leq \sum_{k=1}^{r} \sum_{i \in S_{j+1}} k v_i s_{i k}, \quad \forall j = 1,\ldots,n - 1. \]

Let us denote the symmetry-defeating-based formulations $P^{(R)}_{1,h}$ and $P^{(t)}_{2,h}$ that are augmented with the foregoing symmetry-defeating constraints by $P^{(R)}_{1,h}(\delta)$ and $P^{(t)}_{2,h}(\pi)$, for $h = 1,2,3$, where $(\delta) \in \{ (\delta_1,\delta_2), (\delta_1,\delta_3) \}$ and $(\pi) \in \{ (\pi_1,\pi_2), (\pi_1,\pi_3) \}$. As we shall see in our numerical results presented in Section 8.5, the inclusion of such decision hierarchies within the model proves critical in enhancing the solvability of the problem.

### 8.3.2 Symmetry Compatible Formulations

The concept of symmetry compatible formulations plays a crucial role in the effectiveness of implementing any particular symmetry-defeating constraints. The basic idea here is that if the root node LP solution of the original formulation does not satisfy the proposed decision hierarchies, then a significant branching effort might be required to identify a good solution that is compatible with the selected symmetry-defeating constraints. Therefore, it is advisable to enforce decision hierarchies that conform as much as possible with the initial LP relaxation. In this spirit, an alternative strategy is to first derive some feasible solution that is close to the LP root node solution by means of a heuristic and, subsequently, to impose a hierarchy that is compatible with that solution, and to then solve the resulting model. In Section 8.5.2, we demonstrate how symmetry-defeating constraints might have negative effects on a formulation if this concept of symmetry compatible formulations is overlooked.

### 8.3.3 Nonpreemptive Multi-objective Programming Strategy

Let $P(\pi)$ generically denote any of the foregoing symmetry-enhanced formulations pertaining to Model 2, where $\pi$ includes the symmetry-defeating constraints ($\pi_1$) related to the rounds, and one of the player-induced symmetry-defeating hierarchies ($\pi_2$ or $\pi_3$). For instance, if we take $\pi \equiv (\pi_1,\pi_2)$, then Problem $P(\pi)$ refers to any of the problems $P^{(t)}_{2,h}(\pi_1,\pi_2)$, $h = 1,2,3$. For any such Problem $P(\pi)$, let $F_0$ designate the original objective function, and denote the round-based decision hierarchy functions as $F_1,\ldots,F_r$, and the player-based decision hierarchy functions as $F_{r+1},\ldots,F_{r+n}$, where the symmetry-defeating decision hierarchy constraints ($\pi$) effectively enforce the following
functional relationships: \( F_1 \leq F_2 \leq \ldots \leq F_r \), and \( F_{r+1} \leq F_{r+2} \leq \ldots \leq F_{r+n} \).

By arbitrarily prioritizing the player-based hierarchical objective functions after the round-based ones, the resulting problem \( P(\pi) \) can alternatively be viewed as a multi-objective preemptive programming problem that imposes the following relative priority ranking of the objective functions: \( F_0 \gg F_1 \gg \ldots \gg F_{r+n} \). Sherali and Soyster [126] show that such a preemptive formulation can be cast into an equivalent nonpreemptive multi-objective programming problem that seeks to minimize the weighted aggregate objective function \( \sum_{k=0}^{r+n} \mu_k F_k \), where \( \mu_k, k = 0, \ldots, r + n \), are suitable nonnegative weights. We denote this aggregated problem that is predicated on a vector of weights \( \mu \) as \( AP^\mu(\pi) \), where the symmetry-defeating constraints are weighted in the aggregate objective function, while being explicitly retained in the set of constraints as in \( P(\pi) \). Observe that Problem \( AP^\mu(\pi) \) can potentially assist in sharply differentiating the feasible solutions (that satisfy the selected symmetry-defeating constraints) via their relative impact on the weighted hierarchy of terms introduced in the objective function. Furthermore, it is conceivable to investigate nonpreemptive formulations in the spirit of \( AP^\mu(\pi) \), but where the symmetry-defeating constraints are not enforced within the set of constraints. However, preliminary computational tests indicate that the latter formulations are more computationally intensive than \( AP^\mu(\pi) \) or \( P(\pi) \).

**Remark 8.4.** Sherali and Soyster [126] propose a theoretical derivation of a set of weights \( \mu \) that would make the nonpreemptive multi-objective programming formulation equivalent to the preemptive model. However, the constructed weights can typically become very large in practice for high values of \( r + n \). From this perspective, note that we can arbitrarily select a suitable decreasing set of weights \( \mu_1 > \mu_2 > \ldots > \mu_{r+n} \), and given any such choice, it is only essential then to select \( \mu_0 \) such that the value of \( F_0 \) produced by \( AP^\mu(\pi) \) turns out to be optimal to the original problem \( P(\pi) \). Hence, define \( \tilde{F} = \sum_{k=1}^{r+n} \mu_k F_k \) and consider the aggregate objective function \( \mu_0 F_0 + \tilde{F} \) that arises in Problem \( AP^\mu(\pi) \). Let \( \tilde{F}_{\text{max}} \geq \max\{\tilde{F} : \text{Constraints in Problem } P(\pi)\} \) and \( \tilde{F}_{\text{min}} \leq \min\{\tilde{F} : \text{Constraints in Problem } P(\pi)\} \). Observe that, for practical purposes, we could simply set \( \tilde{F}_{\text{min}} = 0 \) and select \( \tilde{F}_{\text{max}} \) equal to the value of the LP relaxations to the problem \( \max\{\tilde{F} : \text{Constraints in Problem } P(\pi)\} \). Furthermore, compute \( \Delta \) as (a possible lower bound on) the smallest possible deviation from an optimal value for \( F_0 \). Observe that for problem instances where \( \bar{z} \) is integral (hence, so is \( \bar{w} \)), \( \Delta \) can assume the value of 1. On the other hand, if \( \bar{z} \) and \( \bar{w} \) are both fractional, then we can set \( \Delta = \min\{\alpha_{\bar{z}}, 1 - \alpha_{\bar{z}}, \alpha_{\bar{w}}, 1 - \alpha_{\bar{w}}\} \), where \( \alpha_{\bar{z}} = \bar{z} - \lfloor \bar{z} \rfloor \) and \( \alpha_{\bar{w}} = \bar{w} - \lfloor \bar{w} \rfloor \), and in case only \( \bar{z} \) is fractional, then we can set \( \Delta = \min\{\alpha_{\bar{z}}, 1 - \alpha_{\bar{z}}\} \).
Proposition 8.1. If we select \( \mu_0 > \frac{\tilde{F}_{\text{max}} - \tilde{F}_{\text{min}}}{\Delta} \), where \( F_{\text{max}}, F_{\text{min}}, \) and \( \Delta \) are described in Remark 8.4 above, then the value of \( F_0 \) obtained upon solving \( AP^\mu(\pi) \) is the optimal objective value to \( P(\pi) \).

Proof. Follows directly from Sherali and Sosyter [126]. \( \square \)

Proposition 8.1 provides a constructive approach to derive the weight \( \mu_0 \) in concert with a decreasing hierarchy \( \mu_k, k = 1, ..., r + n \), in order to guarantee that the contribution of \( F_0 \) to the optimal objective value of \( AP^\mu(\pi) \) is optimal to \( P(\pi) \) as well. The use of this strategy is demonstrated in Section 8.5.3, where a specific decreasing hierarchy of weights (see Equation (8.5)) is empirically prescribed for \( \mu_k, k = 1, ..., r + n \), to be used along with a derivation of \( \mu_0 \) based on Proposition 8.1.

8.4 Heuristics

Our preliminary computational experience (reported in Section 8.5) revealed that solving Model 1 is computationally prohibitive for moderate-sized problem instances, and also, Model 2 becomes computationally intractable for somewhat larger, realistic problem instances. We therefore develop below four heuristics based on the two models proposed in Section 8.2 with the intent of achieving a good compromise between the quality of the solution derived and the accompanying computational effort.

8.4.1 Greedy Heuristic

This is a fast greedy heuristic (GH) that aims at balancing the overall player partnership and opponentship frequencies, and is executed sequentially for rounds 1, ..., \( r \) as follows.

Step 1. Select the players for the current round from the pool of candidate players based on the number of previous rounds played by each player. In an iterative fashion, the \( p \) players who have played the least number of rounds so far are selected.

Step 2. Match the selected players into pairs as follows. For each unpaired player, choose a partner from the list of selected, unpaired players who has partnered him/her for the least number of rounds.

Step 3. Match the identified partner pairs into games according to the total number of times any two partnered pairs have played against each other in previous rounds. The two pairs having the smallest sum of the number of times the four pairs of opponents have played against each other...
are matched first in this process.

The complexity of this procedure is $O(n^2)$.

### 8.4.2 Sequential Round-based Heuristics

To hedge against the myopic feature of the GH procedure, we propose two sequential round-based heuristics, SRH-A and SRH-B, in order to solve the problem in a stage-wise fashion that includes an optimization subproblem at each stage. The essential strategy adopted in both SRH-A and SRH-B consists of sequentially optimizing a current round subproblem, given the values that have been assigned to the decision variables pertaining to previous rounds in earlier iterations or stages, along with some look-ahead feature, where the latter is incorporated within SRH-B and in two out of three proposed implementation strategies for SRH-A.

**SRH-A**

For any problem instance $(n, p, r)$, we shall designate by $(n, p, k)$, for $k \leq r$, the problem instance where the number of rounds is restricted to $k$. This heuristic is delineated below.

**Step 1.** Set $k = 1$ and solve $P_{2,h}^t(\pi_2, \pi_3)$ for the problem instance $(n, p, k)$. Fix the resulting values of the $t$-variables pertaining to this first round problem, and relax the symmetry-defeating constraints. Set $k = 2$.

While $k \leq r$ repeat:

**Step 2.** Solve $P_{2,h}^t$ for the problem instance $(n, p, k)$, where the values of the $t_{i_1i_2q}$-variables are fixed as previously determined for $q < k$. Fix the values of the $t$-variables pertaining to round $k$ as determined by the resulting optimal solution, and set $k \leftarrow k + 1$.

Note that the alternative player-based symmetry-defeating constraints $(\pi_2)$ and $(\pi_3)$ are *jointly* enforced (*in a heuristic sense*) only in the first iteration of SRH-A in order to enhance the solvability of this first round subproblem, which, in turn, assists in solving the subproblems for the following rounds. In fact, it is empirically observed that the computational effort associated with this heuristic is predominantly incurred in Step 1, i.e., in solving the first round subproblem. Also, observe that maintaining the aforementioned symmetry-defeating decision hierarchies for $k \geq 2$ and perhaps also activating $\pi_1$, would inappropriately highly constrain the formulation, and sacrifice the quality of the generated solutions.
We also consider two implementation variants of SRH-A, denoted by SRH-A(2) and SRH-A(3), that include some look-ahead feature. In SRH-A(2), at any iteration \(2 \leq k \leq r - 1\) the variables that pertain to the subsequent rounds are maintained in the relevant constraints and the objective function, but as continuous variables. On the other hand, SRH-A(3) retains, at any iteration, the variables that pertain to the subsequent rounds as binary-valued in the objective function, while relaxing any constraints that involve these 0-1 variables.

**SRH-B**

Let Problem \(P_{t,k}^{2,h}(\pi), (\pi) \in \{(\pi_1,\pi_2), (\pi_1,\pi_3)\}\), for \(k = 1,...,r\), designate Problem \(P_{t,k}^{2,h}(\pi)\) for which: (a) binary restrictions are enforced for the \(t_{i_1i_2k}\)-variables, \(\forall (i_1,i_2) \in C\); (b) the \(t_{i_1i_2k'}\)-variables are fixed as determined in the solution of the previous Problem \(P_{2,h}^{4,k-1}(\pi)\) for \(k' \leq k - 1, \forall (i_1,i_2) \in C\) (whenever \(k \geq 2\)); and (c) the \(t_{i_1i_2k}\)-variables are relaxed to be continuous for \(k' \geq k + 1, \forall (i_1,i_2) \in C\) (whenever \(k \leq r - 1\)). SRH-B solves Problem \(P_{2,h}^{4,k}(\pi)\) sequentially for \(k = 1,...,r\), stopping for the first \(k \geq 1\) for which all the \(t\)-variables turn out to be binary-valued in the resulting optimal solution.

