Branch-and-Price Method for Stochastic Generalized Assignment Problem, Hospital Staff Scheduling Problem and Stochastic Short-Term Personnel Planning Problem

Seon Ki Kim

Dissertation submitted to the Faculty of
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Industrial and Systems Engineering

Dr. Subhash C. Sarin, Chair
Dr. Robert Hendricks
Dr. Ebru K. Bish
Dr. Barbara Fraticelli

March 5, 2009
Blacksburg, Virginia

Keywords: Branch-and-Price Method, Monte Carlo Method, Stochastic Programming, Branch-and-Cut Method, Dual Stabilization Technique, Greedy Heuristic, Stochastic Generalized Assignment Problem, Hospital Staff Scheduling Problem, Stochastic Hospital Staff Scheduling Problem, Stochastic Short-Term Personnel Planning Problem

Branch-and-Price Method for Stochastic Generalized Assignment Problem, Hospital Staff Scheduling Problem and Stochastic Short-Term Personnel Planning Problem

Seon Ki Kim

Abstract

The work presented in this dissertation has been focused on exploiting the branch-and-price (BNP) method for the solution of various stochastic mixed integer programming problems (MIPs). In particular, we address the stochastic generalized assignment problem (SGAP), a hospital staff scheduling problem (HSSP), a stochastic hospital staff scheduling problem (SHSSP), and a stochastic short-term personnel planning problem (SSTPP). The BNP method has been developed in concert with the dual stabilization technique and other enhancements of this method for each of these problems. In view of an excessive number of scenarios that arise for these problems, we also implement the Monte Carlo method within the BNP scheme. The superiority of the BNP-based method over the branch-and-cut (BNC) method is demonstrated for all of these problems.

The first problem that we address is the SGAP for which the processing time of a job on a machine is assumed to be stochastic. Even though the generalized assignment problem (GAP) has been solved using the BNP method, yet no study has been reported in the literature on the use of the BNP method for the solution of the SGAP. Our work has been motivated by the desire to fill this gap.

We begin by showing that it is better to solve the SGAP as a stochastic program in contrast to solving it by using the expected values of the times required to process the jobs on the machines. Then, we show that the stochastic model of the SGAP is a complete recourse model – a useful property which permits the first stage decisions to produce feasible solutions for the recourse problems. We develop three BNP-based methods for the solution of the SGAP. The first

of these is **BNP-SGAP**, which is a combination of branch-and-bound and column generation methods. The pricing problem of **BNP-SGAP** is separable with regard to each machine, and it is a multiple-constraint knapsack problem. The second method is **BNP-SGAP** implemented in concert with the dual stabilization technique (DST), and it is designated as **BNPDST-SGAP**. We have introduced a new DST by modifying the Boxstep method of Pigatti et al. [76]. We have shown that our method performs better than the method of Pigatti et al. [76] resulting in over two-fold savings in cpu times on average. The third method that we develop for the solution of the SGAP is **BNPDST-SGAP** implemented with an advanced start to obtain an initial feasible solution. We use a greedy heuristic to obtain this solution, and this heuristic is a modification of a similar method used for the knapsack problem. It relies on the information available at a node of the underlying branch-and-bound tree. We have shown that this procedure obtains an initial feasible solution, if it exists at that node. We designate this method as **BNPDSTKP-SGAP**. We have also developed a BNC method to solve the SGAP using CPLEX 9.0. We have compared the performances of the BNP and BNC methods on various problem instances obtained by varying the number of machines, the ratio of the number of machines to the number of jobs, the machine capacity, and the penalty cost per unit of extra resource required at each machine. Our results show that all BNP-based methods perform better than the BNC method, with the best performance obtained for **BNPDSTKP-SGAP**.

An issue with the use of the scenario-based methods that we have employed for the solution of the SGAP is that the number of scenarios generally grows exponentially in problem parameters, which gives rise to a large-size problem. To overcome the complexity caused by the presence of a large number of scenarios for the solution of the SGAP, we introduce the use of the Monte Carlo method (MCM) within the BNP scheme. We designate this method as **BNPDSTKP-SGAP with MCM**. It affords the use of a small subset of scenarios at a time to estimate the "true" optimal objective function value. Replications of the subsets of scenarios are carried out until the objective function value satisfies a stopping criterion. We have established theoretical results for the use of the MCM. These pertain to determining unbiased estimates of: (i) lower and upper bounds of the "true" optimal objective function value, (ii) the "true" optimal solution, and (iii) the optimality gap. We have also provided the $100(1-\alpha)$ confidence interval on the optimality gap. Our experimental investigation has shown the efficacy of using this method. It obtains almost optimal solutions, with the objective function value lying within 5% of the

"true" optimal objective function value, while giving almost ten-fold savings in cpu time. Our experimentation has also revealed that an increment in the number of scenarios in each replication makes a greater impact on the quality of the solution obtained than an increment in the number of replications. We have also observed the impact of a change in the variance of a processing time distribution on cpu time. As expected, the optimal objective function value increases with increment in processing time variability. Also, by comparing the results with the expected value solution, it is observed that the greater the variability in the data, the better it is to use the stochastic program.

The second problem that we study is the hospital staff scheduling problem. We address the following three versions of this problem: HSSP (General): Implementation of schedule incorporating the four principal elements, namely, surgeons, operations, operating rooms, and operation times; HSSP (Priority): Inclusion of priority for some surgeons over the other surgeons regarding the use of the facility in HSSP (General); HSSP (Pre-arranged): Implementation of a completely pre-fixed schedule for some surgeons. The consideration of priority among the surgeons mimics the reality. Our BNP method for the solution of these problems is similar to that for the SGAP except for the following: (i) a feasible solution at a node is obtained with no additional assignment, i.e., it consists of the assignments made in the preceding nodes of that node in the branch-and-bound tree; (ii) the columns with positive reduced cost are candidates for augmentation in the CGM; and (iii) a new branching variable selection strategy is introduced, which selects a fractional variable as a branching variable by fixing a value of which we enforce the largest number of variables to either 0 or 1. The priority problem is separable in surgeons.

The results of our experimentation have shown the efficacy of using the BNP-based method for the solution of each HSSP as it takes advantage of the inherent structure of each of these problems. We have also compared their performances with that of the BNC method developed using CPLEX. For the formulations **HSSP (General)**, **HSSP (Priority)**, and **HSSP (Pre-arranged)**, the BNP method gives better results for 22 out of 30, 29 out of 34, and 20 out 32 experiments over the BNC method, respectively. Furthermore, while the BNC method fails to obtain an optimal solution for 15 experiments, the BNP method obtains optimal solutions for all 96 experiments conducted. Thus, the BNP method consistently outperforms the BNC method for all of these problems.

The third problem that we have investigated in this study is the stochastic version of the HSSP, designated as the Stochastic HSSP (SHSSP), in which the operation times are assumed to be stochastic. We have introduced a formulation for this formulation, designated as **SHSSP2 (General)**, which allows for overlapping of schedules for surgeons and operating rooms, and also, allows for an assignment of a surgeon to perform an operation that takes less than a pre-arranged operation time, but all incurring appropriate penalty costs. A comparison of the solution of **SHSSP2 (General)** and its value with those obtained by using expected values (the corresponding problem is designated as **Expected-SHSSP2 (General)**) reveals that **Expected-SHSSP2 (General)** may end up with inferior and infeasible schedules. We show that the recourse model for **SHSSP2 (General)** is a relatively complete recourse model. Consequently, we use the Monte Carlo method (MCM) to reduce the complexity of solving **SHSSP2 (General)** by considering fewer scenarios. We employ the branch-and-cut (BNC) method in concert with the MCM for solving **SHSSP2 (General)**. The solution obtained is evaluated using tolerance ratio, closeness to optimality, length of confidence interval, and cpu time. The MCM substantially reduces computational effort while producing almost optimal solutions and small confidence intervals.

We have also considered a special case of **SHSSP2 (General)**, which considers no overlapping schedules for surgeons and operating rooms and assigns exactly the same operation time for each assignment under each scenario, and designate it as **SHSSP2 (Special)**. With this, we consider another formulation that relies on the longest operation time among all scenarios for each assignment of a surgeon to an operation in order to avoid scheduling conflicts, and we designate this problem as **SHSSP (Longest)**. We show **SHSSP (Longest)** to be equivalent to deterministic HSSP, designated as **HSSP (Equivalent)**, and we further prove it to be equivalent to **SHSSP (General)** in terms of the optimal objective function value and the optimal assignments of operations to surgeons. The schedule produced by **HSSP (Equivalent)** does not allow any overlap among the operations performed in an operating room. That is, a new operation cannot be performed if a previous operation scheduled in that room takes longer than expected. However, the schedule generated by **HSSP (Equivalent)** may turn out to be a conservative one, and may end up with voids due to unused resources in case an operation in an operating room is completed earlier than the longest time allowed. Nevertheless, the schedule is still a feasible one. In such a case, the schedule can be left-shifted, if possible, because the

scenarios are now revealed. Moreover, such voids could be used to perform other procedures (e.g., emergency operations) that have not been considered within the scope of the SHSSP addressed here. Besides, such a schedule can provide useful guidelines to plan for resources ahead of time.

The fourth problem that we have addressed in this dissertation is the stochastic short-term personnel planning problem, designated as Stochastic STPP (SSTPP). This problem arises due to the need for finding appropriate temporary contractors (workers) to perform requisite jobs. We incorporate uncertainty in processing time or amount of resource required by a contractor to perform a job. Contrary to the SGAP, the recourse model for this problem is not a relatively complete recourse model. As a result, we cannot employ a MCM method for the solution of this problem as it may give rise to an infeasible solution. The BNP method for the SSTPP employs the DST and the advanced start procedure developed for the SGAP, and due to extra constraints and presence of binary decision variables, we use the branching variable selection strategy developed for the HSSP models. Because of the distinctive properties of the SSTPP, we have introduced a new node selection strategy. We have compared the performances of the BNC-based and BNP-based methods based on the cpu time required. The BNP method outperforms the BNC method in 75% of the experiments conducted, and the BNP method is found to be quite stable with smaller variance in cpu times than those for the BNC method. It affords solution of difficult problems in smaller cpu times than those required for the BNC method.

Dedication

This dissertation is dedicated to my parents, Jae Kwon Kim and JeongSook Baek, my brother, SeungJeong Kim, my son, Isaac Kim, and my wife, Mi Hye Hwang. I would not be able to finish this dissertation and obtain Ph.D degree without their support and love for me.

Acknowledgements

This dissertation has been successfully completed only because of Dr. Subhash C. Sarin, who is my mentor as well as my advisor. He has supported me in every possible way that includes scholarship, wonderful ideas, and invaluable advice. I could not have overcome all the difficulties that I ran into over all these years without his keen insights and great encouragement. He did not let me give up, and he always found a way to help me relaxed and focused on my work when I struggle. I would also like to thank my other committee members, including Dr. Robert Hendricks, Dr. Ebru Bish, and Dr. Barbara Fraticelli, for their great support, warm consideration, and valuable suggestions during the course of this dissertation. Also, I could not have finished this dissertation without their support.

I appreciate all the help that I received from the Grado Department of Industrial and Systems Engineering at Virginia Tech. The Graduate Teaching Assistantships and scholarships that I received were not only useful for devoting myself to studies and research, but also they provided financial support that I needed the most. Also, I have to thank the Professors at Virginia Tech who I took class es from. All of them contributed in enhancing my knowledge base that I relied upon in my dissertation. Their kindness as well as their wisdom during the classes that they taught made me focused ever since my first semester at Virginia Tech. Especially, I would like to thank Dr. Hanif Sherali for his great lectures and his kind words and Dr. Patrick Koelling for his great support. I really appreciate all the ISE staff for their timely help, especially, Mrs. Kim Ooms and Mr. Will Vest.

Fortunately, I have made a number of good friends during my stay at Virginia Tech. I thank them for their friendship. I, especially, appreciate the close relationships that I have with all the past and current colleagues of the Electronics Manufacturing Research (EMR) Laboratory, where we worked together under the guidance of Dr. Subhash Sarin. Also, I wish to thank by Korean friends in the Industrial and Systems Engineering department with whom I had fun time.

Finally, I would like to thank my parents, my brother, my son, and my wife, for all their support and sacrifice that I have received from them in pursuit of my Ph.D. degree. They are the source of my drive to work with passion. Especially, I cannot thank enough my wife, Mi Hye Hwang, for her patience and love since the first day I met her.

CONTENTS

Chapter I. Introduction 1

Chapter II. Background Information and Literature Review 5

2.1. Generalized Assignment Problem (GAP) 5

2.2. Branch-and-Cut and Branch-and-Price Methods 6

2.2.1. Branch-and-Cut Method 6

2.2.2. Branch-and-Price Method 8

2.2.3. General Column Generation Method 13

2.2.4. Relationship between the Column Generation and Branch-and-Price Methods 16

2.2.5. Dual Stabilization Technique for the Branch-and-Price Method 19

2.2.6. Branching Variable Selection Strategies for the Branch-and-Price Method 25

2.2.7. Application Areas of the Branch-and-Price Method 25

2.3. Stochastic Programming 26

2.3.1. Optimization with Expected Values 26

2.3.2. Optimization with Wait-and-See Policy 28

2.3.3. Optimization of a SP with a Simple Recourse Model 29

2.3.4. Optimization of a SP with Two General Recourse Models 30

2.3.5. Classification of Recourse Policy in the Stochastic Programming (SP) 32

2.4. Monte Carlo Method 33

2.4.1. Application Areas 33

2.4.2. Basic Statistics for the MCM 34

2.4.3. Steps for Implementing the MCM 35

2.4.4. Sampling Methods for the MCM 35

2.4.5. Exterior Sampling Method (ESM) 37
2.4.5.1. Formulation for the ESM 37
2.4.5.2. Average and Variance of M Replications 37
2.4.5.3. Lower Bound of *h* 38
2.4.5.4. Upper Bound of *h* and the Unbiased Estimator of $h\left(\hat{X}_N^m\right)$ 38
2.4.5.5. Variance of $h_{\tilde{N}}\left(\hat{X}_N^m\right)$ 38
2.4.5.6. Candidate for the Optimal Solution 39
2.4.5.7. Evaluation of the Quality of $\left(\hat{X}_N\right)^*$ 39
2.4.5.8. Application Areas of the Sample Average Approximation (SAA) Method 40
2.5. Stochastic Generalized Assignment Problem (SGAP) 41
2.5.1. Generalized Assignment Problem (GAP) 41
2.5.2. Greedy Heuristic for the Knapsack Problem 42
2.5.3. Literature on the Stochastic Generalized Assignment Problem 43
2.6. Hospital Scheduling Problem 44
2.6.1. Staff Scheduling Problem 44
2.6.2. Operating Room Scheduling Problems 47
2.6.3. Stochastic Hospital Scheduling Problem (SHSP) 48
2.6.3.1. Literature Review on SHSP 48
2.6.3.2. A Formulation for the SHSP 49
2.7. Personnel Planning Problem 50
2.7.1. Short-Term and Long-Term Personnel Planning Problems 50
2.7.2. Literature Review on Personnel Planning Problems 51

Chapter III. A Branch-and-Price Method for the Stochastic Generalized Assignment Problem (SGAP) 53

3.1. Stochastic Generalized Assignment Problem 53
3.1.1. Model Formulation for the SGAP and its Properties 53

3.1.1.1. The SGAP and its Recourse Model 53
3.1.1.2. Stochastic Properties of the Recourse Model for SGAP1 56
3.1.2. An Illustrative Example of the SGAP 57
3.1.3. SGAP with Expected Values 59
3.1.3.1. Formulation for the SGAP with Expected Values of Processing Times 59
3.1.3.2. Comparison between SGAP2 and Expected SGAP2 60

3.2. Branch-and-Price Method for the SGAP 62
3.2.1. Basic Formulation of the Branch-and-Price Method for the SGAP (BNP-SGAP) 62
3.2.2. Procedure for BNP-SGAP 64

3.3. BNP-SGAP Method with Dual Stabilization Technique (BNPDST-SGAP) 68
3.3.1. Implementation of BNPDST-SGAP 68
3.3.1.1. Investigation into Possible Poor Convergence of BNP-SGAP 68
3.3.1.2. Formulation for the BNP Method with Dual Stabilization Technique 70
3.3.2. A Warm-up Period Policy for the Determination of ε^k 73
3.3.3. A New Policy for Updating $\delta_j^{(-)k}$ and $\delta_j^{(+)k}$ and Removing $z_j^{(-)}$ and $z_j^{(+)}$ 76
3.3.4. Effectiveness of New Policy on the Performance of the Dual Stabilization Technique (DST) 78

3.4. BNPDST-SGAP with an Advanced Start (BNPDSTKP-SGAP) 80
3.4.1. A Heuristic Method for the Knapsack Problem 80
3.4.2. A Greedy Heuristic (GH) for the SGAP 81
3.4.3. An Example to Illustrate Procedure GH 83
3.4.4. Procedure BNPDSTKP-SGAP 84

3.5. BNPDSTKP-SGAP with the Monte Carlo Method (BNPDSTKP-SGAP with MCM) 86
3.5.1. Monte Carlo Method (MCM) for the SGAP 86
3.5.2. Statistical Properties of the MCM for the SGAP 88
3.5.2.1. Optimal Objective Function Value of Each Replication 89

3.5.2.2. A Lower Bound of the "True" Optimal Objective Function Value 95
3.5.2.3. An Upper Bound of the "True" Objective Function Value of $\hat{X}_N^m$ 95
3.5.2.4. Variance of $h_{\tilde{N}}^m\left(\hat{X}_N^m\right)$ 97
3.5.2.5. Candidate for "True" Optimal Solution 97
3.5.2.6. Unbiased Estimator of Optimality Gap 97
3.5.2.7. Unbiased Estimator of the Variance of Optimality Gap 98
3.5.2.8. Estimation of 100(1-α) Percent Confidence Interval 99
3.5.2.9. An Unbiased Estimator of the Ratio of Optimality Gap and Optimal Objective Function Value 100
3.5.3. An Exterior Sampling Method for the SGAP 100
3.5.4. Procedure for BNPDSTKP-SGAP with MCM 101
3.5.5. An Example to Illustrate Use of BNPDSTKP-SGAP with MCM 103
3.6. Computational Experimentation 105
3.6.1. Experimental Designs for BNC-SGAP, BNP-SGAP, BNPDST-SGAP, and BNPDSTKP-SGAP 106
3.6.2. Experimental Designs for BNPDSTKP-SGAP with MCM 107
3.6.2.1. Variation in the Values of N, Ñ, and M 109
3.6.2.2. Variation in $|I|$, $|J|$, e_i, and b_i 110
3.7. Results and Analysis 111
3.7.1. Experimental Results for the BNC-SGAP, BNP-SGAP, BNPDST-SGAP, and BNPDSTKP-SGAP Methods When $|J|=|I|$ and $e_1=10$ 111
3.7.2. Experimental Results for the BNC-SGAP, BNP-SGAP, BNPDST-SGAP, and BNPDSTKP-SGAP Methods When $|J|=|I|$ and $e_1=100$ 114
3.7.3. Experimental Results for the BNC-SGAP, BNP-SGAP, BNPDST-SGAP, and BNPDSTKP-SGAP Methods When $|J|=|I|$ and $e_1=1,000$ 117
3.7.4. Effect of Changes in the Values of Coefficients for the SGAP 120
3.7.5. Experimental Results for the BNC-SGAP, BNP-SGAP, BNPDST-SGAP, and BNPDSTKP-SGAP Methods When $|J|=1.5|I|$ and e1=100 123

3.7.6. Experimental Results for the BNC-SGAP, BNP-SGAP, BNPDST-SGAP, and BNPDSTKP-SGAP Methods When $|J|=2|I|$ and $e_1=100$ 126
3.7.7. Effect of Variation in the Number of Jobs per Machine 130
3.7.8. Results for the BNPDSTKP-SGAP with MCM Method 136
3.7.8.1. $|J|=|I|$ and $e_1=10$ 136
3.7.8.2. $|J|=|I|$ and $e_1=100$ 140
3.7.8.3. $|J|=|I|$ and $e_1=1{,}000$ 142
3.7.8.4. $|J|=1.5|I|$ and $e_1=100$ 144
3.7.8.5. $|J|=2|I|$ and $e_1=100$ 146
3.7.9. Results for the BNPDSTKP-SGAP with MCM Method with Variations in N, Ñ, and M 148
3.7.9.1. Variation in N 148
3.7.9.2. Variation in Ñ 151
3.7.9.3. Variation in M 152
3.7.9.4. An Observation on the Relationship between the Quality of Confidence Interval and Size of Feasible Region 155
3.7.10. Impact of Variance of Processing Time Distributions 156
3.8. Concluding Remarks 160

Chapter IV. A Branch-and-Price Method for the Hospital Staff Scheduling Problem (HSSP) 163

4.1. HSSP (General): Integration of Surgeons, Operations, Operating Rooms and Operation Hours 164
4.1.1. Formulation for the HSSP (General) 164
4.1.2. Size of the Formulation HSSP (General) 172
4.1.3. A Branch-and-Price Method for HSSP (General) 173
4.2. Priority in HSSP (General) 175
4.2.1. Incorporation of Priority into Model HSSP (General) 176

4.2.2. Differences between the HSSP (Priority) and Trainee Scheduling Problems 180
4.2.3. A Branch-and-Price Method for HSSP (Priority) 181

4.3. A Special Case of HSSP (Priority) 185
4.3.1. Completely Pre-arranged Schedules for Type I Surgeons 185
4.3.2. Formulation for the HSSP (Pre-arranged) 187
4.3.3. Examples of the HSSP (Pre-arranged) 189
4.3.4. Modification and Reduction of the HSSP (Pre-arranged) 191
4.3.5. A Branch-and-Price Method for the HSSP (Pre-arranged) 193

4.4. A Branch-and-Price Method for the HSSP Models 195
4.4.1. Branching Variable Selection Strategy 195
4.4.1.1. Setting Values of Variables to 0 or 1 195
4.4.1.2. A Numerical Example to Illustrate Proposition 4.6 204
4.4.1.3. Implementation of the New Branching Variable Selection Strategy 206
4.4.2. A Procedure to Find Initial Feasible Solutions at Each Node 208
4.4.3. A BNP Method for the HSSP Models 209

4.5. Computational Experimentation 212
4.5.1. Data Generation for the HSSP (Pre-arranged) 212
4.5.1.1. Data for Type I Surgeons 212
4.5.1.2. Data for Type II Surgeons 213
4.5.1.3. Values of Coefficients 214
4.5.2. Experimental Designs 214
4.5.2.1. BNC-HSSP (General) vs. BNP-HSSP (General) 216
4.5.2.2. BNC-HSSP (Priority) vs. BNP-HSSP (Priority) 216
4.5.2.3. BNC-HSSP (Pre-arranged) vs. BNP-HSSP (Pre-arranged) 216

4.6. Results and Analysis 217
4.6.1. Results for BNC-HSSP (General) vs. BNP-HSSP (General) 217
4.6.2. Results for BNC-HSSP (Priority) vs. BNP-HSSP (Priority) 220
4.6.3. Results for BNC-HSSP (Pre-arranged) vs. BNP-HSSP (Pre-arranged) 220

4.7. Concluding Remarks 225

Chapter V. Stochastic Hospital Staff Scheduling Problem (SHSSP) 226

5.1. Model Description 226

5.2. Formulation for the Stochastic Hospital Staff Scheduling Problem (General) 226

5.3. Classification of Recourse Model for SHSSP1 (General) 231

5.4. A Special Case of SHSSP (General) 231

5.4.1. Formulation for a Special Case of HSSP (General) 232

5.4.2. Classification of Recourse Model for SHSSP1 (Special) 234

5.4.3. An Example of the SHSSP (Special) 234

5.4.4. Another Formulation for the SHSSP (Special) 236

5.4.4.1. Model Description 236

5.4.4.2. Comparison of HSSP (Equivalent) and HSSP (General) 242

5.5. SHSSP (General) with Expected Values 243

5.5.1. Formulation for the SHSSP (General) with Expected Values of Operation Times .. 244

5.5.2. Comparison between SHSSP2 (General) and Expected-SHSSP2 (General) 246

5.6. SHSSP2 (General) with the Monte Carlo Method (SHSSP2-MCM (General)) 249

5.6.1. Monte Carlo Method (MCM) for the SHSSP (General) 249

5.6.2. Statistical Properties of the MCM for the SHSSP (General) 253

5.6.2.1. Optimal Objective Function Value of Each Replication 255

5.6.2.2. Lower and Upper Bounds for the “True” Optimal Objective Function Value, Estimates of the “True” Optimal Solution and Optimality Gap, and Confidence Interval on Optimality Gap for the SHSSP2 (General) 256

5.6.3. An Exterior Sampling Method for the SHSSP (General) 259

5.7. Computational Experimentation 260

5.7.1. Experimental Data to Justify the Use of the MCM 261

5.7.2. Experimental Data for the Experiments in the Second Group 264

5.8. Results and Analysis 265
5.8.1. Results on the Performances of the MCM and Optimum-Seeking Method 265
5.8.2. Results for Experiments in the Second Group 268
5.9. Concluding Remarks 270

Chapter VI. Branch-and-Price Method for a Stochastic Short-Term Personnel Planning Problem 271

6.1. Stochastic Short-Term Personnel Planning Problem (SSTPP) 272
6.1.1. Formulation for SSTPP 272
6.1.2. Stochastic Properties of the Recourse Function of SSTPP1 275
6.1.3. Size of Formulation SSTPP2 276
6.1.4. Stochastic Short Term Personnel Problem with Expected Values 277
6.1.4.1. Formulation for the SSTPP with Expected Values 277
6.1.4.2. An Example of Expected SSTPP2 Producing an Infeasible Solution to SSTPP2 278
6.2. Branch-and-Price Method for SSTPP2 283
6.2.1. Decomposition for SSTPP2 283
6.2.2. Dual Stabilization Technique (DST) for SSTPP2 285
6.2.2.1. Modified SSTPP2 with the DST 285
6.2.2.2. Use of the DST for BNPDST-SSTPP 287
6.2.3. Branching Variable and Node Selection Strategies for the BNPDST-SSTPP Method 288
6.2.3.1. Setting Variables to 0 or 1 288
6.2.3.2. Branching Variable Selection Strategy 290
6.2.3.3. Node Selection Strategy 291
6.2.4. A Greedy Heuristic for the SSTPP 293
6.3. Branch-and-Price Method for the SSTPP 296
6.4. Experimental Data 298

6.5. Results and Analysis 300

6.6. Concluding Remarks 302

Chapter VII. Concluding Remarks and Future Research 303

Appendix 308

A. Tables of Experimental Designs for BNC-SGAP, BNP-SGAP, BNPDST-SGAP, and BNPDSTKP-SGAP 308

B. Tables of Experimental Designs for BNPDSTKP-SGAP with MCM 314

C. Tables of Experimental Designs for HSSP Models 318

References 320

LIST OF TABLES

Table 2.1. Formulation for Example 2.1 with Expected Values, Scenario 1, and Scenario 2 ... 27

Table 2.2. Two Different General Recourse Models for Example 2.1 ... 31

Table 2.3. Application Areas of the MCM in the Literature ... 41

Table 2.4. Various Special Cases of the GAP ... 42

Table 2.5. Literature Review on Staff Scheduling Problem ... 45

Table 2.6. Comparison of Short-Term and Long-Term Personnel Planning Problems ... 51

Table 3.1. Comparison of SGAP2 with Expected SGAP2 ... 61

Table 3.2. Number of Variables with Zero Values at the Optimal Solution ... 69

Table 3.3. Effect of New Policy on the Performance of BNPDSTKP-SGAP ... 79

Table 3.4. Problem Sizes for Different $|I|$, $|J|$ and $|S|$ Values When $|R|=2$ ... 86

Table 3.5. Computational Times for Different Methods When $|J|=|I|$ and $e_1=10$ ... 112

Table 3.6. Comparative Results for the Four Methods When $|J|=|I|$ and $e_1=10$ ... 114

Table 3.7. Computational Times for Different Methods When $|J|=|I|$ and $e_1=100$ ... 115

Table 3.8. Comparative Results for the Four Methods when $|J|=|I|$ and $e_1=100$ ... 117

Table 3.9. Computational Times for Different Methods When $|J|=|I|$ and $e_1=1,000$ ... 118

Table 3.10. Comparative Results for the Four Methods When $|J|=|I|$ and $e_1=1,000$ ... 120

Table 3.11. Experimental Data ... 121

Table 3.12. Computational Times for Different Methods When $|J|=1.5|I|$ and $e_1=100$ ... 124

Table 3.13. Comparative Results for the Four Methods When $|J|=1.5|I|$ and $e_1=100$ ... 126

Table 3.14. Computational Times for Different Methods When $|J|=2|I|$ and $e_1=100$ ... 127

Table 3.15. Comparative Results for the Four Methods When $|J|=2|I|$ and $e_1=100$ ... 130

Table 3.16. Problem Size with Change in the Ratio of $|J|$ to $|I|$ ... 130

Table 3.17. Number of Experiments with the Best CPU Time by a Method ... 135

Table 3.18. Results for the BNPDSTKP-SGAP with MCM Method When $|J|=|I|$, $e_1=10$, and $\varepsilon=0.20$ ... 137

Table 3.19. Results for the BNPDSTKP-SGAP with MCM Method When $|J|=|I|$, $e_1=100$, and $\varepsilon=0.20$ ... 141

Table 3.20. Results for the BNPDSTKP-SGAP with MCM Method When $|J|=|I|$, $e_1=1000$, and $\varepsilon=0.20$ 143
Table 3.21. Results for the BNPDSTKP-SGAP with MCM Method When $|J|=1.5|I|$, $e_1=100$, and $\varepsilon=0.20$ 145
Table 3.22. Results for the BNPDSTKP-SGAP with MCM Method When $|J|=2|I|$, $e_1=100$, and $\varepsilon=0.20$ 147
Table 3.23. Best Objective Values with Variation in N 148
Table 3.24. Confidence Intervals on Optimality Gap with Variation in N 149
Table 3.25. Best Objective Values with Variation in $\tilde{N}$ 151
Table 3.26. Confidence Intervals on Optimality Gap with Variation in $\tilde{N}$ 151
Table 3.27. Results with Variation in Initial M 153
Table 3.28. Effect of Variation in b_1 on Confidence Interval 155
Table 3.29. Results for the BNPDSTKP-SGAP Method with Variation in the Value of Maximum Rate 157
Table 3.30. Number of Experiments with Best CPU Times among the Four Optimization Methods, and Results for the BNPDSTKP-SGAP with MCM Method 160
Table 3.31. BNC-SGAP vs. BNP-SGAP, BNPDST-SGAP, and BNPDSTKP-SGAP with respect to CPU Times 161

Table 4.1. Comparison of the formulations HSSP (Priority) and Trainee Scheduling Problem 180
Table 4.2. Results for BNC-HSSP (General) and BNP-HSSP (General) 218
Table 4.3. Results for BNC-HSSP (Priority) and BNP-HSSP (Priority) 221
Table 4.4. Results for BNC-HSSP (Pre-arranged) and BNP-HSSP (Pre-arranged) 223
Table 4.5. BNC vs. BNP Methods for HSSP Models 225

Table 5.1. Comparison of SHSSP2 (Special) and HSSP (Equivalent) in Size of Problems 242
Table 5.2. Comparison between SHSSP2 (General) and Expected-SHSSP2 (General) 247
Table 5.3. Problem Size of SHSSP2 (General) for Different $|I|$, $|J|$, $|M|$, and $|S|$ Values 251

Table 5.4. Experimental Data for the First Group 262
Table 5.5. Experiments for the Second Group 264
Table 5.6. Results for the Experimental Data Presented in Table 5.4 266
Table 5.7. Results for the Experimental Data Presented in Table 5.5 269

Table 6.1. Feasibility of a Solution to Expected-SSTPP2 282
Table 6.2. Experimental Data for the SSTPP 299
Table 6.3. Results for BNC-SSTPP and BNP-SSTPP 301