### 8.4.3 Two-Phase Heuristic

The binary variables related to opponentship considerations in Model 1 (i.e., the \(y_{j_1j_2k}\)-variables) significantly contribute to the number of constraints and the computational effort. Hence, if these constraints are omitted, the simplified model will be more tractable, and solving this relaxed model to optimality will produce a well-balanced partner-pairing scheme that can be complemented via a second pass to determine opponent-pairing relationships. This motivates the development of a two-phase heuristic algorithm, denoted by TPH. We consider the adapted objective functions \(\bar{f}_1 \equiv \bar{f}_2 \equiv \max_{j_1 < j_2} |z_{j_1j_2} - \bar{z}|\), and construct the following restricted formulations \(\bar{P}_{1,h}, h = 1,2\):

\[
\bar{P}_{1,1} \equiv \bar{P}_{1,2} \equiv \{\min \bar{f}_1 : (1b), (1c), x_{j_1j_2k} \in \{0,1\}, \forall j_1 < j_2, \forall k\}.
\]

Although incorporating symmetry-defeating constraints into \(\bar{P}_{1,h}, h = 1,2\), would present some computational benefits, we derive some results below that permit the problem in Phase I to be alternatively solved as a simpler constraint-satisfaction problem (CSP).

**Proposition 8.2.** Consider some problem instance \((n,p,r)\) where \(n = p\). (i) Suppose that \(\bar{z}\) is not an integer. Then \(\lfloor \bar{z} \rfloor \leq z_{j_1j_2} \leq \lfloor \bar{z} \rfloor + 1, \forall j_1 < j_2\) if and only if the partner-pairing is optimal for \(\bar{P}_{1,1}\). Moreover, (ii) if \(\bar{z}\) is integral, then \(z_{j_1j_2} = \bar{z}, \forall j_1 < j_2\), at optimality for \(\bar{P}_{1,1}\).
Proof. The proof of (i) is in two parts:

(i) Part I. Assume that there exists a feasible solution, \( x^1 \) (with a corresponding \( z \)-solution \( z^1 \)), to \( P_{1,1} \) such that \([\bar{z}] \leq z^1_{j_1j_2} \leq [\bar{z}] + 1, \forall j_1 < j_2 \). The corresponding objective function value, denoted \( f_1(x^1) \), satisfies \( f_1(x^1) \leq \max\{[\bar{z}] + 1 - \bar{z}, \bar{z} - [\bar{z}]\} \). Let us show that this solution is optimal to Problem \( P_{1,1} \). Consider any other feasible solution \( x^2 \) with an accompanying \( z \)-solution \( z^2 \). Since \( \bar{z} \) is not integral, and the average of the \( z \)-values must equal \( \bar{z} \), there exist \( j_1 < j_2 \) and \( j_3 < j_4 \) such that \([\bar{z}] + 1 \leq z^2_{j_1j_2} \) and \( z^2_{j_3j_4} \leq [\bar{z}] \). Therefore, \( \max_{j_1 < j_2} \left| z^2_{j_1j_2} - \bar{z} \right| > \max\{[\bar{z}] + 1 - \bar{z}, \bar{z} - [\bar{z}]\} \geq f_1(x^1) \). Hence, \( x^1 \) is an optimal solution.

(ii) Part II. Consider an optimal solution \( x^* \) to Problem \( P_{1,1} \) with an accompanying \( z \)-solution \( z^* \). Let us show that \( x^* \) satisfies the condition (i) of the Lemma by demonstrating the existence of a feasible solution \( x \) with a corresponding \( z \)-solution that satisfies this condition. Note that \( z^* \) must then also satisfy this condition because otherwise, as in the proof of Part I, we would have \([\bar{z}] \geq z^*_{j_1j_2} \) or \([\bar{z}] + 1 < z^*_{j_1j_2} \) for some \( j_1 < j_2 \), and therefore, \( \max_{j_1 < j_2} \left| z^*_{j_1j_2} - \bar{z} \right| > \max\{[\bar{z}] + 1 - \bar{z}, \bar{z} - [\bar{z}]\} \geq f_1(x) \), which would contradict the optimality of \( x^* \).

Since \( n = p \), \( n \) is even, and therefore by [144], there always exists a perfect partnership-pairing (where \( z_{j_1j_2} = 1, \forall j_1 < j_2 \)) for \( n \) players in \( n - 1 \) rounds. If \( r \leq n - 1 \), which yields \([\bar{z}] = 0\), we can construct a perfect partnership-pairing for \( n - 1 \) rounds, and select the first \( r \) rounds out of the \( n - 1 \) rounds. Clearly, we have \( 0 \leq z_{j_1j_2} \leq 1, \forall j_1 < j_2 \) in the resulting schedule. If \( r \geq n \), we repeat the perfect partnership-pairing over \( (n-1) \) rounds \( \left\lceil \frac{r}{n-1} \right\rceil + 1 \) times, and then select the first \( r \) rounds. The resulting schedule is such that \( \left\lceil \frac{r}{n-1} \right\rceil \leq z_{j_1j_2} \leq \left\lceil \frac{r}{n-1} \right\rceil + 1, \forall j_1 < j_2 \), where \( \bar{z} = \frac{r}{n-1} \).

(ii) Given that \( \bar{z} = \alpha \), where \( \alpha \) is some integer, we have, by Part II of (i), which holds identically for integral \( \bar{z} \) as well, that at optimality, we have \( \alpha \leq z_{j_1j_2} \leq \alpha + 1, \forall j_1 < j_2 \). Assume that there exists some \( j_1 < j_2 \) such that \( z_{j_1j_2} = \alpha + 1 \). This implies that \( \bar{z} > \alpha \), a contradiction. □

The result stated in Proposition 8.2 aims at shedding some light on the structure of the partnership-matching problem. Observe that regardless \( n = p \) or not, the proof of Proposition 8.2 indicates that if we determine a solution that satisfies the stated conditions, then this solution must be optimal to Problem \( P_{1,1} \). Accordingly, recalling that \( \bar{z} = \frac{rP}{n(n-1)} \) in general, we define the following two CSPs in connection with \textbf{Phase I} of the TPH procedure, where \( P^\text{frac}_{\text{CSP}}(\theta) \) and \( P^\text{int}_{\text{CSP}}(\theta) \) are invoked in TPH as described below depending on whether \( \bar{z} \) is fractional or integral,
respectively, and where \( \theta \) is some integer-valued parameter that is initialized at zero and is possibly incremented during the proposed heuristic TPH as necessary.

\[
P_{CSP}^{\text{frac}}(\theta): \left\lfloor \frac{rp}{n(n-1)} \right\rfloor - \theta \leq \sum_{k=1}^{r} x_{j_1j_2k} \leq \left\lfloor \frac{rp}{n(n-1)} \right\rfloor + 1 + \theta, \quad \forall j_1 < j_2 \tag{8.3a}
\]
\[
\sum_{j_1=1}^{n-1} \sum_{j_2=j_1+1}^{n} x_{j_1j_2k} = \frac{p}{2}, \quad \forall k \tag{8.3b}
\]
\[
\sum_{j_2=1}^{n} x_{[j_1j_2]k} \leq 1, \quad \forall j_1, \forall k \tag{8.3c}
\]
\[
x_{j_1j_2k} \in \{0, 1\}, \forall j_1 < j_2, \forall k. \tag{8.3d}
\]

\[
P_{CSP}^{\text{int}}(\theta): \sum_{k=1}^{r} x_{j_1j_2k} \leq \frac{rp}{n(n-1)} + \theta, \quad \forall j_1 < j_2 \tag{8.4a}
\]
\[
\sum_{j_1=1}^{n-1} \sum_{j_2=j_1+1}^{n} x_{j_1j_2k} = \frac{p}{2}, \quad \forall k \tag{8.4b}
\]
\[
\sum_{j_2=1}^{n} x_{[j_1j_2]k} \leq 1, \quad \forall j_1, \forall k \tag{8.4c}
\]
\[
x_{j_1j_2k} \in \{0, 1\}, \forall j_1 < j_2, \forall k. \tag{8.4d}
\]

TPH

- **Phase I**: Set \( \theta = 0 \). If \( n = p \), then by virtue of Proposition 8.2, solve \( P_{CSP}^{\text{frac}}(\theta) \) (respectively, \( P_{CSP}^{\text{int}}(\theta) \)) if \( \bar{z} \) is fractional (respectively, integral), and proceed to Phase II. If \( n \neq p \) solve either \( P_{CSP}^{\text{frac}}(\theta) \) or \( P_{CSP}^{\text{int}}(\theta) \) (according to the value of \( \bar{z} \) as above), and increment the value of \( \theta \) as necessary, until the relevant CSP becomes feasible.

- **Phase II**: Solve Problem \( P^{(R)}_{1,h}, h \in \{1, 2, 3\} \) to optimality in terms of the \( y \)-variables, with the \( x \)-variables fixed as per based on the partnership-pairings previously determined in Phase I, in order to produce the opponentship-pairings.

**Remark 8.5.** For all problem instances investigated in Section 8.5 with \( n \neq p \), it was empirically observed that \( P_{CSP}^{\text{frac}}(\theta) \) and \( P_{CSP}^{\text{int}}(\theta) \) produced feasible solutions with \( \theta = 0 \) itself, similar to the
case \( n = p \), which leads us to conjecture that Proposition 8.2 is true for \( n \neq p \) as well. However, since an extension of this result to the case of \( n \neq p \) would required a more complex proof and would not affect the structure of proposed heuristic, we do not pursue it here.

**Remark 8.6.** An alternative heuristic approach can be designed by solving Problem \( P_{1,h}^{R,CSP} \), which includes, in addition to the constraints and variables of Problem \( P_{1,h}^{R} \), either Constraints (8.3a) or (8.4a) (according to the fractionality or integrality of \( \bar{z} \), respectively). In this formulation, the objective function \( f_h \) is modified to contain only the term that involves the \( y \)-variables, and incorporates an additional term that penalizes \( \theta \) with a penalty coefficient to induce low \( \theta \)-values, where \( \theta \) is now considered as an integer-valued variable in the problem. That is, the partnership frequencies are bounded in \( P_{1,h}^{R,CSP} \) via (8.3a) or (8.4a), and, while permitting this restricted freedom in selecting teams for each round, the objective function focuses exclusively on opponentship considerations with the aforementioned penalty term. This approach provides freedom to sift among alternative optimal solutions to \( P_{CSP}^{frac} \) or \( P_{CSP}^{int} \) in the course of optimizing Problem \( P_{1,h}^{R,CSP} \). Preliminary computational tests indicated that although this approach yields competitive results, it becomes impractical for problem instances of the size of \((n,p,r) = (6,4,7)\) and larger.

### 8.5 Computational Experience

We now provide some computational experience to study the solvability of Models 1 and 2, as well as to illustrate the critical role of the proposed symmetry-defeating constraints for a set of realistic problem instances inspired by training sessions in the tennis club in Virginia, USA, which motivated this research. We consider the following three sets of problem instances.

- **Set 1** consists of seven small- to moderate-sized problem instances: \((n,p,r) = (6,4,4), (6,4,5), (6,4,6), (6,4,7), (7,4,4), (7,4,5), \) and \((8,8,7)\). This set is designed to reflect the growth in the computational effort associated with solving problem instances to optimality for increasingly larger values of \( n \) and \( r \).

- **Set 2** is composed of three relatively larger problem instances: \((n,p,r) = (10,8,10), (12,12,11), \) and \((16,16,16)\).

All mathematical programs were coded in AMPL and solved using CPLEX 9.0 on a Dell Precision 650 workstation having a Xeon(TM) CPU 2.40 GHz processor and 1.50 GB of RAM.

#### 8.5.1 Computational Effort for Solving LP Relaxations

Table 8.2 presents the CPU time required to solve the LP relaxations of Models 1 and 2 for the instances in Set 2. Note that although the LP relaxations of \( P_{1,h}^{R} \) are less computationally
Table 8.2: CPU times in seconds for LP relaxations, Set 2

<table>
<thead>
<tr>
<th></th>
<th>(n, r, p) = (10,8,10)</th>
<th>(n, r, p) = (12,12,11)</th>
<th>(n, r, p) = (16,16,16)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_{1,1}$</td>
<td>11.68</td>
<td>63.18</td>
<td>24.04</td>
</tr>
<tr>
<td>$P_{1,2}$</td>
<td>10.85</td>
<td>64.42</td>
<td>27.70</td>
</tr>
<tr>
<td>$P_{1,3}$</td>
<td>12.17</td>
<td>63.88</td>
<td>39.67</td>
</tr>
<tr>
<td>$P_{R_{1,1}}$</td>
<td>3.29</td>
<td>27.92</td>
<td>412.0</td>
</tr>
<tr>
<td>$P_{R_{1,2}}$</td>
<td>3.11</td>
<td>28.14</td>
<td>420.17</td>
</tr>
<tr>
<td>$P_{R_{1,3}}$</td>
<td>3.53</td>
<td>30.03</td>
<td>442.55</td>
</tr>
<tr>
<td>$P_{2,1}$</td>
<td>6.35</td>
<td>34.56</td>
<td>711.15</td>
</tr>
<tr>
<td>$P_{2,2}$</td>
<td>5.25</td>
<td>31.29</td>
<td>692.89</td>
</tr>
<tr>
<td>$P_{2,3}$</td>
<td>7.35</td>
<td>33.37</td>
<td>668.61</td>
</tr>
<tr>
<td>$P_{2,1}'$</td>
<td>0.65</td>
<td>2.51</td>
<td>30.54</td>
</tr>
<tr>
<td>$P_{2,2}'$</td>
<td>0.70</td>
<td>3.20</td>
<td>34.37</td>
</tr>
<tr>
<td>$P_{2,3}'$</td>
<td>1.86</td>
<td>3.76</td>
<td>156.30</td>
</tr>
</tbody>
</table>

8.5.2 Effects of Model Structure and Symmetry-defeating Constraints

When solved to optimality, no clear dominance is empirically observed between the formulations $P_{1,h}$ and $P_{1,h}^R$, $h = 1, 2, 3$ (see Table 8.3). However, the integration of symmetry-defeating constraints gives an edge to the reduced Problem $P_{1,h}^R(\delta_1, \delta_2)$ over Problem $P_{1,h}(\delta_1, \delta_2)$, $h = 1, 2, 3$, where the former formulations achieve an average savings in computational effort of 48.8% over the latter. In addition, Model 2 clearly outperforms Model 1 in all the problem instances that have been tested, as evident from Tables 8.3 and 8.4. In particular, $P_{2,h}$, and $P_{2,h}^t$, $h = 1, 2, 3$, respectively achieve an average savings in computational effort of 78.8% and 77.2% over $P_{1,h}^R$, $h = 1, 2, 3$, for the reported instances. These results also indicate that the compact mathematical programs formulated in the $t$-variables only, $P_{2,h}^t$, $h = 1, 2, 3$, have varying effects on the solver performance, depending on the problem instance and the selected objective function. Interestingly, the consideration of model structure becomes decisive when combined with decision hierarchies, and greatly enhances the pruning effect of the symmetry-defeating constraints. For example, Problems $P_{2,h}^t(\pi_1)$, $h = 1, 2, 3$, achieve an average savings in computational effort of 71.2% over $P_{2,h}(\pi_1)$, $h = 1, 2, 3$.

It is observed that for problem instances where $\bar{z}$ is fractional (e.g., $(n, p, r) = (6, 4, 4)$ or $(6, 4, 5)$), the inclusion of symmetry-defeating decision hierarchies has a dramatic pruning effect on the search space for both Models 1 and 2. Formulations $P_{1,h}^R(\delta_1, \delta_2)$, $h = 1, 2, 3$, achieve an overall average computational savings of 93.9% over $P_{1,h}^R$, $h = 1, 2, 3$, and likewise, the formulations $P_{2,h}^t(\pi_1, \pi_2)$, $h = 1, 2, 3$, achieve an average savings of 96.9% over $P_{2,h}^t$, $h = 1, 2, 3$. 

168
The formulations $P_{2,h}^t(\pi_1, \pi_2), h = 1, 2, 3$, tend to outperform, by several orders of magnitude, other formulations whenever $\bar{z}$ is fractional. For instance, Problems $P_{2,h}^t(\pi_1, \pi_2), h = 1, 2, 3$, save, in average, 91.9% of the computational effort reported for $P_{1,h}^R(\delta_1, \delta_2), h = 1, 2, 3$ (see Table 8.3). Also, the additive benefit and cumulative pruning effect of jointly appended symmetry-defeating constraints should be highlighted. For instance, the formulations $P_{2,h}^t(\pi_1, \pi_2), h = 1, 2, 3$, achieve an overall average computational savings of 73.5% and 95.5% over $P_{2,h}^t(\pi_1)$ and $P_{2,h}^t(\pi_2), h = 1, 2, 3$, respectively.

The problem instance $(n, p, r) = (8, 8, 7)$ (see Table 8.4), where $\bar{z}$ is integral, is particularly worthy of scrutiny as it provides a strong case for the importance of symmetry compatible formulations (see Section 8.3.2). It can be observed that the set of alternative optimal solutions is exclusively composed by solutions that correspond to permutations of rounds of some optimal schedule. This result contrasts with problem instances where $\bar{z}$ is fractional, in which case, the set of alternative optimal solutions is larger because of symmetry effects. While this instance could not be solved in 10 hours with Model 1, it is almost instantaneously solved to optimality with Model 2, as reported in Table 8.4. Furthermore, observe that the consideration of compactness as in Problems $P_{2,h}^t$, versus $P_{2,h}, h = 1, 2, 3$, has somewhat mixed effects, and symmetry-defeating strategies have a negative effect on the model solvability for this model formulation. In fact, for this specific problem instance, an optimal solution was obtained at the root node of the search tree for Problems $P_{2,1}^t$, $P_{2,2}^t$, $P_{2,3}$, and $P_{2,3}^t$, and after enumerating four nodes for Problem $P_{2,2}$. However, the root node LP solution does not satisfy all the proposed decision hierarchies and, therefore, a significant branching effort is required to identify a solution compatible with the selected symmetry-defeating constraints.

More specifically, the root node analysis for formulations $P_{2,h}^t, h = 1, 2, 3$, reveals that the LP solution available at node 0 is compatible with $(\pi_2)$ and $(\pi_3)$, while it violates $(\pi_1)$. Likewise, the decision hierarchy $(\pi_1)$ is not satisfied by the root node solution for the formulations $P_{2,h}, h = 1, 2, 3$. Hence, the inclusion of $(\pi_1)$, as in $P_{2,h}^t(\pi_1)$ or $P_{2,h}(\pi_1), h = 1, 2, 3$, modifies the LP root node solution, and induces a costly branching effort to achieve optimality as shown in Table 8.4. On the other hand, when $(\pi_2)$ or $(\pi_3)$ are employed, a satisfactory performance is obtained. This highlights the importance of the concept of symmetry compatible models.

We also employed the symmetry compatible strategy alluded to in Section 8.3.2, by applying a heuristic to derive a feasible solution based on the LP root node solution, and then imposing a
hierarchy that is compatible with that solution. The resulting formulation was subsequently solved to optimality. By applying this strategy to \((n, r, p) = (8, 8, 7)\), a feasible solution that satisfies the following symmetry-defeating constraints (call this \(\pi_1\)) is derived:

\[
F_2 \leq F_4 \leq F_3 \leq F_1 \leq F_5 \leq F_6 \leq F_7,
\]

where \(F_k\), is as defined in Section 8.4.4, for \(k = 1,\ldots,7\). Implementing this strategy, Problem \(P_{2,1}(\pi_1)\) was solved in 61.5 seconds, resulting in a 24.6% savings in computational effort in comparison with solving \(P_{2,1}(\pi_1)\).

Table 8.5 reports the computational effort associated with solving \(P^t_{2,h}(\pi_1, \pi_2), h = 1, 2, 3\), for the problem instances in Set 1. The plot in Figure 8.1 displays the growth in computational effort required to solve the problem instances \((n, p, r) = (6, 4, r), r = 4, \ldots, 7\), with \(P^t_{2,1}(\pi_1, \pi_2)\).

8.5.3 Nonpreemptive Multi-objective Programming-based Approach

We demonstrate in this section the potential computational benefits of the nonpreemptive multi-objective programming approach based on, and combined with, the symmetry-defeating constraints. The decreasing hierarchy of weights associated with the functions \(F_k, k = 1,\ldots, r + n\), as discussed in Remark 8.4 of Section 8.3.3 is empirically and selected to be

\[
\mu_k = \frac{1}{k^5 + k}, k = 1,\ldots, r + n.
\] (8.5)

Given the weight hierarchy in Equation (8.5), \(\mu_0\) is computed for every problem instance in concordance with Proposition 8.1. Table 8.6 presents the computational experience obtained with
<table>
<thead>
<tr>
<th>((n, r, p) = (6.4.4), f_1^* = g_1^* = 1.067)</th>
<th>((n, r, p) = (6.4.5), f_2^* = g_2^* = 1.333)</th>
<th>((n, r, p) = (7.4.4), f_3^* = g_3^* = 17.524)</th>
<th>(\text{CPU time (seconds)})</th>
<th>(\text{MIP simplex iterations, B&amp;B nodes})</th>
</tr>
</thead>
<tbody>
<tr>
<td>(P_{1.1})</td>
<td>675.7</td>
<td>(4196401, 128140)</td>
<td>(P_{1.2})</td>
<td>4848.0</td>
</tr>
<tr>
<td>(P_{1.1}(\delta_1))</td>
<td>236.9</td>
<td>(1523888, 35461)</td>
<td>(P_{1.2}(\delta_1))</td>
<td>831.2</td>
</tr>
<tr>
<td>(P_{1.1}(\delta_1, \delta_2))</td>
<td>149.4</td>
<td>(8534555, 14387)</td>
<td>(P_{1.2}(\delta_1, \delta_2))</td>
<td>657.6</td>
</tr>
<tr>
<td>(P_{1.1}^R(\delta_1))</td>
<td>256.1</td>
<td>(2604787, 142918)</td>
<td>(P_{1.2}^R(\delta_1))</td>
<td>9703.9</td>
</tr>
<tr>
<td>(P_{1.1}^R(\delta_1, \delta_2))</td>
<td>194.3</td>
<td>(1762529, 74639)</td>
<td>(P_{1.2}^R(\delta_1, \delta_2))</td>
<td>1172.7</td>
</tr>
<tr>
<td>(P_{1.1}(\delta_1, \delta_2))</td>
<td>33.1</td>
<td>(206121, 8330)</td>
<td>(P_{1.2}(\delta_1, \delta_2))</td>
<td>206.6</td>
</tr>
<tr>
<td>(P_{1.1}(\delta_1))</td>
<td>92.2</td>
<td>(1591990, 195964)</td>
<td>(P_{1.2}(\delta_1))</td>
<td>388.0</td>
</tr>
<tr>
<td>(P_{2.1}(\delta_1))</td>
<td>26.4</td>
<td>(458420, 32063)</td>
<td>(P_{2.2}(\delta_1))</td>
<td>111.2</td>
</tr>
<tr>
<td>(P_{2.1}(\delta_1, \delta_2))</td>
<td>142.6</td>
<td>(2486680, 602619)</td>
<td>(P_{2.2}(\delta_1, \delta_2))</td>
<td>268.1</td>
</tr>
<tr>
<td>(P_{2.1}^R(\delta_1))</td>
<td>8.5</td>
<td>(152797, 23292)</td>
<td>(P_{2.2}^R(\delta_1))</td>
<td>27.1</td>
</tr>
<tr>
<td>(P_{2.1}^R(\delta_1, \delta_2))</td>
<td>33.1</td>
<td>(530137, 94957)</td>
<td>(P_{2.2}^R(\delta_1, \delta_2))</td>
<td>3.1</td>
</tr>
<tr>
<td>(P_{2.1}(\delta_1, \delta_2))</td>
<td>7.3</td>
<td>(127926, 17045)</td>
<td>(P_{2.2}(\delta_1, \delta_2))</td>
<td>3.1</td>
</tr>
</tbody>
</table>

Table 8.3: Comparison of Models 1 and 2 solved to optimality
<table>
<thead>
<tr>
<th>((n, r, p) = (8,8,7)), (g_1^* = 0)</th>
<th>CPU time (seconds)</th>
<th>(MIP simplex iterations, B&amp;B nodes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(P_{1,1}^2(\delta_1, \delta_2))</td>
<td>&gt;36000</td>
<td>-</td>
</tr>
<tr>
<td>(P_{2,1})</td>
<td>0.8</td>
<td>(677, 0)</td>
</tr>
<tr>
<td>(P_{2,1}(\pi_1))</td>
<td>81.6</td>
<td>(166963, 1225)</td>
</tr>
<tr>
<td>(P_{2,1}^2(\pi_1))</td>
<td>1.2</td>
<td>(661, 0)</td>
</tr>
<tr>
<td>(P_{2,1}(\pi_3))</td>
<td>1.3</td>
<td>(952, 0)</td>
</tr>
<tr>
<td>(P_{2,1}(\pi_1, \pi_2))</td>
<td>212.0</td>
<td>(715407, 19108)</td>
</tr>
<tr>
<td>(P_{1,2}^3(\delta_1, \delta_2))</td>
<td>&gt;36000</td>
<td>-</td>
</tr>
<tr>
<td>(P_{2,2})</td>
<td>8.7</td>
<td>(7056, 4)</td>
</tr>
<tr>
<td>(P_{2,2}(\delta_1))</td>
<td>1.2</td>
<td>(656, 0)</td>
</tr>
<tr>
<td>(P_{2,2}^1(\delta_1))</td>
<td>1.1</td>
<td>(1049, 0)</td>
</tr>
<tr>
<td>(P_{2,2}(\delta_2))</td>
<td>1317.5</td>
<td>(5357316, 88357)</td>
</tr>
<tr>
<td>(P_{2,2}^1(\delta_2))</td>
<td>1.1</td>
<td>(1027, 0)</td>
</tr>
<tr>
<td>(P_{2,2}(\pi_3))</td>
<td>4.2</td>
<td>(4778, 9)</td>
</tr>
<tr>
<td>(P_{2,2}(\pi_1, \pi_2))</td>
<td>465.1</td>
<td>(1630802, 29745)</td>
</tr>
<tr>
<td>(P_{2,2}^1(\pi_1, \pi_3))</td>
<td>262.1</td>
<td>(913267, 18343)</td>
</tr>
<tr>
<td>(P_{1,3}^3(\delta_1, \delta_2))</td>
<td>&gt;36000</td>
<td>-</td>
</tr>
<tr>
<td>(P_{2,3})</td>
<td>1.1</td>
<td>(1244, 0)</td>
</tr>
<tr>
<td>(P_{2,3}(\delta_1))</td>
<td>535.8</td>
<td>(1260490, 9515)</td>
</tr>
<tr>
<td>(P_{2,3}^1(\delta_1))</td>
<td>1.4</td>
<td>(1428.0)</td>
</tr>
<tr>
<td>(P_{2,3}(\delta_2))</td>
<td>8.2</td>
<td>(12847, 20)</td>
</tr>
<tr>
<td>(P_{2,3}(\delta_3))</td>
<td>1.8</td>
<td>(1524, 0)</td>
</tr>
<tr>
<td>(P_{2,3}(\pi_3))</td>
<td>4.3</td>
<td>(2456, 3)</td>
</tr>
<tr>
<td>(P_{2,3}(\pi_1, \pi_2))</td>
<td>93.0</td>
<td>(220897, 1732)</td>
</tr>
<tr>
<td>(P_{2,3}^1(\pi_1, \pi_3))</td>
<td>112.9</td>
<td>(308448, 6697)</td>
</tr>
</tbody>
</table>