Table A.1. Experimental Designs for the SGAP with $|J|/|I|=1.0$ and $e_1=10$, and Changes in $|I|$, $|J|$, and b_1 309
Table A.2. Experimental Designs for the SGAP with $|J|/|I|=1.0$ and $e_1=100$, and Changes in $|I|$, $|J|$, and b_1 310
Table A.3. Experimental Designs for the SGAP with $|J|/|I|=1.0$ and $e_1=1{,}000$, and Changes in $|I|$, $|J|$, and b_1 311
Table A.4. Experimental Designs for the SGAP with $|J|/|I|=1.5$ and $e_1=100$, and Changes in $|I|$, $|J|$, and b_1 312
Table A.5. Experimental Designs for the SGAP with $|J|/|I|=2.0$ and $e_1=100$, and Changes in $|I|$, $|J|$, and b_1 313
Table B.1. Experimental Designs for BNPDSTKP-SGAP with MCM Considering ε=0.20 for the Experiments in Table A.1 315
Table B.2. Experimental Designs for BNPDSTKP-SGAP with MCM Considering ε=0.20 for the Experiments in Table A.2 315
Table B.3. Experimental Designs for BNPDSTKP-SGAP with MCM Considering ε=0.20 for the Experiments in Table A.3 316
Table B.4. Experimental Designs for BNPDSTKP-SGAP with MCM Considering ε=0.20 for the Experiments in Table A.4 316
Table B.5. Experimental Designs for BNPDSTKP-SGAP with MCM Considering ε=0.20 for the Experiments in Table A.5 317
Table C.1. Experimental Designs for Comparison between BNC-HSSP (General) and BNP-HSSP (General) 318

Table C.2. Experimental Designs for Comparison between BNC-HSSP (Priority) and BNP-HSSP (Priority)..... 318
Table C.3. Experimental Designs for Comparison between BNC-HSSP (Pre-arranged) and BNP-HSSP (Pre-arranged)..... 319

LIST OF FIGURES

Figure 3.1. Scenarios for the SGAP with 2 Machines, 3 Jobs, and 3 Speed Factors for Each Machine 57

Figure 3.2. Branch-and-Price Method for the SGAP (BNP-SGAP) 65

Figure 3.3. BNP-SGAP with the Dual Stabilization Technique (BNPDST-SGAP) 75

Figure 3.4. BNPDST-SGAP with a New Policy for Updating the Dual Stabilization Technique Parameters and Removing $z_j^{(-)}$ and $z_j^{(+)}$ Variables 77

Figure 3.5. BNPDST-SGAP with the Greedy Heuristic Procedure GH (BNPDSTKP-SGAP) 85

Figure 3.6. BNPDSTKP-SGAP with Monte Carlo Method (BNPDSTKP-SGAP with MCM) 102

Figure 3.7. Effect of Change in Machine Capacity When e_1=10 122

Figure 3.8. Effect of Change in Machine Capacity When e_1=100 122

Figure 3.9. Effect of Change in Machine Capacity When e_1=1,000 123

Figure 3.10. BNC-SGAP vs. BNPDSTKP-SGAP When $|J|=|I|$ 132

Figure 3.11. BNC-SGAP vs. BNPDSTKP-SGAP When $|J|=1.5|I|$ 133

Figure 3.12. BNC-SGAP vs. BNPDSTKP-SGAP When $|J|=2|I|$ 134

Figure 3.13. Values of Ratio= $\frac{\text{Best CPU Time Value Given in Column 7 of Table 3.16}}{\text{Best CPU Time Value Given in Column 14 of Table 3.16}}$ vs. ($|I|$, $|J|$, b_1) 139

Figure 3.14. Values of Ratio= $\frac{\text{Best CPU Time Value Given in Column 7 of Table 3.17}}{\text{Best CPU Time Value Given in Column 14 of Table 3.17}}$ vs ($|I|$, $|J|$, b_1) 140

Figure 3.15. Values of Ratio= $\frac{\text{Best CPU Time Value Given in Column 7 of Table 3.18}}{\text{Best CPU Time Value Given in Column 14 of Table 3.18}}$ vs. ($|I|$, $|J|$, b_1) 142

Figure 3.16. Values of Ratio= $\frac{\text{Best CPU Time Value Given in Column 7 of Table 3.19}}{\text{Best CPU Time Value Given in Column 14 of Table 3.19}}$ vs. ($|I|$, $|J|$, b_1) 144

Figure 3.17. Values of Ratio= $\frac{\text{Best CPU Time Value Given in Column 7 of Table 3.20}}{\text{Best CPU Time Value Given in Column 14 of Table 3.20}}$ vs. ($|I|$, $|J|$, b_1) 146
Figure 3.18. Confidence Interval of Optimality Gap with Changes in N 150
Figure 3.19. Confidence Interval of Optimality Gap with Changes in Ñ 152
Figure 3.20. Confidence Interval of Optimality Gap with Changes in M 154

Figure 4.1. An Example of Feasible Schedules for Type I Surgeons 175
Figure 4.2. An Example of Type I Surgeons' Schedule 185
Figure 4.3. 2 Type I Surgeons, 2 Type II Surgeons, 8 Operations, and 4 Operating Rooms .. 189
Figure 4.4. 2 Type I Surgeons, 2 Type II Surgeons, 12 Operations, and 4 Operating Rooms 190
Figure 4.5. 2 Type I Surgeons, 2 Type II Surgeons, 16 Operations, and 5 Operating Rooms 190
Figure 4.6. An Example of Branching on y_{ijmt} 203
Figure 4.7. A Numerical Example for Branching Variable Selection Strategy 204
Figure 4.8. Branch-and-Price Method for HSSP Models 210

Figure 5.1. Schedule of Type I Surgeons' Operations 235
Figure 5.2. Schedule of Type I and Type II Surgeons' Operations 235
Figure 5.3. Full Schedule Considering Stochastic Operation Times 235
Figure 5.4. $\bar{s}_{ij}$ for Example in Figure 5.3 238
Figure 5.5. Value of Ratio= $\frac{\text{CPU Time Value Given in Column 3 of Table 5.6}}{\text{CPU Time Value Given in Column 9 of Table 5.6}}$ vs. ($|M|$, $|I|$, $|J|$)268

Figure 6.1. BNP Method for the SSTPP with Dual Stabilization Technique, Greedy Heuristic, and Branching Variable and Node Selection Strategies 297
Figure 6.2. Ratios of CPU Times for BNC-SSTPP and BNP-SSTPP 302

Chapter I. Introduction

The main objective of this research is to explore the use of the branch-and-price (BNP) method for the solution of the following four problems: Stochastic Generalized Assignment Problem (SGAP), Hospital Staff Scheduling Problem (HSSP), Stochastic Hospital Staff Scheduling Problem (SHSSP), and Stochastic Short-Term Personnel Planning Problem (Stochastic STPP). Compared to the branch-and-cut (BNC) method, which is a popular method for the solution of mixed integer programming (MIP) problems, the BNP method is relatively new, and can be quite effective depending upon the inherent structure of the problem on hand.

We can view the BNP method as a combination of the column generation method (CGM) and the branch-and-bound (BNB) method. While the BNB method is used for branching on fractional variables, the CGM is employed to solve a relaxed problem at each node. Consequently, the BNP method could be regarded as a generalized type of the CGM, also known as the Dantzig-Wolfe decomposition method, since it could be applied to the MIP problems as well as to continuous linear programming (LP) problems. The CGM consists of a master problem and one or multiple pricing problems, also called subproblems. In the CGM, information is transferred back and forth between the master problem and the pricing problems until no additional valid columns are generated by the pricing problems.

The Generalized Assignment Problem (GAP) is a well-known problem that arises in a wide variety of machine-job (or agent-job) assignment type of situations belonging to:

- Scheduling problems (e.g., crew scheduling problem),
- Multiple traveling salesman problem,
- Vehicle routing problem,
- Set partitioning problem, and
- Set covering problem.

The stochastic generalized assignment problem (SGAP) is a variation of the GAP in which the time required to process a job on a machine is stochastic. Even though the SGAP could be

encountered in the application areas mentioned above, due to the stochasticity introduced, it would require a lot more computational effort for its solution than that required for the GAP. Nonetheless, the inclusion of stochasticity adds an important and realistic feature to the GAP.

Interestingly, even though a considerable amount of effort has been devoted to the solution of the GAP, there has been little effort reported in the literature for the solution of the SGAP. Three studies have been reported with regard to the SGAP, and except for one paper (Albareda-Sambola et al. [2]), which presents three exact algorithms for the SGAP based on branch-and-cut algorithms, the other two studies (Fukushima et al., 1983 and Albareda-Sambola et al. [1]) have proposed heuristic procedures for the SGAP. Albareda-Sambola et al. [2] have assumed that an event happens with a Bernoulli probability distribution function (p.d.f.), i.e., each event will happen or may not happen with the probability of p and (1-p), respectively. In this study, we relax this assumption and consider a general, discrete p.d.f. for processing times. We also consider the possibility of a machine requiring more resources (times) than available via the use of extra work hours or extra amount of resources through a penalty function. This feature does not normally arise in deterministic circumstances.

Our study contributes not only to introducing a two-stage, complete, fixed recourse model for the SGAP, which can be applied to any kind of discrete probability distribution function for processing times, but also, to implementing the BNP method to the SGAP for the first time as an exact algorithm that combines the BNP method with a dual stabilization technique. We have also added features that improve the performance of the BNP as it is applied for the solution of the SGAP. A stochastic program generally involves a large number of scenarios in its formulation that limits its application to small-size problems. To overcome this shortcoming, we integrate the use of the Monte Carlo method within the BNP method, which allows the use of only a small set of scenarios. Our computational investigation shows a large amount of savings in cpu time that can be achieved by this integrated method while obtaining almost optimal solutions.

The second problem that we address is the hospital staff scheduling problem (HSSP). The health care industry is one of the fastest growing industries in the world. It is also an industry that requires implementation of effective methods in its day-to-day operations in order to minimize the cost incurred or to maximize net revenue (from the perspective of hospitals). To that end, we address a staff scheduling problem that is commonly encountered in a hospital. In

particular, we consider the problem of scheduling surgeons to hospital facilities (rooms) to perform operations such that the revenue generated by the hospital is maximized. We consider two types of surgeons, namely, Type I and Type II, with Type I surgeons having priority over Type II surgeons in scheduling their operations. Given the inherent structure of the problem, we are able to exploit it to effectively implement the BNP method for its solution. To that end, we have addressed the following three HSSPs: (i) HSSP (General): No priority is considered between surgeons; (ii) HSSP (Priority): Type I surgeons are given priority over Type II surgeons for scheduling; and (iii) HSSP (Pre-arranged): The priority between surgeons is considered, and the schedule for all Type I surgeons is completely predetermined.

The third problem investigated in this study is a stochastic version of the HSSP, in which we consider the operation times to be stochastic. We allow overlapping of schedules for surgeons and operating rooms, and also, allow for an assignment of a surgeon to perform an operation that takes less than a pre-arranged operation time, but all incurring appropriate penalty costs. In order to justify the use of the recourse model, we compare its solution with that obtained using expected values. The Monte Carlo method (MCM) has been used in order to reduce the complexity of the stochastic program (SP) by considering fewer number of scenarios. We employ the branch-and-cut (BNC) method in concert with the MCM to solve this problem. The solution obtained is evaluated using the tolearance ratio, the confidence interval, and cpu time required.

We have also considered a special case of this stochastic HSSP, which permits no overlapping schedule for any surgeon and any operating room, and assigns the same operation time for each assignment under each scenario. In this regard, we consider another formulation that relies on the longest operation times among all scenarios for each assignment of a surgeon to an operation in order to avoid scheduling conflicts. Subsequently, we develop a deterministic HSSP which we show to be identical in complexity to that of the formulation for HSSP (General).

The fourth problem that we address is the stochastic short-term personnel planning problem (SSTPP). This problem arises due to the need for hiring subcontractors (workers) to perform short-term work (jobs). We also assume the time required to perform a job by a subcontractor to be stochastic, which is a realistic feature given the temporary nature of the job. Such a problem has not been addressed in the literature. The BNP method is devoted for the

solution of this problem in concert with features such as the dual stabilization technique, new strategies for branching and node selection, and a greedy heuristic to obtain an advanced start. This problem lacks relatively complete recourse. Hence, the MCM cannot be incorporated in the BNP for this problem.

Contributions of This Research

- Development of the BNP method as an exact algorithm for the solution of the SGAP, the HSSP, the SHSSP, and the SSTPP.
- Introduction of a two-stage, complete, fixed recourse model for the SGAP.
- Introduction of a two-stage, relatively complete recourse model for the SHSSP.
- A new dual stabilization technique for the SGAP and the SSTPP.
- Use of an advanced start for the SGAP and the SSTPP to find an initial basic feasible solution.
- Use of the Monte Carlo method (MCM) in concert with the BNP and BNC methods for the solution of the SGAP and the SHSSP, respectively.
- Development of necessary convergence results for the MCM regarding the use of the SGAP and the SHSSP.

Remainder of the Dissertation

We have presented our work in four chapters. In Chapter II, we provide background information and review of literature on the BNP method, the SGAP, the HSSP, and the Stochastic STPP. Our work on the SGAP is presented in Chapter III. Chapter IV addresses the HSSP. Chapters V and VI are devoted to the SHSSP and the Stochastic STPP, respectively. Each of these chapters contains details of the model formulations, various algorithms that are developed for the solutions of the corresponding problems, analytical results, and the results of the experimental investigations conducted. Finally, Chapter VII contains concluding remarks and ideas for future research.

Chapter II. Background Information and Literature Review

In this chapter, we formally introduce the generalized assignment problem (GAP). We also present the methods of branch-and-cut (BNC) and branch-and-price (BNP), as well as the BNP method through its application to the GAP. Then, the stochastic programming problem is introduced along with the methods of its solution including the Monte Carlo Method (MCM). This is followed by presentation of the operating room and staff scheduling problem encountered in a hospital for both deterministic and stochastic ones. Finally, we introduce the personnel planning problem. We also provide literature review pertinent to each of these topics.

2.1. Generalized Assignment Problem (GAP)

We use the following notation.

Parameters

i : Machine (resource) i, $\forall\, i \in I$, where I is the set of machines

j : Job j, $\forall\, i \in J$, where J is the set of jobs

c_{ij} : Cost of assigning job j to machine i, $\forall\, i \in I$, $\forall\, j \in J$

d_{ij} : Amount of resource needed by machine i to process job j, $\forall\, i \in I$, $\forall\, j \in J$

b_i : Resource capacity of machine i, $\forall\, i \in I$

Variables

x_{ij} : =1, if job j is assigned to machine i, otherwise, =0, $\forall\, i \in I$, $\forall\, j \in J$

GAP

Minimize $\sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij}$ (2.1a)

Subject to $\sum_{i \in I} x_{ij} = 1 \qquad \forall j \in J$ (2.1b)

$\sum_{j \in J} d_{ij} x_{ij} \leq b_i \qquad \forall i \in I$ (2.1c)

$x_{ij} \in \{0,1\} \qquad \forall i \in I, \forall j \in J$ (2.1d)

The objective is to minimize the cost of processing the given set of jobs (J) on the machines. Constraint set (2.1b) captures the fact that a job is assigned to only one machine, while constraint set (2.1c) imposes available machine capacities. Constraint set (2.1d) enforces the binary nature of the variables.

2.2. Branch-and-Cut and Branch-and-Price Methods

In this section, we briefly introduce the branch-and-cut (BNC) and branch-and-price (BNP) methods used for the solution of the mixed integer programming problems (MIPs). We also address the column generation method (CGM), relationship between the column generation and BNP methods, dual stabilization technique for the BNP method, properties of branching variable selection strategies for the BNP method, and finally, some application areas of the BNP method.

2.2.1. Branch-and-Cut Method

The BNC method is essentially a cutting plane method in which an appropriate cut is introduced to remove infeasible, fractional solutions at every node of the underlying branch-and-bound tree. Such cuts are added at a node until a feasible solution is obtained or no more valid cuts exist, in which case the node is branched from. The BNC method is generally effective for the MIPs involving a large number of constraints. The following steps are implemented at each node of the BNC method.

Step 1. Solve the relaxed problem at the node.

Step 2. Evaluate the node.

a. In case the solution of the problem at the node, before it is relaxed, is feasible and its objective function value is better than that of the current incumbent solution, then it becomes the new incumbent solution, and the node is pruned.

b. If the solution is infeasible and its objective function value is not better than that of the current incumbent solution, then the node is pruned.

c. If the solution is infeasible and its objective function value is better than that of the current incumbent solution, a valid cut is determined, which would direct the solution towards feasibility. If a valid cut is found, then go to step 3; otherwise, if no valid cut exists, then go to step 4.

Step 3. Add the cut to the problem, and go to step 1

Step 4. Branch the current node into its children nodes.

The schemes for choosing the next node, branching from a node, and pruning of a node basically follow those of the branch-and-bound method. The effectiveness of the BNC method basically depends on the quality of valid cuts used, which remove infeasible solution space while maintaining the original feasible solution space intact. There are various types of BNC algorithms (see Albareda-Sambola et al. [2]), and the following criteria can be used to determine the effectiveness of a BNC algorithm:

- The number of constraints added to the formulation.
- The computational effort required in terms of cpu times.
- The number of nodes required until an optimal solution is obtained.
- The complexity of the problem that a BNC algorithm can solve effectively.

2.2.2. Branch-and-Price Method

The branch-and-price (BNP) method, on the other hand, is a combination of the branch-and-bound (BNB) and column generation methods. Since the column generation method (CGM) adds columns (vertices) to the relaxed problem in a systematic manner, it is applicable to a problem involving a large number of columns since we end up using only a relatively small number of these columns. The relaxed problem consisting of a subset of columns is called the restricted master problem, and the columns are generated using a pricing problem, also called the subproblem.

Although the CGM has been successfully applied to various problems, we cannot arbitrarily apply it to a MIP because the duality theory does not hold true for the constraints that include integer variables, i.e., the dual variable values corresponding to the constraints containing integer variables might not be valid. However, it can be employed in conjunction with the BNB method for the solution of a MIP by relaxing the master problem, and also, by implementing a branching scheme until all the nodes are pruned. In the CGM, the master problem consists of continuous variables, and the pricing problem is of less complexity than that of the original MIP. The benefits of the CGM are as follows:

- It results in tighter bounds than those obtained by the LP relaxation of the original formulation (at least as tight).
- It effectively handles the "Integrality Property." A MIP is said to possess this property if the optimal solution of its relaxed version guarantees integrality. As a result, its objective function value is the same as that of the relaxed MIP. Generally, this property restricts us from obtaining good-quality bounds at each node of the underlying BNB tree. However, the CGM is effective in circumventing this situation by retaining in the master problem the constraints that cause this to occur. Consequently, the pricing problem enables generation of an effective column.

Some fundamental ideas of the branch-and-price method are summarized in Vanderbeck et al. [90]. These include generation of a set of columns, convexification, discretization, generalization

of a MIP with discretization of integer variables and convexification of continuous variables, and restriction on variables in the pricing problem.

Many of these issues need to be appropriately addressed before applying the branch-and-price method. The most important of these issues is the decision on which constraints to keep in the master problem and which constraints to transfer to the pricing problem(s). This is critical because it directly impacts the tightness of the bounds obtained. We illustrate this by using the GAP.

There are two types of column generation models for the formulation **GAP**. In the first model, designated as **GAPCG1**, the job assignment constraints are kept in the master problem while the machine capacity constraints are relegated to the pricing problem. To illustrate this, we use the following additional notation. Let

λ_i^k : k^{th} convexity variable for machine i, $k = 1,...,k_i$,

x_{ij}^k : k^{th} column of x_{ij} transferred from **Pricing Problem** i, $\forall i \in I$,

$\overline{\pi}_j$: Dual price for constraint *j*, $\forall j \in J$, of constraint set (2.2b) (below), and

$\overline{\pi}_i$: Dual price for constraint *i*, $\forall i \in I$, of convexity constraint set (2.2c) (below).

We have:

GAPCG1

Master Problem

Minimize $\sum_{i \in I} \sum_{j \in J} \sum_{k=1}^{K_i} (c_{ij} x_{ij}^k) \lambda_i^k$ (2.2a)

Subject to $\sum_{i \in I} \sum_{k=1}^{K_i} (x_{ij}^k) \lambda_i^k = 1 \qquad \forall j \in J \qquad (\overline{\pi}_j)$ (2.2b)

$\sum_{k=1}^{K_i} \lambda_i^k \le 1 \qquad \forall i \in I, \qquad (\overline{\pi}_i)$ (2.2c)

$\lambda_i^k \in \{0,1\} \qquad \forall i \in I, \; k = 1,...,K_i$ (2.2d)

Pricing Problem i, $\forall i \in I$

Minimize $-\bar{\pi}_i + \sum_{j \in J} (c_{ij} - \bar{\pi}_j) x_{ij}$ (2.2e)

Subject to $\sum_{j \in J} d_{ij} x_{ij} \le b_i$ (2.2f)

$x_{ij} \in \{0,1\}$ $\forall j \in J$ (2.2g)

In the second model, designated as **GAPCG2**, the machine capacity constraints stay in the master problem while the job assignment constraints become a part of the pricing problem. Let

$\bar{\pi}_i$: Dual prices for constraint *i*, $\forall i \in I$, of constraint set (2.3b) (below), and

$\bar{\pi}_j$: Dual prices for convexity constraint *j*, $\forall j \in J$, of constraint set (2.3c) (below).

We have:

GAPCG2

Master Problem

Minimize $\sum_{i \in I} \sum_{j \in J} \sum_{k=1}^{K_j} (c_{ij} x_{ij}^k) \lambda_j^k$ (2.3a)

Subject to $\sum_{j \in J} \sum_{k=1}^{K_j} (d_{ij} x_{ij}^k) \lambda_j^k \le b_i$ $\forall i \in I$ $(\bar{\pi}_i)$ (2.3b)

$\sum_{k=1}^{K_j} \lambda_j^k = 1$ $\forall j \in J$ $(\bar{\pi}_j)$ (2.3c)

$\lambda_j^k \in \{0,1\}$ $\forall j \in J$, $k = 1,...,K_j$ (2.3d)

Pricing Problem j, $\forall j \in J$

Minimize $-\bar{\pi}_j + \sum_{i \in I} (c_{ij} - \bar{\pi}_i d_{ij}) x_{ij}$ (2.3e)

Subject to $\sum_{i \in I} x_{ij} = 1$ (2.3f)

$$x_{ij} \in \{0,1\} \qquad \forall i \in I \qquad \text{(2.3g)}$$

Constraint sets (2.2b) and (2.3b) are called linking constraints, and constraint sets (2.2c) and (2.3c) are convexity constraints for their respective master problems. Note that the constraint set (2.2c) consists of inequalities since all the machines do not have to be busy, i.e., some of the machines could be idle as long as all the jobs are assigned. An equation per machine will result if every machine were to process at least a job. On the other hand, the convexity constraint set (2.3c) consists of equations since every job should be assigned to one and only one machine. In general, **GAPCG1**, in which the number of pricing problems is equal to the number of machines (=$|I|$), is preferred over **GAPCG2**. This is because the presence of machine capacity constraints in the pricing problems for **GAPCG1** tends to generate more valuable columns, which lead to tighter bounds than those generated by the pricing problems for **GAPCG2** consisting of job assignment constraints. The job assignment constraints for the pricing problems of **GAPCG2** constitute a unitary matrix, with all entities being zeroes or ones, which give rise to the determinants of all of its sub-matrices to be zero or one. The resulting Integrality Property guarantees that the polyhedron of **GAPCG2**, **conv(Pricing Problem** j**)**, has integral extreme points. As a result, **GAPCG2** does not give tight bounds, just like those obtained for the relaxation of the original problem.

On the other hand, if the pricing problem includes the machine capacity constraints as in **GAPCG1**, each pricing problem is a knapsack problem. The fractional solutions that will be feasible to the corresponding master problem are convex combinations of the 0-1 solutions of the knapsack problem. In other words, some fractional solutions feasible to the relaxation of the original problem might not be feasible to the master problem. This property is likely to make the bounds of **GAPCG2** tighter. Consequently, since the objective function is minimization, the lower bounds for **GAPCG1** will be greater than or at least equal to those for **GAPCG2**. Nonetheless, it is possible that the number of vertices dealt with in the master problem is so large that, overall, it takes a lot longer to solve knapsack problems than to solve the relaxation of the restricted master problem since those vertices are generated by the pricing problems. Our primary objective at each node of the column generation method is not to obtain solutions as fast as possible, but to obtain better bounds.

Both branch-and-cut and branch-and-price methods take advantage of the branch-and-bound method. The deeper a node is situated in its branch-and-bound tree, the greater the number of variables that we can fix to a value or a range of values. The branch-and-price method, however, can deal with multiple pricing problems, which could be useful to reduce the complexity of the problem. For example, in the GAP, if we apply the column generation method where the number of pricing problems is equal to the number of machines, the most difficult problem that we need to deal with is a knapsack problem with one constraint regardless of the number of jobs and the number of machines. Thus, the complexity of solving the GAP reduces to that of solving the knapsack problem. Other topics related to the branch and price method include:

- Branch-and-bound scheme for the master problem,
- Lower bounds for early termination of column generation,
- Generation of columns for 0-1 integer programming (IP) problems,
- Generation of an initial solution,
- Column pool management schemes,
- Trade-offs between the computation efforts required using strong bounds and weak bounds
- Depth of branch-and-bound tree considering the tailing-off effect
- Ways of obtaining valuable dual solutions
- Reformulation of master problem to decrease the number of columns generated
- Primal heuristics,
- Branching strategy,
- Mixture of column generation and row generation, and
- Implementation of branch-and-price with CPLEX, OSL, and MINTO.

For more detail on the above issues, see Vanderbeck et al. [91].

2.2.3. General Column Generation Method

In the column generation method, the master problem consists of convexity variables. As long as the original problem is a continuous linear programming problem, their coefficients correspond to the convex combinations of vertices feasible to the original problem as well as the pricing problems used to generate those vertices. The dual solution acquired from the master problem is used to construct the objective function of the pricing problem. If the pricing problem finds a column with a negative reduced cost, then that column is added to the master problem. This procedure leads to an optimal solution, if it exists, under the condition that all variables are continuous. This, however, is not likely to happen for a MIP for the following two reasons:

- The dual solution for the constraints containing integer variables is not well understood. The concept of duality for such situations has been recognized, but it fails to produce exact dual solutions for the pricing problem.
- The convex combination of vertices in the master problem might not warrant the integrality of variables. Even if the pricing problem introduces an integer solution to the master problem, the resulting solution could be fractional. The integrality, as a result, is not guaranteed by the column generation method alone.

The column generation method-based algorithm can be outlined as follows (Barnhart et al. [9]):

Step 1. Determine an initial feasible solution.

Step 2. Solve the restricted master problem after it is updated with the columns introduced at the preceding step. Obtain the dual values corresponding to all the constraints including both the linking constraints and convexity constraint(s), and transfer them to the corresponding pricing problem, and go to step 3.

Step 3. Solve each pricing problem. If one or more columns with negative reduced costs (for the minimization problem) are found, pass them to the restricted master problem, and go to step 2; otherwise, stop.

We start with one or more initial feasible solutions, but their determination is not that easy. A heuristic method or the two-phase method is normally used to that end. The former can be convenient if we are able to find a collection of feasible solutions, which would lead to a desired restricted master problem in step 2. However, the number of iterations required to find such columns may be highly variable. On the other hand, the latter usually provides only one feasible solution. Moreover, the quality of this feasible solution might not be good. Notwithstanding these limitations, the two-phase method guarantees a solution if the problem is feasible. However, we can also implement a combination of these methods, that is, the application of a heuristic method first followed by the two-phase method.

Determination of the number of columns to be maintained in the restricted master problem is also an important issue. It is possible that having an excessive number of columns could merely contribute to increasing the computational effort required as the convexity variables corresponding to many of these columns could be nonbasic in the optimal solution. Then, the question is if all nonbasic columns exhibiting nonnegative reduced costs in the current restricted master problem (RMP) should be deleted. However, after deletion from the RMP, it is possible that they are reintroduced in future pricing problems. Therefore, additional controls are required to prevent this from happening. One of the ways to implement this control is the 'column pool management,' as follows:

Step 1. Create a column pool which contains all the columns generated from pricing problems that have not been added yet,

Step 2. Update and solve the restricted master problem by adding columns from this pool,

Step 3. Delete the columns associated with the nonbasic convexity variables,

Step 4. If there are columns with negative reduced costs in the column pool, then select a subset of these columns, and go to step 2; and

Step 5. If all the columns in the column pool have nonnegative reduced costs or the column pool is empty, stop; and pass the dual information to the pricing problem.

In step 1, there are two ways of determining columns that should be included in the column pool:

- Columns generated from all the previous iterations
- Columns generated from the past few iterations, the number of which is predetermined.

The second method is based on the assumption that the greater the distance of the iteration at which a column is generated from the current iteration, the less correlated that column might be to the optimal solution. Furthermore, in step 3, we can enhance the variety of the column pool by allowing a predetermined portion of the columns to return to the column pool instead of removing all of them. Pigatti et al. [76] show that addition of all the columns from earlier iterations makes the convergence of the algorithm faster. They also show that the strategy of keeping all the columns is better for the GAP than that of deleting half of the columns corresponding to nonbasic convexity variables.

Since the relaxed master problem is solved for the convexity variables, we can apply the Simplex method for its solution. The pricing problem, however, mostly remains a MIP. Even though its size could be much smaller than that of the original MIP, it is probable that solving the pricing problem repeatedly is the biggest burden of the algorithm. Finding an efficient way to solve the pricing problem is one of the greatest challenges that need to be resolved.

By and large, a pricing problem contains a special structure for which an efficient algorithm can be used. The pricing problem in **GAPCG1** is a knapsack problem. Also, note that, it is not necessary to find the column with the most negative reduced cost in the pricing problem. As long as it has a negative reduced cost, it is valid for the master problem. Therefore, we can utilize an approximation method to solve the pricing problem.

2.2.4. Relationship between the Column Generation and Branch-and-Price Methods

Vanderbeck et al. [90] list the following three perspectives on the column generation method (CGM):

- First, due to the presence of convexity variables in the master problem, it is considered to be a technique of exploring an LP formulation expressed by the Lagrangian duals.
- Second, we can transform a MIP into another formulation with integral convexity variables. Imposition of integrality restrictions on the CGM makes it a more challenging problem. However, it is possible to build a solution framework to deal with integer variables. The BNP method is one of these solution frameworks, which utilizes suitable branching-variable and node-selection strategies. It is also possible to augment valid cuts as in the branch-and-cut method.
- Third, the CGM could be considered in conjunction with reformulating the pricing problem to generate tighter bounds. Thus, the convexity variables can be thought of as the weights affiliated with the solutions or the extreme rays transferred from the pricing problem.

The importance of correctly selecting complicating constraints and linking constraints cannot be overemphasized in the column generation method. The constraints in the original formulation are separated into the master problem and the pricing problem. The linking constraints can ideally have a block diagonal structure. They are much easier to handle when they are alone than in the presence of additional constraints.

With regard to the GAP, whether we choose the job assignment constraints or the machine capacity constraints as the linking constraints, the resulting pricing problems in both **GAPCG1** and **GAPCG2** are readily solvable. In fact, **GAPCG2** is easier to solve than **GAPCG1** since the pricing problem for **GAPCG1** is a knapsack problem with one constraint while that for **GAPCG2** is an assignment problem. However, pseudo polynomial-time

algorithms have been developed for the knapsack problem with single constraint in Martello et al. [64], Pisinger ([76] and [77]), and Vazirani [92].

Another reason for choosing **GAPCG1** over **GAPCG2** is the generation of better bounds (columns) from the pricing problem. For the case of **GAPCG2**, the pricing problem possesses Integrality Property, and hence, can be solved by a linear programming algorithm. However, the disadvantage is that the master problem produces no better bounds than those resulting from the relaxation of the original problem, which contributes to excessive computational effort and overall run time. If we choose **GAPCG1**, on the other hand, the polytope of the relaxed pricing problem embodies the convex hull, **conv(Pricing problem i)**. Some of the feasible integer solutions are not extreme points of this polytope. The Integrality Property does not hold true for this case and the bounds generated from the CGM are at least as tight as those of the relaxation of the original problem, often even tighter.

Geoffrion [38] has shown this relationship between Integrality Property and the degree of tightness of bounds obtained using Lagrangian relaxation. The Lagrangian dual problem is the dual of the master problem. Therefore, the effect of Integrality Property also can be applied to the CGM. In other words, the CGM generates a better bound, which is at least as tight as or tighter than that of the relaxation of the original problem (see Wilhelm [97] for more details).