Table 8.4: Results for \((n, r, p) = (8,8,7)\)
<table>
<thead>
<tr>
<th>$(n, r, p)$</th>
<th>CPU time (seconds)</th>
<th>(MIP simplex iterations, B&amp;B nodes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(6, 4, 4)$</td>
<td>$3.18$</td>
<td>$(59143, 4610)$</td>
</tr>
<tr>
<td>$(g^<em>_1, g^</em>_2, g^*_3) = (1.067, 1.6, 11.2)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P^{2, 1}_{2.1}(\pi_1, \pi_2)$</td>
<td>$5.96$</td>
<td>$(112866, 7746)$</td>
</tr>
<tr>
<td>$P^{2, 2}_{2.1}(\pi_1, \pi_2)$</td>
<td>$11.09$</td>
<td>$(193901, 15745)$</td>
</tr>
<tr>
<td>$P^{2, 3}_{2.1}(\pi_1, \pi_2)$</td>
<td>$41.39$</td>
<td>$(728675, 50990)$</td>
</tr>
<tr>
<td>$(6, 4.5)$</td>
<td>CPU time (seconds)</td>
<td>(MIP simplex iterations, B&amp;B nodes)</td>
</tr>
<tr>
<td>$(g^<em>_1, g^</em>_2, g^*_3) = (0.667, 1.333, 13.333)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P^{2, 1}_{2.1}(\pi_1, \pi_2)$</td>
<td>$5.87$</td>
<td>$(113329, 6505)$</td>
</tr>
<tr>
<td>$P^{2, 2}_{2.1}(\pi_1, \pi_2)$</td>
<td>$11.09$</td>
<td>$(193901, 15745)$</td>
</tr>
<tr>
<td>$P^{2, 3}_{2.1}(\pi_1, \pi_2)$</td>
<td>$41.39$</td>
<td>$(728675, 50990)$</td>
</tr>
<tr>
<td>$(6, 4.6)$</td>
<td>CPU time (seconds)</td>
<td>(MIP simplex iterations, B&amp;B nodes)</td>
</tr>
<tr>
<td>$(g^<em>_1, g^</em>_2, g^*_3) = (0.8, 1.4, 12)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P^{2, 1}_{2.1}(\pi_1, \pi_2)$</td>
<td>$21.70$</td>
<td>$(406615, 24685)$</td>
</tr>
<tr>
<td>$P^{2, 2}_{2.1}(\pi_1, \pi_2)$</td>
<td>$28.17$</td>
<td>$(497950, 27383)$</td>
</tr>
<tr>
<td>$P^{2, 3}_{2.1}(\pi_1, \pi_2)$</td>
<td>$156.84$</td>
<td>$(2533817, 169435)$</td>
</tr>
<tr>
<td>$(6, 4.7)$</td>
<td>CPU time (seconds)</td>
<td>(MIP simplex iterations, B&amp;B nodes)</td>
</tr>
<tr>
<td>$(g^<em>_1, g^</em>_2, g^*_3) = (1.333, 2.067, 7.067)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P^{2, 1}_{2.1}(\pi_1, \pi_2)$</td>
<td>$626.83$</td>
<td>$(9440479, 9440479)$</td>
</tr>
<tr>
<td>$P^{2, 2}_{2.1}(\pi_1, \pi_2)$</td>
<td>$369.65$</td>
<td>$(5447793, 435242)$</td>
</tr>
<tr>
<td>$P^{2, 3}_{2.1}(\pi_1, \pi_2)$</td>
<td>$111.48$</td>
<td>$(1756526, 82538)$</td>
</tr>
<tr>
<td>$(7, 4.4)$</td>
<td>CPU time (seconds)</td>
<td>(MIP simplex iterations, B&amp;B nodes)</td>
</tr>
<tr>
<td>$(g^<em>_1, g^</em>_2, g^*_3) = (0.761, 1.380, 17.523)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P^{2, 1}_{2.1}(\pi_1, \pi_2)$</td>
<td>$14.68$</td>
<td>$(149829, 14626)$</td>
</tr>
<tr>
<td>$P^{2, 2}_{2.1}(\pi_1, \pi_2)$</td>
<td>$11.21$</td>
<td>$(147474, 9346)$</td>
</tr>
<tr>
<td>$P^{2, 3}_{2.1}(\pi_1, \pi_2)$</td>
<td>$184.26$</td>
<td>$(2644076, 124181)$</td>
</tr>
<tr>
<td>$(7, 4.5)$</td>
<td>CPU time (seconds)</td>
<td>(MIP simplex iterations, B&amp;B nodes)</td>
</tr>
<tr>
<td>$(g^<em>_1, g^</em>_2, g^*_3) = (1.047, 1.571, 14.285)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P^{2, 1}_{2.1}(\pi_1, \pi_2)$</td>
<td>$202.65$</td>
<td>$(2822087, 147839)$</td>
</tr>
<tr>
<td>$P^{2, 2}_{2.1}(\pi_1, \pi_2)$</td>
<td>$247.45$</td>
<td>$(3039274, 182797)$</td>
</tr>
<tr>
<td>$P^{2, 3}_{2.1}(\pi_1, \pi_2)$</td>
<td>$559.44$</td>
<td>$(7219044, 338989)$</td>
</tr>
<tr>
<td>$(8, 8.7)$</td>
<td>CPU time (seconds)</td>
<td>(MIP simplex iterations, B&amp;B nodes)</td>
</tr>
<tr>
<td>$(g^<em>_1, g^</em>_2, g^*_3) = (0.0, 0.0)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P^{2, 1}_{2.1}(\pi_1, \pi_2)$</td>
<td>$573.61$</td>
<td>$(3397972, 58150)$</td>
</tr>
<tr>
<td>$P^{2, 2}_{2.1}(\pi_1, \pi_2)$</td>
<td>$465.15$</td>
<td>$(1630802, 29745)$</td>
</tr>
<tr>
<td>$P^{2, 3}_{2.1}(\pi_1, \pi_2)$</td>
<td>$93.0$</td>
<td>$(220897, 1732)$</td>
</tr>
</tbody>
</table>

Table 8.5: Performance of Problem $P_{2,h}^r(\pi_1, \pi_2)$
Table 8.6: Results for \( AP^\mu(\pi_1, \pi_2) \), Set 1

<table>
<thead>
<tr>
<th>((n, r, p) = (6,4,4),) ((g_1^<em>, g_2^</em>, g_3^*) = (1.067, 1.6, 11.2))</th>
<th>CPU time (seconds)</th>
<th>([\text{MIP simplex iterations, B&amp;B nodes}])</th>
<th>Objective value ( F_0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( AP^\mu_1(\pi_1, \pi_2) )</td>
<td>3.60</td>
<td>(57210, 7554)</td>
<td>1.067</td>
</tr>
<tr>
<td>( AP^\mu_2(\pi_1, \pi_2) )</td>
<td>4.04</td>
<td>(64906, 7069)</td>
<td>1.6</td>
</tr>
<tr>
<td>( AP^\mu_3(\pi_1, \pi_2) )</td>
<td>6.67</td>
<td>(123795, 8501)</td>
<td>11.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>((n, r, p) = (6,4,5),) ((g_1^<em>, g_2^</em>, g_3^*) = (0.667, 1.333, 13.333))</th>
<th>CPU time (seconds)</th>
<th>([\text{MIP simplex iterations, B&amp;B nodes}])</th>
<th>Objective value ( F_0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( AP^\mu_1(\pi_1, \pi_2) )</td>
<td>4.12</td>
<td>(55646, 7388)</td>
<td>0.667</td>
</tr>
<tr>
<td>( AP^\mu_2(\pi_1, \pi_2) )</td>
<td>10.12</td>
<td>(163597, 16150)</td>
<td>1.333</td>
</tr>
<tr>
<td>( AP^\mu_3(\pi_1, \pi_2) )</td>
<td>53.18</td>
<td>(894224, 65731)</td>
<td>13.333</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>((n, r, p) = (6,4,6),) ((g_1^<em>, g_2^</em>, g_3^*) = (0.8, 1.4, 12))</th>
<th>CPU time (seconds)</th>
<th>([\text{MIP simplex iterations, B&amp;B nodes}])</th>
<th>Objective value ( F_0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( AP^\mu_1(\pi_1, \pi_2) )</td>
<td>20.21</td>
<td>(484413, 41561)</td>
<td>0.8</td>
</tr>
<tr>
<td>( AP^\mu_2(\pi_1, \pi_2) )</td>
<td>25.35</td>
<td>(418691, 30744)</td>
<td>1.4</td>
</tr>
<tr>
<td>( AP^\mu_3(\pi_1, \pi_2) )</td>
<td>150.51</td>
<td>(2417719, 176769)</td>
<td>12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>((n, r, p) = (6,4,7),) ((g_1^<em>, g_2^</em>, g_3^*) = (1.133, 2.067, 7.067))</th>
<th>CPU time (seconds)</th>
<th>([\text{MIP simplex iterations, B&amp;B nodes}])</th>
<th>Objective value ( F_0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( AP^\mu_1(\pi_1, \pi_2) )</td>
<td>781.59</td>
<td>(9753689, 1301121)</td>
<td>1.133</td>
</tr>
<tr>
<td>( AP^\mu_2(\pi_1, \pi_2) )</td>
<td>491.28</td>
<td>(6975550, 713068)</td>
<td>2.067</td>
</tr>
<tr>
<td>( AP^\mu_3(\pi_1, \pi_2) )</td>
<td>130.84</td>
<td>(1851238, 114930)</td>
<td>7.067</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>((n, r, p) = (7,4,4),) ((g_1^<em>, g_2^</em>, g_3^*) = (0.761, 1.380, 17.523))</th>
<th>CPU time (seconds)</th>
<th>([\text{MIP simplex iterations, B&amp;B nodes}])</th>
<th>Objective value ( F_0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( AP^\mu_1(\pi_1, \pi_2) )</td>
<td>10.20</td>
<td>(85055, 12425)</td>
<td>0.761</td>
</tr>
<tr>
<td>( AP^\mu_2(\pi_1, \pi_2) )</td>
<td>11.61</td>
<td>(126888, 10602)</td>
<td>1.380</td>
</tr>
<tr>
<td>( AP^\mu_3(\pi_1, \pi_2) )</td>
<td>21.87</td>
<td>(2946003, 142635)</td>
<td>17.523</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>((n, r, p) = (7,4,5),) ((g_1^<em>, g_2^</em>, g_3^*) = (1.047, 1.571, 14.285))</th>
<th>CPU time (seconds)</th>
<th>([\text{MIP simplex iterations, B&amp;B nodes}])</th>
<th>Objective value ( F_0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( AP^\mu_1(\pi_1, \pi_2) )</td>
<td>82.51</td>
<td>(843438, 99018)</td>
<td>1.047</td>
</tr>
<tr>
<td>( AP^\mu_2(\pi_1, \pi_2) )</td>
<td>149.36</td>
<td>(1624929, 164727)</td>
<td>1.571</td>
</tr>
<tr>
<td>( AP^\mu_3(\pi_1, \pi_2) )</td>
<td>987.14</td>
<td>(11785083, 649266)</td>
<td>14.285</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>((n, r, p) = (8,8,7),) ((g_1^<em>, g_2^</em>, g_3^*) = (0, 0, 0))</th>
<th>CPU time (seconds)</th>
<th>([\text{MIP simplex iterations, B&amp;B nodes}])</th>
<th>Objective value ( F_0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( AP^\mu_1(\pi_1, \pi_2) )</td>
<td>0.59</td>
<td>(1800, 0)</td>
<td>0</td>
</tr>
<tr>
<td>( AP^\mu_2(\pi_1, \pi_2) )</td>
<td>0.56</td>
<td>(1704, 0)</td>
<td>0</td>
</tr>
<tr>
<td>( AP^\mu_3(\pi_1, \pi_2) )</td>
<td>0.56</td>
<td>(1532, 0)</td>
<td>0</td>
</tr>
</tbody>
</table>

Problem \( AP^\mu(\pi_1, \pi_2) \), and reports the objective value \( F_0 \) for various problem instances which is verified to be optimal to \( P(\pi) \). This preliminary computational test reveals that this exact solution method exhibits computational times comparable to those obtained for Problem \( P(\pi) \) in Table 8.5.

Observe that this strategy induces via the augmented objective terms a root node solution that is compatible with the imposed symmetry-defeating hierarchy, and hence affords an automated alternative to the symmetry compatible strategy of Section 8.3.2. In particular, this enforced compatibility via the model structure itself that is therefore manifested at all nodes produced a dramatic decrease in computational effort for the problem instance \((n, p, r) = (8,8,7)\) as evident from Tables 8.5 and 8.6.
8.5.4 Comparison of Heuristics

The numerical results reported in Tables 8.7-8.9 provide a comparison between the performance of the greedy heuristic, GH, the two sequential round-based heuristics, SRH-A and SRH-B, and the two-phase heuristic, TPH, for several problem instances. For these examples, there is no clear dominance between GH and TPH, and both heuristics can be used to quickly generate reasonable upper bounds for the problem instances under investigation. It can also be noted that for problem instances where \( p = 4 \), TPH has zero degrees of freedom in Phase II, since the full schedule is already determined by the output of Phase I. The reported results reveal that both SRH-A and SRH-B provide the best trade-off between the quality of the solution and the resulting computational effort.

Table 8.9 compares the performances of SRH-A, SRH-A(2), SRH-A(3), SRH-B, TPH, and GH for solving problem instances in Set 1 and \((n, p, r) = (8, 8, 7)\), which have been solved to optimality using Problem \( P_{2,h}^t(\pi_1, \pi_2), h = 1, 2, 3 \) (see Section 8.5.2). This comparison reveals that SRH-A(3) strikes an excellent compromise between the quality of the solution derived and the associated computational effort for small- and moderate-sized test instances. This heuristic detected an optimal solution for 95.2% of the tested problems for which an optimum has been found above, and saved 96.8% of the computational effort required for solving \( P_{2,h}^t(\pi_1, \pi_2), h = 1, 2, 3 \), which were empirically observed to be the most promising formulations for deriving exact solutions. In addition, it appears that SRH-B is a competitive heuristic; this method saved 99.2% of the computational effort associated with \( P_{2,h}^t(\pi_1, \pi_2), h = 1, 2, 3 \), while discovering an optimum for 76.2% of the test problem instances. In contrast, TPH and GH both saved 99.9% of the computational effort required for the exact solution method, at the expense of discovering an optimum for only 9.5% of the test instances.

Despite the excellent performance of SRH-A(3) for test instances in Set 1, this implementation strategy becomes impractical for larger problem instances. Observe that SRH-A solved all problem instances in Sets 1-3 in manageable times, discovering an optimum for 57.1% of the time for the problem instances for which an optimum was provably found with Problem \( P_{2,h}^t(\pi_1, \pi_2), h = 1, 2, 3 \), and, therefore, offers an attractive alternative for larger problem instances. In Table 8.8, the symbol † indicates that the computational times for each of the two phases of TPH and every iteration of SRH-A(2/3) were bounded by 10000 seconds for larger problem instances, and that a 5% optimality tolerance was prescribed while invoking CPLEX 9.0.
<table>
<thead>
<tr>
<th>((n, p, r))</th>
<th>((g^<em>_h, g^</em>_l, g^*_v))</th>
<th>(f_1, g_1)</th>
<th>CPU time (s)</th>
<th>(f_2, g_2)</th>
<th>CPU time (s)</th>
<th>(f_3, g_3)</th>
<th>CPU time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GH</td>
<td>1.067</td>
<td>0.43</td>
<td>1.6</td>
<td>0.28</td>
<td>12.267</td>
<td>0.01</td>
<td></td>
</tr>
<tr>
<td>SRH-A</td>
<td>1.067</td>
<td>0.56</td>
<td>1.6</td>
<td>0.28</td>
<td>12.267</td>
<td>0.20</td>
<td></td>
</tr>
<tr>
<td>SRH-A(2)</td>
<td>1.067</td>
<td>0.35</td>
<td>1.6</td>
<td>1.26</td>
<td>11.2</td>
<td>0.57</td>
<td></td>
</tr>
<tr>
<td>SRH-B</td>
<td>1.067</td>
<td>0.65</td>
<td>1.6</td>
<td>0.71</td>
<td>12.267</td>
<td>0.75</td>
<td></td>
</tr>
<tr>
<td>TPH</td>
<td>1.067</td>
<td>0.65</td>
<td>2.467</td>
<td>0.01</td>
<td>15.067</td>
<td>0.01</td>
<td></td>
</tr>
<tr>
<td>((n, p, r))</td>
<td>((g^<em>_h, g^</em>_l, g^*_v))</td>
<td>(f_1, g_1)</td>
<td>CPU time (s)</td>
<td>(f_2, g_2)</td>
<td>CPU time (s)</td>
<td>(f_3, g_3)</td>
<td>CPU time (s)</td>
</tr>
<tr>
<td>GH</td>
<td>1.667</td>
<td>0.01</td>
<td>1.0</td>
<td>0.01</td>
<td>15.333</td>
<td>0.01</td>
<td></td>
</tr>
<tr>
<td>SRH-A</td>
<td>1.333</td>
<td>0.23</td>
<td>2</td>
<td>0.29</td>
<td>13.333</td>
<td>0.76</td>
<td></td>
</tr>
<tr>
<td>SRH-A(2)</td>
<td>0.667</td>
<td>0.28</td>
<td>1.333</td>
<td>0.34</td>
<td>14.667</td>
<td>0.43</td>
<td></td>
</tr>
<tr>
<td>SRH-A(3)</td>
<td>0.667</td>
<td>0.48</td>
<td>1.333</td>
<td>1.31</td>
<td>13.333</td>
<td>0.26</td>
<td></td>
</tr>
<tr>
<td>SRH-B</td>
<td>0.667</td>
<td>0.81</td>
<td>1.333</td>
<td>0.82</td>
<td>14.667</td>
<td>0.92</td>
<td></td>
</tr>
<tr>
<td>TPH</td>
<td>1.067</td>
<td>0.01</td>
<td>2.333</td>
<td>0.01</td>
<td>18.667</td>
<td>0.01</td>
<td></td>
</tr>
<tr>
<td>((n, p, r))</td>
<td>((g^<em>_h, g^</em>_l, g^*_v))</td>
<td>(f_1, g_1)</td>
<td>CPU time (s)</td>
<td>(f_2, g_2)</td>
<td>CPU time (s)</td>
<td>(f_3, g_3)</td>
<td>CPU time (s)</td>
</tr>
<tr>
<td>GH</td>
<td>1.4</td>
<td>0.02</td>
<td>2.6</td>
<td>0.02</td>
<td>16.0</td>
<td>0.02</td>
<td></td>
</tr>
<tr>
<td>SRH-A</td>
<td>1.2</td>
<td>0.31</td>
<td>2.2</td>
<td>0.34</td>
<td>12</td>
<td>0.29</td>
<td></td>
</tr>
<tr>
<td>SRH-A(2)</td>
<td>0.8</td>
<td>0.48</td>
<td>1.4</td>
<td>0.46</td>
<td>12</td>
<td>0.60</td>
<td></td>
</tr>
<tr>
<td>SRH-A(3)</td>
<td>0.8</td>
<td>4.42</td>
<td>1.4</td>
<td>6.92</td>
<td>12</td>
<td>1.18</td>
<td></td>
</tr>
<tr>
<td>SRH-B</td>
<td>0.8</td>
<td>0.99</td>
<td>1.4</td>
<td>1.01</td>
<td>13.2</td>
<td>1.04</td>
<td></td>
</tr>
<tr>
<td>TPH</td>
<td>1.4</td>
<td>0.01</td>
<td>2.2</td>
<td>0.01</td>
<td>13.2</td>
<td>0.01</td>
<td></td>
</tr>
<tr>
<td>((n, p, r))</td>
<td>((g^<em>_h, g^</em>_l, g^*_v))</td>
<td>(f_1, g_1)</td>
<td>CPU time (s)</td>
<td>(f_2, g_2)</td>
<td>CPU time (s)</td>
<td>(f_3, g_3)</td>
<td>CPU time (s)</td>
</tr>
<tr>
<td>GH</td>
<td>2.133</td>
<td>0.02</td>
<td>3.2</td>
<td>0.02</td>
<td>14.133</td>
<td>0.02</td>
<td></td>
</tr>
<tr>
<td>SRH-A</td>
<td>1.133</td>
<td>0.35</td>
<td>2.067</td>
<td>0.42</td>
<td>8.8</td>
<td>0.31</td>
<td></td>
</tr>
<tr>
<td>SRH-A(2)</td>
<td>1.133</td>
<td>0.67</td>
<td>2.067</td>
<td>0.67</td>
<td>8.8</td>
<td>0.70</td>
<td></td>
</tr>
<tr>
<td>SRH-A(3)</td>
<td>1.133</td>
<td>0.92</td>
<td>2.067</td>
<td>2.34</td>
<td>8.933</td>
<td>0.73</td>
<td></td>
</tr>
<tr>
<td>SRH-B</td>
<td>1.133</td>
<td>1.22</td>
<td>2.067</td>
<td>1.31</td>
<td>8.933</td>
<td>1.15</td>
<td></td>
</tr>
<tr>
<td>TPH</td>
<td>1.133</td>
<td>0.01</td>
<td>2.067</td>
<td>0.01</td>
<td>8.8</td>
<td>0.01</td>
<td></td>
</tr>
<tr>
<td>((n, p, r))</td>
<td>((g^<em>_h, g^</em>_l, g^*_v))</td>
<td>(f_1, g_1)</td>
<td>CPU time (s)</td>
<td>(f_2, g_2)</td>
<td>CPU time (s)</td>
<td>(f_3, g_3)</td>
<td>CPU time (s)</td>
</tr>
<tr>
<td>GH</td>
<td>1.238</td>
<td>0.01</td>
<td>1.857</td>
<td>0.01</td>
<td>20.571</td>
<td>0.01</td>
<td></td>
</tr>
<tr>
<td>SRH-A</td>
<td>0.761</td>
<td>0.23</td>
<td>1.380</td>
<td>0.29</td>
<td>17.523</td>
<td>0.28</td>
<td></td>
</tr>
<tr>
<td>SRH-A(2)</td>
<td>0.761</td>
<td>0.24</td>
<td>1.380</td>
<td>0.29</td>
<td>17.523</td>
<td>1.01</td>
<td></td>
</tr>
<tr>
<td>SRH-A(3)</td>
<td>0.761</td>
<td>0.97</td>
<td>1.380</td>
<td>3.56</td>
<td>17.523</td>
<td>1.18</td>
<td></td>
</tr>
<tr>
<td>SRH-B</td>
<td>1.238</td>
<td>1.14</td>
<td>1.380</td>
<td>1.62</td>
<td>17.523</td>
<td>2.70</td>
<td></td>
</tr>
<tr>
<td>TPH</td>
<td>2.238</td>
<td>0.01</td>
<td>2.857</td>
<td>0.01</td>
<td>25.142</td>
<td>0.01</td>
<td></td>
</tr>
<tr>
<td>((n, p, r))</td>
<td>((g^<em>_h, g^</em>_l, g^*_v))</td>
<td>(f_1, g_1)</td>
<td>CPU time (s)</td>
<td>(f_2, g_2)</td>
<td>CPU time (s)</td>
<td>(f_3, g_3)</td>
<td>CPU time (s)</td>
</tr>
<tr>
<td>GH</td>
<td>1.047</td>
<td>0.02</td>
<td>1.571</td>
<td>0.02</td>
<td>21.904</td>
<td>0.02</td>
<td></td>
</tr>
<tr>
<td>SRH-A</td>
<td>1.047</td>
<td>0.34</td>
<td>1.571</td>
<td>0.36</td>
<td>14.285</td>
<td>0.31</td>
<td></td>
</tr>
<tr>
<td>SRH-A(2)</td>
<td>1.047</td>
<td>0.34</td>
<td>1.571</td>
<td>0.59</td>
<td>14.285</td>
<td>1.21</td>
<td></td>
</tr>
<tr>
<td>SRH-A(3)</td>
<td>1.047</td>
<td>2.21</td>
<td>1.571</td>
<td>1.99</td>
<td>14.285</td>
<td>3.40</td>
<td></td>
</tr>
<tr>
<td>SRH-B</td>
<td>1.047</td>
<td>1.42</td>
<td>1.571</td>
<td>2.15</td>
<td>14.285</td>
<td>4.42</td>
<td></td>
</tr>
<tr>
<td>TPH</td>
<td>3.047</td>
<td>0.01</td>
<td>3.571</td>
<td>0.01</td>
<td>23.809</td>
<td>0.01</td>
<td></td>
</tr>
<tr>
<td>((n, p, r))</td>
<td>((g^<em>_h, g^</em>_l, g^*_v))</td>
<td>(f_1, g_1)</td>
<td>CPU time (s)</td>
<td>(f_2, g_2)</td>
<td>CPU time (s)</td>
<td>(f_3, g_3)</td>
<td>CPU time (s)</td>
</tr>
<tr>
<td>GH</td>
<td>1</td>
<td>0.03</td>
<td>1</td>
<td>0.03</td>
<td>16</td>
<td>0.03</td>
<td></td>
</tr>
<tr>
<td>SRH-A</td>
<td>1</td>
<td>1.26</td>
<td>2</td>
<td>0.46</td>
<td>8</td>
<td>0.39</td>
<td></td>
</tr>
<tr>
<td>SRH-A(2)</td>
<td>0</td>
<td>1.22</td>
<td>0</td>
<td>1.62</td>
<td>0</td>
<td>0.54</td>
<td></td>
</tr>
<tr>
<td>SRH-A(3)</td>
<td>0</td>
<td>2.01</td>
<td>0</td>
<td>2.23</td>
<td>0</td>
<td>1.87</td>
<td></td>
</tr>
<tr>
<td>SRH-B</td>
<td>0</td>
<td>1.93</td>
<td>0</td>
<td>1.87</td>
<td>0</td>
<td>2.85</td>
<td></td>
</tr>
<tr>
<td>TPH</td>
<td>1</td>
<td>1.04</td>
<td>1</td>
<td>0.19</td>
<td>12</td>
<td>1.01</td>
<td></td>
</tr>
</tbody>
</table>