In order to decompose a problem involving some degree of complexity, we need to test a variety of models constituting different master problems and pricing problems in order to determine the best configuration with each pricing problem belonging to an easily solvable optimization problem. Vanderbeck et al. [90] give an example of multiple structures for capacitated multi-item, lot-sizing problem where we can derive either a knapsack pricing problem or a single-item lot-sizing pricing problem as its pricing problem depending on which constraints are chosen as the complicating constraints and the linking constraints.

Vanderbeck et al. [90] also refer to the difficulty in the convergence of dual values. They describe this as one of the main challenges encountered in implementing a proper column generation method. As a result, a good performance of a branch-and-price algorithm is not always guaranteed. The performance of the branch-and-price-based methodology can be negatively impacted by the following reasons:

- Poor initial performance of dual variable values (heading-in effect)
- Delayed convergence near an optimal solution (tailing-off effect)
- Unstable dual bounds hopping around extreme points (bang-bang effect)
- Degeneracy, where the objective value of master problem stays the same (plateau effect)
- Dual values that continuously increase (or decrease), i.e., those that do not converge monotonically (yo-yo effect)
- Difficulty in solving the pricing problem

Some techniques presented in the literature to make the branch-and-price method a more efficient one are as follow:

- A warm start (Vanderbeck et al. [91]), which pertains to the supplementary information added at the preprocessing step. This can be applied to the CGM to improve its efficiency. It helps the algorithm in approaching the optimal solution quickly. Valuable columns or tighter gaps between lower and upper bounds by developing the algorithm with the relaxed pricing problem could constitute the warm start for the CGM.
- A primal-dual heuristic for the pricing problem (Vanderbeck et al. [91]) relaxes the pricing problem to get dual bounds and provides a heuristic properly designed to obtain near-optimal primal and dual solutions. This reduces the computational time required to solve the pricing problem.
- One of the most popular techniques to resolve the difficulties mentioned above is the dual stabilization technique. Vanderbeck et al. [91] also present this technique to improve the branch-and-price method.

2.2.5. Dual Stabilization Technique for the Branch-and-Price Method

As we alluded to earlier, one of the biggest challenges of a branch-and-price-based methodology is the plateau effect, i.e., cycling among the same bases. This might be caused by degeneracy, where one or more basic variables take on a value of zero and prevent improvement in the objection function value. Hence, this effect needs to be avoided. An avoidance of this effect could possibly prevent occurrences of some of the other problems as well, such as heading-in effect by skipping useless dual variable values at the early stages of the algorithm. The bang-bang effect can be mitigated by penalizing deviation between the dual variable values and a predetermined reference value.

The dual stabilization technique is used in concert with the CGM to alleviate the bang-bang and plateau effects. This technique bounds the values of dual variables by looking for dual solutions constrained by penalties on their variance. Pigatti et al. [76] present a dual stabilization technique for the GAP, and Westerland et al. [96] apply a dual stabilization technique to a single vehicle routing-allocation problem (SVRAP).

A typical way of avoiding degeneracy is to use the perturbation method where bounded surplus and slack variables are added to each constraint. Gill et al. [39] present the exact penalty method to constrain the ranges of dual variable values. This method includes penalties for the surplus and slack variables in the objective function. Next, we present two examples that illustrate application of the perturbation method and the exact penalty method to the GAP. The formulations **GAP-Perturbation** and **GAP-Exact Penalty** given below exemplify the GAP model adjusted to the perturbation method and the exact penalty method, respectively. Note that the dual prices $\bar{\pi}_i$ and $\bar{\pi}_j$ are obtained after relaxing the binary variables, x_{ij}, in **GAP-Exact Penalty**.

Consider the following notation.

$y_j^-\left(y_j^+\right)$: Slack (surplus) variable for the job assignment constraint, $\forall j \in J$

$w_j^-\left(w_j^+\right)$: Slack (surplus) variable for the machine capacity constraints, $\forall j \in J$

$\varepsilon_j^-\left(\varepsilon_j^+\right)$: Parameter to restrain $y_j^-\left(y_j^+\right)$, $\forall j \in J$

$\delta_j^-\left(\delta_j^+\right)$: Parameter to restrain $w_j^-\left(w_j^+\right)$, $\forall j \in J$

GAP-Perturbation

Minimize $\sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij}$ (2.4a)

Subject to $\sum_{i\in I} x_{ij} - y_j^- + y_j^+ = 1 \qquad \forall j \in J$ (2.4b)

$\sum_{j\in J} d_{ij}x_{ij} - w_i^- + w_i^+ \le b_i \qquad \forall i \in I$ (2.4c)

$y_j^- \le \varepsilon_j^- \qquad \forall j \in J$ (2.4d)

$y_j^+ \le \varepsilon_j^+ \qquad \forall j \in J$ (2.4e)

$w_i^- \le \delta_i^- \qquad \forall i \in I$ (2.4f)

$w_i^+ \le \delta_i^+ \qquad \forall i \in I$ (2.4g)

$x_{ij} \in \{0,1\} \qquad \forall i \in I\ ,\ \forall j \in J$ (2.4h)

$y_j^-\ ,\ y_j^+ \ge 0 \qquad \forall j \in J$ (2.4i)

$w_i^-\ ,\ w_i^+ \ge 0 \qquad \forall i \in I$ (2.4j)

GAP-Exact Penalty

Minimize $\sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} + \sum_{j\in J}(\varepsilon_j^- y_j^- + \varepsilon_j^+ y_j^+) + \sum_{i\in I}(\delta_i^- w_i^- + \delta_i^+ w_i^+)$ (2.5a)

Subject to $\sum_{i\in I} x_{ij} - y_j^- + y_j^+ = 1 \qquad \forall j \in J \qquad (\bar{\pi}_j)$ (2.5b)

$\sum_{j\in J} d_{ij}x_{ij} - w_i^- + w_i^+ \le b_i \qquad \forall i \in I \qquad (\bar{\pi}_i)$ (2.5c)

$x_{ij} \in \{0,1\} \qquad \forall i \in I, \forall j \in J$ (2.5d)

$y_j^-,\ y_j^+ \ge 0 \qquad \forall j \in J$ (2.5e)

$w_i^-,\ w_i^+ \ge 0 \qquad \forall i \in I$ (2.5f)

where

$\bar{\pi}_j$: Dual price for constraint *j*, $\forall j \in J$, of constraint set (2.5b) obtained after relaxing **GAP Exact Penalty**

$\bar{\pi}_i$: Dual price for constraint *i*, $\forall i \in I$, of constraint set (2.5c) obtained after relaxing **GAP Exact Penalty**

The formulation **Dual of Relaxed GAP-Exact Penalty** given below represents the dual of the relaxation of **GAP-Exact Penalty**. Note that the dual prices π_j, $\forall j \in J$ are limited to $-\varepsilon_j^- \le \pi_j \le \varepsilon_j^+$ by (2.6c), (2.6d), and (2.6g), and π_i, $\forall i \in I$ are limited to $0 \le \pi_i \le \delta_i^-$ by (2.6e), (2.6f), and (2.6h), to keep the dual-feasible status. With proper values of the parameters, which are $\varepsilon_j^-, \varepsilon_j^+, \delta_j^-,$ and δ_j^+, the target regions of the dual variables can be reduced. Then, the relaxation of **GAP-Exact Penalty** will match with one of the problems dealt with by the iterative process of the Boxstep method using the trust-region idea (see Marsten et al. [65]). Tran et al. [87] extend this exact penalty method to Boxstep method in order to stabilize dual prices, and their method also exploits all possible properties of a stabilization technique for the CGM, and the method has been used for the airline crew pairing problem.

Dual of Relaxed GAP-Exact Penalty

Maximize $\sum_{j\in J}\pi_j - \sum_{i\in I} b_i\pi_i$ (2.6a)

Subject to $\pi_j - d_{ij}\pi_i \le c_{ij}$ $\forall i \in I, \forall j \in J$ (2.6b)

$-\pi_j \le \varepsilon_j^-$ $\forall j \in J$ (2.6c)

$\pi_j \le \varepsilon_j^+$ $\forall j \in J$ (2.6d)

$\pi_i \le \delta_i^-$ $\forall i \in I$ (2.6e)

$-\pi_i \le \delta_i^+$ $\forall i \in I$ (2.6f)

π_j unrestricted $\forall j \in J$ (2.6g)

$\pi_i \ge 0$ $\forall i \in I$ (2.6h)

Merle et al. [68] propose a procedure to combine the perturbation method and the exact penalty method into one dual stabilization technique. They demonstrate updating the parameters for penalties and bounds, and apply the technique to the CGM. The formulations **GAP-DST** and **Dual of Relaxed GAP-DST** given below are the GAP with the dual stabilization technique and the GAP with the dual of the relaxation of **GAP-DST**, respectively.

GAP-DST

Minimize $\sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} + \sum_{j\in J}(\varepsilon_j^- y_j^- + \varepsilon_j^+ y_j^+) + \sum_{i\in I}(\delta_i^- w_i^- + \delta_i^+ w_i^+)$ (2.7a)

Subject to $\sum_{i\in I} x_{ij} - y_j^- + y_j^+ = 1$ $\forall j \in J$ $(\bar{\pi}_j)$ (2.7b)

$\sum_{j\in J} d_{ij}x_{ij} - w_i^- + w_i^+ \le b_i$ $\forall i \in I$ $(\bar{\pi}_i)$ (2.7c)

$y_j^- \le \alpha_j^-$ $\forall j \in J$ $(\bar{\theta}_j^-)$ (2.7d)

$y_j^+ \le \alpha_j^+$ $\forall j \in J$ $(\bar{\theta}_j^+)$ (2.7e)

$w_i^- \le \beta_i^-$ $\forall i \in I$ $(\bar{\phi}_i^-)$ (2.7f)

$w_i^+ \le \beta_i^+$ $\forall i \in I$ $(\bar{\phi}_i^+)$ (2.7g)

$$x_{ij} \in \{0,1\} \qquad \forall i \in I, \forall j \in J \qquad (2.7h)$$

$$y_j^-,\ y_j^+ \geq 0 \qquad \forall j \in J \qquad (2.7i)$$

$$w_i^-,\ w_i^+ \geq 0 \qquad \forall i \in I \qquad (2.7j)$$

Dual of Relaxed GAP-DST

Maximize $$\sum_{j \in J} \pi_j - \sum_{i \in I} b_i \pi_i - \sum_{j \in J} \alpha_j^- \theta_j^- - \sum_{j \in J} \alpha_j^+ \theta_j^+ - \sum_{i \in I} \beta_i^- \phi_i^- - \sum_{i \in I} \beta_i^+ \phi_i^+ \qquad (2.8a)$$

Subject to $$\pi_j - d_{ij} \pi_i \leq c_{ij} \qquad \forall i \in I, \forall j \in J \qquad (2.8b)$$

$$-\pi_j - \theta_j^- \leq \varepsilon_j^- \qquad \forall j \in J \qquad (2.8c)$$

$$\pi_j - \theta_j^+ \leq \varepsilon_j^+ \qquad \forall j \in J \qquad (2.8d)$$

$$\pi_i - \phi_i^- \leq \delta_i^- \qquad \forall i \in I \qquad (2.8e)$$

$$-\pi_i - \phi_i^+ \leq \delta_i^+ \qquad \forall i \in I \qquad (2.8f)$$

$$\pi_j \quad \text{unrestricted} \qquad \forall j \in J \qquad (2.8g)$$

$$\pi_i \geq 0 \qquad \forall i \in I \qquad (2.8h)$$

$$\theta_j^-,\ \theta_j^+ \geq 0 \qquad \forall j \in J \qquad (2.8i)$$

$$\phi_i^-,\ \phi_i^+ \geq 0 \qquad \forall i \in I \qquad (2.8j)$$

Note that the surplus and slack variables are bounded and also penalized in the objective function. The dual variables are limited in the dual formulation as follows:

$$-\theta_j^- - \varepsilon_j^- \leq \pi_j \leq \theta_j^+ + \varepsilon_j^+ \qquad \forall j \in J \qquad \text{by (2.8c), (2.8d) and (2.8g)}$$

$$-\phi_i^+ - \delta_i^+ \leq \pi_i \leq \phi_i^- + \delta_i^- \qquad \forall i \in I \qquad \text{by (2.8e), (2.8f) and (2.8h)}$$

If π_j, $\forall j \in J$ does not fall into the range $[-\varepsilon_j^-, \varepsilon_j^+]$, i.e., $\theta_j^- > 0$ or $\theta_j^+ > 0$, the objective function in the dual will be penalized by $\alpha_j^- \theta_j^-$ or $\alpha_j^+ \theta_j^+$. A similar property is also true for π_i,

$\forall i \in I$. This property prohibits the dual variables from taking values lying beyond the specified range, which leads to the stabilization of dual prices in the CGM. Also, note that **GAP-Exact Penalty** is identical with **GAP-DST** when $\alpha_j^- = \alpha_j^+ = \beta_i^- = \beta_i^+ = 0$ or $-\varepsilon_j^- \le \pi_j \le \varepsilon_j^+$ and $-\delta_i^+ \le \pi_i \le \delta_i^-$.

The Boxstep method, on the other hand, sets infinity as the upper bounds of the surplus and slack variables in the primal $(\alpha_j^-, \alpha_j^+, \beta_i^-, \text{and } \beta_i^+)$, where we can delete (2.7d), (2.7e), (2.7f) and (2.7g). Thus, for the dual to be feasible, the following condition should be met: $\theta_j^- = \theta_j^+ = \phi_i^- = \phi_i^+ = 0$, i.e., both π_j and π_i should fall in the range $-\varepsilon_j^- \le \pi_j \le \varepsilon_j^+$ and $-\delta_i^+ \le \pi_i \le \delta_i^-$, respectively. The dual stabilization technique by Merle et al. [68] iterates until it satisfies (2.8b) and $y_j^- = y_j^+ = w_i^- = w_i^+ = 0$. They also introduce several strategies for updating $(\varepsilon_j^-, \varepsilon_j^+, \delta_i^-, \delta_i^+)$ and $(\alpha_j^-, \alpha_j^+, \beta_i^-, \beta_i^+)$ using dual prices.

Pigatti et al. [76] make use of dual prices by relaxing the GAP at the root node. The dual prices are passed from a parent node down to its children nodes in order to use them for initial penalty parameters in the objective function for those nodes. They vary the upper bounds from 0.1, 0.01 to 0.001, and then, set them to zero after three iterations. Tran et al. [87], who address crew pairing problems, employ the Boxstep strategy. We adapt the method of Pigatti et al [76] for the SGAP. This will be explained in detail later in Chapter III.

2.2.6. Branching Variable Selection Strategies for the Branch-and-Price Method

In the BNP method, the branching step is implemented when the solution obtained from the column generation method is fractional and its objective value is better than that of the incumbent solution. Like in the branch-and-bound method, this step plays an important role in the BNP method.

Note that the variables in the CGM are not the ones in the original formulation, but convexity variables. For example, in the formulation **GAPCG1** given earlier, x_{ij} is not the variable in the master problem. Instead, it is λ_i^k. Accordingly, we branch on a convexity variable by fixing its value or by assigning an appropriate range to it. However, this branching strategy could be incompatible with the CGM. It would be better to branch on the original variables whose values are obtained by using the convexity variables. The information about the fixed variables is transferred to the pricing problem, and it reduces the size (or complexity) of the problem.

2.2.7. Application Areas of the Branch-and-Price Method

Even though the branch-and-price method has been illustrated here for the GAP, yet there have been a number of application areas of this method. Barnhart et al. [9] apply it to a set partitioning problem involving *m* patrol cars and *n* districts where every patrol car has to be allotted to one and only one district. It has also been applied to set covering problems in the context of vehicle routing and scheduling. Kallenhauge et al. ([49] and [50]) have developed a branch-and-price method for vehicle routing problem with time windows (VRPTW). The pricing problem is the shortest path problem with time windows and capacity constraints. Danna et al. [28] have devised a heuristic method (a guided tabu search) to implement the BNP approach for the solution of VRPTW. Degraeve et al. [29] have used the BNP method for solving the capacitated lot sizing problem with setup times (CLST).

2.3. Stochastic Programming

While deterministic programming (or DP) starts with the assumption that the data is known with certainty, stochastic programming (or SP) deals with uncertain situations. Some examples of stochastic situations are as follows.

- Traveling times between nodes or for the entire path in a traveling salesman problem
- Time required to process a job on a machine for a scheduling problem
- Periodic demand of a product in a production planning problem
- Number of patrol cars or ambulances available in a set covering problem
- Job assignment cost in an assignment problem

Typically, it is assumed that the probability distribution function, which could be either continuous or discrete, is known or could be estimated for the corresponding stochastic element. The stochasticity of a problem makes it more difficult to solve than its deterministic counterpart. However, a SP problem can provide feasible, valid solutions and optimize the expected objective value over future occurrences, called scenarios. In the following sections, we illustrate four ways of solving a SP problem with a simple example. For more details, please see Sen et al. [83].

2.3.1. Optimization with Expected Values

The simplest way to solve a SP problem is to replace parameters with their expected values. Consider the following example where the coefficients of the second constraint, (a,b), are the source of the uncertainty. The values of (a,b) for scenarios 1 and 2 are (1,3) and (0,2), respectively.

Example 2.1. Stochastic Programming

Minimize x_1+x_2 (2.9a)

Subject to $-x_1+x_2\leq 1$ (2.9b)

$$ax_1+bx_2=3 \quad (2.9c)$$

$$x_1\geq 0,\ x_2\geq 0 \quad (2.9d)$$

where $(a,b)=\begin{cases}(1,3) & \text{with probability}=0.5\\ (0,2) & \text{with probability}=0.5\end{cases}$

Note that $(\bar{a},\bar{b})$, the expected values of coefficients (a,b), are (0.5,2.5). Table 2.1 presents three formulations: the first for the formulation with expected values, the second for the formulation with scenario 1, and the second for the formulation with scenario 2. It also shows the optimal solutions to each formulation. Note that, it is possible for the optimal solution of the SP with expected values to be infeasible for the scenario that is acutally encountered. For instance, for the problem on hand, $(x_1^*,x_2^*)=\left(\frac{1}{6},\frac{7}{6}\right)$ and $z^*=\frac{4}{3}$, which is the optimal solution obtained with the expected values, is not feasible to any of the scenarios (second and third formulations). Hence, the use of expected values may not be a viable alternative for all SP problems.

Table 2.1. Formulation for **Example 2.1** with Expected Values, Scenario 1, and Scenario 2

Type of Formulation	Formulation with Expected Values	Formulation with Scenario 1	Formulation with Scenario 2
Formulation	Minimize x_1+x_2 Subject to $-x_1+x_2\leq 1$ $0.5x_1+2.5x_2=3$ $x_1,x_2\geq 0$	Minimize x_1+x_2 Subject to $-x_1+x_2\leq 1$ $x_1+3x_2=3$ $x_1,x_2\geq 0$	Minimize x_1+x_2 Subject to $-x_1+x_2\leq 1$ $2x_2=3$ $x_1,x_2\geq 0$
Optimal Solution	$(x_1^*,x_2^*)=\left(\frac{1}{6},\frac{7}{6}\right)$ $z^*=\frac{4}{3}$	$(x_1^*,x_2^*)=(0,1)$ $z^*=1$	$(x_1^*,x_2^*)=\left(\frac{1}{2},\frac{3}{2}\right)$ $z^*=2$

2.3.2. Optimization with Wait-and-See Policy

Under this policy, all the decisions are held until the uncertainty is revealed. In other words, we pretend that we can wait to make all decisions until all the information we need is available. We end up dealing with a deterministic programming problem for a scenario, which, in all likelihood, will be infeasible for the others.

From the first two approaches based on expected values and wait-and-see policy, if we do not consider all the scenarios properly, there is a possibility that we might end up with an infeasible solution for a future event. Hence, we need to make a proper response to the decisions that have already been made. This response is called a recourse policy in SP. A recourse problem provides a decision making procedure for SP. The decisions that are made for a stage help in solving another problem in the subsequent stage depending upon the scenario that has been revealed.

Consider the problem of investing in stocks over two months. Suppose that, at the beginning of the first month, we have enough information to decide on how much stock of which company to buy for that month. For the second month, we have information about possible scenarios that can happen, but we do not know which one. We wish to make a decision to maximize the profit for both months. The objective is to make a decision right now to give ourselves the best gain for both periods. This is unlike wait-and-see policy, which delays the decision for the second month until the start of the second month.

Another difference between the wait-and-see policy and the recourse model is the nonanticipativity property. The wait-and-see policy pretends we know which scenario is going to happen in the future and makes a decision based on that information. The recourse model, on the other hand, assumes that the decision we make is completely independent of future information since we do not know which event will happen. The policy with the recourse model is called the here-and-now policy, which is nonanticipative, against the wait-and-see policy, which mimics anticipativity. We present two recourse models in the following two sections.

2.3.3. Optimization of a SP with a Simple Recourse Model

One way to integrate a recourse policy is to include *a priori* the penalties for the deviations from a predetermined value, e.g., the penalty for unsatisfied demands in a supply demand problem, the penalty for jobs that could not be assigned to any machine due to the lack of total machine capacity in an assignment problem, or the penalty for extra work hours required in a crew scheduling problem. This kind of stochastic program is referred to as a simple recourse model. Consider the formulation **Example 2.1** introduced earlier. Let x_1 and x_2 be the first stage variables, and suppose at the second stage, we allow the deviation from 3, the RHS of the second constraint, to be penalized at \$1 per unit difference from 3. The second stage variables $s_1^+, s_1^-(s_2^+, s_2^-)$ are introduced for the first (second) scenario. We provide a degree of flexibility for the problem by taking a recourse policy. The values of the first stage variables are independent of the scenarios, while those for the second stage are defined with regard to each scenario. The penalty function associated with the first stage variables is $\$1*\mathrm{E}[3-(ax_1+bx_2)]$. We can formulate the stochastic programming problem with a simple recourse model as follows:

Stochastic Programming with Simple Recourse Model for Example 2.1

Minimize $x_1 + x_2 + \mathrm{E}[3-(ax_1+bx_2)]$ (2.10a)

Subject to $-x_1 + x_2 \le 1$ (2.10b)

$x_1, x_2 \ge 0$ (2.10c)

Observe that the recourse model considers compensation for uncertainty *a priori*, in contrast to the previous two methods, and an infeasible solution to the constraint set (2.9c) might be acceptable. Since the random event is discrete and the penalty is a constant, we can restate $\mathrm{E}[3-(ax_1+bx_2)]$ as follows:

$$\left(0.5(s_1^+ + s_1^-) + 0.5(s_2^+ + s_2^-)\right)$$

Note that, it is assumed that the recourse model follows a discrete probability distribution function under which each scenario has the same possibility of occurrence. The second stage variables, the variables for the recourse model, are nonnegative and satisfy the following constraints:

$$x_1 + 3x_2 + s_1^+ - s_1^- = 3$$
$$2x_2 + s_2^+ - s_2^- = 3$$

Then, the nonlinear program for the SP problem above can be reduced to the following linear programming problem:

LP Stochastic Programming with Simple Recourse Model for Example 2.1.

Minimize $x_1 + x_2 + 0.5(s_1^+ + s_1^-) + 0.5(s_2^+ + s_2^-)$ (2.11a)

Subject to $-x_1 + x_2 \le 1$ (2.11b)

$x_1 + 3x_2 + s_1^+ - s_1^- = 3$ (2.11c)

$2x_2 + s_2^+ - s_2^- = 3$ (2.11d)

$x_1, x_2, s_1^+, s_1^-, s_2^+, s_2^- \ge 0$ (2.11e)

The optimal solution and the optimal value are as follows: $(x_1^*, x_2^*, s_1^+, s_1^-, s_2^+, s_2^-) = (0,1,0,0,1,0)$, and $z^* = 1.5$.

2.3.4. Optimization of a SP with Two General Recourse Models

In the simple recourse model, the recourse actions that we can take are quite limited since all the original variables are set at the first stage. In the general recourse model, all the original variables are separated into the first stage variables and the second stage variables. Due to the resulting flexibility, we do not have to add the penalty cost for infeasibility, unlike the simple recourse

model. The values of the first stage variables are determined first, and we can delay decisions on the second stage variables until after the uncertainty has been revealed. Table 2.2 depicts two different general recourse models. In the first general recourse model, only x_1 is the first stage variable while x_2 is the second stage variable and can take two values, say, x_{21} and x_{22}. In the second model, there are no first stage variables and both variables are the second stage variables and as a result can take two values each, namely, x_{11}, x_{12}, x_{21}, and x_{22} based on two scenarios. Note that the objective value of the second model is less than that of the first. A greater flexibility leads to better results even though the problem could be more complicated.

Table 2.2. Two Different General Recourse Models for **Example 2.1**

1st Stage Variables	x_1	None
2nd Stage Variables	x_2	$x_1,\ x_2$
Formulation	Minimize $x_1 + 0.5x_{21} + 0.5x_{22}$ Subject to $-x_1 + x_{21} \le 1$ $-x_1 + x_{22} \le 1$ $x_1 + 3x_{21} = 3$ $2x_{22} = 3$ $x_1, x_{21}, x_{22} \ge 0$	Minimize $0.5x_{11} + 0.5x_{12} + 0.5x_{21} + 0.5x_{22}$ Subject to $-x_{11} + x_{21} \le 1$ $-x_{12} + x_{22} \le 1$ $x_{11} + 3x_{21} = 3$ $2x_{22} = 3$ $x_{11}, x_{12}, x_{21}, x_{22} \ge 0$
Optimal Solution	$\left(x_1^*, x_{21}^*, x_{22}^*\right) = \left(3, \frac{5}{6}, \frac{3}{2}\right)$ $z^* = \frac{5}{3}$	$\left(x_{11}^*, x_{12}^*, x_{21}^*, x_{22}^*\right) = \left(0, \frac{1}{2}, 1, \frac{3}{2}\right)$ $z^* = \frac{3}{2}$

2.3.5. Classification of Recourse Policy in the Stochastic Programming (SP)

The recourse policy in a SP is categorized depending upon the nature of coefficients in the second stage problem, and degree of feasibility and infeasibility of the second stage variables. We explain this next. Consider the following stochastic program.

General SP Problem

$$\text{Minimize} \quad \mathbf{Cx} + E[f(\mathbf{x})] \tag{2.12a}$$

$$\text{Subject to} \quad \mathbf{Ax} = \mathbf{b} \tag{2.12b}$$

$$\mathbf{x} \geq \mathbf{0} \tag{2.12c}$$

The function $\boldsymbol{f}_s(\boldsymbol{x})$ is the recourse function for each scenario s, $s \in S$, and is expressed as follows:

Recourse Function

$$f_s(\mathbf{x}) = \text{Minimize} \quad \mathbf{C}_s\mathbf{y}_s \tag{2.13a}$$

$$\text{Subject to} \quad \mathbf{D}_s\mathbf{y}_s = \mathbf{b}_s - \mathbf{A}_s\mathbf{x} \tag{2.13b}$$

$$\mathbf{y}_s \geq \mathbf{0} \tag{2.13c}$$

The formulation above is a general recourse model where **x** and **y** are vectors for the first stage and the second stage variables, respectively. Note that if the matrix $\mathbf{D}_s$ changes with each scenario, then we call the formulation as a random recourse model; otherwise, we call it as a fixed recourse model. When $\mathbf{D}_s=[\mathbf{I},-\mathbf{I}]$, it is a special case of fixed recourse model, called a simple recourse model.

We can consider ***x*** to be the decision variables which must be put into operation before the random parameters are revealed. Thus, the **x** variables are independent of the random occurrence of the uncertain parameters. On the other hand, the second stage variables, **y**, are reactive rather than proactive. We postpone decisions on these variables until the uncertainty is revealed. If a scenario follows a discrete probability distribution function, say, p_s, we can incorporate the two formulations above into one as follows:

Discrete SP Problem

$$\text{Minimize} \quad \mathbf{C}\mathbf{x} + \mathbf{P}_s\mathbf{C}_s\mathbf{y}_s \tag{2.14a}$$

$$\text{Subject to} \quad \mathbf{A}\mathbf{x} = \mathbf{b} \tag{2.14b}$$

$$\mathbf{A}_s\mathbf{x} + \mathbf{D}_s\mathbf{y}_s = \mathbf{b}_s \qquad \forall\, s \in S \tag{2.14c}$$

$$\mathbf{x} \geq \mathbf{0} \tag{2.14d}$$

$$\mathbf{y}_s \geq \mathbf{0} \qquad \forall\, s \in S \tag{2.14e}$$

If there is at least one feasible solution to the second stage variables, $\mathbf{y}_s$, for any value to the first stage variables, $\mathbf{x}$, considering the entire formulation above, we call the recourse problem as a complete recourse problem. If a feasible solution $\mathbf{y}_s$, $\forall\ s \in S$, exists for all scenarios with respect to $\boldsymbol{x}$ satisfying the constraint sets (2.14b) and (2.14d), then we call it a relatively complete recourse problem.

2.4. Monte Carlo Method

2.4.1. Application Areas

Along with the Dual Stabilization Technique, the Monte Carlo Method (MCM) is another procedure that we implement in the branch-and-price method to deal with the complex problems involving greater number of variables, constraints, and scenarios. Estimation of target values in the MCM is made in an arbitrary, stochastic, and nondeterministic way by spawning the set of random numbers produced by a random number generator. The objective of the MCM is to guide the search by using a subset of data set. It is useful when the amount of data set is too large to deal with at a time and when an algorithm is applied repeatedly to the data set. In fact, the MCM is one of the most practical methods used in the areas of physics, chemistry, mathematics, economics, and other computational areas where unpredictable activities need to be simulated. For example:

- In physics, the MCM is quite helpful in understanding and predicting the movement of millions of atoms, where we have to deal with millions of equations or inequalities,
- In order to build a financial portfolio, we have to consider not only deterministic parameters, but also, stochastic ones including the market status in asset management, interest rate for saving account, return on investment, stock price, among others,
- In traffic management, we have to deal with stochastic parameters such as the number of cars using the road, the average speed of the cars, the busiest time/day, the number of accidents, among others, and
- If we think of a simple supply and demand problem, there can be myriads of products made available by a supplier, and then, a variety of their demands, which can lead to a large number of scenarios to deal with.

2.4.2. Basic Statistics for the MCM

The MCM produces a set of random numbers and assesses the degree of validity of the results. It is efficient when the structure of the problem is so complicated that it is difficult even to obtain a feasible solution. With the Law of Large Numbers, the MCM provides us with comprehensible solutions even though we cannot determine whether those are optimal or not.

The MCM requires estimation of the properties of the values of a random variable vector, **x**. Denote by μ, σ^2, $\overline{X}(N)$, and $S^2(N)$ the average and variance of **x**, and the estimators of μ and σ^2, respectively. Let $x_1, x_2, \ldots, x_N$ be the independent, identically distributed (IID) random values of **x**. Then, $\overline{X}(N)$ and $S^2(N)$ are as follows:

$$\overline{X}(N) = \frac{\sum_{i=1}^{N} x_i}{N} \tag{2.15a}$$

$$S^2(N) = \frac{\sum_{i=1}^{N} \left(x_i - \bar{x}(N)\right)^2}{N-1} \tag{2.15b}$$

The 100(1-α) percent confidence interval, to estimate the accuracy of $\bar{X}(N)$ and $S^2(N)$, where $t_{N-1,1-\alpha/2}$ is the upper (1-α/2) critical point of the t-distribution with N-1 degrees of freedom, is given by

$$\bar{X}(N) \pm t_{N-1,1-\alpha/2}\sqrt{\frac{S^2(N)}{N}}\,. \tag{2.15c}$$

2.4.3. Steps for Implementing the MCM

The necessary steps for implementing the MCM to stochastic optimization problems are as follows (Mak et al. [60], Shapiro [84], and Verweij et al. [94]):

Step 1. Create a scenario-based model.

Step 2. Specify the number of scenarios to be used in a subset as well as the number of iterations.

Step 3. Randomly generate a subset of scenarios by a random number generator.

Step 4. Solve the model with this subset of scenarios.

Step 5. If the number of iterations meets a pre-specified value in step 2, go to step 6; otherwise, go to step 3.

Step 6. Evaluate the statistical results such as the average and variance; and determine the standard deviation of the observed variable and confidence interval.

2.4.4. Sampling Methods for the MCM

There are two sampling methods that can be used to determine scenarios, namely, interior sampling method (ISM) and exterior sampling method (ESM). The procedure for the ISM is as follows:

Step 1. Build the stochastic programming model with a recourse policy.