Table 8.7: Comparison between heuristics, Set 1
Table 8.8: Comparison between heuristics, Set 2

<table>
<thead>
<tr>
<th>Heuristic</th>
<th>% of optimal solutions</th>
<th>% of CPU time savings over exact solution method</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRH-A</td>
<td>57.1</td>
<td>99.7</td>
</tr>
<tr>
<td>SRH-A(2)</td>
<td>90.4</td>
<td>99.5</td>
</tr>
<tr>
<td>SRH-A(3)</td>
<td>95.2</td>
<td>96.8</td>
</tr>
<tr>
<td>SRH-B</td>
<td>76.2</td>
<td>99.2</td>
</tr>
<tr>
<td>TPH</td>
<td>9.5</td>
<td>99.9</td>
</tr>
<tr>
<td>GH</td>
<td>9.5</td>
<td>99.9</td>
</tr>
</tbody>
</table>

Table 8.9: Percentage of optimal solutions and computational savings
8.6 Summary, Conclusions, and Future Research

In this chapter, we have proposed two alternative 0-1 programming models to address the problem of scheduling a doubles tennis training tournament, where it is desirable to equitably balance the partnership and opponentship frequencies between pairs of players. Model 1 can be viewed as a low-level formulation that focuses on pairing individual players, whereas Model 2 proposes a higher-level aggregate formulation aimed at matching teams. Various inherent symmetries (between rounds, players, and pairs of players), which can pose a formidable computational burden on branch-and-bound solvers, are suppressed using several classes of decision hierarchies that are judiciously derived and jointly enforced within the proposed models. These strategies are empirically shown to enhance the algorithmic convergence by several orders of magnitude. Although the pruning effect of symmetry-defeating constraints might be compared to that of valid inequalities, they are conceptually different. Such symmetry-defeating constraints can be viewed as cuts that efficiently eliminate alternative feasible solutions that do not conform with some specified round- and player-related structures, while always preserving a (smaller) subset of valid optimal solutions.

Our computational experience demonstrates the superiority of Model 2 over Model 1, and illustrates the advantage of incorporating symmetry-defeating constraints within both models. The resulting enhanced formulations $P^{t_{2,h}}(\pi_1, \pi_2), h = 1, 2, 3$, yielded an average CPU time savings of 96.9% over the respective original models $P^{t_{2,h}}, h = 1, 2, 3$, and moreover, saved an average of 91.9% of the computational effort reported for Problems $P^{R_{1,h}}(\delta_1, \delta_2), h = 1, 2, 3$. We also demonstrated that the concept of symmetry-defeating constraints can be exploited in conjunction with nonpreemptive multi-objective programming strategies in order to design effective exact solution methods. However, our computational experience indicates that the computational effort for solving $AP^\mu(\pi_1, \pi_2)$ is almost identical to that observed for solving $P^{t_{2,h}}(\pi_1, \pi_2), h = 1, 2, 3$.

To solve relatively larger problem instances in practice, several heuristics have also been proposed. A greedy heuristic (GH) generates very quick, but relatively lower-quality solutions via myopic decisions in an iterative fashion. The first sequential round-based heuristic, SRH-A, and its two implementation variants, SRH-A(2) and SRH-A(3), optimize the decisions pertaining to the current round with those for the earlier rounds fixed as determined during the previous iterations, and incorporates symmetry-defeating constraints in the first round only. Both SRH-A(2) and SRH-A(3) integrate some look-ahead feature that employs the variables pertaining to subsequent rounds as relaxed (continuous) and binary-valued variables, respectively. The second sequential
round-based heuristic, SRH-B, defines subproblems similar to those adopted in SRH-A(2), but incorporates different symmetry-defeating constraints. The two-phase heuristic, TPH, is based on Model 1, and composes teams for each round in Phase I, and subsequently, optimizes the pairing of the selected teams in Phase II. For relatively small test instances (Set 1 and \((n, p, r) = (8, 8, 7)\)), SRH-A(3) dominated all the other proposed heuristics, generating optimal solutions for 95.2% of the test problem instances, while saving 96.2% of the computational effort of the best proposed exact solution approach. The second best performance was witnessed for Heuristic SRH-A(2), which produced an optimal solution for 90.4% of the test instances, while saving 99.5% of the computational effort of the best proposed exact solution approach. SRH-A terminated in manageable times for all test problems in Sets 1-3, discovering an optimum for 57.1% of the time for the problem instances that were solved to optimality with Problem \(P_{t,h}^{\pi_1,\pi_2}, h = 1, 2, 3\), and is, therefore, an attractive alternative as the size of the problem grows.

The following additional investigations are recommended for future research:

1. For exact solution approaches, it would be interesting to further develop decision hierarchies that are jointly compatible with respect to players and rounds, and that conform as closely as possible with the schedule that the initial relaxation is tending to produce. This would trigger the cumulative benefits of the natural branch-and-bound process and the symmetry-defeating constraints.

2. For Heuristic SRH-A and its variants (SRH-A(2) and SRH-A(3)), which were the most effective among those tested, solving the first round’s subproblem is typically the most computationally challenging, but also one that assists in solving the subproblems for the subsequent rounds. Hence, it would be interesting to further enhance the solvability of the first round’s subproblem, possibly by carefully combining several decision hierarchies.
Chapter 9

Conclusions and Directions for Future Research

Broad classes of continuous and discrete nonconvex programming problems continue to pose stimulating theoretical and practical challenges. Notwithstanding several recent contributions, there is a persisting and pressing need to develop strong formulations that enable the design of efficient solution methodologies for such problems. In the perspective of solving mixed-integer programming problems to optimality, three crucial concepts come to prominence, and serve as the guiding motivation for this dissertation. First, we have reinforced the concept of developing \textit{lifted and enhanced polyhedral representations} that yield tight relaxations, thereby improving problem solvability. Second, we have demonstrated that \textit{symmetry-defeating strategies}, which impart specific identities to certain indistinguishable subsets of model defining objects, can drastically improve algorithmic performance by obviating the wasteful exploration of equivalent symmetric reflections of various classes of solutions. In particular, we have discussed the novel concept of \textit{symmetry compatible formulations} and the importance of hierarchical constraints that conform as much as possible with the model root node LP relaxation. Third, the pursuit of \textit{efficient solution methodologies} and optimization strategies has been intimately coupled with exploiting the structure and the strength of the mathematical programming formulations under investigation.

This dissertation embraces and develops modeling approaches along with algorithmic strategies for minimax problems as well as operational and logistical decision problems that arise in various industrial systems, including machine scheduling, joint vehicle assembly and routing, joint partitioning and sequencing in mixed-model assembly lines, subassembly part assignment problems, and sports scheduling. For each of these research topics, we have proposed suitable formulations that
are enhanced via lifted polyhedral representations or symmetry-defeating strategies. This effort has prompted the design of novel algorithmic features that specialize traditional optimization frameworks and that are applicable in the broader context of solving formidable combinatorial problems.

The first part of this dissertation has addressed the general class of minimax mixed-integer 0-1 programs (MMIP), and amongst the plethora of minimax optimization problems that are discussed in the literature, we have focused on the notoriously difficult job-shop scheduling problem (JSSP) with the objective of minimizing the makespan. In Chapter 3, we conducted an extensive polyhedral analysis of the general class of MMIP problems in order to tighten its representation using the Reformulation-Linearization Technique (RLT) of Sherali and Adams [106]. Many such MMIP problems are formulated as mathematical programs that are plagued by the weakness of their continuous relaxations, inducing a large optimality gap. Therefore, we explored the possibility of reducing this gap by employing the RLT methodology as well as some alternative lifting mechanisms to provide an automatic partial convexification and to generate valid inequalities that serve to strengthen the relationship between the minimax objective function and the model defining constraints. We have demonstrated that the level-1 RLT relaxation significantly tightens the MMIP representation, thereby expanding the spectrum of nonconvex problems for which the application of low-level RLT relaxations, even in partial form, has enabled improved, practical solution approaches. In addition, we have proposed RLT-enhanced Lagrangian dual formulations for this class of problems in concert with a suitable deflected/conjugate subgradient algorithm. The latter are further exploited within a novel RLT-based lifting procedure (RLP) that sequentially augments the MMIP formulation with strongest surrogate type constraints. Moreover, we have developed a sequential lifting procedure (SLP) that iteratively strengthens each “minimax constraint” via the incorporation of a nonnegative term that involves a single model-defining binary variable. It is interesting to observe that for a test-bed of randomly generated problem instances the level-1 RLT relaxation consistently outperformed Procedure SLP by yielding tighter lower bounds, while achieving appreciable computational savings.

In Chapter 4, we proposed novel continuous nonconvex as well as lifted discrete formulations for the challenging class of job-shop scheduling problems with the objective of minimizing the maximum completion time (i.e., minimizing the makespan). Although several mathematical programming formulations have been proposed for the JSSP since the late fifties, little progress has been realized with this trend of research, principally because of the weakness of the underlying continuous re-
laxations of the formulated models and the tremendous consequent computational effort required to solve the associated pure or mixed-integer programs. Motivated by the encouraging results that are reported in Chapter 3, and more generally in the literature on the benefits of the RLT methodology for minimax and discrete optimization problems, we developed an RLT-enhanced continuous nonconvex model for the job-shop problem based on a quadratic formulation of the job sequencing constraints on machines due to Nepomiastchy [95]. The lifted linear programming relaxation that is induced by this formulation was then embedded in a globally convergent branch-and-bound algorithm. Furthermore, we designed another novel formulation for the job-shop scheduling problem that possesses a tight continuous relaxation, where the non-overlapping job sequencing constraints on machines are modeled via a lifted asymmetric traveling salesman problem (ATSP) construct, and specific sets of valid inequalities and RLT-based enhancements are incorporated to further tighten the resulting mathematical program.

Our extensive computational experience revealed that the LP relaxations produced by the lifted ATSP-based models provide very tight lower bounds, and substantially reduce the optimality gap that characterizes the LP relaxation of the popular formulation due to Manne [81]. Notably, our lifted ATSP-based formulation produced a 0% optimality gap via the root node LP relaxation for 50% of the classical problem instances due to Lawrence [74], thereby substantially dominating other alternative mixed-integer programming models available for this class of problems. In addition, our results concerning the tightness of the LP relaxations of the lifted formulations echo earlier observations made in the literature on the influence of the number of machines, in particular, on the hardness of test instances. Explicitly, denoting \( n \) as the number of jobs and \( m \) as the number of machines, our computational experience confirmed that the best results are obtained for rectangular instances that are characterized by a relatively large \( \frac{n}{m} \) ratio. To take this promising empirical evidence one step further, we suggest that a theoretical investigation of dominance relationships between our ATSP-based formulation and alternative MIP formulations of the JSSP be conducted for future research. We also propose to evaluate the RLT-based Lagrangian dual formulations, and possibly integrate these within our B&B algorithm in lieu of the RLT-based linear programming relaxation to accelerate the computational performance and enhance the B&B pruning effect. Finally, it would be worthwhile to apply the general purpose lifting procedures introduced in Chapter 3 for strengthening the JSSP formulation, and compare the induced relaxation against our ATSP-based formulations that were lifted using specialized valid inequalities and RLT constructs.
The second part of this dissertation has investigated enhanced model formulations and specialized, efficient solution methodologies for applications arising in four particular industrial and sports scheduling settings. Chapter 5 addressed an integrated assembly and routing problem that was posed to us by a major trucking company (Volvo Logistics North America). Such assembly-routing problems occur in the truck manufacturing industry, as well as in the business of shipping goods via boat-towed barges along inland waterways, or via trains through railroad networks. Considering this broader context, we examined the general class of logistical systems where it is desirable to appropriately ascertain the joint composition of the sequences of vehicles that are to be physically connected along with determining their delivery routes. We presented a novel modeling framework and column generation-based optimization approach for this challenging class of joint vehicle assembly-routing problems. In addition, we suggested several extensions to accommodate particular industrial restrictions where assembly sequence-dependent delivery routes are necessary, as well as those where limited driver- and equipment-related resources are available.

This approach automates decisions with respect to assembly, inventory, delivery, driver-return, and lateness penalty costs, thereby promoting more consistent, cost-effective decisions and eliminating the labor-intensive, manual ad hoc decision-making process. Our computational results demonstrated the efficiency of the proposed methodology for the basic model, as well as its extensions and specializations. In particular, problem instances involving up to 50 vehicles over a planning horizon of three days were efficiently solved by all the proposed models. For larger problem instances, we demonstrate the benefits of employing clustering strategies. The observed efficiency of our column generation approach is essentially due to the sparsity of the columns generated as well as the flexibility in the number of vehicles that can be selected to compose any pattern within the column generation subproblem, which varies between one and four in the context of our truck manufacturing application. These features tend to mitigate the computational effort associated with the subproblem, thereby allowing the problem to be solved in manageable times. In addition, these characteristics of the problem tend to produce continuous solutions to the set partitioning problem where the selected patterns are relatively sparse and do not significantly overlap and, hence, the associated binary-restricted solutions are generally optimal or near-optimal (with an optimality gap within 1.3% when load-factor considerations are not included). For future research, we recommend investigating the use of a deflected subgradient optimization strategy along with a primal solution recovery scheme as discussed in Lim and Sherali [76] for solving the Lagrangian dual associated with the set partitioning formulations that lie at the heart of our modeling approach, in
lieu of using LP solvers, because of the following reasons. First, this can generate dual solutions for use in the subproblems relatively much faster. Second, as discussed in Subramanian and Sherali [132], this approach can reduce the dual-noise and stalling phenomena that are common with LP solvers when tackling set partitioning problems using column generation techniques (even using interior point algorithms), and can promote the generation of an improved set of columns.

Chapter 6 examined a production planning and scheduling problem faced by a major motorcycle manufacturing firm (Harley Davidson Motor Company). In this context, we addressed the problem of partitioning and sequencing the production of different manufactured items in mixed-model assembly lines, where each model has various specific options and designated destinations. We proposed two mixed-integer programming formulations for this problem that seek to sequence the manufactured goods within production batches in order to balance the motorcycle model and destination outputs as well as the load demands on material and labor resources. Recognizing the symmetry inherent to the second MIP formulation, we enforced an additional set of hierarchical symmetry-defeating constraints that impart specific identities amongst batches of products under composition. The latter model inspired a third set partitioning-based formulation in concert with an efficient column generation approach that directly achieves the joint partitioning of jobs into batches along with ascertaining the sequence of jobs within each composed batch. Finally, we explored a subgradient-based optimization strategy that exploits a non-differentiable optimization formulation, which is prompted by the flexibility in the production process as reflected in the model via several soft-constraints, thereby providing a real-time decision-making tool. Our computational experience demonstrated the relative effectiveness of the different proposed formulations and the associated optimization strategies for solving a set of realistic problem instances. We also exhibited the insightful benefits of a complementary column generation (CCG) feature that is further discussed below. The versatility of our integrated modeling and optimization framework renders it attractive to capture additional specifications that arise in mixed-model assembly lines such as the incorporation of due-dates and set-up cost considerations. Future research can address the interesting problem of integrating our partitioning-sequencing model with the problem of line balancing in assembly lines.

Chapter 7 was devoted to the subassembly parts assignment problem (SPAP) that aims at composing final assemblies in a fashion that minimizes the total deviation from a vector of nominal and mean values of certain quality characteristics that are of interest to the manufacturer in
the assembled products. Upon recognizing the symmetry inherent within this problem, we revisited and enhanced an earlier assignment-based formulation (SPAP1) [92] via symmetry-defeating strategies. The latter either incorporates tailored hierarchical constraints (Models SPAP1(σ) and SPAP1(φ)) that impart specific identities to sets of originally indistinguishable variables, or a priori assigns (without loss of optimality) the subassembly parts that belong to a single group to the final products (Model SPAP2). Although Sherali and Smith [124] have discussed the conceptual and computational benefits of appending such symmetry-defeating constraints, they have also called for caution against the negative effects that might result from altering certain specially structured formulations by enforcing general hierarchical constraints. Indeed, our study revealed that whereas our proposed symmetry-defeating enhanced formulation SPAP2 significantly outperforms Model SPAP1, both the sets of hierarchical constraints (σ) and (φ) consistently deteriorate the algorithmic performance. Despite imposing unique identities to the products under assembly, such (σ) and (φ) hierarchical constraints do not preserve the structure underlying the assignment-based formulations, thereby rendering the popular mathematical programming software (CPLEX) ineffective. In contrast, Model SPAP2 offers an appealing alternative that combines the benefits of combating the inherent problem symmetry and preserving the problem structure. In addition, we developed several heuristic procedures.

A novel concept of complementary column generation (CCG) was also proposed in this connection, which aims at iteratively complementing any column that prices out favorably in the subproblem, thereby simultaneously augmenting the restricted master program with a block of patterns that constitutes a feasible solution to the set partitioning problem. We have provided insights into the vital role of this strategy for the SPAP, and recommend that this algorithmic feature be judiciously employed in the broader context of set partitioning-based formulations that are characterized by columns having relatively dense non-zero values.

Finally, we have studied a doubles tennis scheduling problem in Chapter 8 in the context of a training tournament as prompted by a tennis club in Virginia, and have developed two alternative 0-1 mixed-integer programming models, each with three different objective functions that attempt to balance the partnership and the opponentship pairings among the players. Model 1 can be viewed as a low-level formulation that focuses on pairing individual players, whereas Model 2 proposed a higher-level aggregate formulation aimed at matching teams. Various inherent symmetries (between rounds, players, and pairs of players), which can pose a formidable computational burden
on branch-and-bound solvers, were suppressed using several classes of judicious decision hierarchies that were jointly enforced within the proposed models. These strategies are empirically shown to enhance the algorithmic convergence by several orders of magnitude. Although the pruning effect of symmetry-defeating constraints might be compared to that of valid inequalities, they are conceptually different. Such symmetry-defeating constraints can be viewed as cuts that efficiently eliminate alternative feasible solutions that do not conform with some specified round- and player-related structures, while always preserving a (smaller) subset of valid optimal solutions. We also investigated objective function perturbation strategies to combat symmetry, and have proposed a novel nonpreemptive multi-objective programming strategy in concert with decision hierarchies. We have highlighted the conceptual value of alternatively defeating symmetry by using either hierarchical constraints or objective perturbation strategies to enhance problem solvability. Our study has provided the insight that the special structure of the mathematical program to which specific tailored symmetry-defeating constraints are appended can greatly influence their pruning effect. For exact solution approaches, it would be interesting to further develop decision hierarchies that are jointly compatible with respect to players and rounds, and that conform as closely as possible with the schedule that the initial relaxation is tending to produce. This would trigger the cumulative benefits of the natural branch-and-bound process and the symmetry-defeating constraints.

Beyond our contributions to the different topics addressed in this dissertation, it is our belief and hope that many generalizable elements of our work will serve as an inspiration for solving difficult combinatorial optimization problems. Clearly, the lifting concepts and procedures, the alternative hierarchical constraint as well as objective function perturbation-based symmetry-defeating strategies, the complementary column generation feature, and the early termination criteria within column generation frameworks that we have designed can be readily applied to several combinatorial optimization problems, and in particular, to different types of scheduling problems.
Appendix A

Mathematical Programming

Formulations for the JSSP

This appendix provides a review of several mathematical programming formulations that have been proposed for the JSSP. The notation introduced in Section 4.1.1 is employed here as well, in addition to the following additional decision variables pertaining to the models presented herein.

Decision Variables

- \( C_{ij} = t_{ij} + p_{ij} \) = completion time of job \( j \) on machine \( i \).
- \( s_{ij\ell} = \begin{cases} 1 & \text{if the operation of job } j \text{ is scheduled in the } \ell^{th} \text{ position for processing on machine } i \\ 0 & \text{otherwise} \end{cases} \), \( \forall i \in M, j \in J, \ell = 1, \ldots, n \)
- \( v_{ijk} = \begin{cases} 1 & \text{if } C_{ij} = k \\ 0 & \text{otherwise} \end{cases} \), \( \forall i \in M, j \in J, k = 1, \ldots, T \)
- \( w_{ijk} = \begin{cases} 1 & \text{if the operation of job } j \text{ is being processed during the } k^{th} \text{ time-unit on machine } i \\ 0 & \text{otherwise} \end{cases} \), \( \forall i \in M, \forall j \in J, k = 1, \ldots, T \)

Also, let \( T_0 = \) be some lower bound on the makespan.

A.1 Wagner’s Model (1959)

Wagner [142] presented a general model to capture various constraints that arise in scheduling problems. In particular, the following model is suitable for the JSSP. In this formulation, the set of binary variables, \( s \), reflects the position of each job in the processing sequence, and the set of nonnegative real variables, \( I \), corresponds to idle times on machines. Let \( t_{i(\ell)} \) be the starting time
of the \( \ell \)th job to be processed on machine \( i \), \( I_{i\ell} \) be the idle time between the \( \ell \)th and the \((\ell + 1)\)th job scheduled on machine \( i \), and \( I_{i0} \) be the time elapsed on machine \( i \) between the initiation of the production and the start time of the first item to be processed on machine \( i \). Note that \( t_{i(1)} = I_{i0} \), \( i = 1, \ldots, m \) and \( t_{i(\ell)} = I_{i\ell} + p_{i,1(1)} + I_{i(1)} + \ldots + p_{i,\ell-1} + I_{i,\ell-1} \). The model is summarized below.

\begin{align*}
\text{Minimize} & \quad C_{\text{max}} \tag{A.1a} \\
\text{subject to} & \quad \sum_{j=1}^{n} s_{ij\ell} = 1, \quad \forall i \in M, \ell = 1, \ldots, n \tag{A.1b} \\
& \quad \sum_{\ell=1}^{\ell-1} s_{ij\ell} = 1, \quad \forall i \in M, \forall j \in J \tag{A.1c} \\
& \quad t_{i(\ell)} = \sum_{h=0}^{\ell-1} I_{ih} + \sum_{h=1}^{\ell-1} \sum_{j=1}^{n} p_{ij}s_{ijh}, \quad \forall i \in M, \ell = 2, \ldots, n \tag{A.1d} \\
& \quad t_{i(1)} = I_{i0}, \quad \forall i \in M \tag{A.1e} \\
& \quad t_{i_{1}(\ell_{1})} + p_{i_{1}j}s_{i_{1}j\ell_{1}} \leq t_{i_{2}(\ell_{2})} + K(2 - s_{i_{1}j\ell_{1}} - s_{i_{2}j\ell_{2}}), \quad \forall \ell_{1}, \ell_{2} = 1, \ldots, n, (i_{1}j, i_{2}j) \in A_{j}, \forall j \in J \tag{A.1f} \\
& \quad t_{i(n)} + \sum_{j=1}^{n} p_{ij}s_{ijn} \leq C_{\text{max}}, \quad \forall i \in M \tag{A.1g} \\
& \quad s \quad \text{binary,} \tag{A.1h}
\end{align*}

where \( K \) is a suitable large number.