Step 2. Randomly create a subset of scenarios out of the set of all scenarios.

Step 3. Implement one iteration of the procedure by using the current subset of scenarios.

Step 4. If the number of iterations is reached at a pre-specified value or the stopping criterion is satisfied, stop; otherwise, go to step 5.

Step 5. Add new scenarios to the subset, remove some of the scenarios, which are in the subset, or create a whole new subset by a pre-specified procedure; and go to step 3.

The general procedure for the ESM is as follows:

Step 1. Build the stochastic programming model with a recourse policy.

Step 2. Randomly create a subset of scenarios out of the set of all scenarios.

Step 3. Solve the problem to optimality using the current subset of scenarios.

Step 4. If the predetermined number of replications is arrived at or the stopping criterion is satisfied, stop, otherwise, go to step 5.

Step 5. If we have to, change the size of the subset according to a pre-specified policy, and go to step 2.

The ISM adds new scenarios to the current subset, removes some scenarios from the subset, or creates a whole new subset of scenarios at each iteration; and selection of which one of these actions to take depends on the results of the optimization algorithm at each iteration (Jirutitijaroen et al. [48]). Therefore, each subset relies on the performance of the optimization algorithm and the pre-specified rules.

Although the ESM also allows us to change the size of the subset and the number of replication at each iteration, it generates a whole new subset of scenarios at every replication before the optimization algorithm is applied (Jirutitijaroen et al. [48]). Therefore, each subset is independent of the other irrespective of the performance of the optimization algorithm. We use the ESM in this study, and present it in more detail in the next section.

2.4.5. Exterior Sampling Method (ESM)

2.4.5.1. Formulation for the ESM

The ESM often employs the sample average approximation (SAA) method (see Verweij et al. [94] for detail), where E[f(**x**)] is estimated by $\frac{1}{N}\sum_{n=1}^{N} f$ (**x**,s_n). Then, the sample average approximation problem is as follows:

SAA Formulation

$$h_N = \quad \text{Minimize} \quad \mathbf{Cx} + \frac{1}{N}\sum_{n=1}^{N} f\,(\mathbf{x},s_n) \tag{2.16a}$$

$$\text{Subject to} \quad \mathbf{Ax} = \mathbf{b} \tag{2.16b}$$

$$\mathbf{x} \geq \mathbf{0} \tag{2.16c}$$

If we denote by $\hat{X}_N$ the optimal solution to the formulation above, then h_N and $\hat{X}_N$ are utilized for the estimator of h and $\hat{X}$, respectively. Suppose that we make M replications, and the sample size of each sample is N. Then, we have $h_N^1, h_N^2, \cdots, h_N^M$ as the objective function values of N replications, respectively, and $\hat{X}_N^1, \hat{X}_N^2, \cdots, \hat{X}_N^M$ the corresponding optimal solutions for the replications. Next, we present some statistical information on the SAA method.

2.4.5.2. Average and Variance of the *M* Replications

Let $\bar{h}_N$ be the average of $h_N^1, h_N^2, \cdots, h_N^M$. Then, the equation for $\bar{h}_N$ and its variance are as follows:

$$\bar{h}_N = \frac{1}{M}\sum_{m=1}^{M} h_N^m \tag{2.16d}$$

$$\mathrm{V}\left(\bar{h}_N\right) = \frac{1}{M(M-1)}\sum_{m=1}^{M}\left(h_N^m - \bar{h}_N\right)^2 \tag{2.16e}$$

2.4.5.3. Lower Bound of h

Note that, $\bar{h}_N$ is the unbiased estimator of the objective value, *h*, and also, it is well known (see Mak et al. [60]) that the relationship between $\bar{h}_N$ and *h* satisfies $\mathrm{E}(\bar{h}_N) \le h$. Therefore, $\bar{h}_N$ can be considered as a lower bound of *h*.

2.4.5.4. Upper Bound of h and the Unbiased Estimator of $h(\hat{X}_N^m)$

It is also obvious that any solution $\hat{X}_N^1, \hat{X}_N^2, \cdots, \hat{X}_N^M$ provides an upper bound, since $h(\hat{X}) \le h(\hat{X}_N^m)$, $\forall m = 1,...,M$, where the equality holds when $\hat{X}_N^m$ is the optimal solution (or one of alternate optimal solutions). The quality of this upper bound can be improved by using a large number of scenarios, say, $\tilde{N}$, by the Central Limit Theorem (CLT) (Mak et al. [60]). Then, an estimator of the upper bound is obtained if $\hat{X}_N^m$ is feasible to the function *h* (Verweij et al. [94]), as follows:

$$h_{\tilde{N}}(\hat{X}_N^m) = C\hat{X}_N^m + \frac{1}{\tilde{N}} \sum_{n=1}^{\tilde{N}} f(\hat{X}_N^m, s_n) \qquad (2.16g)$$

Note that $h_{\tilde{N}}(\hat{X}_N^m)$ might be considered as the unbiased estimator of the objective value of $\hat{X}_N^m$, $h(\hat{X}_N^m)$, when $\tilde{N} >> N$.

2.4.5.5. Variance of $h_{\tilde{N}}(\hat{X}_N^m)$

The variance of $h_{\tilde{N}}(\hat{X}_N^m)$ is given as follows (see Verweij et al. [94]):

$$\mathrm{V}\left(h_{\tilde{N}}\left(\hat{X}_N^m\right)\right) = \frac{1}{\tilde{N}(\tilde{N}-1)}\sum_{n=1}^{\tilde{N}}\left[C\hat{X}_N^m + f(\hat{X}_N^m, s_n) - h_{\tilde{N}}\left(\hat{X}_N^m\right)\right]^2$$

$$= \frac{1}{\tilde{N}(\tilde{N}-1)}\sum_{n=1}^{\tilde{N}}\left[C\hat{X}_N^m + f(\hat{X}_N^m, s_n) - C\hat{X}_N^m - \frac{1}{\tilde{N}}\sum_{n=1}^{\tilde{N}} f(\hat{X}_N^m, s_n)\right]^2 \qquad (2.16h)$$

$$= \frac{1}{\tilde{N}(\tilde{N}-1)}\sum_{n=1}^{\tilde{N}}\left[f(\hat{X}_N^m, s_n) - \frac{1}{\tilde{N}}\sum_{n=1}^{\tilde{N}} f(\hat{X}_N^m, s_n)\right]^2$$

2.4.5.6. Candidate for the Optimal Solution

To choose a "proper" solution out of the candidates, $\hat{X}_N^1, \hat{X}_N^2, \cdots, \hat{X}_N^M$, we select the one with the least objective value. In other words, choose $\left(\hat{X}_N\right)^*$ that satisfies the following equation:

$$\left(\hat{X}_N\right)^* \equiv \left[\left(\hat{X}_N\right)^* \in \left\{\hat{X}_N^m, m = 1,\ldots,M\right\} : h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right) \le h_{\tilde{N}}\left(\hat{X}_N^m\right), \forall m = 1,\ldots,M\right] \qquad (2.16i)$$

2.4.5.7. Evaluation of the Quality of $\left(\hat{X}_N\right)^*$

The closer the objective value of $\left(\hat{X}_N\right)^*$ is to the "real" optimal objective value, the better is the quality of $\left(\hat{X}_N\right)^*$. Ideally, we can use $h\left(\left(\hat{X}_N\right)^*\right) - h$ as a measure of the quality of $\left(\hat{X}_N\right)^*$. Since h is, typically, not known, we can, instead, consider

$$h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right) - \bar{h}_N \ . \qquad (2.16j)$$

Since each sample of a subset of scenarios is independently generated, the variance of the estimator of the optimality gap can be determined as follows:

$$\mathrm{V}\left[h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)-\bar{h}_N\right]=\mathrm{V}\left[h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)\right]+\mathrm{V}\left[\bar{h}_N\right]. \tag{2.16k}$$

Mak et al. [60] introduce an approximate 100(1-α) percent confidence interval for the optimality gap by applying the Central Limit Theorem (CLT) as follows:

$$\left[0, h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)-\bar{h}_N+z_{1-\alpha}\sqrt{\mathrm{V}\left[h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)\right]+\mathrm{V}\left[\bar{h}_N\right]}\right]. \tag{2.16l}$$

However, since the optimality gap, $h\left(\left(\hat{X}_N\right)^*\right)-h$, is supposed to be nonnegative, we can express this as presented in Mak et al. [60]:

$$\left[0, \mathrm{Max}\left[0, h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)-\bar{h}_N\right]+z_{1-\alpha}\sqrt{\mathrm{V}\left[h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)\right]+\mathrm{V}\left[\bar{h}_N\right]}\right], \tag{2.16m}$$

where $0<\alpha<1$ and $z_{1-\alpha}$ is the cumulative distribution function value of the standard normal distribution at the (1-α) critical point.

2.4.5.8. Application Areas of the Sample Average Approximation (SAA) Method

Shapiro [84] has observed various properties of the Monte Carlo Method (MCM), especially of the SAA method, when it is applied to the stochastic programming problems (SPs). Their study includes the rates of convergence and the complexity of the SAA method. Ahmed and Shapiro [85] extend the work of Shapiro [84] to the SAA method for stochastic integer programming problems (SIPs). They analyze the convergence, and propose an approximation method based on branch-and-bound algorithm, and they validate the solutions obtained by the approximation method with the lower and upper bounds presented above. We adopt their algorithm for the solution of the Stochastic Generalized Assignment Problem (SGAP) in this study. This is described in detail in chapter III. Table 2.3 summarizes the application areas of the MCM that have been discussed in the literature.

Table 2.3. Application Areas of the MCM in the Literature

Literature	Application Area
Mak et al [60]	A stochastic vehicle-allocation model in a single-commodity network
	An electric power system capacity-expansion model with uncertain demand forecasts and generator reliability
	A motor freight carrier's operations
	Capacity expansion and management of call-demands network in telecommunication industry
Kleywegt et al. [51]	A resource allocation problem, where each resource required to finish a job is uncertain
Verweij et al. [94]	The shortest path problem with random travel times
	The shortest path problem with arc failures
	The traveling salesman problem with random travel times

2.5. Stochastic Generalized Assignment Problem (SGAP)

2.5.1. Generalized Assignment Problem (GAP)

The GAP introduced in Section 2.1 is an NP-hard combinatorial problem. It consists of a number of machines ($\forall\ i = 1,\ldots,|I|$) and jobs ($\forall\ j = 1,\ldots,|J|$). Usually, a job is assigned to only one machine, but a machine could process multiple jobs. Each assignment requires a certain amount of resource (d_{ij}) and incurs a cost (c_{ij}). Moreover, each machine has a limitation on its capacity (b_i). There are special cases for the GAP as follows. First, if the cost for each job is the same for all the machines, i.e., $c_{ij}=c_j$, $d_{ij}=d_j$, and $b_i=b\ \forall i$, then it is called the GAP for identical machines. Note that, it has the same structure as the knapsack problem with multiple constraints (that is, multiple knapsacks). In addition, if $|I|=1$, the problem would be equivalent to a single knapsack

Table 2.4. Various Special Cases of the GAP

GAP for Identical Machines	Maximum Assignment Problem	Knapsack Problem
Minimize $\sum_{i\in I}\sum_{j\in J} c_j x_{ij}$	Maximize $\sum_{i\in I}\sum_{j\in J} x_{ij}$	Maximize $\sum_{j\in J} c_j x_j$
Subject to $\sum_{i\in I} x_{ij} = 1 \quad \forall j \in J$	Subject to $\sum_{i\in I} x_{ij} = 1 \quad \forall j \in J$	Subject to $\sum_{j\in J} d_j x_j \le b$
$\sum_{j\in J} d_j x_{ij} \le b \quad \forall i \in I$	$\sum_{j\in J} x_{ij} = 1 \quad \forall i \in I$	$x_j \in \{0,1\} \quad \forall j \in J$
$x_{ij} \in \{0,1\}, \forall i \in I, \forall j \in J$	$x_{ij} \in \{0,1\} \quad \forall i \in I, \forall j \in J$	

problem. Furthermore, if $c_j=d_j=1$ $\forall j$, $b=1$, and the objective is to maximize profit, it reduces to a maximum assignment problem, where normally $|I| << |J|$. Unlike the other problems mentioned above, the maximum assignment problem is a linear problem, which can be solved using polynomial algorithms, such as the Hungarian algorithm. Table 2.4 presents three formulations, namely, the GAP for identical machines, the maximum assignment problem, and the knapsack problem.

2.5.2. Greedy Heuristic for the Knapsack Problem

There are two reasons that we need to mention the greedy heuristic for the knapsack problem (KP). First, the greedy heuristic provides a polynomial-time algorithm to find a feasible and near-optimal solution to the knapsack problem, and second, we use a modified version of this greedy heuristic to obtain a starting solution for the SGAP. The greedy heuristic for the KP is as follows:

Step 1. Assume J to be the set of all jobs.

Step 2. Find j such that $\frac{c_j}{d_j} \geq \frac{c_{j'}}{d_{j'}}, \forall j' \in J \setminus j.$

Step 3. Check whether the capacity of the knapsack is exceeded, i.e., if $b - d_j$ is nonnegative. If it is nonnegative, go to step 4; otherwise, stop.

Step 4. Select job *j*, i.e., set $x_j = 1$.

Step 5. Reduce the capacity of the knapsack by subtracting d_j from *b*, i.e., set $b = b - d_j$.

Step 6. Remove *j* from the set *J*; and go to step 2.

We discuss the application of this greedy algorithm to the SGAP later in Chapter III.

2.5.3. Literature on the Stochastic Generalized Assignment Problem

As we alluded to earlier, there has not been much reported on the SGAP while the GAP has been one of the core problems for a variety of branch-and-price algorithms. Only three studies, due to Mine et al. [70], Albareda-Sambola et al. [1], and Albareda-Sambola et al. [2], have been reported on this problem. The stochastic element in the work reported by Mine et al. [70] is the machine capacity (b_i), which is the right hand side of the machine capacity constraint. On the other hand, in the other two studies, the processing time or the amount of resource required by a machine to process a job (d_{ij}) is assumed to be stochastic. For example, the amount of time for an insurance agent to deal with an accident can vary based on his/her ability and the degree of severity of the accident. When a computer software company tries to create an online game, finishing each part of the program is dependent on the ability of a programmer. This stochastic behavior is also encountered by building-contractors to process an activity, or by a policeman to arrive at an accident site from different police stations.

Mine et al. [70] have developed a Lagrangian heuristic to solve a two-stage stochastic programming formulation for the GAP. The other two studies have assumed the stochastic demand to follow the Bernoulli distribution, $B(p)$, where a job arrives for processing with the probability p, i.e., only a random set of jobs are considered. The first stage variables represent the assignment of jobs to the machines before the arrival of the jobs, and the second stage variables represent the reassignment of the jobs that really arrived. At the first stage, all the jobs

are assigned to the set of machines, and then, once the demand has been realized, the machines may become either overloaded or relatively "underloaded". The jobs on the overloaded machines are reassigned, which incurs a reassignment penalty. Albareda-Sambola et al. [1] have developed heuristic methods, while Albareda-Sambola et al. [2] present three exact algorithms consisting of branch-and-bound, use of optimality cuts generated by the branch-and-cut methods provided by CPLEX, and a special-purpose lower bound.

Although the GAP has been alluded to in the literature as one of the ideal problems for implementing the branch-and-price method, yet no study has been reported on the use of the BNP method for the solution of the SGAP. This has been our motivation to work on the SGAP.

2.6. Hospital Scheduling Problem

The studies on hospital scheduling problems have been focused on two areas: staff scheduling and operating room scheduling. Some efforts have also been devoted to integrate these two areas as well in order to develop a comprehensive approach for hospital scheduling. The term, staff usually refers to nurses (see Trivedi [88], Berrada et al.[14], and Burke et al. [19]) or trainees (see Beliën [10]). Litvak et al. [58] and Beliën et al. [12] deal with the scheduling of operating rooms. These studies have been motivated by the desire to improve efficiency of a hospital system, and reduce cost, which is imperative to make health care costs affordable for all, and also, for hospitals to offer quality service, in view of growing competition in health care.

2.6.1. Staff Scheduling Problem

Staff scheduling problems are categorized into two groups: long-term scheduling and short-term scheduling problems. The former addresses the long-term needs of a hospital and determines when to hire or lay-off the hospital staff. Thus, it is analogous to capacity planning in manufacturing industry. On the other hand, the latter is related to short-term production planning. It schedules doctors to perform operations in operating rooms on a daily basis. The planning period could be for a month, a week, or a day. The literature devoted to this problem is summarized in Table 2.5.

Table 2.5. Literature Review on Staff Scheduling Problem

Literature	Application Area	Method
Trivedi [88]	Expense budgeting in a hospital nursing department	Mixed-integer goal programming
Venkataraman et al. [93]	An integrated planning for both long-term and short-term scheduling problems	Staffing and scheduling policies
Berrada et al. [14]	Nurse scheduling	Multi-objective approach
Burke et al. [19]	Nurse rostering problem	Multi-criteria evolutionary approach
Cheang et al. [24] Burke et al. [20] Ernst et al. [35] Blëchliger [17]	Staff (especially nurse) scheduling problems	Beneficial surveys
Ozkarahan [73] Chen et al. [25] Azaiez et al. [6]	Nurse scheduling	Goal programming methods
Jaumard et al. [46]	Nurse scheduling	Generalized linear programming model and 0-1 column generation model
Mehrotra et al [67]	Shift scheduling	Branch-and-price approach
Bard et al. [7]	Nurse scheduling	Heuristic column generation approach
Beliën ([10] and [11])	trainee scheduling	Branch-and-price approach

There are different types of staff involved in a hospital, including nurses, trainees, and surgeons. The nurse scheduling problem pertains to determining work periods (or shifts) for nurses on a daily basis. Typically, there are three shifts, namely, day-shift, night-shift, and late-night shift, which require the use of nurses. The possible objective functions include the minimization of operating costs, the maximization of each nurse's preferences, or the minimization of the conflicts between the schedule and the nurse's requirements. Constraint logic programming methods (Bosi et al. [18]) or heuristics such as genetic algorithms (Burke et

al. [19]) have been applied to the nurse scheduling problems to find a feasible solution that attains the desired objective.

Beliën [11] has studied a trainee scheduling problem. The problem can be described as follows. A hospital needs to provide training on a pre-specified set of operations to the trainees. Each trainee has to participate in only one operation in a given period, and the length of time for which a trainee is involved must be within a given range of minimum and maximum periods. Consider the following notation and formulation.

i : Surgeon i, $\forall i \in I$

j : Operation j, $\forall j \in J$

t : Period t, $\forall t \in T$

$I(j)$: Set of trainees who have to participate in operation j

c_{ij} : Cost for trainee i assigned for period t

l_{ij} : Minimum number of periods for which trainee i is involved with operation j

u_{ij} : Maximum number of periods for which trainee i is involved with operation j

x_{ijt} : $x_{ijt} = 1$, if trainee i starts with operation j in period t; otherwise, $x_{ijt} = 0$

y_{ijt} : $y_{ijt} = 1$, if trainee i is involved with operation j in period t; otherwise, $y_{ijt} = 0$

Trainee Scheduling Problem

Minimize $$\sum_{j \in J} \sum_{i \in I(j)} \sum_{t \in T} c_{it} y_{ijt} \qquad (2.17a)$$

Subject to $$\sum_{j \in J} y_{ijt} \le 1 \qquad \forall i \in I, \forall t \in T \qquad (2.17b)$$

$$\sum_{i \in I(j)} y_{ijt} = 1 \qquad \forall j \in J, \forall t \in T \qquad (2.17c)$$

$$\sum_{t \in T} y_{ijt} \ge l_{ij} \qquad \forall j \in J, \forall i \in I(j) \qquad (2.17d)$$

$$\sum_{t \in T} y_{ijt} \le u_{ij} \qquad \forall j \in J, \forall i \in I(j) \qquad (2.17e)$$

$$x_{ij1} = y_{ij1} \qquad \forall j \in J, \forall i \in I(j) \qquad (2.17f)$$

$$x_{ijt} \geq y_{ijt} - y_{ij(t-1)} \qquad \forall j \in J, \forall i \in I(j), \forall t \geq 2 \qquad (2.17g)$$

$$\sum_{t \in T} x_{ijt} = 1 \qquad \forall j \in J, \forall i \in I(j) \qquad (2.17h)$$

$$x_{ijt}, y_{ijt} \in \{0,1\} \qquad \forall j \in J, \forall i \in I(j), \forall t \in T \qquad (2.17i)$$

The objective in (2.17a) is to minimize the scheduling cost. Constraint set (2.17b) restraints each trainee from attending more than one operation at-a-time, while constraint set (2.17c) limits only one trainee to an operation in a period. Constraint sets (2.17d) and (2.17e) specify a lower bound and an upper bound on training hours for a trainee for each operation. Constraint sets (2.17f) and (2.17g) define the relationship between the x_{ijt} and y_{ijt} variables. Constraint set (2.17h) asserts that there is only one start time for an operation to which a trainee is assigned.

Beliën et al. [10] applies the branch-and-price method to this trainee scheduling problem. Constraint set (2.17b) is considered as linking constraints and all the other constraint sets, i.e., (2.17c) to (2.17i), are treated as complicating constraints. As a result, it has $|J|$ pricing problems.

2.6.2. Operating Room Scheduling Problems

The operating room scheduling problems address the assignment of a surgical group to operating rooms. The pertinent entities include surgical groups, operating rooms, and the duration for which each surgical group needs an operating room. Blake et al. [15] have developed an integer programming model to assign each surgical group to an operating room, and they have also considered assigning the group as close to the most desired operation times as possible by penalizing the gap between the preferable hour and the real assigning hour. Santibanez et al. [81] have described a model that addresses operating rooms, bed availability, and the order of patients on the waiting list.

A number of case studies on the operating room scheduling problem have also been reported (Weiss [95], Ozkarahan ([74] and [75]), Guinet et al. [40], Jebali et al. [47], Lapierre et al. [54], Dexter et al. ([30], [31], and [32]) and Marcon et al. [62]). The goal of operating room

scheduling is to guarantee availability of operating rooms for sufficient time slots to each surgical group, and to schedule the operating rooms as efficiently as possible. Since revenues from surgeries contribute significantly to the profits made by a hospital, this scheduling plays a critical role in the fiscal status of a hospital by preventing it from keeping its operating rooms unoccupied. The typical objectives of the operating room scheduling problem are:

- Maximization of the efficiency of the operating rooms,
- Maximization of the number of patients served by the hospital,
- Minimization of the over-workload surgical groups, or
- Maximization of the profit generated from surgeries.

2.6.3. Stochastic Hospital Scheduling Problem (SHSP)

2.6.3.1. Literature Review on SHSP

There are a number of elements of the hospital scheduling problem (HSP) that are stochastic in nature. This includes the time that a surgeon requires to perform an operation, and also, the arrival time of the patients. To offer a high quality of service, it is essential to allow sufficient time, although uncertain, for a procedure to be completed satisfactory. Typically, hospitals have maintained records of time required for each procedure in order to minimize gap between the predetermined time slot and the real time required by a procedure. However, a better approach is to consider stochastic operation times *a priori* while generating a schedule.

Belien et al. [10] have considered two other sources of stochasticity: the number of patients for each operating room and the length of stay for each patient. Their work has been focused on the cyclic master surgery schedule and has resulted from two case studies in Belgian hospitals. They employ MIP and a simulated annealing method for the solution of their problem.

Lamiri et al. [53] have presented a stochastic model where operation times are uncertain, and they have also considered two types of surgeries: elective surgeries such as plastic surgeries and routine surgeries and emergency surgeries such as those pertaining to acute abdomen surgery and acute appendicitis surgery. They have proposed a new stochastic mathematical programming

model for the schedule of operating rooms, and they have developed a combination of the Monte Carlo Method and mixed integer programming for the solution of the problem.

2.6.3.2. A Formulation for the SHSP

Next, we present the formulation proposed by Lamiri et al. [53] for the SHSP mentioned above. They separate operations into two groups: elective operations that are planned *a priori* (before the start of a day), and emergency operations that occur unexpectedly. We use the following notation.

s : Scenario s, $\forall s \in S$, where $|S|$ is the number of total scenarios

i : Elective operation i, $i = 1, \ldots, |I|$, where I is the set of elective operations

t : Period t in a planning horizon, $t = 1, \ldots, TP, TP+1$

TP : Number of periods in a planning horizon

B_i : Earliest period when operation i could start

c_{it} : Cost for conducting elective operation i in period t

e_t : Penalty for unit workload that exceeds a hospital's capacity in period t

a_t : Capacity that needs to be reserved for emergency operations in period t

d_i : Duration (in hours) for elective operation i

b_t : Total available capacity of a hospital in period t (in hours)

a_{ts} : Capacity reserved for emergency operations in period t under scenario s

y_t : Amount of excessive workload in period t

x_{it} : Binary variable, =1, if operation i is conducted in period t; =0, otherwise

y_{ts} : Continuous variable representing the excessive workload in period t under scenario s

Instance of SHSP

Minimize $$\sum_{i \in I} \sum_{t=B_i}^{TP+1} c_{it} x_{it} + \sum_{t=1}^{TP} e_t y_t \qquad (2.18a)$$

Subject to $y_t = \mathrm{EXP}\left[\mathrm{Max}\left[0, a_t + \sum_{i \in I} d_i x_{it} - b_t \right] \right],\ \forall t = 1,...,TP$ (2.18b)

$$\sum_{t=B_i}^{TP+1} x_{it} = 1,\ i \in I \tag{2.18c}$$

$$x_{it} \in \{0,1\},\ i \in I,\ \forall t = 1,...,TP \tag{2.18d}$$

The first term in the objective function represents the cost for elective operations. Note that $x_{i,\,TP+1} = 1$ implies that operation i is not assigned to the current planning horizon. The second term sums up the penalty cost for over-workload periods. Constraint set (2.18b) determines the excessive workload during period t, that is, if the sum of the capacity that is reserved for emergency operations and consumed by elective operations is greater than the total capacity available during period t, then y_t will be positive. Constraint set (2.18c) guarantees that all elective operations will be assigned to one of the periods, $t = 1,...,TP,TP+1$. The formulation above, **Instance of SHSP**, can be equivalently transformed to the following LP problem by replacing (2.18b) with the following constraints:

$$y_{ts} \geq a_{ts} + \sum_{i \in I} d_i x_{it} - b_t \qquad \forall t = 1,...,TP,\ \forall s \in S \tag{2.18e}$$

$$y_t = \frac{\sum_{s \in S} y_{ts}}{|S|} \qquad \forall t = 1,...,TP \tag{2.18f}$$

$$y_{ts} \geq 0 \qquad \forall t = 1,...,TP,\ \forall s \in S \tag{2.18g}$$

2.7. Personnel Planning Problem

2.7.1. Short-Term and Long-Term Personnel Planning Problems

A Personnel Planning Problem (PPP), also called Workforce Planning Problem or Manpower Planning Problem, may be categorized into either a short-term PPP or a long-term PPP according

to the length of its planning horizon. In a short-term PPP, the workforce level can be affected by using temporary workers (e.g., contractors). The short-term and long-term PPPs are mainly associated with the shop-floor operational management and the upper level management of an organization, respectively. A comparison of short-term and long-term personnel planning problems is presented in Table 2.6.

2.7.2. Literature Review on Personnel Planning Problems

In an earlier study, Holt et al.[43] have incorporated both the optimal hiring-and-firing policy and the optimal production-and-inventory policy in their model. They provide linear decision rules to determine both production and workforce levels. Their model minimizes the sum of regular payroll costs, hiring-and-firing costs, overtime costs, and inventory costs. Singhal et al. [86] have evaluated the contribution of their work on operational management.

Table 2.6. Comparison of Short-Term and Long-Term Personnel Planning Problems

	Short-Term Personnel Planning	Long-Term Personnel Planning
Length of Planning Horizon	Daily, Weekly, Monthly Planning	Yearly, Multi-Year Planning
Level of Management	Shop-floor operational management	Top strategic management
Decisions	Detailed schedule of workers with respect to employment agreements and labor costs	(1) Optimal hiring-and-firing policy (2) Opening or closing of organization units (3) New formation or cancellation of positions
Associated Plans	(1) Short-term personnel recruitment plans (2) Short-term personnel assignment plans (3) Overtime plans (4) Training plans for contractors	(1) Personnel allotment plans over years (2) Training plans over years

A similar study is also reported by Hanssmann et al.[41]. They determine the amount of monthly production and the number of employees on the basis of monthly demands for products. For a planning horizon, the objective function of their model consists of the costs given in Holt et al. [43]. They start with quadratic cost functions, but end up approximating them with linear cost functions.

Martel et al. [63] have presented a two-stage stochastic programming problem for a short-term personnel planning in the job shops of an oil company. Because of the workers with special skills required in the industry, hiring and layoff costs tend to be large. Therefore, a stable employment of workers is very important, and the best case scenario is not to change workforce level. The stochastic element in this model is the workload, and it follows a known joint probability distribution function determined by the number of jobs and the job processing time.
Charnes et al. [23], Liang et al. [56], Liang et al. [57], Ali et al. [3], Blanco et al. [16], and Holder [42] have addressed the personnel planning problem encountered by U.S. Navy. Charnes et al. [23] have presented a collection of mathematical models for personnel planning. Liang et al. [56] create a large-scale network model in order to help assign sailors to missions, and Liang et al. [57] have presented the network model that maximizes the utilization of training resources. Holder [42] has also presented personnel planning problem in order to determine the optimal assignments of sailors to missions. The objective is to maximize the satisfaction of sailors by assigning them to sets of missions based on their preferences. They have employed interior-point algorithm to find an optimal policy for the determination of such missions, which helps in specifying a job list for each sailor.

Eiselt et al. [34] have developed a mixed integer programming (MIP) problem in which jobs are assigned to employees under the condition that each employee is capable of performing diverse jobs, but at different skill levels. While the objective in the study is to minimize the difference in each employee's capacity and job assignment, it contains hiring-and-firing policy as well as training schedule of employees. The constraint programming (CP) method and multi-objective optimization method are employed to solve the underlying MIP.

Chapter III. A Branch-and-Price Method for the Stochastic Generalized Assignment Problem (SGAP)

In this chapter, we provide the following three versions of the branch-and-price (BNP) method for the solution of the SGAP.

- "Basic" Branch-and-Price Method (designated as **BNP-SGAP**)
- Branch-and-Price Method with Dual Stabilization Technique (designated as **BNPDST-SGAP**)
- Branch-and-Price Method with Dual Stabilization Technique and a Greedy Heuristic (designated as **BNPDSTKP-SGAP**)
- Branch-and-Price Method with Dual Stabilization Technique, a Greedy Heuristic, and Monte Carlo Method (designated as **BNPDSTKP-SGAP with MCM**)

We, first, present the model formulation of the SGAP.

3.1. Stochastic Generalized Assignment Problem

3.1.1. Model Formulation for the SGAP and its Properties

3.1.1.1. The SGAP and its Recourse Model

We re-introduce the generalized assignment problem (GAP) from Section 2.1.

GAP

Minimize $\sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij}$ (3.1a)

Subject to $\sum_{i \in I} x_{ij} = 1 \qquad \forall j \in J$ (3.1b)

$$\sum_{j \in J} d_{ij} x_{ij} \leq b_i \qquad i \in I \qquad (3.1c)$$

$$x_{ij} \in \{0,1\} \qquad i \in I,\ \forall j \in J \qquad (3.1d)$$

For the SGAP, we assume the processing time of job j on a machine i, d_{ij}, to be stochastic. Let d_{ij} follow a discrete probability distribution function. Note that, the machine capacity constraints need not remain valid under the stochasticity of d_{ij} since the consumption may exceed the available capacity, b_i. We introduce slack variables to handle this situation. To that end, consider the following additional notation.

y_{is} : Slack variable to capture the excessive amount of resource utilized on machine i under scenario s, $i \in I$, $\forall s \in S$, where S is the set of all scenarios

ℓ_i : Unit penalty for the excessive utilization of machine i, $i \in I$

d_{ijs} : Processing time of job j on machine i under scenario s, $i \in I$, $\forall j \in J$, $\forall s \in S$

We have the following formulation for the SGAP.