Constraints (A.1b) and (A.1c) enforce classical assignment constraints. Constraints (A.1d) and (A.1e) guarantee that the processing of the \((\ell + 1)\)th ordered operation on machine \( i \) cannot start unless the \( \ell \)th ordered operation on the same machine is fully processed. Constraint (A.1f) expresses job constraints, that is, the second of two consecutive operations of any job cannot start unless the first one is processed (Consider some job \( j \) and some pair of consecutive operations \((i_{1}j, i_{2}j) \in A_{j}\). Assume that \( \ell_{1} \) and \( \ell_{2} \) are the positions of job \( j \) on machine \( i_{1} \) and machine \( i_{2} \) respectively). Constraint (A.1g) reflects the definition of the makespan and Constraint (A.1h) enforces logical binary restrictions on the \( s \)-variables.
A.2 Bowman’s Model (1959)

Bowman [22] suggested a time-indexed formulation for the job-shop scheduling problem, where the scheduling horizon is divided into $T$ time-units. Binary variables, $w$, are introduced to reflect the processing of operations over time-units. The model is stated below.

\[
\text{Minimize } \sum_{k=0}^{T-T_0} \sum_{ij \in E^*} (n+1)^k w_{ij,T_0+k} \tag{A.2a}
\]

subject to

\[
\sum_{k=1}^{T} w_{ijk} = p_{ij}, \quad \forall i \in M, j \in J \tag{A.2b}
\]

\[
p_{ij}(w_{ijk} - w_{ij,k+1}) + \sum_{h=k+2}^{T} w_{ijh} \leq p_{ij},
\]

\[
k = 1, \ldots, T, \forall i \in M, \forall j \in J \tag{A.2c}
\]

\[
p_{i_1j}w_{i_2j} \leq \sum_{h=1}^{k-1} w_{i_1jh}, \quad \forall (i_1j, i_2j) \in A_j, j \in J, k = 1, \ldots, T \tag{A.2d}
\]

\[
\sum_{j=1}^{n} w_{ijk} \leq 1, \quad \forall i \in M, k = 1, \ldots, T \tag{A.2e}
\]

$w$ binary, \tag{A.2f}

The full processing of operations over the scheduling horizon is ensured via Constraint (A.2b). Constraint (A.2c) forbids job splitting or preemption. Job constraints are enforced in constraints (A.2d), and machine constraints are imposed in (A.2e). The objective function in (A.2a) involves the last operation of each job. For those operations, an increasing cost is associated for time-units between $T_0$ and $T$. Since operations processed at the end of the scheduling horizon become most costly, the optimization effort leads to scheduling the last operation of each job as early as possible. This formulation forces schedules to be as compact as possible and, therefore, minimizes the makespan. Bowman suggests the sum of the processing times of all operations and the sum of the processing times of the longest job as possible values for $T$ and $T_0$ respectively. Constraint (A.2f) enforces logical binary restrictions on the $w$-variables. Note that $nmT$ binary variables are required in this model.
A.3 Pritsker et al.’s Model (1969)

In a spirit similar to that of Bowman’s model, Pritsker et al. [99] developed a more compact formulation for multiproject scheduling under resource constraints. In particular, their approach is suitable for the JSSP. Let $\bar{t}_{ij}$ and $\bar{u}_{ij}$ be some lower bound and upper bound, respectively, on the completion time of operation $O_{ij}$. We take $\bar{t}_{ij} = \sum_{k:O_{kj} \in P(O_{ij})} p_{kj} + p_{ij}$ and $\bar{u}_{ij} = T - \sum_{k:O_{kj} \in S(O_{ij})} p_{kj}$.

We state this model as follows.

$$\text{Minimize } C_{\text{max}} \quad (A.3a)$$

subject to

$$\sum_{k = \bar{t}_{ij}} \bar{u}_{ijk} = 1, \forall i \in M, j \in J$$

$$\sum_{k = \bar{t}_{i1j}} \bar{u}_{i1j} + p_{i2j} \leq \sum_{k = \bar{t}_{i2j}} \bar{u}_{i2j}, \forall (i1j, i2j) \in A_j, j \in J$$

$$\sum_{j=1}^{n} \sum_{h=k}^{k+p_{ij}-1} v_{ijh} \leq 1, \forall k = 1, ..., T - \min_{j \in J} \{p_{ij}\}, i = 1, ..., m$$

$$\sum_{k = \bar{t}_{ij}} \bar{u}_{ij} \sum_{k = \bar{t}_{ij}} \bar{u}_{ijk} \leq C_{\text{max}}, \forall j \in E^*_i, i \in M$$

$v$ binary. \quad (A.3f)

Constraint (A.3b) prevents job splitting or preemption. Constraint (A.3c) enforces the precedence constraints among pairs of consecutive operations that belong to job $j$, whereas Constraint (A.3d) imposes the non-overlapping job sequencing constraints on machines. The objective function in (A.3a) along with Constraint (A.3e) express the objective of minimizing the makespan. Constraint (A.3f) introduces logical binary restrictions on the $v$-variables.

A.4 Von Lanzenauer and Himes’ Model (1970)

The integer linear programming formulation by Von Lanzenauer and Himes [140] is very close to the early model introduced by Bowman. The originality of this model, however, is that the job splitting constraints and the job constraints, which involve coefficients other than zero or one in Bowman’s model, are reformulated so that the new model involves coefficients that are zero or one only. The hope is that the use of some linear programming code on the resulting zero-one coefficient
matrix would produce an integer, optimal solution. This result cannot be guaranteed in general, some favorable special cases notwithstanding.

Again, the main set of variables is that introduced in Bowman’s model. Provision is made by introducing the set of variables \( \kappa_{ijk}, \forall i \in M, j \in J, k = 1, ..., T \), so as to reformulate the job splitting constraints and the precedence relationship for two consecutive operations of the same job. The zero-one variable \( \kappa_{ijk} = 1 \) if and only if \( w_{ijk} - w_{ij,k+1} = 1 \), that is, operation \( O_{ij} \) is completed or interrupted at the end of the \( k \)-th time-unit. Also, \( l_{ij} \) and \( u_{ij} \) provide bounds on the feasible range of processing as in Pritsker et al.’s model.

Minimize
\[
\sum_{k=1}^{T} \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ijk} w_{ijk}
\]  \hspace{1cm} (A.4a)

subject to
\[
\sum_{j=1}^{n} w_{ijk} \leq 1, \quad \forall i \in M, k = 1, ..., T \]  \hspace{1cm} (A.4b)

\[
\sum_{k=l_{ij}}^{u_{ij}} w_{ijk} = p_{ij}, \quad \forall i \in M, j \in J \]  \hspace{1cm} (A.4c)

\[
\sum_{i=1}^{m} w_{ijk} \leq 1, \quad \forall j \in J, k = \min_{i=1, ..., m} \{l_{ij}\}, ..., \max_{i=1, ..., m} \{u_{ij}\} \]  \hspace{1cm} (A.4d)

\[
\kappa_{ijk} \geq w_{ijk} - w_{ij,k+1}, \quad \forall i \in M, j \in J, k = l_{ij}, ..., u_{ij} \]  \hspace{1cm} (A.4e)

\[
\sum_{k=l_{ij}}^{u_{ij}} \kappa_{ijk} = 1, \quad \forall i \in M, j \in J \]  \hspace{1cm} (A.4f)

\[
\sum_{h=l_{i1j}}^{k} \kappa_{i1jh} - \sum_{h=l_{i1j}+1}^{k+1} \kappa_{i2jh} \geq 0, \quad \forall (i_{1j}, i_{2j}) \in A_{j}, \quad j \in J, k \in \{(l_{i1j}, ..., u_{i1j}) \cap (l_{i2j}, ..., u_{i2j})\} \]  \hspace{1cm} (A.4g)

\( w, \kappa \) binary.  \hspace{1cm} (A.4h)

Constraint (A.4b) enforces machine constraints, whereas the full processing constraint for each operation over the scheduling horizon is introduced in (A.4c). Constraint (A.4d) ensures that any job can be processed by at most one machine at a time. Prevention of job splitting is achieved through Constraints (A.4e) and (A.4f). Since \( \kappa_{ijk} \) assumes a value of 1 only once over the scheduling horizon for any operation \( O_{ij} \), \( (w_{ijk} - w_{ij,k+1}) \) is also equal to one only once, that is, when the
operation is fully processed. Note that if \( w_{ijk} - w_{ij,k+1} = 1 \) before \( O_{ij} \) is fully processed, Constraint (A.4c) will be violated. Job constraints are introduced in (A.4g). The objective function is expressed in (A.4a), where \( a_{ijk} \) is some positive weight that increases for \( k \) varying between \( l_{ij} \) and \( u_{ij} \) to make the schedule as compact as possible.

A.5 Wilson’s Model (1989)

Based on Wagner’s definition of binary variables, Wilson [151] has suggested an alternative formulation for the permutation flow-shop scheduling problem. His model can be adapted to the classical job-shop scheduling problem. The main difference between the two models resides in the formulation of the machine constraints. Note that the Wilson-like model does not involve nonnegative continuous variables that represent idleness on machines. The full model is given below, and its description is similar to that provided in Section A.1.

\[
\text{Minimize } C_{\text{max}} \quad (A.5a)
\]

subject to \( \sum_{j=1}^{n} s_{ij\ell} = 1, \quad \forall i \in M, \ell = 1, \ldots, n \) \( (A.5b) \)

\( \sum_{\ell=1}^{n} s_{ij\ell} = 1, \quad \forall i \in M, \forall j \in J \) \( (A.5c) \)

\( t_{i(\ell)} + \sum_{j=1}^{n} p_{ij}s_{ij\ell} \leq t_{i,(\ell+1)}, \quad \forall \ell = 1, \ldots, n - 1, i \in M \) \( (A.5d) \)

\( t_{i_1(\ell_1)} + p_{i_1j}s_{i_1j\ell_1} \leq t_{i_2(\ell_2)} + K(2 - s_{i_1j\ell_1} - s_{i_2j\ell_2}), \)

\( \forall \ell_1, \ell_2 = 1, \ldots, n, (i_1j, i_2j) \in A_j, j \in J \) \( (A.5e) \)

\( t_{i(n)} + \sum_{j=1}^{n} p_{ij}s_{ijn} \leq C_{\text{max}}, \quad \forall i \in M \) \( (A.5f) \)

\( s \quad \text{binary.} \) \( (A.5g) \)

A.6 Morton and Pentico’s Model (1993)

The time-indexed formulation proposed by Morton and Pentico [91] for the JSSP combines various ideas from earlier formulations. Continuous variables that nonetheless have a binary connotation are introduced to reflect the processing of operations over time-units as in Bowman’s model; \( w_{ijk} = 1 \) if and only if operation job \( j \) is being processed on machine \( i \) during the \( k^{th} \) time-unit. An
additional set of binary variables is used to reflect the completion times of operations; $v_{ijk} = 1$ if and only if $C_{ij} = k$, as in Pritsker et al.’s model. The associated mathematical program is stated below.

Minimize $C_{max}$

subject to $C_{max} \geq C_{ij}$, $\forall j \in E_i^*, i \in M$  

$C_{ij} = \sum_{k=1}^{T} kv_{ijk}$, $\forall i \in M, j \in J$  

$w_{ijk} = \sum_{h=k}^{k+p_{ij}-1} v_{ijh}$, $\forall i \in M, j \in J, k = 1, ..., T$  

$C_{i2j} \geq C_{i1j} + p_{i2j}$, $\forall j \in J, (i1j, i2j) \in A_j$  

$\sum_{j=1}^{n} w_{ijk} \leq 1$, $\forall i \in M, k = 1, ..., T$  

$v$ binary.

The objective function (A.6a) in conjunction with Constraint (A.6b) express the objective of minimizing the makespan. Constraint (A.6c) provides an expression of completion time of any operation in terms of the model defining variables. Constraints (A.6d) and (A.6f) enforce the non-overlapping job sequencing constraints on machines, whereas Constraint (A.6e) imposes the precedence constraints between any pair of consecutive operations that belong to job $j$. Finally, Constraint (A.6f) enforces logical binary restrictions on the $v$-variables. Observe that Problem (A.6) induces the binariness of the $w$-variables and that this model can be entirely formulated in terms of the $v$-variables only.
Bibliography


[74] Lawrence, S. (1984), Supplement to resource constrained project scheduling: An experimental investigation of heuristic scheduling techniques, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA.


[133] Suzuka, A., Miyashiro, R., Yoshise, A. and Matsui, T. (2005), Semidefinite programming based approaches to home-away assignment problems in sports scheduling, *METR2005-07*, Department of Mathematical Engineering and Information Physics, Faculty of Engineering, The University of Tokyo.


Vita

Ahmed Ghoniem was born in 1980 in Egypt. He has received a French education in Kuwait, Egypt, and Qatar, and obtained in 1997 his French Baccalaureat (A-level Science equivalent in Mathematics and Physics) with distinction, ranking 1st at the French schools in the Middle East. He was awarded an Excellence Scholarship from the French Government, and started his engineering studies in France in 1997. In 2001, he pioneered the dual M.S. degree co-developed by the Industrial and Systems Engineering Department at Virginia Tech and the École des Mines de Nantes (EMN, France). In 2002, he received an M.S. in Operations Management in Production and Logistics from EMN, and obtained his M.S. in Industrial and Systems Engineering from Virginia Tech in 2003. He started his Ph.D. program in 2004 and, upon completing his Ph.D. program, he will assume a Postdoctoral Fellow position at Virginia Tech in August 2007.