SGAP1

Minimize $$\sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} + \mathrm{E}[\tilde{f}(\vec{x})] \qquad (3.2a)$$

Subject to $$\sum_{i} x_{ij} = 1 \qquad \forall j \in J \qquad (3.2b)$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in I, \forall j \in J \qquad (3.2c)$$

where

$\vec{x}$: Vector of x_{ij}, $i \in I$, $\forall j \in J$, that satisfies (3.2b) and (3.2c)

$\mathrm{E}[\tilde{f}(\vec{x})]$: Expected penalty cost for extra resource requirement

Then, the recourse model for scenario s, $f_s(\vec{x})$, is as follows:

Recourse Model of SGAP1 for Scenario *s*

Minimize $\sum_{i\in I} e_i y_{is}$ (3.3a)

Subject to $y_{is} = \text{Max}\left(0, \sum_{j\in J} d_{ijs} x_{ij} - b_i\right)$ $\forall i \in I$ (3.3b)

$y_{is} \geq 0$ $\forall i \in I$ (3.3c)

Remark 3.1. If machine i is used beyond its capacity, i.e., $\sum_{j\in J} d_{ijs} x_{ij} > b_i$, then by constraints (3.3b), $y_{is} = \sum_{j\in J} d_{ijs} x_{ij} - b_i$. However, the objective function (3.3a) minimizes the sum of the product of e_i and y_{is}. Therefore, constraints (3.3b), in view of constraints (3.1c), can be expressed as

$$\sum_j d_{ijs} x_{ij} - y_{is} \leq b_i \qquad \forall i \in I\,. \tag{3.3d}$$

Then, we have $\mathrm{E}[\tilde{f}(X)] = \sum_s p_s \sum_i e_i y_{is}$, where p_s is the probability of occurrence of scenario s, and y_{is} satisfies constraints (3.3d). Consequently, the model formulation **SGAP1** can be re-written as

SGAP2

Minimize $\sum_{i\in I}\sum_{j\in J} c_{ij} x_{ij} + \sum_{s\in S} p_s \sum_{i\in I} e_i y_{is}$ (3.3e)

Subject to $\sum_{i \in I} x_{ij} = 1$ $\forall j \in J$ (3.3f)

$\sum_{j \in J} d_{ijs} x_{ij} - y_{is} \leq b_i$ $\forall i \in I, \forall s \in S$ (3.3g)

$x_{ij} \in \{0,1\}$ $\forall i \in I, \forall j \in J$ (3.3h)

$y_{is} \geq 0$ $\forall i \in I, \forall s \in S$ (3.3i)

In **SGAP2**, x_{ij}, $\forall i \in I$, $\forall j \in J$, are the first stage variables, and y_{is}, $\forall i \in I$, $\forall s \in S$, are the second stage variables. The coefficient, e_i, is the unit cost of acquiring additional resource type *i*. Formulation **SGAP2** is an adaptation of the formulation **GAP**, which is known to be NP-hard. Consequently, **SGAP2** is more difficult than **GAP** due to the introduction of additional constraints in (3.1c), which now becomes constraint set (3.3g) in **SGAP2**, and addition of continuous variables, y_{is}, $\forall i \in I$, $\forall s \in S$, which makes it a mixed binary integer programming problem from a pure binary integer program for the GAP. Consequently, the formulation **SGAP2** is at least as difficult as the formulation **GAP**, and is also a NP-hard problem.

3.1.1.2. Stochastic Properties of the Recourse Model for SGAP1

We can categorize the formulation **Recourse Model of SGAP1 for Scenario s**, the resource model of the SGAP, as follows:

- it is a two-stage recourse model as we have two different stage variables, and
- it is a fixed recourse model since all the coefficients of the second stage variables, y_{is}, being equal to $[-1]$, remain the same irrespective of i and s, but it is not a simple recourse model.

Moreover, we can show the following property.

Proposition 3.1. Recourse Model of SGAP1 for Scenario s is a complete recourse model.

Proof. When $\sum_{j\in J} d_{ijs}x_{ij} - b_i \geq 0$, the second stage variable, $y_{is} = \sum_{j\in J} d_{ijs}x_{ij} - b_i \geq 0$. In case $\sum_{j\in J} d_{ijs}x_{ij} - b_i < 0$ for some i and s, then $y_{is} \geq \sum_{j\in J} d_{ijs}x_{ij} - b_i (< 0)$, and y_{is} will be equal to zero due to its non-negativity requirement. Therefore, the formulation **Recourse Model of SGAP1 for Scenario s** remains feasible for all values of x_{ij}. □

3.1.2. An Illustrative Example of the SGAP

Suppose that the processing time of a job on a machine changes according to different random events depicting different speeds. If we have $|I|$ machines and $|R|$ different speeds (or rates), the number of possible scenarios is equal to $|R|^{|I|}$, irrespective of the number of jobs. Let $r(i,s)$ denote the speed factor for machine i under scenario s, then the value of $r(i,s)$ is given by

$$r(i,s) = \left\lfloor \frac{(s-1) \bmod \left(|R|^{|I|-i+1} \right)}{|R|^{|I|-i}} \right\rfloor + 1,$$

where *a mod (b)* is the remainder obtained when *a* is divided by *b*, and $\lfloor a \rfloor$ is the round-down value of *a*.

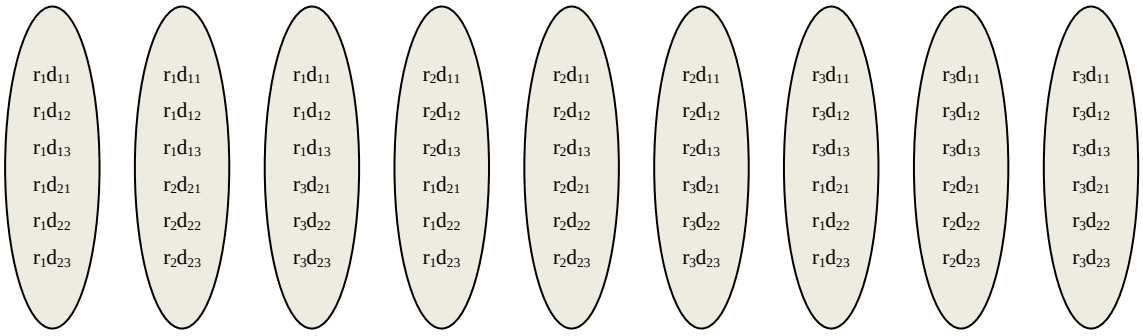

Figure 3.1. Scenarios for the SGAP with 2 Machines, 3 Jobs, and 3 Speed Factors for Each Machine

Example 3.1. Stochastic GAP

Consider an example consisting of 2 machines, 3 jobs, and 3 speed rates illustrated in Figure 3.1. Then, the speed factor for a pair of machine and scenario, $r(i,s)$, is as follows:

$$r(1,1)=r(1,2)=r(1,3)=1,\ r(1,4)=r(1,5)=r(1,6)=2,\ r(1,7)=r(1,8)=r(1,9)=3,$$
$$r(2,1)=r(2,4)=r(1,7)=1,\ r(2,2)=r(2,5)=r(2,8)=2,\ r(2,3)=r(2,6)=r(2,9)=3.$$

We have the following formulation.

Minimize

$$(c_{1,1}x_{1,1}+c_{1,2}x_{1,2}+c_{1,3}x_{1,3}+e_1(p_1y_{1,1}+p_2y_{1,2}+p_3y_{1,3}+p_4y_{1,4}+p_5y_{1,5}+p_6y_{1,6}+p_7y_{1,7}+p_8y_{1,8}+p_9y_{1,9}))+$$
$$(c_{2,1}x_{2,1}+c_{2,2}x_{2,2}+c_{2,3}x_{2,3}+e_2(p_1y_{2,1}+p_2y_{2,2}+p_3y_{2,3}+p_4y_{2,4}+p_5y_{2,5}+p_6y_{2,6}+p_7y_{2,7}+p_8y_{2,8}+p_9y_{2,9}))$$

Subject to

$$x_{1,1}+x_{2,1}=1$$
$$x_{1,2}+x_{2,2}=1$$
$$x_{1,3}+x_{2,3}=1$$
$$-y_{1,1}+d_{1,1,1}x_{1,1}+d_{1,2,1}x_{1,2}+d_{1,3,1}x_{1,3}\le b_1$$
$$\vdots$$
$$-y_{1,9}+d_{1,1,9}x_{1,1}+d_{1,2,9}x_{1,2}+d_{1,3,9}x_{1,3}\le b_1$$
$$-y_{2,1}+d_{2,1,1}x_{2,1}+d_{2,2,1}x_{2,2}+d_{2,3,1}x_{2,3}\le b_2$$
$$\vdots$$
$$-y_{2,9}+d_{2,1,9}x_{2,1}+d_{2,2,9}x_{2,2}+d_{2,3,9}x_{2,3}\le b_2$$
$$x_{ij}\in\{0,1\}\quad \forall i=1,2,\forall j=1,2,3$$
$$y_{is}\ge 0\quad \forall i=1,2,\ \forall 1,...,9$$

where $d_{ijs}=r(i,s)d_{ij}$, the processing time required by machine i to process job j under scenario s.

Note that, we have the same block diagonal structure for this formulation as that for the formulation **GAP**. In other words, the number of pricing problems for the formulation **SGAP2** is equal to the number of machines. In the sequel, we present the following four branch-and-price methods for **SGAP2**: (1) basic branch-and-price method (**BNP-SGAP**)**,** (2) **BNP-SGAP** in conjunction with the dual stabilization technique (**BNPDST-SGAP**), (3) **BNPDST-SGAP** with an advanced start (**BNPDSTKP-SGAP**), and (4) **BNPDSTKP-SGAP** with the Monte Carlo method (**BNPDSTKP-SGAP with MCM**). However, first, we show the potential advantage of solving the SGAP using a stochastic program in contrast to solving it using expected values of processing times.

3.1.3. SGAP with Expected Values

In this section, the SGAP is solved using the expected values of processing times. First, we provide the underlying formulation of the SGAP for this case, and then, compare the solution obtained by solving the SGAP as a stochastic program with the solution obtained by using expected values of processing times.

3.1.3.1. Formulation for the SGAP with Expected Values of Processing Times

Consider the following notation.

$\hat{y}_i$: Slack variable to capture the expected value of excessive amount of resource utilized on machine i, $\forall i \in I$

$\hat{d}_{ij}$: Expected value of processing time of job j on machine i, $\forall i \in I$, $\forall j \in J$

Note that $\hat{d}_{ij} = \sum_{s \in S} p_s d_{ijs}$. We have the following formulation for the SGAP with expected values of processing times, designated as **Expected SGAP2**.

Expected SGAP2

Minimize $\sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} + \sum_{i\in I} e_i\hat{y}_i$ (3.4a)

Subject to $\sum_{i\in I} x_{ij} = 1 \quad \forall j \in J$ (3.4b)

$\sum_{j\in J} \hat{d}_{ij}x_{ij} - \hat{y}_i \le b_i \quad \forall i \in I$ (3.4c)

$x_{ij} \in \{0,1\} \quad \forall i \in I,\ \forall j \in J$ (3.4d)

$\hat{y}_i \ge 0 \quad \forall i \in I$ (3.4e)

Note that **Expected SGAP2** has $|I|$ ($|S|$-1) less number of constraints (compare (3.3g) with (3.4c)) and $|I|$ ($|S|$-1) less number of continuous variables (compare y_{is} in **SGAP2** with $\hat{y}_i$ in **Expected SGAP2**) than **SGAP2**.

3.1.3.2. Comparison between SGAP2 and Expected SGAP2

We solve both **SGAP2** and **Expected SGAP2** using different values of $|I|$, $|J|$, $|S|$, and b_1. The values used for these parameters are presented in Table 3.1. The 1st, 2nd, 3rd, 4th, and 5th columns represent the experiment number, the number of machines, the number of jobs, the number of scenarios in **SGAP2**, and the machine capacity, respectively. The optimal objective function values for **SGAP2** are shown in the 6th column and those for **Expected SGAP2** are shown in the 7th column. Let x_{ij}^E, $i \in I, j \in J$ be the optimal solution of x_{ij}, $i \in I, j \in J$, for **Expected SGAP2**. Since the constraint sets (3.3f) and (3.4b) are the same, x_{ij}^E, $i \in I, j \in J$, satisfy (3.3f) as well. By substituting x_{ij}^E for x_{ij}, $i \in I, j \in J$, in (3.3g), we can obtain corresponding values y_{is}, $i \in I, s \in S$, denoted by y_{is}^E, $i \in I, s \in S$. Then, the solution (x_{ij}^E, y_{is}^E), $i \in I, j \in J, s \in S$, is feasible to the formulation **SGAP2**. Therefore, the objective function value obtained by (x_{ij}^E, y_{is}^E) gives an upper bound for each experiment, which is presented in the 8th column, named as the "Upper Bound" column. Accordingly, the values in Column 8 represent the "true" objective function values obtained by implementing x_{ij}^E, $i \in I, j \in J$. The ratios of the values in Column 8 to those in Column 6 give the magnitude by which the expected value solutions are worse. The values of these ratios are presented in percentages.

Table 3.1. Comparison of **SGAP2** with **Expected SGAP2**

Experiment #	$\|I\|$	$\|J\|$	$\|S\|$	b_1	Objective Function Value			Ratio of Values in Column 8 to Values in Column 6 (%)
					SGAP2	**Expected SGAP2**	Upper Bound	
1	3	3	8	3	825.187	754.875	834.5625	101.14%
2	3	3	8	6	464.25	37.6875	590.8125	127.26%
3	3	3	8	9	300.188	33	410.34375	136.70%
4	3	3	8	12	220.5	33	295.5	134.01%
5	3	3	8	15	150.188	33	276.75	184.27%
6	3	3	8	18	70.5	33	258	365.96%
7	3	3	8	21	51.75	33	239.25	462.32%
8	3	3	8	24	33	33	220.5	668.18%
9	4	4	16	3	931.846	851.84717	931.84595	100.00%
10	4	4	16	6	420.89	46.018311	601.30396	142.86%
11	4	4	16	9	176.536	46	502.47278	284.63%
12	4	4	16	12	69.4375	46	653.14111	940.62%
13	4	4	16	15	68.2656	46	72.953125	106.87%
14	4	4	16	18	60.0625	46	71.78125	119.51%
15	4	4	16	21	58.8906	46	70.609375	119.90%
16	4	4	16	24	48.3438	46	69.4375	143.63%
17	5	5	32	3	1236.1	1156.1013	1236.1013	100.00%
18	5	5	32	6	432.903	60.045776	615.26665	142.13%
19	5	5	32	9	95.2255	60	234.03983	245.77%
20	5	5	32	12	69.4301	60	605.80911	872.55%
21	5	5	32	15	61.2818	60	766.99474	1251.59%
22	5	5	32	18	60.1099	60	688.18036	1144.87%
23	5	5	32	21	60.1053	60	61.281756	101.96%
24	5	5	32	24	60.0641	60	60.109863	100.08%
25	6	6	64	3	1651.06	1571.0632	1651.0632	100.00%
26	6	6	64	6	547.903	175.04578	730.26665	133.28%
27	6	6	64	9	92.5784	75	244.37972	263.97%
28	6	6	64	12	77.4079	75	357.18814	461.44%
29	6	6	64	15	75.0687	75	283.06592	377.08%
30	6	6	64	18	75.0092	75	656.12154	874.72%
31	6	6	64	21	75.0046	75	1365.5882	1820.67%
32	6	6	64	24	75	75	1187.9412	1583.92%

For instance, consider Experiment #12. The percentage value of the ratio is 940.62, which indicates that the use of the expected value solution results in the objective function value that is about 9.41 times the objective function value obtained by solving the SGAP as a stochastic program. The average value of ratio is 428.50%, i.e., the stochastic program does better than the use of **Expected SGAP2** over four folds.

Another reason for the use of the stochastic program is that, even though the expected value solution is feasible to **SGAP2**, yet the expected value of a processing time may never be realized. For instance, if the processing time of a job on a machine can take two possible values, say, 1 and 2, then their average of 1.5 will never be realized.

3.2. Branch-and-Price Method for the SGAP

3.2.1. Basic Formulation of the Branch-and-Price Method for the SGAP (BNP-SGAP)

Referring to model **SGAP2**, it consists of $|I|*(|J|+|S|)$ variables and $|J|+|I|*|S|$ constraints. The former is greater than the latter by $|J|*(|I|-1)$, for $|I|>1$. Hence, the column generation method is an attractive alternative for solving the SGAP. We designate this method as **BNP-SGAP**.

In order to implement **BNP-SGAP**, we assume the job assignment constraints to be the linking constraints and the machine capacity constraints to be the complicating constraints. We have the following formulation where **MP-SGAP2** and **PP-SGAP2 for Machine *i*** are the master problem and the pricing problem for machine *i*, respectively.

BNP-SGAP2

MP-SGAP2

$$\text{Minimize} \quad \sum_{i\in I}\sum_{k=1}^{k_i}\left(\sum_{j\in J} c_{ij}\left(x_{ij}^k\right)+e_i\sum_{s\in S} p_s\left(y_{is}^k\right)\right)\lambda_i^k \tag{3.5a}$$

$$\text{Subject to} \quad \sum_{i\in I}\sum_{k=1}^{k_i}\left(x_{ij}^k\right)\lambda_i^k = 1 \qquad \forall j\in J \qquad \left(\bar{\pi}_j\right) \tag{3.5b}$$

$$\sum_{k=1}^{k_i} \lambda_i^k \le 1 \qquad \forall i \in I \qquad (\bar{\pi}_i) \tag{3.5c}$$

$$\lambda_i^k \in \{0,1\} \qquad \forall i \in I,\ k = 1,...,k_i \tag{3.5d}$$

PP-SGAP2 for Machine i, $\forall i \in I$

Minimize $$-\bar{\pi}_i + \sum_{j \in J} (c_{ij} - \bar{\pi}_j) x_{ij} + \sum_{s \in S} p_s e_i y_{is} \tag{3.5e}$$

Subject to $$\sum_{j \in J} d_{ijs} x_{ij} - y_{is} \le b_i \qquad \forall s \in S \tag{3.5f}$$

$$x_{ij} \in \{0,1\} \qquad \forall j \in J \tag{3.5g}$$

$$y_{is} \ge 0 \qquad \forall s \in S \tag{3.5h}$$

where

k_i : Number of columns generated corresponding to machine *i*,

λ_i^k : k^{th} convexity variable for machine i, $\forall i \in I$, $k = 1,...,k_i$,

x_{ij}^k : k^{th} column of x_{ij} transferred from **PP-SGAP2** i, $\forall i \in I$, $\forall j \in J$, $k = 1,...,k_i$,

y_{is}^k : k^{th} column of y_{is} transferred from **PP-SGAP2** i, $\forall i \in I$, $\forall s \in S$, $k = 1,...,k_i$,

$\bar{\pi}_j$: Dual price of constraints (3.5b), $\forall j \in J$, and

$\bar{\pi}_i$: Dual price of convexity constraints (3.5c), $\forall i \in I$.

Let *K* be the number of iterations executed thus far. Then, $k_i \le K$, because it is not necessary that a column is generated for a machine at every iteration. Also, note that the convexity constraints (3.5c) hold as inequality because machine *i* may not be utilized just like for similar constraints in the formulation **GAPCG1** in Section 2.2.2. Note that **PP-SGAP2** is a multiple-constraint knapsack problem for which effective algorithms are available for solution. Hence, the above decomposition scheme is an attractive one to use.

3.2.2. Procedure for BNP-SGAP

A flowchart for the BNP-SGAP method is shown in Figure 3.2. Note that the column generation method aspect of the procedure is executed in steps 1 through 9, which are in blue, and the branch-and-bound aspect is implemented in steps 10 through 18, which are in yellow. In steps 2, 6, and 8, **Phase I** is set to "Yes" when the algorithm is in Phase I and to "No" when it is in Phase II. Next, we describe each of the steps presented in the flowchart.

- **Phase I:** (Determination of an Initial Feasible Solution)
    - **Step 1 (BNP):** Solve the relaxed **SGAP2** after updating the relevant information from the column generation procedure.
    - **Step 2 (BNP, Preparation for Phase I):** Add artificial variables to the job assignment constraints in the master problem, and make the objective function the sum of the artificial variables.
    - **Step 3 (BNP):** Relax the restricted master problem (**MP-SGAP2**) by making the convexity variables continuous. Solve the relaxed RMP and set the objective function value as a lower bound. Transfer the dual prices $\left(\overline{\pi}_i^k\right)$, $\forall i \in \mathrm{I}$, and $\left(\overline{\pi}_j^k\right)$, $\forall j \in J$, to the pricing problem, **PP-SGAP2 for Machine *i***.
    - **Step 4 (BNP):** Using the dual prices from the restricted master problem, construct the pricing problem corresponding to each machine. Solve each pricing problem. Note that it is not necessary for the columns to be the optimal solutions to the pricing problems. As long as a column has a negative reduced cost, i.e., it can improve the objective function value of the next restricted master problem, we can use it for the next iteration. As a result, there are two ways to solve the MIP pricing problems:

        - A heuristic method such as a greedy procedure, which does not guarantee achievement of an optimal solution but might be quite effective in terms of computational effort.

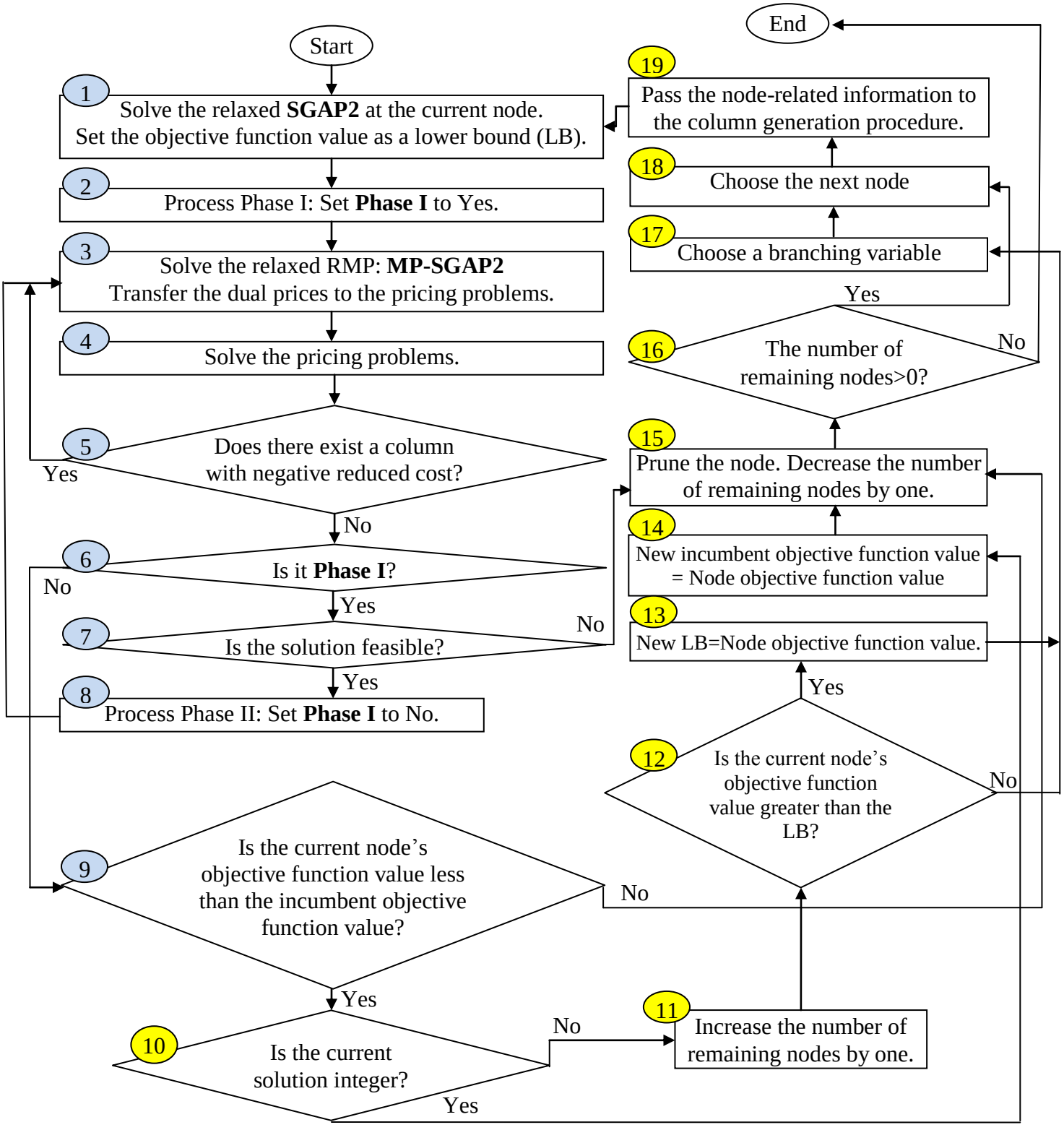

Figure 3.2. Branch-and-Price Method for the SGAP (**BNP-SGAP**)

- An optimum-seeking algorithm such as a branch-and-bound method, which guarantees achievement of an optimal solution, but it might be quite time consuming, especially when the dual prices approach near-optimal values because of the tailing effect (that results in a relatively small change in the objective function value). Furthermore, it requires excessive cpu time to find a suitable column since columns with negative reduced costs get scarce with increase in iterations.

Thus, we start with a heuristic algorithm to find valid columns, but once it fails, we switch over to an optimization algorithm.

- **Step 5 (BNP, Check for the availability of a column with a negative reduced cost):** If such a column exists, then go to step 3; otherwise, if the objective function values of all the pricing problems are nonnegative, then go to step 6.
- **Step 6 (BNP):** If we are in Phase I, go to step 7; otherwise, we are in Phase II, and go to step 9.
- **Step 7 (BNP, Check for the availability of an initial solution):** If we have an initial feasible solution, go to step 8; otherwise, go to step 14.
- **Step 8 (BNP, Preparation for Phase II):** Build the master problem with the initial solution found in Phase I.

- **Phase II:** Repeat steps 3, 4, and 5 until no valid columns exist. Go to step 9.
  - **Step 9 (BNP):** Obtain values of the original variables by combining the columns and values of convexity variables. If the objective function value of the current node is better than the incumbent objective function value, go to step 10; otherwise, go to step 15.
  - **Step 10 (BNP, Check for integrality):** Check for integrality of the original variables. If integrality is satisfied, go to step 14; otherwise, go to step 11.
  - **Step 11 (BNP, Integrality not satisfied):** Increase the number of remaining nodes by one.
  - **Step 12 (BNP, Check for new lower bound):** Check if the objective function value is greater than the current lower bound. If so, go to step 13; otherwise, go to step 17.

- **Step 13 (BNP, New lower bound):** Claim the objective function value as the new lower bound. Go to step 17.
- **Step 14 (BNP, New incumbent solution and objective function value):** Claim the solution and the objective function value as new incumbent solution and objective function value, respectively, and go to step 15.
- **Step 15 (BNP):** Prune the node. Decrease the number of remaining nodes by 1, and go to step 16.
- **Step 16 (BNP, Check for the number of the remaining nodes):** If the number of remaining nodes is zero, generate the incumbent solution and the objective function value as the optimal solution and the optimal objective function value, respectively, and stop; otherwise, go to Step 18.
- **Step 17 (BNP, Branching variable selection strategy):** Choose an original variable out of the fractional variables as the branching variable. Note that, we branch on an original variable, and not on a convexity variable. Increase the number of remaining nodes by one. For branching variable, we choose a fractional variable that is nearest to 0.5. Go to step 18.
- **Step 18 (BNP, Node selection strategy):** Choose the next node out of the set of the remaining nodes by the node selection strategy and go to step 19. The selected node is the one that has the greatest lower bound.
- **Step 19 (BNP, Node-related information):** Collect the node-related information, such as the variables which have been fixed at their respective values, the current lower bound, and the incumbent objective function value as an upper bound, and go to step 1 to start a new column generation process.

3.3. BNP-SGAP Method with Dual Stabilization Technique (BNPDST-SGAP)

In spite of the fact that the solution of the SGAP is aided by decomposing the problem via the column generation method, the performance of this method can be further improved by appropriately addressing the issues related to the values of dual variables and other aspects that are discussed in Section 2.2.4. These pertain to the heading-in effect, tailing-off effect, bang-bang effect, plateau effect, yo-yo effect, and the difficulty in solving the NP-hard pricing problem. We employ relevant techniques to alleviate the impact of these issues.

To that end, one of the methods is the dual stabilization technique (DST). Westerlund et al. [96] have demonstrated the use of this technique for the travelling salesman problem. The idea is to avoid extreme variations in the values of dual variables from one iteration to another, which can otherwise cause poor convergence. We have adapted the Boxstep method in the column generation procedure for the SGAP that restricts the range of dual prices generated from one iteration to another.

3.3.1. Implementation of BNPDST-SGAP

3.3.1.1. Investigation into Possible Poor Convergence of BNP-SGAP

We performed a preliminary investigation to determine if BNP-SGAP could possibly possess poor convergence properties. Table 3.2 presents results of this investigation. Information on the problems that we used is contained in the first seven columns. This includes the number of machines ($|I|$), the number of jobs ($|J|$), the number of scenarios ($|S|$), the machine capacity, the number of constraints, and the number of variables in Columns 1 to 7, respectively. The results pertaining to the optimal solution for each experiment, which includes the number of variables having a positive value, are presented in the 8^{th} and 9^{th} columns and those pertaining to the variables having zero values in the 10^{th} and 11^{th} columns.

Table 3.2. Number of Variables with Zero Values at the Optimal Solution

$\|I\|$	$\|J\|$	$\|S\|$	b_1	Number of Constraints	Number of Variables		Number of Variables Having Positive Values		Number of Variables Having Zero Values		Percent of Variables Having Zero Values (%)
					Number of Binary Variables (x_{ij})	Number of Continuous Variables (y_{is})	x_{ij}	y_{is}	x_{ij}	y_{is}	
7	7	128	3	903	49	896	7	704	42	192	24.8
7	7	128	6	903	49	896	7	384	42	512	58.6
7	7	128	9	903	49	896	7	128	42	768	85.7
7	7	128	12	903	49	896	7	0	42	896	99.3
8	8	256	3	2056	64	2048	8	1664	56	384	20.8
8	8	256	6	2056	64	2048	8	1024	56	1024	51.1
8	8	256	9	2056	64	2048	8	384	56	1664	81.4
8	8	256	12	2056	64	2048	8	128	56	1920	93.6
8	8	256	15	2056	64	2048	8	0	56	2048	99.6
9	9	512	3	4617	81	4608	9	3840	72	768	17.9
9	9	512	6	4617	81	4608	9	2560	72	2048	45.2
9	9	512	9	4617	81	4608	9	1280	72	3328	72.5
9	9	512	12	4617	81	4608	9	512	72	4096	88.9
9	9	512	15	4617	81	4608	9	0	72	4608	99.8
10	10	1024	9	10250	100	10240	10	3584	90	6656	65.2
10	10	1024	12	10250	100	10240	10	1536	90	8704	85.0
10	10	1024	15	10250	100	10240	10	512	90	9728	95.0
10	10	1024	18	10250	100	10240	10	0	90	10240	99.9

Note that, generally, the number of variables having positive values at the optimal solution, which is the sum of the values in the 8th and 9th columns, is much less than the number of constraints shown in the 5th column. Clearly, only a small percentage (about 10 to 14 %) of binary variables (x_{ij}, $i \in I$, $j \in J$) take on positive values in the optimal solution, for every experiment. Moreover, the percent of positive continuous variables ($y_{is}>0$, $i \in I$, $s \in S$) varies with the machine capacity, which is as expected. When the machine capacity is insufficient, the percent of y_{is} variables with zero value increases up to about 85%, while when the capacity is sufficient, that simply becomes zero, i.e., most of them become positive.

This observation leads us to believe that there may be quite a few redundant constraints in each SGAP problem, which causes degeneracy. Moreover, if the parameter values corresponding to each assignment of a job to a machine are not much different, the SGAP ends up generating solutions with almost identical objective function values. Both of these phenomena can lead to poor convergence behavior of **BNP-SGAP**. One of the most popular methods to address these is the dual stabilization technique (DST). Next, we present an adaptation of the DST technique suggested by Westerlund et al. [96] to our problem. Some new features regarding the implementation of this technique are then developed in Sections 3.3.2 and 3.3.3.

3.3.1.2. Formulation for the BNP Method with the Dual Stabilization Technique

The formulation **BNPDST-SGAP2** below presents the master problem for the BNP method with dual stabilization technique for the SGAP. Note that the use of the DST only impacts the master problem, while the pricing problem is unaffected.

K : Number of iterations thus far,

$z_j^{(-)}(z_j^{(+)})$: Slack (surplus) variable, $\forall j \in J$, and

$\delta_j^{(-)K}(\delta_j^{(+)K})$: Bound parameters (lower and upper bounds) for dual price $\overline{\pi}_j$, $\forall j \in J$, at the K^{th} iteration.

Then, the formulation of the BNP method in the presence of the dual stabilization technique is as follows.

BNPDST-SGAP2

MPDST-SGAP2

$$\text{Minimize} \quad \sum_{i\in I}\sum_{j\in J} c_{ij} \sum_{k}^{k_i} \left(x_{ij}^k\right)\lambda_i^k + \sum_{s\in S} p_s \sum_{i\in I} e_i \sum_{k}^{k_i} \left(y_{is}^k\right)\lambda_i^k - \sum_j \delta_j^{(-)K} z_j^{(-)} + \sum_j \delta_j^{(+)K} z_j^{(+)} \tag{3.6a}$$

$$\text{Subject to} \quad \sum_{i\in I}\sum_{k}^{k_i} \left(x_{ij}^k\right)\lambda_i^k - z_j^{(-)} + z_j^{(+)} = 1 \qquad \forall j \in J \qquad \left(\overline{\pi}_j\right) \tag{3.6b}$$

$$\sum_{k}^{k_i} \lambda_i^k \le 1 \qquad \forall i \in I \qquad \left(\overline{\pi}_i\right) \tag{3.6c}$$

$$\lambda_i^k \in \{0,1\} \qquad \forall i \in I,\ \forall k = 1,\ldots,k_i \tag{3.6d}$$

$$z_j^{(-)},\ z_j^{(+)} \ge 0 \qquad \forall j \in J \tag{3.6e}$$

Note that **MPDST-SGAP2** is the restricted master problem because of the consideration of only a subset of all possible columns at every iteration K, and $k_i \le K$, $\forall i \in I$.

Proposition 3.2. The introduction of variables, $z_j^{(-)}$ and $z_j^{(+)}$, and the bound parameters, $\delta_j^{(-)K}$ and $\delta_j^{(+)K}$, also restrain the dual prices, $\overline{\pi}_j$, $\forall j \in J$, and $\overline{\pi}_i$, $\forall i \in I$, corresponding to constraint sets (3.6b) and (3.6c), respectively.

Proof. Consider the dual of the relaxed **MPDST-SGAP2**, as follows:

Dual of MPDST-SGAP2

$$\text{Maximize} \quad \sum_{j\in J} \pi_j - \sum_{i\in I} \pi_i \tag{3.7a}$$

$$\text{Subject to} \quad \sum_{j\in J} \left(x_{ij}^k\right)\pi_j - \pi_i \le \sum_{j\in J} c_{ij}\left(x_{ij}^k\right) + e_i \sum_{s\in S} p_s \left(y_{is}^k\right) \qquad \forall i \in I,\ \forall k = 1,\ldots,k_i \qquad \left(\overline{\lambda}_i^k\right) \tag{3.7b}$$

$$-\pi_j \le -\delta_j^{(-)K} \qquad \forall j \in J \qquad \left(\overline{z}_j^{(-)}\right) \tag{3.7c}$$

$$\pi_j \le \delta_j^{(+)K} \qquad \forall j \in J \qquad \left(\overline{z}_j^{(+)}\right) \tag{3.7d}$$

$$\pi_j \text{ unrestricted} \qquad \forall j \in J \tag{3.7e}$$

$$\pi_i \ge 0 \qquad \forall i \in I \tag{3.7f}$$

In view of the constraints (3.7c) and (3.7d), we have the following range for each dual price.

$$\delta_j^{(-)K} \le \bar{\pi}_j \le \delta_j^{(+)K}, \qquad \forall j \in J. \tag{3.7g}$$

Hence, the dual prices for the linking constraints (3.6b) are limited by the bounding parameters, which are the coefficients of the new variables in the objective function. Consider constraints (3.7b) and (3.7g), we have

$$\bar{\pi}_i \ge \sum_j \left(x_{ij}^k\right)\bar{\pi}_j - \sum_j c_{ij}\left(x_{ij}^k\right) - e_i \sum_s p_s\left(y_{is}^k\right) \ge \sum_j \left(x_{ij}^k\right)\delta_j^{(-)K} - \sum_j c_{ij}\left(x_{ij}^k\right) - e_i \sum_s p_s\left(y_{is}^k\right), \tag{3.7h}$$

and thus, the dual prices for convexity constraints are also restrained. □

In order to update the values of the parameters, we modify the method of Pigatti et al [76] by using the dual prices from the previous iteration and a positive pre-specified value for the K^{th} iteration as follows:

$$\delta_j^{(-)K} = \bar{\pi}_j^{K-1} - \varepsilon^K \qquad \forall j \in J \tag{3.7i}$$

$$\delta_j^{(+)K} = \bar{\pi}_j^{K-1} + \varepsilon^K \qquad \forall j \in J \tag{3.7j}$$

where $\bar{\pi}_j^{K-1}$ is the dual price obtained at the (K-1)$^{\text{th}}$ iteration, and ε^K is a positive pre-specified value.

The difference between the method of Pigatti et al [76] and our dual stabilization technique (DST) lies in the method of managing the DST variables, $z_j^{(-)}$ and $z_j^{(+)}$, and the DST bound parameters, $\delta_j^{(-)K}$ and $\delta_j^{(+)K}$. We elaborate on this next. However, first, we make the following remark.

Remark 3.2. If $\bar{\pi}_j^0$ is the dual optimal solution to the relaxed **MPDST-SGAP2** obtained during the K$^{\text{th}}$ iteration, then based on the inequalities (3.7g), (3.7i) and (3.7j), the following inequalities hold true:

$$\bar{\pi}_j^{K-1} - \varepsilon^K \le \bar{\pi}_j^0 \le \bar{\pi}_j^{K-1} + \varepsilon^K \qquad \forall j \in J\,. \qquad (3.7k)$$

3.3.2. A Warm-Up Period Policy for the Determination of ε^K

We propose the following procedure for determining ε^K. Choose a value of ε^K, and then, proceed as follows:

- Increase ε^K when one or more slack ($z_j^{(+)}$) or surplus ($z_j^{(-)}$) variables take a positive value after the column generation method is completed, which implies that the solution given by the dual stabilization technique is not a feasible solution. In other words, the predetermined range for the dual values is too small and the "true" dual optimal solution lies outside this range, thus, requiring an increment in the value of ε^K. Note that, in the extreme case, if the value of ε^K increases to infinity, then the resulting situation is identical to the one without the dual stabilization technique, i.e., the dual values could assume any value. There would be no effect of the dual stabilization technique on the problem in that case.
- Decrease ε^K when the effect of the dual stabilization technique is hardly observed during the column generation method, i.e., when the computational effort is about the same or greater than without the DST. In the extreme case, if we set ε^K to zero, then $\bar{\pi}_j = \bar{\pi}_j^{K-1}$.

The iterations required to determine the value of ε^K is termed the warm-up period. In our experimentation, we started with a value of ε^K=0.001, and increased it to 0.01, 1, 2, 3, …., 10, as need be, until a feasible solution was obtained. A flowchart of the branch-and-price (BNP) method with the dual stabilization technique for the SGAP, after having determined ε^K during the warm-up period, is shown in Figure 3.3. Steps 1-1, 3-1, 3-2, 8-1, 9-1, and 9-2 in the flow

chart, which are shown in orange color, belong to the dual stabilization technique. The other steps are the same as those presented in Figure 3.2. We describe these steps next.

- **Phase I**
  - **Step 1-1 (DST):** Store the dual prices $\left(\bar{\pi}_j^0\right)$, and set **DST** to No. Go to Step 2.
- **Phase II**
  - **Step 8-1 (DST):** Set the number of iterations in Phase II, *K*, to one. Build the first RMP with the initial feasible solution found in Phase I. Add the slack and surplus variables, $z_j^{(-)}$ and $z_j^{(+)}$, to the job assignment constraints in the master problem. Update $\delta_j^{(-)1}$ and $\delta_j^{(+)1}$ using $\bar{\pi}_j^0$, determined in step 1-1, and ε^1, determined in a warm-period, and substitute their values in equations (3.7i) and (3.7j). Add the slack and surplus variables, respectively, $z_j^{(-)}$ and $z_j^{(+)}$, corresponding to $\delta_j^{(-)1}$ and $\delta_j^{(+)1}$. Go to step 3.
  - **Steps 3 to 8 (BNP):** Repeat steps 3 to 8 until no valid column exists.
  - **Steps 3-1 and 3-2 (DST):** After step 3, update the number of iterations in Phase II, $K = K + 1$. Store the dual variable values $\left(\bar{\pi}_j^K\right)$. At each iteration *K*, update $\delta_j^{(-)K}$ and $\delta_j^{(+)K}$ using $\bar{\pi}_j^{K-1}$ and ε^K. Note that these steps are performed only in Phase II.
  - **Step 9-1 (DST, Check for feasibility):** If there is a positive slack or surplus variable in the solution, go to step 9-2; otherwise, go to step 9.
  - **Step 9-2 (DST):** Increase the values of the parameters by ε^K based on the feasibility of the solution. If there are slack or surplus variables with positive values, the solution is infeasible, and then increase ε^K. Go to step 8-1.

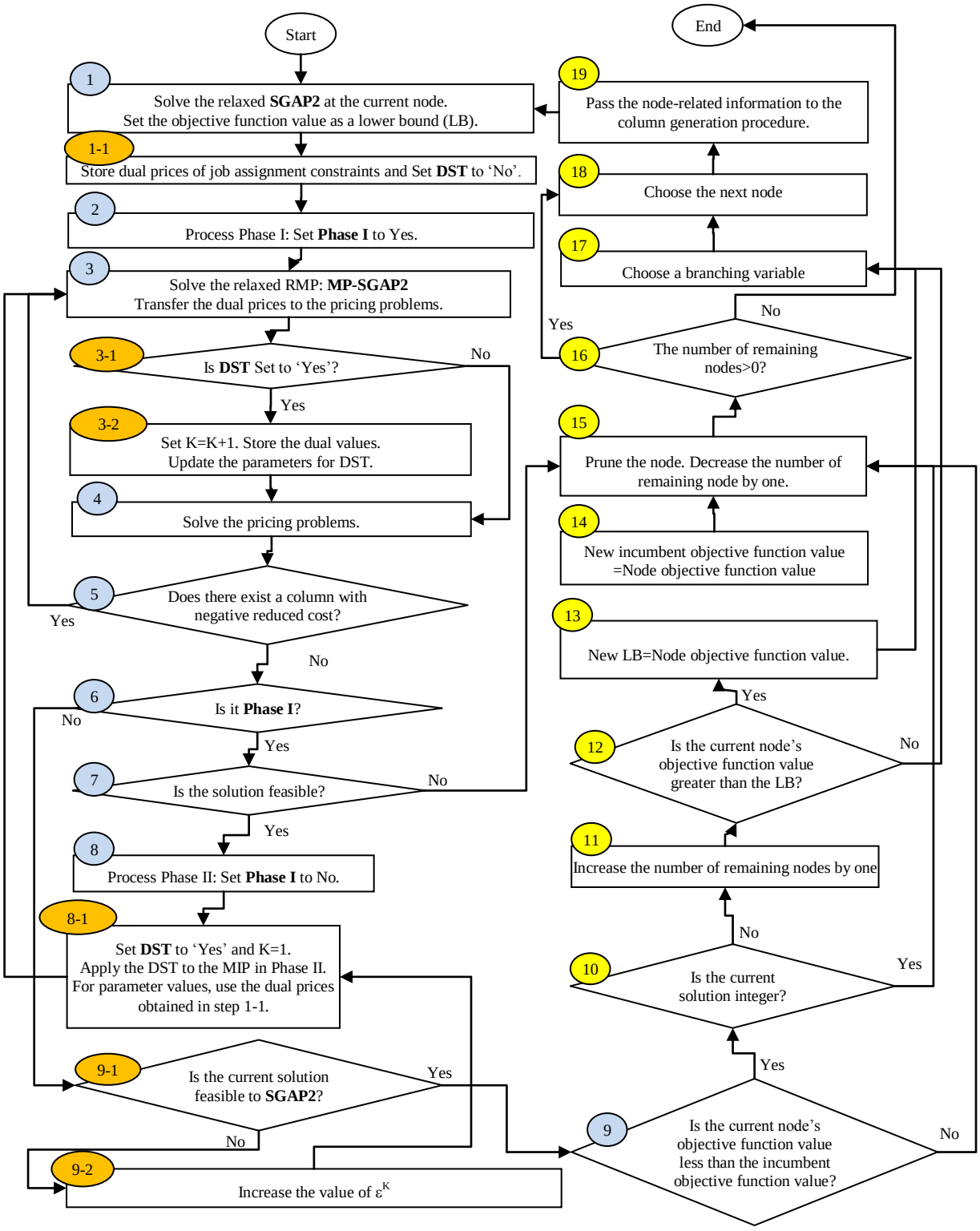

Figure 3.3. **BNP-SGAP** with the Dual Stabilization Technique (**BNPDST-SGAP**)

3.3.3. A New Policy for Updating $\delta_j^{(-)K}$ and $\delta_j^{(+)K}$ and Removing $z_j^{(-)}$ and $z_j^{(+)}$

Note that in the algorithm presented above, the dual stabilization technique is applied only in Phase II since, generally, little difficulty is encountered in Phase I to get an initial feasible solution. Updating of $\delta_j^{(-)K}$ and $\delta_j^{(+)K}$, the coefficients of the slack and surplus variables $z_j^{(-)}$ and $z_j^{(+)}$ in the objective function at the K^{th} iteration, respectively, is based on dual prices $\bar{\pi}_j^{K-1}$, $\forall j \in J$, and the nonnegative pre-specified values of ε^K (see (3.7i) and (3.7j)).

Once the columns with negative reduced costs are found in step 5, the restricted master problem is modified to include the new convexity variables corresponding to the new columns and the objective function coefficients corresponding to the slack and surplus variables (step 6). This is a standard practice for the implementation of the branch-and-price method with dual stabilization technique. However, the value of ε^K, which is predetermined during the warm-up period, may not always be very effective with regard to the convergence of the algorithm and in removing $z_j^{(-)}$ and $z_j^{(+)}$ from the basis.

Therefore, we have devised the following new strategy to lessen the effort required in finding a proper value for ε^K. As the column generation method is started at a node, we keep track of the number of iterations executed in Phase II. The $z_j^{(-)}$ and $z_j^{(+)}$ variables are kept in the system as long as this number is less than a pre-determined number, itDS. Once the number is equal to itDS, however, we check whether the $z_j^{(-)}$ and $z_j^{(+)}$ variables are basic. We delete those that are nonbasic. We also delete those $z_j^{(-)}$ and $z_j^{(+)}$ variables that are basic and have zero value. In case, the current solution is infeasible (i.e., at least a $z_j^{(-)}$ or $z_j^{(+)}$ has a positive value), we increase the value of ε^K, and solve the problem again. We repeat this process until all $z_j^{(-)}$ and $z_j^{(+)}$ variables become zero. The following steps are added to the algorithm, as shown in Figure 3.4. These steps are depicted in green color in the flow chart.

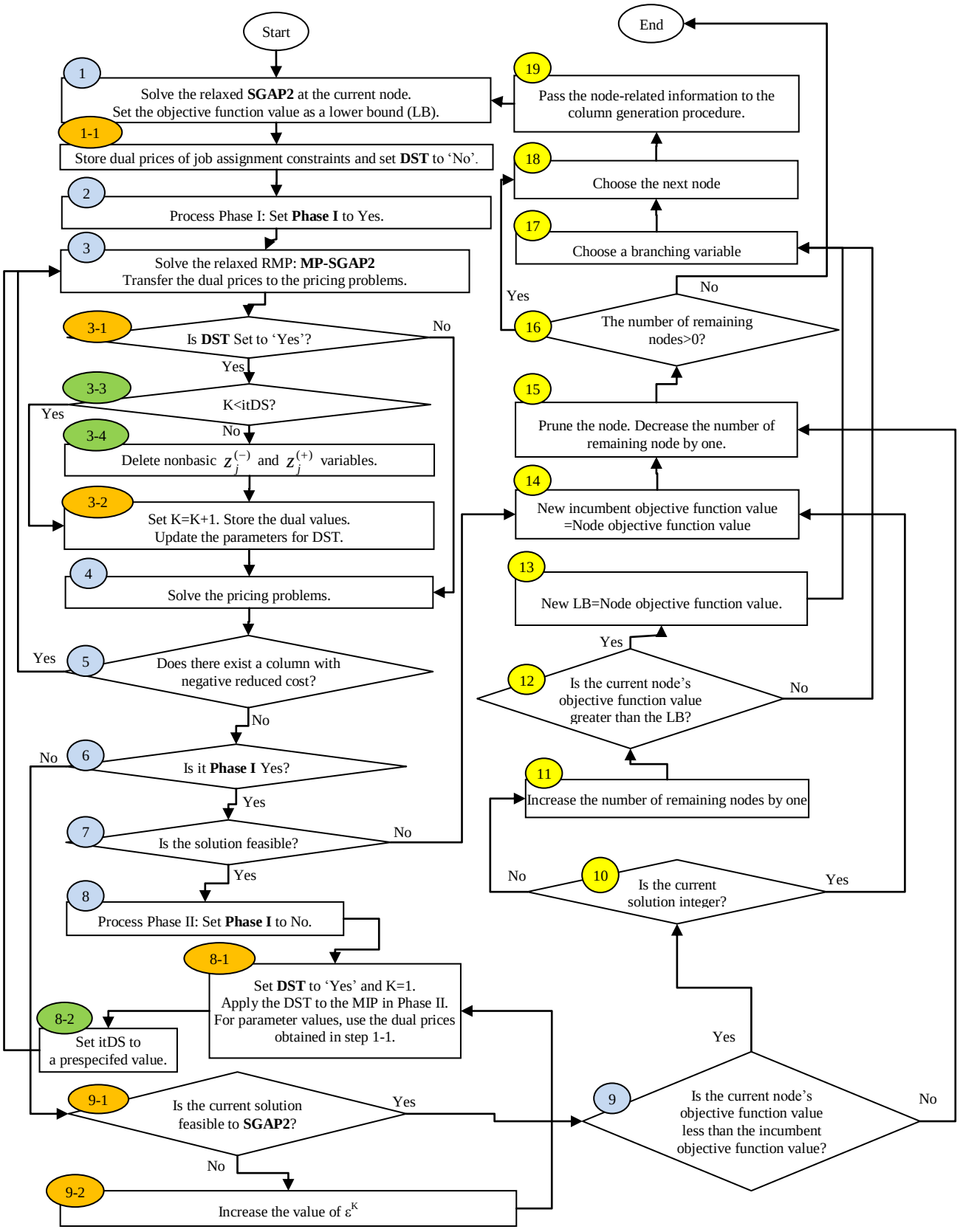

Figure 3.4. **BNPDST-SGAP** with a New Policy for Updating the Dual Stabilization Technique Parameters and Removing $z_j^{(-)}$ and $z_j^{(+)}$ Variables

- **Step 8-2 (DST) :** After step 8-1, set itDS to some positive integer. Go to step 3.
- **Step 3-3 (DST) :** After step 3-1, if the number of iterations in Phase II is less than itDS, go to step 3-2; otherwise, go to step 3-4.
- **Step 3-4 (DST) :** Delete nonbasic $z_j^{(-)}$ or $z_j^{(+)}$ variables. Go to step 3-2.

Note that, at each node, **BNPDST-SGAP** will yield the same objective function value as given by **BNP-SGAP**, and hence, will obtain the "true" optimal objective function value.

3.3.4. Effectiveness of New Policy on the Performance of the Dual Stabilization Technique (DST)

In this section, we demonstrate effectiveness of the new policy that we presented in the previous section. To that end, we compare the performance of our method with that proposed by Pigatti et al [76]. Table 3.3 summarizes results of 25 experiments that we used. The first, second, third, fourth, fifth, sixth, seventh, and eighth columns contain information on the experiment number, the number of machines, the number of jobs, the number of scenarios, the number of variables, the number of constraints, the first machine capacity, and the optimal objective function value for each experiment, respectively. The cpu times required by **BNPDST-SGAP** using the method of Pigatti et al. [76] and the new policy are presented in the next two columns. The highlighted number indicates the best of the value of cpu times for each experiment. The last column depicts the ratio of the cpu times obtained using the method of Pigatti et al. [76] and the new policy (in percent). For example, in the first experiment, the new policy performs about four times better than the method of Pigatti et al. [76]. Overall, the new policy performs better in 20 out of 25 experiments. This clearly indicates that the new policy tends to perform better than the method of Pigatti et al. [76].

Table 3.3. Effect of New Policy on the Performance of **BNPDSTKP-SGAP**

#	$\|I\|$	$\|J\|$	$\|S\|$	Number of Variables	Number of Constraints	b_1	Optimal Objective Function Value	CPU Time (Seconds)		Ratio (%)
								With the Method of Pigatti et al.	With the New Policy	
1	7	7	128	945	903	3	2416	0.93119	0.22978	405.26
2	7	7	128	945	903	6	991	0.21181	0.96744	21.89
3	7	7	128	945	903	9	316	0.56616	0.01384	4090.02
4	7	7	128	945	903	12	91	0.07347	0.07266	101.12
5	8	8	256	2112	2056	3	3258	2.27702	2.09413	108.73
6	8	8	256	2112	2056	6	1533	1.0738	1.13455	94.65
7	8	8	256	2112	2056	9	558	0.45522	0.4093	111.22
8	8	8	256	2112	2056	12	183	1.09313	2.21369	49.38
9	8	8	256	2112	2056	15	108	0.18537	0.17195	107.81
10	9	9	512	4689	4617	3	4226	10.6353	7.50807	141.65
11	9	9	512	4689	4617	6	2201	10.3255	9.73066	106.11
12	9	9	512	4689	4617	9	926	15.3685	8.16466	188.23
13	9	9	512	4689	4617	15	126	0.56569	0.55595	101.75
14	10	10	1024	10340	10250	3	5320	175.192	170.783	102.58
15	10	10	1024	10340	10250	6	2995	69.803	52.3906	133.24
16	10	10	1024	10340	10250	9	1420	60.2319	44.5622	135.16
17	10	10	1024	10340	10250	12	595	40.7404	21.7268	187.51
18	10	10	1024	10340	10250	15	220	23.0922	23.4798	98.35
19	10	10	1024	10340	10250	18	145	2.69772	3.4003	79.34
20	11	11	2048	22649	22539	3	6540	986.035	938.171	105.10
21	11	11	2048	22649	22539	6	3915	402.962	290.285	138.82
22	11	11	2048	22649	22539	9	2040	201.787	175.626	114.90
23	11	11	2048	22649	22539	12	915	303.662	198.413	153.05
24	11	11	2048	22649	22539	15	390	2983.55	2183.88	136.62
25	11	11	2048	22649	22539	18	165	19.2181	12.1092	158.71

3.4. BNPDST-SGAP Method with an Advanced Start (BNPDSTKP-SGAP)

In the two-phase method, the column generation method is used in Phase I to find an initial feasible solution which, invariably, requires a large amount of effort. Instead, we present a procedure to obtain an advanced start, which would help in reducing computational time. As noted earlier, model **SGAP2** is like a knapsack problem. Our procedure is based on the greedy heuristic procedure for the single constraint knapsack problem, which has been shown to generate almost optimal solutions (Martello et al. [64], Pisinger ([77], [78]), and Vazirani [92]). To that end, we first present this procedure next.

3.4.1. A Heuristic Method for the Knapsack Problem

Consider the following single constraint knapsack problem.

KP

Maximize $\sum_{i \in I} c_i x_i$ (3.8a)

Subject to $\sum_{i \in I} d_i x_i \le b$ (3.8b)

$x_i \in \{0,1\}, \ \forall i \in I$ (3.8c)

The greedy heuristic to obtain a feasible solution for model **KP** is as follows:

Step 1. Order the items in the decreasing order of c_i/d_i. Let this order be $\{[1],[2],...,[|I|]\}$, where $[i]$ represents the job in position i.

Step 2. Let $x_{[1]} = 1$, if the constraint is not violated; otherwise, $x_{[1]} = 0$.

Step 3. Set $b = b - d_{[1]}$.

Step 4. Select the next item in the order of items specified in step1, and repeat steps 2 and 3 until no more items are left.

3.4.2. A Greedy Heuristic for the SGAP

Note that for the minimization objective, we need to consider the items in the increasing order of c_i/d_i. We develop a procedure, designated as **GH**, to find an initial feasible solution at each node for **SGAP2** by modifying the above heuristic as follows.

Procedure GH

Step 1. Set $I \otimes J = \{(i, j), \forall i \in I, \forall j \in J\}$ and $b_{i(s)} = b_i$, $\forall i \in I$, $\forall s \in S$.

Step 2. Set x_{ij} to zero or one based on the information available at the node. Adjust the set $I \otimes J$ and $b_{i(s)}$, $\forall i \in I$, $\forall s \in S$ as follows: Remove all the elements related to job *j* for which x_{ij} have been specified a value of 0 or 1. Let $\xi_{ij} = \{(i, j) \mid x_{ij} = 1\}$. Then, let $b_{i(s)} = b_i - \sum_{(i,j') \in \xi_{ij}} d_{ij's}$, $\forall i \in I$, $\forall s \in S$.

Step 3. Find $(i, j) \in I \otimes J$ that minimizes

$$\left\{ \frac{c_{ij}}{\sum_{s \in S} p_s d_{ijs}} + \frac{\sum_{s \in S} p_s e_i \mathrm{Max}(0, d_{ijs} - b_{i(s)})}{\sum_{s \in S} p_s d_{ijs}} \right\}. \tag{3.9a}$$

Step 4. Break ties in the following order of preference:

i. increasing order of $\dfrac{\sum_{s \in S} p_s e_i \mathrm{Max}(0, d_{ijs} - b_{i(s)})}{\sum_{s \in S} p_s d_{ijs}}$, (3.9b)

ii. increasing order of $\sum_{s\in S} p_s e_i \text{Max}(0, d_{ijs} - b_{i(s)})$, (3.9c)

iii. increasing order of e_i,

iv. increasing order of $\text{Max}(0, d_{ijs} - b_{i(s)})$, (3.9d)

v. increasing order of $\frac{c_{ij}}{\sum_{s\in S} p_s d_{ijs}}$, (3.9e)

vi. increasing order of c_{ij}, and

vii. randomly.

Step 5. Set $x_{ij} = 1$, and $x_{i'j} = 0 \ \forall i' \in I \setminus i$. Adjust $b_{i(s)} = b_{i(s)} - d_{ijs}$, and remove all the elements related to job j from the set $I \otimes J$. If $I \otimes J = \phi$, go to step 6; otherwise, go to step 3.

Step 6. Find the values of $y_{is} \ \forall i \in I$, $\forall s \in S$ based on the values of $x_{ij} \ \forall i \in I, \forall j \in J$ that are fixed in step 1 through step 5.

Proposition 3.3. The greedy heuristic procedure **GH** guarantees a feasible solution to **SGAP2**, and consequently, its objective function value provides an upper bound for **SGAP2**.

Proof. Suppose that $\bar{x}_{ij}$, $\forall i \in I$, $\forall j \in J$ is the solution obtained by **GH**, but it is not feasible to **SGAP2**. First, note that, for a given solution $\bar{x}_{ij}$, $\forall i \in I$, $\forall j \in J$, corresponding values of y_{is}, denoted by $\bar{y}_{is} \ \forall i \in I$, $\forall s \in S$, can be obtained by using constraints (3.3g) in **SGAP2**, as mentioned in step 6 above. That is,

$$\bar{y}_{is} = \text{Max}(0, \sum_{j\in J} d_{ijs}\bar{x}_{ij} - b_i) \qquad \forall i \in I, \ \forall s \in S \tag{3.9f}$$

However, since $\bar{x}_{ij}$ is not feasible, it must violate constraints (3.3f) in **SGAP2**. Therefore, there must exist a $j' \in J$ such that either $\sum_{i\in I} \bar{x}_{ij'} = 0$ or $\sum_{i\in I} \bar{x}_{ij'} \geq 2$. However, $x_{ij'}$, $j' \in J$, is eventually set to 1 for only one *i* in accordance with steps 4 and 5, so $\sum_{i\in I} \bar{x}_{ij'} = 1 \ \forall j' \in J$. Therefore, constraint

set (3.3f) is also satisfied, which contradicts the assumption. Hence, a solution obtained by the greedy heuristic is always feasible to **SGAP2**, and, as a result, its objective function value provides an upper bound on the optimal value of **SGAP2**. □

3.4.3. An Example to Illustrate Procedure GH

As an illustration, consider a node for which $x_{11} = x_{22} = 1$, and $x_{33} = 0$. Then, in accordance with step 2 for this node, we have, $x_{i'1} = 0,\ \forall i' \neq 1$ and $x_{i'2} = 0,\ \forall i' \neq 2$. Then, the set $I \otimes J$, and the machine capacities are as follows:

$$I \otimes J = I \otimes J - \{(i,1),(i,2) \forall i \in I, and (3,3)\},$$
$$b_{1(s)} = b_1 - d_{11s} \quad \forall s \in S, \text{ and}$$
$$b_{2(s)} = b_2 - d_{22s} \quad \forall s \in S.$$

Assume that $(i, j) = (1,3)$, which we find from step 1 to step 4 at the first iteration. Then, in step 5, we set $x_{13} = 1$, and $x_{i'3} = 0,\ \forall i' \neq 1$, adjust $b_{1(s)} = b_{1(s)} - d_{13s}, \forall s \in S$, and remove all the elements related to job 3 from the set $I \otimes J$.

Assume that all the rest of the jobs are assigned to machine 2. Then, step 6 leads to: $x_{2j} = 1$, $\forall j = 4,..,n$, and $x_{i'j} = 0,\ \forall i' \neq 2, \forall j = 4,..,n$. Consequently,

$$b_{2(s)} = b_{2(s)} - d_{24s} - ... - d_{2ns}, \forall s \in S,$$
$$y_{1s} = \text{Max}\{0, d_{11s} + d_{13s} - b_1\}, \forall s \in S, \text{ and}$$
$$y_{2s} = \text{Max}\{0, d_{22s} + d_{24s} + \cdots + d_{2ns} - b_1\}, \forall s \in S.$$

Next, we present a procedure for implementing **BNPDST-SGAP** in concert with the advanced start method presented above. We designate the corresponding method as **BNPDSTKP-SGAP**.

3.4.4. Procedure BNPDSTKP-SGAP

This is depicted in Figure 3.5. Appropriate changes are made in the BNP method to accommodate the use of the greedy heuristic to obtain an advanced starting solution. After solving the relaxed problem at the current node, we skip Phase I, and, as a result, the steps related to this phase, namely, steps 2, 6, 7, and 8, are removed. Consequently, we incorporate the greedy heuristic into the BNP method in step 2-1, which is shown in green color, as follows.

- **Step 2-1 (KP):** After steps 1 and 1-1, go to step 2-1. Apply the greedy heuristic, and transfer the initial feasible solution to the first restricted master problem in Phase II.

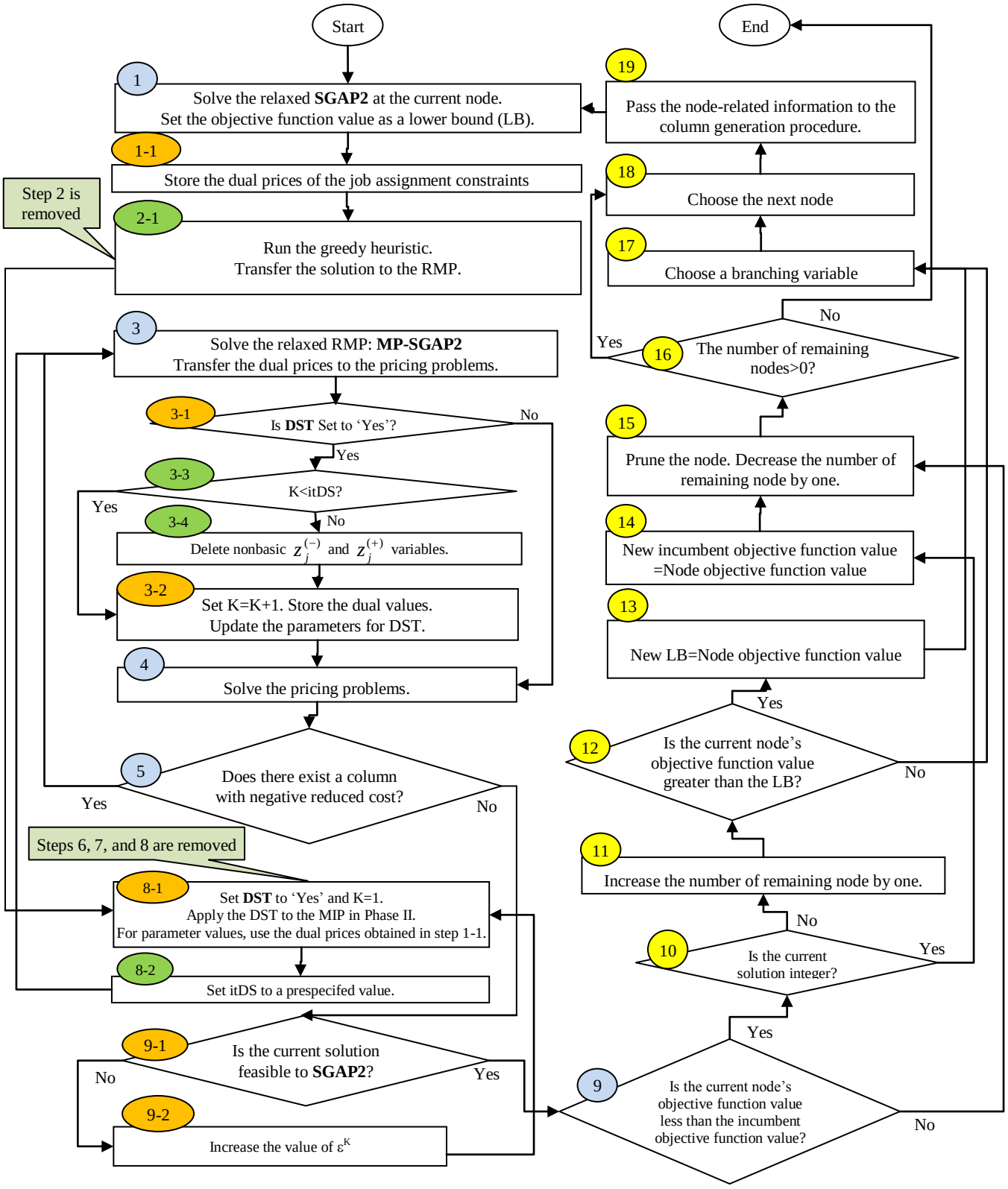

Figure 3.5. **BNPDST-SGAP** with the Greedy Heuristic Procedure **GH** (**BNPDSTKP-SGAP**)

3.5. BNPDSTKP-SGAP with the Monte Carlo Method (BNPDSTKP-SGAP with MCM)

3.5.1. Monte Carlo Method (MCM) for the SGAP

The MCM has been applied to the stochastic programming problems (SPs) to avoid enumeration of a large number of scenarios, which makes the SPs difficult to solve. The situation is not too different for the SGAP. The number of scenarios for the SGAP is

$$|S|=(\text{Number of Rates})^{\text{Number of Machines}}=|R|^{|I|}.$$

The numbers of variables and constraints are $|I|(|S|+|J|)$ and $|I||S|+|J|$, respectively. As an illustration, if we have 2 different speed factors (rates), the problem sizes for different values of $|I|$, $|J|$, and $|S|$ are shown in Table 3.4. Obviously, the size of the problem increases exponentially with increment in $|I|$, $|J|$, and $|S|$.

When we deal with 14 machines and 14 jobs, both the number of variables and

Table 3.4. Problem Sizes for Different $|I|$, $|J|$ and $|S|$ Values When $|R|=2$

$\|I\|$	$\|J\|$	$\|S\|$	Number of Variables	Number of Constraints
3	3	8	33	27
4	4	16	80	68
5	5	32	185	165
6	6	64	420	390
7	7	128	945	903
8	8	256	2,112	2,056
9	9	512	4,689	4,617
10	10	1,024	10,340	10,250
11	11	2,048	22,649	22,539
12	12	4,096	49,296	49,164
13	13	8,192	106,665	106,509
14	14	16,384	229,572	229,390

constraints are about a quarter of a million each. In fact, CPLEX cannot handle problems involving more than 13 machines and 13 jobs. Also, the branch-and-cut (BNC) method, the branch-and-price (BNP) method, the BNP method with dual stabilization technique (DST), and the BNP method with DST and heuristic algorithm are limited by the number of scenarios. The MCM is, therefore, a viable alternative for solving large problem instances. By dealing with only a subset of all the scenarios, it enables solution of large-size problems efficiently.

Let M and N be the number of replications and the number of independent random scenarios in each replication, respectively. We can express the formulation for the m^{th} replication of **SGAP2** as follows.

m^{th} Replication of SGAP2

$$\text{Minimize} \quad \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} + \frac{1}{N}\sum_{s\in S^m}\sum_{i\in I} e_i y_{is} \tag{3.10a}$$

$$\text{Subject to} \quad \sum_{i\in I} x_{ij} = 1 \qquad \forall j \in J \tag{3.10b}$$

$$\sum_{j\in J} d_{ijs}x_{ij} - y_{is} \le b_i \qquad \forall i \in I, \forall s \in S^m \tag{3.10c}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in I, \forall j \in J \tag{3.10d}$$

$$y_{is} \ge 0 \qquad \forall i \in I, \forall s \in S^m \tag{3.10e}$$

where $S^m = \{s_1^m, s_2^m, ..., s_N^m\}$ is the set of scenarios randomly generated for the m^{th} replication.

Proposition 3.4. The formulation ***m*th Replication of SGAP2** guarantees a feasible solution to the original formulation.

Proof. Let $\left(x_{ij}^*\right)_N^m$ and $\left(y_{is}^*\right)_N^m$ designate an optimal (and feasible) solution to the formulation ***m*th Replication of SGAP2**. Since (3.10b) and (3.3f) are identical, $\left(x_{ij}^*\right)_N^m$ satisfies (3.3f). All we need to show is if $\left(x_{ij}^*\right)_N^m$ and $\left(y_{is}^*\right)_N^m$ satisfy (3.3g), that is, if the following inequalities are true.

$$\sum_{j\in J} d_{ijs}\left(x_{ij}^{*}\right)_N^m - y_{is} \le b_i \qquad \forall i \in I,\ \forall s \in S \qquad (3.10f)$$

Separating (3.10f) into two groups, we have,

$$\sum_{j\in J} d_{ijs}\left(x_{ij}^{*}\right)_N^m - \left(y_{is}^{*}\right)_N^m \le b_i \qquad \forall i \in I,\ \forall s \in S^m \qquad (3.10g)$$

$$\sum_{j\in J} d_{ij\bar{s}}\left(x_{ij}^{*}\right)_N^m - y_{i\bar{s}} \le b_i \qquad \forall i \in I,\ \forall \bar{s} \in S \setminus S^m \qquad (3.10h)$$

The inequalities (3.10g) are true by (3.10c). Now, if $\sum_{j\in J} d_{ij\bar{s}}\left(x_{ij}^{*}\right)_N^m > b_i$, then $y_{i\bar{s}} = \sum_{j\in J} d_{ij\bar{s}}\left(x_{ij}^{*}\right)_N^m - b_i$; otherwise, $y_{i\bar{s}} = 0$. Thus, the values of $y_{i\bar{s}}$ are given by

$$y_{i\bar{s}}^{*} = \mathrm{Max}\left(0, \sum_{j\in J} d_{ij\bar{s}}\left(x_{ij}^{*}\right)_N^m - b_i\right) \qquad \forall i \in I,\ \forall \bar{s} \in S \setminus S^m, \qquad (3.10i)$$

which are feasible. Therefore, $\left\{\left(x_{ij}^{*}\right)_N^m, \left(y_{is}^{*}\right)_N^m, y_{i\bar{s}}^{*}\right\}$ is a feasible solution to the formulation **SGAP2**. □

3.5.2. Statistical Properties of the MCM for the SGAP

In view the properties presented in Section 2.3, we have the following statistical properties for the SGAP. First, we present the following additional notations that we use in the sequel.

$\hat{X}_N^m$: Vector of x_{ij} that is the optimal solution to the formulation ***m*th Replication of SGAP2**

$\hat{Y}_N^m$: Vector of y_{is} that is the optimal solution to the formulation ***m*th Replication of**

SGAP2

$\hat{x}^m_{N(i,j)}$: An element of $\hat{X}^m_N$, $\forall i \in I$, $\forall j \in J$

$\hat{y}^m_{N(i,s)}$: An element of $\hat{Y}^m_N$, $\forall i \in I$, $\forall s \in S^m$

X^* : Vector of x_{ij} that is the optimal to the formulation **SGAP2**

Y^* : Vector of y_{is} that is the optimal to the formulation **SGAP2**

h^m_N : Objective function value for the formulation ***m*th Replication of SGAP2**

h : Objective function value for the formulation **SGAP2**

$h^m_N(\hat{X}^m_N)$: Optimal objective function value for the formulation ***m*th Replication of SGAP2**

$h(\hat{X}^m_N)$: "True" objective function value corresponding to $\hat{X}^m_N$ for the formulation **SGAP2**

$h(X^*)$: "True" optimal objective function value for the formulation **SGAP2**

$\bar{h}_N$: Average of $h^1_N(\hat{X}^1_N), h^2_N(\hat{X}^2_N), \cdots, h^M_N(\hat{X}^M_N)$ determined by (2.16d)

$\mathrm{V}(\bar{h}_N)$: Variance of $h^1_N(\hat{X}^1_N), h^2_N(\hat{X}^2_N), \cdots, h^M_N(\hat{X}^M_N)$ determined by (2.16e)

$\tilde{N}$: Number of scenarios used to determine an unbiased estimate of the objective function value once a solution is found using N scenarios, $\tilde{N} >> N$

$h_{\tilde{N}}(\hat{X}^m_N)$: Unbiased estimate of $h(\hat{X}^m_N)$ determined by (2.16g)

$\mathrm{V}(h_{\tilde{N}}(\hat{X}^m_N))$: Variance of $h^m_{\tilde{n}}(\hat{X}^m_N)$, $\tilde{n} = 1,\ldots,\tilde{N}$, determined by (2.16h)

3.5.2.1. Optimal Objective Function Value of Each Replication

The optimal objective function value of the formulation ***m*th Replication of SGAP2**, $h^m_N(\hat{X}^m_N)$, is as follows:

$$h^m_N(\hat{X}^m_N) = \sum_{i\in I}\sum_{j\in J} c_{ij}\hat{x}^m_{N(i,j)} + \frac{1}{N}\sum_{s\in S^m}\sum_{i\in I} e_i \hat{y}^m_{N(i,s)} \qquad (3.11a)$$

Mak et al. [60] have shown that an unbiased estimate of a lower bound of the "true" optimal objective function value of a stochastic program can be obtained by its expected value. This is true for the formulation ***m*th Replication of SGAP2** as well, as we show next.

Proposition 3.5. The optimal objective function value of the formulation **mth Replication of SGAP2**, $h_N^m(\hat{X}_N^m)$, provides an unbiased estimate of a lower bound of the "true" optimal objective function value of **SGAP2**, $h(X^*)$.

Proof. Let X be the set of $\vec{x}$, which is the vector of x_{ij} satisfying constraints (3.3f) and (3.3h) in **SGAP2,** and also, $N = |S^m|$. We have,

$$h(X^*) = \operatorname{Min}_{\vec{x}\in X}\left(\vec{c}\vec{x} + \mathrm{E}[\tilde{f}(\vec{x})]\right) \tag{3.11b}$$

$$= \operatorname{Min}_{\vec{x}\in X}\left(\vec{c}\vec{x} + \mathrm{E}[\frac{1}{N}\sum_{s\in S^m} f_s(\vec{x})]\right) \tag{3.11c}$$

$$= \operatorname{Min}_{\vec{x}\in X}\left(\vec{c}\vec{x}\right) + \operatorname{Min}_{\vec{x}\in X}\left(\mathrm{E}[\frac{1}{N}\sum_{s\in S^m} f_s(\vec{x})]\right) \tag{3.11d}$$

$$\geq \operatorname{Min}_{\vec{x}\in X}\left(\vec{c}\vec{x}\right) + \mathrm{E}\left(\operatorname{Min}_{\vec{x}\in X}[\frac{1}{N}\sum_{s\in S^m} f_s(\vec{x})]\right) \tag{3.11e}$$

$$= \mathrm{E}\left(\operatorname{Min}_{\vec{x}\in X}\left(\vec{c}\vec{x}\right) + \operatorname{Min}_{\vec{x}\in X}[\frac{1}{N}\sum_{s\in S^m} f_s(\vec{x})]\right) \tag{3.11f}$$

$$= \mathrm{E}\left(\operatorname{Min}_{\vec{x}\in X}\left(\vec{c}\vec{x} + \frac{1}{N}\sum_{s\in S^m} f_s(\vec{x})\right)\right) \tag{3.11g}$$

$$= \mathrm{E}[h_N^m(\hat{X}_N^m)] \tag{3.11h}$$

Note that (3.11d) holds since the first term in the objective function is a constant, not a random variable, and we can separate its contribution from the objective function value, and (3.11e) holds because $\operatorname{Min}[\mathrm{E}(f_1(x)+\cdots+f_N(x))] \geq \mathrm{E}(\operatorname{Min}[f_1(x)+\cdots+f_N(x)])$. □

By the Law of Large Numbers, it is reasonable to assume that the bigger the sample size of scenarios that we have, the closer is the optimal objective function value of the replication to the "true" optimal objective function value (see Shapiro [84]). We, formally, show this is also true for our case, ***m*th Replication of SGAP2**.

Proposition 3.6. As N increases, $E[h_N^m(\hat{X}_N^m)]$, the expected value of the optimal objective function value of replications, approaches $h(X^*)$ from below.

Proof. We need to show that $E[h_{N+1}^m(\hat{X}_{N+1}^m)] \geq E[h_N^m(\hat{X}_N^m)]$ since by Proposition 3.5 $h_N^m(\hat{X}_N^m)$ is an unbiased estimate of a lower bound of $h(X^*)$.

$$\mathrm{E}\left[h_{N+1}^m(\hat{X}_{N+1}^m)\right]$$

$$= \mathrm{E}\left[\min_{\vec{x}\in X}\left(\vec{c}\vec{x} + \frac{1}{N+1}\sum_{s\in(S+1)^m} f_s(\vec{x})\right)\right] \tag{3.12a}$$

$$= \min_{\vec{x}\in X}\left(\sum_{i\in I}\sum_{j\in J}\vec{c}\vec{x}\right) + \mathrm{E}\left[\min_{\vec{x}\in X}\left(\frac{1}{N+1}\sum_{s\in(S+1)^m} f_s(\vec{x})\right)\right] \tag{3.12b}$$

$$= \min_{\vec{x}\in X}(\vec{c}\vec{x}) + \mathrm{E}\left[\min_{\vec{x}\in X}\left(\frac{1}{N+1}\frac{1}{N}N\{f_1(\vec{x})+\cdots+f_{N+1}(\vec{x})\}\right)\right] \tag{3.12c}$$

$$= \min_{\vec{x}\in X}(\vec{c}\vec{x}) + \mathrm{E}\left[\min_{\vec{x}\in X}\left(\frac{1}{N+1}\sum_{s\in(S+1)^m}\frac{1}{N}\sum_{s'\in(S+1)^m\setminus s} f_{s'}(\vec{x})\right)\right] \tag{3.12d}$$

$$= \min_{\vec{x}\in X}(\vec{c}\vec{x}) + \frac{1}{N(N+1)}\mathrm{E}\left[\min_{\vec{x}\in X}\left(\sum_{s\in(S+1)^m}\sum_{s'\in(S+1)^m\setminus s} f_{s'}(\vec{x})\right)\right] \tag{3.12e}$$

$$= \min_{\vec{x}\in X}(\vec{c}\vec{x}) + \frac{1}{N(N+1)}\mathrm{E}\left[\min_{\vec{x}\in X}\left(\sum_{s'\in(S+1)^m\setminus s_1^m} f_{s'}(\vec{x}) + \sum_{s'\in(S+1)^m\setminus s_2^m} f_{s'}(\vec{x}) + \cdots + \sum_{s'\in(S+1)^m\setminus s_{S+1}^m} f_{s'}(\vec{x})\right)\right] \tag{3.12f}$$

$$\geq \min_{\vec{x}\in X}(\vec{c}\vec{x}) + \frac{1}{N(N+1)}\mathrm{E}\left[\min_{\vec{x}\in X}\left(\sum_{s'\in(S+1)^m\setminus s_1^m} f_{s'}(\vec{x})\right) + \min_{\vec{x}\in X}\left(\sum_{s'\in(S+1)^m\setminus s_2^m} f_{s'}(\vec{x})\right) + \cdots + \min_{\vec{x}\in X}\left(\sum_{s'\in(S+1)^m\setminus s_{S+1}^m} f_{s'}(\vec{x})\right)\right] \tag{3.12g}$$

$$= \underset{\vec{x}\in X}{\text{Min}}(\vec{c}\vec{x}) + \frac{1}{N(N+1)}$$
$$\left\{ \text{E}\left[\underset{\vec{x}\in X}{\text{Min}} \left(\sum_{s'\in(S+1)^m \setminus s_1^m} f_{s'}(\vec{x}) \right) \right] + \text{E}\left[\underset{\vec{x}\in X}{\text{Min}} \left(\sum_{s'\in(S+1)^m \setminus s_2^m} f_{s'}(\vec{x}) \right) \right] + \cdots + \text{E}\left[\underset{\vec{x}\in X}{\text{Min}} \left(\sum_{s'\in(S+1)^m \setminus s_{S+1}^m} f_{s'}(\vec{x}) \right) \right] \right\} \quad (3.12h)$$

$$= \underset{\vec{x}\in X}{\text{Min}}(\vec{c}\vec{x}) + \frac{1}{N(N+1)} \sum_{s\in(S+1)^m} \text{E}\left[\underset{\vec{x}\in X}{\text{Min}} \left(\sum_{s'\in(S+1)^m \setminus s} f_{s'}(\vec{x}) \right) \right] \quad (3.12i)$$

$$= \underset{\vec{x}\in X}{\text{Min}}(\vec{c}\vec{x}) + \frac{1}{N+1} \sum_{s\in(S+1)^m} \text{E}\left[\underset{\vec{x}\in X}{\text{Min}} \left(\frac{1}{N} \sum_{s'\in(S+1)^m \setminus s} f_{s'}(\vec{x}) \right) \right] \quad (3.12j)$$

$$= \underset{\vec{x}\in X}{\text{Min}}(\vec{c}\vec{x}) + \text{E}\left[\underset{\vec{x}\in X}{\text{Min}} \left(\frac{1}{N} \sum_{s\in S^m} f_s(\vec{x}) \right) \right] \quad (3.12k)$$

$$= \text{E}\left[\underset{\vec{x}\in X}{\text{Min}} \left(\vec{c}\vec{x} + \frac{1}{N} \sum_{s\in S^m} f_s(\vec{x}) \right) \right] \quad (3.12l)$$

$$= \text{E}\left[h_N^m \left(\hat{X}_N^m \right) \right] \quad (3.12m)$$

where $(S+1)^m$ is a set of i.i.d. sample of scenarios of size N+1, and s_l^m is the l^{th} -chosen scenario for the m^{th} replication. □

Furthermore, we can show the following convergence result for **SGAP2**. A generalization of such a result has been shown by Mak et al [60] and Shapiro ([84] and [85]).

Proposition 3.7. As N approaches infinity, the following statements are true with probability one:

(1) The optimal objective function value of the formulation ***m*th Replication of SGAP2**, $h_N^m(\hat{X}_N^m)$, converges to the "true" optimal objective function value of **SGAP2**, $h(X^*)$.

(2) The objective function value of any solution, say X_N^m, to the formulation ***m*th Replication of SGAP2** converges to the objective function value of **SGAP2**, i.e., $h_N^m(X_N^m)$ converges to $h(X_N^m)$.

(3) The optimal solution to the formulation ***m*th Replication of SGAP2**, $\hat{X}_N^m$, converges to the optimal solution to **SGAP2**, X^*, if it is unique.

Proof.

(1) Since scenarios in $S^m(\subset S)$ are i.i.d., the samples $f_s(\vec{x})$, $s \in S^m$ from $\tilde{f}(\vec{x})$ are i.i.d. The Law of Large Numbers alleges that if a size of a sample of a random variable that are i.i.d. is big enough, the average of the sample, with probability 1, converges to the expected value of the random variable that is finite. Therefore, if $|S^m|$ approaches a sufficiently large number, i.e., N tends to infinity, $\frac{1}{N}\sum_{s\in S^m} f_s(\vec{x})$ converges to $\mathrm{E}[\tilde{f}(\vec{x})]$, which implies that $h_N^m(\hat{X}_N^m)$ converges to $h(X^*)$.

(2) The arguments used in the proof of part (1) can be applied to this case as well where X_N^m replaces X^*.

(3) Based on (1) and (2), there exists an $\ddot{N}$, for which the following inequality holds true for any $\varepsilon > 0$,

$$|h_N^m(X_N^m) - h(X_N^m)| < \varepsilon \qquad \forall N \geq \ddot{N} \tag{3.13a}$$

In order to prove the third statement, we need to show that the following two inequalities hold true for any $N \geq \ddot{N}$ and any $\varepsilon > 0$:

$$|h_N^m(\hat{X}_N^m) - h_N^m(X^*)| < \varepsilon \tag{3.13b}$$

$$|h(\hat{X}_N^m) - h(X^*)| < \varepsilon \tag{3.13c}$$

By (3.13b) and (3.13c), both $\hat{X}_N^m$ and X^* are the optimal solutions to h_N^m and h, respectively, for any $N \geq \ddot{N}$. Hence, if the optimal solution is unique, $\hat{X}_N^m$ converges to X^*. We have

$$|h_N^m(\hat{X}_N^m) - h_N^m(X^*)| = h_N^m(X^*) - h_N^m(\hat{X}_N^m) \tag{3.13d}$$

$$\leq h_N^m\left(X^*\right) - h_N^m\left(\hat{X}_N^m\right) + h\left(\hat{X}_N^m\right) - h\left(X^*\right) \tag{3.13e}$$

$$= \left| h_N^m\left(X^*\right) - h\left(X^*\right)\right| + \left| h\left(\hat{X}_N^m\right) - h_N^m\left(\hat{X}_N^m\right)\right| < 2\varepsilon \tag{3.13f}$$

Equation (3.13d) holds true as $\hat{X}_N^m$ denotes the optimal solution to h_N^m, i.e., $h_N^m\left(X^*\right) - h_N^m\left(\hat{X}_N^m\right) \geq 0$. Since X^* is the optimal solution to h, $h\left(\hat{X}_N^m\right) - h\left(X^*\right) \geq 0$. Therefore, (3.13e) holds true. Equation (3.13f) holds true since $h_N^m\left(X^*\right) - h\left(X^*\right)$ and $h\left(\hat{X}_N^m\right) - h_N^m\left(\hat{X}_N^m\right)$ are nonnegative for any $N \geq \ddot{N}$ by (1), and both terms are less than $\varepsilon > 0$ by (3.13a). We have,

$$\left| h\left(\hat{X}_N^m\right) - h\left(X^*\right)\right| = h\left(\hat{X}_N^m\right) - h\left(X^*\right) \tag{3.13g}$$

$$\leq h\left(\hat{X}_N^m\right) - h\left(X^*\right) + h_N^m\left(X^*\right) - h_N^m\left(\hat{X}_N^m\right) \tag{3.13h}$$

$$= \left| h_N^m\left(X^*\right) - h\left(X^*\right)\right| + \left| h\left(\hat{X}_N^m\right) - h_N^m\left(\hat{X}_N^m\right)\right| < 2\varepsilon \tag{3.13i}$$

Equation (3.13g) is satisfied since X^* is the optimal solution to h, and (3.13h) holds true as $\hat{X}_N^m$ is the optimal solution to h_N^m, i.e., $h_N^m\left(X^*\right) - h_N^m\left(\hat{X}_N^m\right) \geq 0$. Equation (3.13i) holds for the same reason for which Equation (3.13f) is true. □

Lamiri et al. [53] have studied a stochastic operating room scheduling problem and have provided similar relationships between the optimal objective function value (and solution) of each replication, the "true" objective function value (and solution), and the number of scenarios in each replication for that problem.

3.5.2.2. A Lower Bound of the "True" Optimal Objective Function Value

As used in Mak et al. [60], the average of $h_N^m\left(\hat{X}_N^m\right),\ m=1,...,M$, $\bar{h}_N$, is given by

$$\bar{h}_N = \frac{1}{M}\sum_{m=1}^{M} h_N^m\left(\hat{X}_N^m\right) \tag{3.14a}$$

Remark 3.4. By Proposition 3.7, as $N\to\infty$, $h_N^m\left(\hat{X}_N^m\right),\ \forall m=1,...,M$, converge to $h\left(X^*\right)$. Therefore, regardless of the number of replications, $\bar{h}_N \to h\left(X^*\right)$ as $N\to\infty$.

The following proposition extends Mak et al. [60]'s work and Proposition 3.5 to the increment in the number of replications (*M*).

Proposition 3.8. As *M* approaches infinity, $\bar{h}_N$ provides an unbiased estimate of a lower bound of the "true" optimal objective function value of **SGAP2**, $h\left(X^*\right)$.

Proof. The determination of $h_N^m(\hat{X}_N^m)$, $m=1,...,M$, relies on the generated scenarios in S^m, which are i.i.d. Each replication is independent of the others, so $\left\{h_N^m\left(\hat{X}_N^m\right)^*, m=1,...,M\right\}$ is an i.i.d. sample. By the Law of Large Number, as $M\to\infty$, then $\bar{h}_N$, the average of the sample, with probability 1, converges to the expected value of the random variable if it is finite, i.e., $\bar{h}_N = \mathrm{E}[h_N^m\left(\hat{X}_N^m\right)]$, and by Proposition 3.5, we have

$$h\left(X^*\right) \geq \mathrm{E}[h_N^m\left(\hat{X}_N^m\right)] = \bar{h}_N\ .\ \square \tag{3.14b}$$

3.5.2.3. An Upper Bound of the "True" Objective Function Value of $\hat{X}_N^m$

The "true" objective function value of $\hat{X}_N^m$, $h\left(\hat{X}_N^m\right)$, is an upper bound of h , i.e., $h\left(\hat{X}_N^m\right)\geq h$, and the equality holds true when $\hat{X}_N^m$ is an optimal solution. Given $\hat{X}_N^m$, the true value of $h\left(\hat{X}_N^m\right)$ is

obtained by considering all the scenarios. However, it is not easy to do so for a large-size problem. However, we can determine its unbiased estimate as follows, as long as $\tilde{N} >> N$.

$$h_{\tilde{N}}^{m}\left(\hat{X}_{N}^{m}\right)=\sum_{i\in I}\sum_{j\in J}c_{ij}\hat{x}_{N(i,j)}^{m}+\frac{1}{\tilde{N}}\sum_{\tilde{n}=1}^{\tilde{N}}\sum_{i\in I}e_{i}y_{\tilde{N}(i,\tilde{s}_{\tilde{n}}^{m})}^{m} \tag{3.15a}$$

where

$\tilde{S}^{m}=\left\{\tilde{s}_{1}^{m},\tilde{s}_{2}^{m},...,\tilde{s}_{\tilde{N}}^{m}\right\}$ is the set of $\tilde{N}$ random scenarios, and

$$y_{\tilde{N}(i,\tilde{s}_{\tilde{n}}^{m})}^{m}=\mathrm{Max}\left\{0,\sum_{j}d_{ij\tilde{s}_{\tilde{n}}^{m}}\hat{x}_{N(i,j)}^{m}-b_{i}\right\},\forall i\in I,\forall\tilde{s}_{\tilde{n}}^{m}\in\tilde{S}^{m} \tag{3.15b}$$

The following result supports the claim that $h_{\tilde{N}}^{m}\left(\hat{X}_{N}^{m}\right)$ is an unbiased estimate of an upper bound of $h\left(X^{*}\right)$ for the formulation **SGAP2**, which is generalized in Mak et al. [60] and Shapiro ([84] and [85]).

Proposition 3.9. As $\tilde{N}$ approaches infinity, $h_{\tilde{N}}^{m}\left(\hat{X}_{N}^{m}\right)$ converges to $h\left(\hat{X}_{N}^{m}\right)$, the objective function value of $\hat{X}_{N}^{m}$ in **SGAP2**, from above with probability one.

Proof. Note that $h_{\tilde{N}}^{m}\left(\hat{X}_{N}^{m}\right)=C\hat{X}_{N}^{m}+\frac{1}{\tilde{N}}\sum_{s\in\tilde{S}^{m}}f_{s}(\hat{X}_{N}^{m})$, where $\tilde{N}=|\tilde{S}^{m}|$. Because the scenarios in $\tilde{S}^{m}(\subset S)$ are i.i.d, $\left\{f_{s}(\hat{X}_{N}^{m}),s\in\tilde{S}^{m}\right\}$ are samples from $\tilde{f}(\hat{X}_{N}^{m})$ that are i.i.d. By the Law of Large Numbers, as $\tilde{N}\to\infty$, $\frac{1}{\tilde{N}}\sum_{s\in\tilde{S}^{m}}f_{s}(\hat{X}_{N}^{m})$ converges to $\mathrm{E}[\tilde{f}(\hat{X}_{N}^{m})]$, which implies that $h_{\tilde{N}}^{m}\left(\hat{X}_{N}^{m}\right)$ approaches $h\left(\hat{X}_{N}^{m}\right)$. That is,

$$\lim_{\tilde{N}\to\infty}h_{\tilde{N}}^{m}\left(\hat{X}_{N}^{m}\right)=\lim_{\tilde{N}\to\infty}\left(C\hat{X}_{N}^{m}+\frac{1}{\tilde{N}}\sum_{s\in\tilde{S}^{m}}f_{s}(\hat{X}_{N}^{m})\right)\equiv C\hat{X}_{N}^{m}+\mathrm{E}[\tilde{f}(\hat{X}_{N}^{m})]=h\left(\hat{X}_{N}^{m}\right).\ \square \tag{3.15c}$$

Remark 3.5. By Proposition 3.7, as $N\to\infty$, $\hat{X}_{N}^{m}$, $\forall m=1,...,M$ converge to X^{*}. Combining it with Proposition 3.9 guarantees that as $N\to\infty$, $h_{\tilde{N}}^{m}\left(\hat{X}_{N}^{m}\right)$ converges to $h\left(X^{*}\right)$ since $\tilde{N} >> N$.

3.5.2.4. Variance of $h_{\tilde{N}}^{m}\left(\hat{X}_{N}^{m}\right)$

As used in Mak et al. [60] and Verweij et al. [94], the expression to determine the variance of $h_{\tilde{N}}^{m}\left(\hat{X}_{N}^{m}\right)$ in the formulation **SGAP2** is as follows:

$$\begin{aligned} \mathrm{V}[h_{\tilde{N}}^{m}\left(\hat{X}_{N}^{m}\right)] &= \frac{1}{\tilde{N}(\tilde{N}-1)}\sum_{\tilde{n}=1}^{\tilde{N}}\left[\begin{array}{l}\sum_{i\in I}\sum_{j\in J}c_{ij}\hat{x}_{N(i,j)}^{m}+\sum_{i\in I}e_{i}y_{\tilde{N}(i,\tilde{s}_{\tilde{n}}^{m})}^{m} \\ -\sum_{i\in I}\sum_{j\in J}c_{ij}\hat{x}_{N(i,j)}^{m}-\frac{1}{\tilde{N}}\sum_{s\in\tilde{S}_{m}}\sum_{i\in I}e_{i}y_{\tilde{N}(i,\tilde{s}_{\tilde{n}}^{m})}^{m}\end{array}\right]^{2} \\ &= \frac{1}{\tilde{N}(\tilde{N}-1)}\sum_{\tilde{n}=1}^{\tilde{N}}\left[\sum_{i\in I}e_{i}y_{\tilde{N}(i,\tilde{s}_{\tilde{n}}^{m})}^{m}-\frac{1}{\tilde{N}}\sum_{s\in\tilde{S}^{m}}\sum_{i\in I}e_{i}y_{\tilde{N}(i,\tilde{s}_{\tilde{n}}^{m})}^{m}\right]^{2} \end{aligned} \tag{3.16a}$$

3.5.2.5. Candidate for "True" Optimal Solution

We choose a candidate for the "true" optimal solution from the set of optimal solutions to each replication, as follows (see Verweij et al. [94]):

$$\left(\hat{X}_{N}\right)^{*} \equiv \left[\left(\hat{X}_{N}\right)^{*} \in \left\{\hat{X}_{N}^{m}, m=1,...,M\right\}: h_{\tilde{N}}\left(\left(\hat{X}_{N}\right)^{*}\right) \le h_{\tilde{N}}^{m}\left(\hat{X}_{N}^{m}\right), \forall m=1,...,M\right] \tag{3.16b}$$

The following remark extends Mak et al. [60] and Proposition 3.7 to the increment in the number of scenarios (N).

Remark 3.6. By Proposition 3.7, as $N\to\infty$, $\hat{X}_{N}^{m}$, $\forall m=1,...,M$, converge to optimal solution. Therefore, as $N\to\infty$, $\left(\hat{X}_{N}\right)^{*}\to X^{*}$, and, accordingly, $h\left(\left(\hat{X}_{N}\right)^{*}\right)\to h\left(X^{*}\right)$.

3.5.2.6. Unbiased Estimate of Optimality Gap

The optimality gap represents the distance of an objective function value from the "true" optimal objective function value. The lesser the gap, the better is the solution quality. We can estimate a

lower bound and an upper bound from (3.14a) and (3.15a), respectively, and $h(X^*)$ lies between these two values. Thus, it is reasonable to use them for calculating an unbiased estimate of the optimality gap as follows (see Verweij et al. [94]):

$$\mathrm{E}[h\left(\left(\hat{X}_N\right)^*\right) - h(X^*)] = h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right) - \bar{h}_N \tag{3.17}$$

The following remark incorporates Propositions 3.8 and 3.9 and Remarks 3.4 and 3.5 in order to see the effect of the increment in the number of scenario (N).

Remark 3.7. Note that by Propositions 3.8 and 3.9, $h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right) - \bar{h}_N$ is an unbiased estimate of the gap between lower and upper bounds. Furthermore, by Remarks 3.4 and 3.5, as $N \to \infty$, $h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right) - \bar{h}_N \to 0$.

3.5.2.7. Unbiased Estimate of Variance of Optimality Gap

Considering Equation (3.17), as generalized in Mak et al. [60], an unbiased estimate of the variance of the optimality gap for **SGAP2** is given by

$$\mathrm{V}[h\left(\left(\hat{X}_N\right)^*\right) - h(X^*)] = \mathrm{V}[h\left(\left(\hat{X}_N\right)^*\right)] + \mathrm{V}[h(X^*)] \tag{3.18a}$$

where

$$\mathrm{V}[\bar{h}_N] = \frac{1}{M(M-1)}\sum_{m=1}^{M}\left[h_N^m\left(\left(\hat{X}_N^m\right)\right) - \frac{1}{M}\sum_{m=1}^{M}h_N^m\left(\left(\hat{X}_N^m\right)\right)\right]^2 = \frac{1}{M(M-1)}\sum_{m=1}^{M}\left[h_N^m\left(\left(\hat{X}_N^m\right)\right) - \bar{h}_N\right]^2 \tag{3.18b}$$

Remark 3.8. By Proposition 3.7 and Remark 3.4, both $h_N^m\left(\left(\hat{X}_N^m\right)\right)$, $\forall m = 1,\ldots,M$, and $\bar{h}_N$ converge to $h(X^*)$, i.e., $h_N^m\left(\left(\hat{X}_N^m\right)\right) - \bar{h}_N \to 0 \ \ \forall m = 1,\ldots,M$ as $N \to \infty$. Therefore, $\mathrm{V}[\bar{h}_N] \to 0$ as $N \to \infty$.

3.5.2.8. Estimation of 100(1-α) Percent Confidence Interval

The quality of the best solution, $\left(\hat{X}_N\right)^*$, is evaluated by an estimation of confidence interval of the optimality gap for the solution. There are two possible ways to calculate 100(1-α) percent confidence interval as follows (see Shapiro [84] and Verweij et al [94]):

$$\left[0, h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right) - \bar{h}_N + z_{1-\alpha}\sqrt{\mathrm{V}\left[h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)\right] + \mathrm{V}\left[\bar{h}_N\right]}\right] \tag{3.19a}$$

$$\left[0, \mathrm{Max}\left[0, h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right) - \bar{h}_N\right] + z_{1-\alpha}\sqrt{\mathrm{V}\left[h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)\right] + \mathrm{V}\left[\bar{h}_N\right]}\right] \tag{3.19b}$$

Note that the confidence interval in (3.19b) is at most as good as the one in (3.19a) since $\mathrm{Max}[0, a-b] \geq a-b$. Since the optimality gap represents the difference between upper and lower bounds, it is supposed to be nonnegative. Therefore, we use (3.19b) even though (3.19b) produces more conservative results. We drop the entire set of replications if $h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right) < \bar{h}_N$, and repeat the process.

Remark 3.9. As $N \to \infty$, $h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right) - \bar{h}_N \to 0$ by Remark 3.7, and $\mathrm{V}[\bar{h}_N] \to 0$ by Remark 3.8. Therefore, both (3.19a) and (3.19b) are equivalent to (3.19c) as follows:

$$\left[0, z_{1-\alpha}\sqrt{\mathrm{V}\left[h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)\right]}\right] \tag{3.19c}$$

3.5.2.9. An Unbiased Estimate of the Ratio of Optimality Gap and Optimal Objective Function Value

The stopping criterion that we use for the **BNPDSTKP-SGAP with the MCM** method is the ratio of the optimality gap over the "true" optimal objective function value, that is, $\frac{h\left(\left(\hat{X}_N\right)^*\right)-h}{h}$ to be less than a pre-specified tolerance value, denoted by ε. Since the unbiased estimates of h and $h\left(\left(\hat{X}_N\right)^*\right)-h$ are given by (3.14a), (3.15a), and (3.16b), respectively, the unbiased estimate of $\frac{h\left(\left(\hat{X}_N\right)^*\right)-h}{h}$ is as follows:

$$\mathrm{E}\left[\frac{h\left(\left(\hat{X}_N\right)^*\right)-h}{h}\right]=\mathrm{E}\left[\frac{h\left(\left(\hat{X}_N\right)^*\right)}{h}-1\right]=\frac{\mathrm{E}\left[h\left(\left(\hat{X}_N\right)^*\right)\right]}{\mathrm{E}[h]}-1=\frac{h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)}{\bar{h}_N}-1 \tag{3.20}$$

Remark 3.10. By Remark 3.7, as $N\to\infty$, $\frac{h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)}{\bar{h}_N}$ converges to one with probability one, i.e., the stopping criterion will be satisfied.

3.5.3. An Exterior Sampling Method for the SGAP

Our algorithm is an adaptation of the methods suggested by Kleywegt et al. [51], Shapiro ([84] and [85]) and Verweij et al. [94]. We introduce a parameter $\varepsilon\,(0<\varepsilon<1)$ as a tolerance in step 11 to determine if $\left(\hat{X}_N\right)^*$ is an acceptable value. The following steps describe the implementation of the MCM for the solution of the SGAP.

Step 1. Specify the values of *N*, $\tilde{N}$, and *M*.

Step 2. Generate $S^m=\left\{s_1^m,s_2^m,...,s_N^m\right\}$ with a random number generator.

Step 3. Build the formulation ***m*th Replication of SGAP2** with S^m.

Step 4. Employ the **BNPDSTKP-SGAP** method presented in Section 3.4, respectively, to solve the formulation ***m*th Replication of SGAP2**.

Step 5. Store $\hat{X}_N^m$ and h_N^m.

Step 6. Generate $\tilde{S}^m = \{\tilde{s}_1^m, \tilde{s}_2^m, ..., \tilde{s}_{\tilde{N}}^m\}$.

Step 7. Calculate $h_{\tilde{N}}^m(\hat{X}_N^m)$ and $\mathrm{V}(h_{\tilde{N}}(\hat{X}_N^m))$.

Step 8. If $m = M$, go to step 9; otherwise, set $m = m+1$, and go to step 2.

Step 9. Estimate $\bar{h}_N$ and $\mathrm{V}(\bar{h}_N)$.

Step 10. Determine $(\hat{X}_N)^*$, and estimate $\mathrm{E}\left[h\left((\hat{X}_N)^*\right) - h\right]$ and $\mathrm{V}\left[h_{\tilde{N}}\left((\hat{X}_N)^*\right) - \bar{h}_N\right]$.

Step 11. If $\dfrac{h_{\tilde{N}}\left((\hat{X}_N)^*\right)}{\bar{h}_N} \le 1+\varepsilon$, go to step 12; otherwise, increase the values of *N*, $\tilde{N}$, and *M*, and go to step 2.

Step 12. Estimate $100(1-\alpha)$ percent confidence interval, and stop.

3.5.4. Procedure for BNPDSTKP-SGAP with MCM

Figure 3.6 depicts **BNPDSTKP-SGAP with MCM**. The additional steps introduced as a result of including the MCM in the branch-and-price algorithm in the presence of dual stabilization technique and the heuristic method for the SGAP are shown in purple color. The changes resulting as a result of including the MCM are as follows.

- **Steps 0-1, 0-2, 0-3, 0-4 (MCM):** The BNP method for SGAP is initialized by providing the following information regarding the MCM: the number of replications (M) at each trial, the sample sizes of scenarios $(N, \tilde{N})$ for each replication, and the formulation ***m*th Replication of SGAP2** at the corresponding replication. Go to step 1.
- **Steps 1 to 19 (BNP, DST, KP):** Perform the BNPDSTKP-SGAP method to solve the problem.

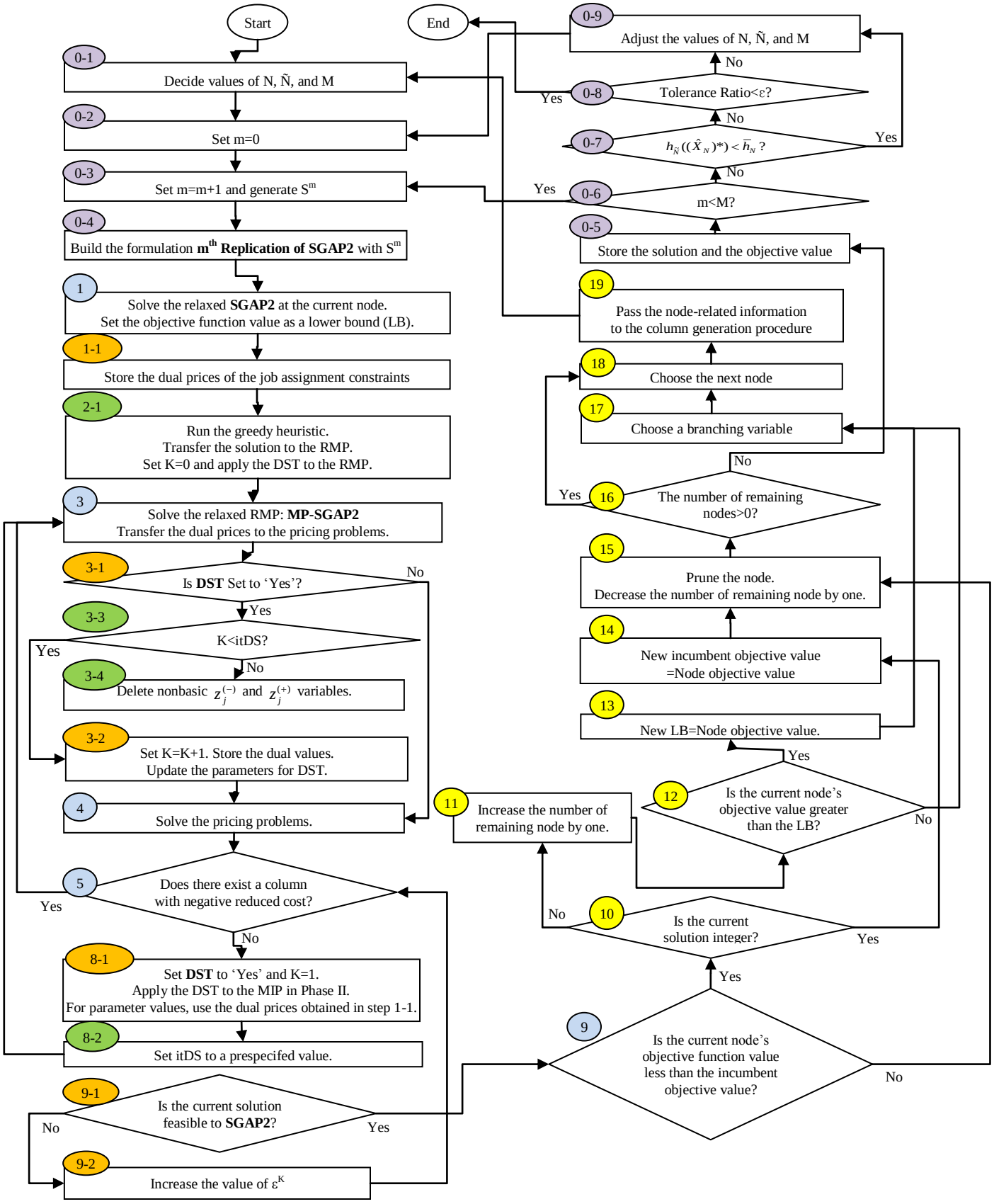

Figure 3.6. **BNPDSTKP-SGAP** with Monte Carlo Method (**BNPDSTKP-SGAP with MCM)**

- **Step 0-5 (MCM):** Store the solution and the objective function value for later computations of the average, variance, and confidence interval of the objective function value.
- **Step 0-6 (MCM):** If the number of replications is less than the value specified either in step 0-1 or in step 0-9, go to step 0-7; otherwise, return to step 0-3.
- **Step 0-7 (MCM):** If the best objective function value is less than the average of the objective function values for each replication, go to step 0-9; otherwise, go to step 0-8.
- **Step 0-8 (MCM):** If the stopping criterion is satisfied, then end the process after reporting the statistical properties on the results; otherwise, go to step 0-9.
- **Step 0-9 (MCM):** Adjust the values of N, $\tilde{N}$, and M, and go to step 0-1.

3.5.5. An Example to Illustrate the Use of BNPDSTKP-SGAP with MCM

We present a simple example to illustrate the above procedure. Let $|I|=2$, $|J|=2$, and $|S|=3$. The underlying formulation is as follows:

Example 3.2. SGAP with MCM

Minimize $c_{11}x_{11}+c_{12}x_{12}+c_{21}x_{21}+c_{22}x_{22}+p_1e_1y_{11}+p_1e_2y_{21}+p_2e_1y_{12}+p_2e_2y_{22}+p_3e_1y_{13}+p_3e_2y_{23}$

Subject to $x_{11}+x_{21}=1;\quad x_{12}+x_{22}=1$

$d_{111}x_{11}+d_{121}x_{12}-y_{11}\le b_1;\quad d_{112}x_{11}+d_{122}x_{12}-y_{12}\le b_1;\quad d_{113}x_{11}+d_{123}x_{12}-y_{13}\le b_1$

$d_{211}x_{21}+d_{221}x_{22}-y_{21}\le b_2;\ d_{212}x_{21}+d_{222}x_{22}-y_{22}\le b_2;\ d_{213}x_{21}+d_{223}x_{22}-y_{23}\le b_2$

Step 1. Let $M=1$, $N=2$, and $\tilde{N}=2$.

Step 2. Suppose $S^1=\{1,2\}$.

Step 3. The formulation **1st Replication of SGAP2** is as follows:

Minimize $c_{11}x_{11}+c_{12}x_{12}+c_{21}x_{21}+c_{22}x_{22}+\frac{1}{2}(e_1y_{11}+e_2y_{21}+e_1y_{12}+e_2y_{22})$

Subject to $x_{11}+x_{21}=1;\qquad x_{12}+x_{22}=1$

$d_{111}x_{11}+d_{121}x_{12}-y_{11}\le b_1;\qquad d_{112}x_{11}+d_{122}x_{12}-y_{12}\le b_1$

$d_{211}x_{21}+d_{221}x_{22}-y_{21}\le b_2;\qquad d_{212}x_{21}+d_{222}x_{22}-y_{22}\le b_2$

Step 4. Run the **BNPDSTKP-SGAP** method.

Step 5. Suppose

$\hat{X}_2^1=\left(\hat{x}_{2(1,1)}^1,\hat{x}_{2(1,2)}^1,\hat{x}_{2(2,1)}^1,\hat{x}_{2(2,2)}^1\right),\hat{Y}_2^1=\left(\hat{y}_{2(1,1)}^1,\hat{y}_{2(1,2)}^1,\hat{y}_{2(2,1)}^1,\hat{y}_{2(2,2)}^1\right)$, and consequently,

$$h_2^1=\sum_{i=1}^{2}\sum_{j=1}^{2}c_{ij}\hat{x}_{2(i,j)}^1+\frac{1}{2}\sum_{i=1}^{2}\sum_{s=1}^{2}e_i\hat{y}_{2(i,s)}^1\ .$$

Step 6. Since $|S|=\tilde{N}=3$, $\tilde{S}^1=\{1,2,3\}$.

Step 7. We have,

$$h_3^1\left(\hat{X}_2^1\right)=\sum_{i=1}^{2}\sum_{j=1}^{2}c_{ij}\hat{x}_{2(i,j)}^1+\frac{1}{3}\sum_{\tilde{n}=1}^{3}\sum_{i=1}^{2}e_i\max\left(0,\sum_{j=1}^{2}d_{ij\tilde{s}_{\tilde{n}}^1}\hat{x}_{2(i,j)}^1-b_1\right),\text{ and}$$

$$\mathrm{V}\left(h_3\left(\hat{X}_2^1\right)\right)=\frac{1}{6}\sum_{\tilde{n}=1}^{3}\sum_{i=1}^{2}e_i\left[\mathrm{Max}\left(0,\sum_{j=1}^{2}d_{ij\tilde{s}_{\tilde{n}}^1}\hat{x}_{2(i,j)}^1-b_1\right)-\frac{1}{3}\sum_{\tilde{n}=1}^{3}\mathrm{Max}\left(0,\sum_{j=1}^{2}d_{ij\tilde{s}_{\tilde{n}}^1}\hat{x}_{2(i,j)}^1-b_1\right)\right]^2.$$

Step 8. Go to step 9 since M=1.

Step 9. $\bar{h}_N=h_2^1$ and $Var\left(\bar{h}_N\right)=0$.

Step 10. Then, $\left(\hat{X}_N\right)^*=\hat{X}_2^1$, and

$$\mathrm{E}\left[h\left(\left(\hat{X}_N\right)^*\right)-h\right]=h_3\left(\hat{X}_2^1\right)-h_2^1,\text{ and }\mathrm{V}\left[h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)-\bar{h}_N\right]=\mathrm{V}\left[h_3\left(\hat{X}_2^1\right)\right].$$

Step 11. $\dfrac{h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)}{\bar{h}_N}=\dfrac{h_3\left(\hat{X}_2^1\right)}{h_2^1}$, and assume that it is less than $1+\varepsilon$. Go to step 12.

Step 12. $100(1-\alpha)$ percent confidence interval is as follows:

$$\left[0,\mathrm{Max}\left[0,h_3\left(\hat{X}_2^1\right)-h_2^1\right]+z_{1-\alpha}\sqrt{V\left[h_3\left(\hat{X}_2^1\right)\right]}\right]$$

3.6. Computational Experimentation

In this section, we present experimental investigation that we have conducted to compare the performances of the five algorithms developed in previous sections. These are:

- **BNC-SGAP**: Branch-and-Cut (BNC) method for the SGAP
- **BNP-SGAP**: Branch-and-Price (BNP) method for the SGAP
- **BNPDST-SGAP**: **BNP-SGAP** in concert with the dual stabilization technique (DST)
- **BNPDSTKP-SGAP**: **BNPDST-SGAP** with a fast start to obtain a feasible starting solution of the underlying knapsack problem using a greedy heuristic
- **BNPDSTKP-SGAP with MCM**: Monte Carlo method (MCM) applied to **BNPDSTKP-SGAP**

Note that the first four methods are optimum seeking methods, while the last one is a heuristic procedure. We compare the cpu times required by these methods as a function of the number of jobs, the number of machines, the capacities of machines, the processing time of each job on each machine, and the ratio of penalty costs to job assignment costs. CPLEX is well-known software, and it uses a variety of features to suitably structure the BNC method for a problem. It also relies on various preprocessing steps to find a good starting solution, applies various heuristic methods to save CPLEX from undertaking time-consuming steps, and utilizes a variety of cuts including:

- Clique Cuts,
- Cover Cuts,
- Disjunctive Cuts,
- Flow Cover Cuts,
- Flow Path Cuts,
- Gomory Fractional Cuts,
- Generalized Upper Bound (GUB) Cover Cuts,

- Implied Bound Cuts, and
- Mixed Integer Rounding (MIR) Cuts.

Even though CPLEX gives the user control over which cuts to use, our experimentation for the SGAP revealed that the best results are obtained when all the cuts are included. For the solution of the SGAP, therefore, we uses CPLEX as such. All the methods introduced here were implemented using CPLEX 9.0 and CONCERT 2.0 built in Java 1.5.0_07, and the software embedding Java is NetBeans IDE 5.0. All the runs were made on a computer containing Intel Pentium® 4 CPU 3.60 GHz, 1.00 GB of RAM.

3.6.1. Experimental Designs for BNC-SGAP, BNP-SGAP, BNPDST-SGAP, and BNPDSTKP-SGAP

The size of a problem in a stochastic programming problem is inherently affected by the number of scenarios. The number of scenarios for the SGAP depends on the number of machines ($|I|$) and the number of different machine rates ($|R|$), and it is given by $|R|^{|I|}$. We assume that there are only two machine rates, i.e., $|R|=2$, such as normal status and slow status. For instance, in an airline scheduling problem, it can represent a normal duration or an abnormal duration, where severe weather conditions could be one of the factors delaying arrivals and departures of the planes. Similarly, in a production scheduling problem, the efficiencies of machines can affect job processing times, and in an insurance service problem, the time to process a claim for an accident might depend on whether the agent is experienced or not. For each of the five methods, we vary the following elements in our experimentation:

- Number of machines
- Ratio of the number of machines to the number of jobs
- Capacities of the machines
- Penalty cost per extra hour at each machine

As noted above, the number of events ($|R|$) is set to 2. The number of machines, $|I|$, is varied from 7 to 11. The number of jobs, $|J|$, depends on the ratio of the number of machines to the number of

jobs, which is expressed as $|J|/|I|$. We change the ratio from 1, 1.5 to 2. For instance, when, $|I|=8$, the values of $|J|$ are 8, 12, and 16.

The change in capacities of the machines reflects upon the complexity of the problem. If the machines have enough capacities to process all the jobs, then we do not encounter overtime penalty. Hence, the SGAP reduces to a pure binary integer programming (PBIP) problem. Conversely, if the machines do not have enough capacity, then the penalty is incurred, and the SGAP becomes a mixed BIP (MBIP). Needless to say, the MBIP is more difficult to solve than the PBIP. Therefore, we anticipate the cpu time required for the MBIP to be greater than that for the PBIP. We also expect the size of the feasible region of the MBIP to be smaller than that for the PBIP if all the other parameters are kept the same, and as a result, the optimal objective function value of the MBIP to be greater than that of the PBIP.

Finally, we change the value of the penalty per extra hour from 10 to 1000. With a small penalty cost, a compromise will be sought between the penalty cost and the job processing cost since they are of identical magnitude. However, if the penalty cost is high, the tendency will be to delay use of extra hours as much as possible. We expect the former to be more computationally intensive than the latter since the second stage variables, y_{is}, for all $i \in I$, $s \in S$, will not come into play for the latter unless really required.

The complete experimental design for our experimentation to determine the performances of **BNC-SGAP**, **BNP-SGAP**, **BNPDST-SGAP**, and **BNPDSTKP-SGAP** is given in Tables A.1, A.2, A.3, A.4, and A.5, included in Appendix A.

3.6.2. Experimental Designs for BNPDSTKP-SGAP with MCM

In this section, we present the structure of experiments for **BNPDSTKP-SGAP with MCM**. The reason we separate them from the others is that **BNPDSTKP-SGAP with MCM** is not an optimum seeking method, which requires observance of additional properties. Besides the cpu time, we also need to compare the results of **BNPDSTKP-SGAP with MCM** with those of the optimum-seeking algorithms with regard to their degree of near-optimality, which measures how close the solution, $\left(\hat{X}_N\right)^*$, is to the "true" optimal solution found by other methods. Along with the optimal objective function values obtained by the optimum-seeking algorithms, we also need to keep track of the following parameters (see Section 3.5.2 for more details):

- Initial values of N, $\tilde{N}$, and M
- Final values of N, $\tilde{N}$, and M
- $\hat{X}_N^m$: The optimal solution to each replication, m=1,…,M
- h_N^m : The objective function value of each replication, m=1,…,M, where average value is used for the unbiased estimator of the lower bound of the "true" optimal objective function value, that is, h
- $h_{\tilde{N}}^m\left(\hat{X}_N^m\right)$: The unbiased estimators of h for each replication, m=1,…,M
- $h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)$: The best objective function value found from among all the replications, and an unbiased estimator of the upper bound of h, where $\left(\hat{X}_N\right)^*$ is the solution such that

$$\left(\hat{X}_N\right)^* \equiv \left[\left(\hat{X}_N\right)^* \in \left\{\hat{X}_N^m, m=1,...,M\right\}: h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right) \le h_{\tilde{N}}\left(\hat{X}_N^m\right), \forall m=1,...,M\right]$$

- $\mathrm{E}\left[\dfrac{h\left(\left(\hat{X}_N\right)^*\right)-h}{h}\right]=\dfrac{h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)}{\bar{h}_N}-1$: The tolerance ratio, where $\bar{h}_N$ is the average of

$$h_N^1, h_N^2, \cdots, h_N^M$$

In case a very small tolerance value (ε) is chosen, there will be a greater chance for the results of replications to be rejected. On the other hand, for a big value of ε, the MCM could be stopped too soon and with a poor-quality solution. Based on our primary investigation, we choose a tolerance value of 0.2. Therefore, if the optimal gap is greater than the average of the objective function values of replications by more than 20%, all the results of the replications are rejected; otherwise, the stopping criterion will be satisfied.

The following sections describe how the values of other parameters were varied in our experimentation.

3.6.2.1. Variation in the Values of *N*, $\tilde{N}$, and *M*

The ratio of the number of scenarios in a subset, *N*, to the total number of scenarios, |*S*|, is an important measure to evaluate the efficiency of the Monte Carlo method. The number of replications, *M*, also plays an important part because the total number of scenarios observed in an iteration is equal to *N***M*. The nearer the sample objective function, $h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)$, is to the "true" optimal objective function value, *h*, and the smaller is the value of the total number of scenarios used before a problem is terminated, the better is the performance of the algorithm. The cpu time tends to increase with increment in the values of *N*, $\tilde{N}$, and *M*, but, as proven in Section 3.5.2, the objective function value also tends to approach the optimum. Therefore, a tradeoff between the cpu time and the near-optimality of the solution is sought. We apply the MCM to two groups of problem instances as follows:

- The first group of instances is those for which the optimal solutions are known and those that are presented in Section 3.6.1. The tolerance value (ε) is set at 0.2, i.e., an experiment will be stopped only if the tolerance ratio of that experiment is less than 0.2. We consider the cases belonging to larger values of number of machines, |*I*|, namely, 10 and 11, for which the results are more discriminating.
- The second group consists of instances for which all of the optimum-seeking algorithms such as **BNC-SGAP**, **BNP-SGAP**, **BNPDST-SGAP**, and **BNPDSTKP-SGAP** fail to obtain optimal solutions.

As alluded to earlier, the values of *N*, $\tilde{N}$, and *M* are varied so as to observe improvements in the final values of the tolerance ratio and the cpu times for both groups of experiments. The near-optimality can only be observed for the problems in the first group since the optimal solutions are not known for those in the second group. The values of *N*, $\tilde{N}$, and *M* in the first group of instances are varied as follows:

- Starting values : (*N*, $\tilde{N}$, *M*)=(0.125|*S*|, 0.25|*S*|, 2).
- If the optimal gap of the r^{th} iteration fails to satisfy the tolerance level (ε), (*N*, $\tilde{N}$, *M*) are increased to (0.125|*S*|, 0.25|*S*|, 2+r) for the $(r+1)^{th}$ iteration.

Basically, for the experiments in the first group, we start with one eighth of the total number of scenarios randomly generated for the formulation ***m*th Replication of SGAP** to determine $\hat{X}_N^m$, and one fourth of the total number of scenarios to compute the objective function value $h_{\tilde{N}}^m\left(\hat{X}_N^m\right)$ for the $\hat{X}_N^m$ determined using N scenarios. The number of replications used is two. If the stopping criterion is not satisfied, instead of increasing N and $\tilde{N}$, we increase M by one every time this happens. On the other hand, the 100(1-α) percent confidence interval is observed for the problems in the second group in order to provide a trustworthy range for the best solution that the Monte Carlo method yields. When it fails to satisfy the stopping criterion, the problem is re-run with the value of M increased by one until the criterion is satisfied.

Finally, Appendix B provides information on the experimental designs for the first group in Tables B.1, B.2, B.3, B.4, and B.5.

3.6.2.2. Variations in $|I|$, $|J|$, e_i, and b_i

As alluded to earlier, in the first group of experiments, we consider the cases where $|I|$=10 and $|I|$=11. All other parameter values are the same as those of the experiments for the optimum-seeking algorithms described in Section 3.6.2. The ratios of $|J|$ to $|I|$ used are 1, 1.5, or 2, the values of unit penalty are 10, 100, or 1,000, and the machine capacity is varied from 3 to 27. In the second group, $|I|$=13, $|J|$=26, and the machine capacity is varied from 3 to 27.

3.7. Results and Analysis

3.7.1. Experimental Results for the BNC-SGAP, BNP-SGAP, BNPDST-SGAP, and BNPDSTKP-SGAP Methods When $|J|=|I|$ and $e_1=10$

Table 3.5 depicts the results for the experiments of Table A.1 in Appendix A. The first seven columns of the table contain, respectively, the experiment number, the number of machines, the number of jobs, the number of scenarios, the number of variables, the number of constraints, and the maximum capacity of machine 1. The optimal objective function value is displayed in the eighth column. Columns nine to twelve contain, respectively, the cpu times for **BNC-SGAP**, **BNP-SGAP**, **BNPDST-SGAP**, and **BNPDSTKP-SGAP**.

The cell with the best value for a given experiment is highlighted with yellow color in order to indicate the method that attains that value. For example, for experiment 1, **BNPDSTKP-SGAP** requires the least cpu time to solve that problem. Referring to Table 3.5, each method gives the best performance for 5, 2, 5, and 14 experiments, respectively. Also, note that all **BNP-SGAP** methods are successful in attaining optimal solutions for all the experiments while **BNC-SGAP** fails to do so for experiments 15, 16, 21, and 22 due to excessive memory requirements. Clearly, **BNPDSTKP-SGAP** performs the best among all the methods. Also, any of the BNP-based methods gives better results more often than the BNC-based method.

We also compare the cpu times for the following pairs to determine the incremental benefit of the additional feature.

- **BNC-SGAP** vs. **BNP-SGAP**
- **BNP-SGAP** vs. **BNPDST-SGAP**
- **BNPDST-SGAP** vs. **BNPDSTKP-SGAP**

Table 3.5. Computational Times for Different Methods When $|J|=|I|$ and $e_1=10$

#	$\|I\|$	$\|J\|$	$\|S\|$	Number of Variables	Number of Constraints	b_1	Optimal Objective Function Value	CPU Time (Seconds)			
								BNC-SGAP	**BNP-SGAP**	**BNPDST-SGAP**	**BNPDSTKP-SGAP**
1	7	7	128	945	903	3	323.5	30.25214397	0.718145681	0.703040506	0.348706591
2	7	7	128	945	903	6	181	0.230672707	0.229063767	0.224889637	0.266756552
3	7	7	128	945	903	9	113.5	0.256282395	0.786007189	0.768467948	0.281886032
4	7	7	128	945	903	12	91	0.085362655	0.068502921	0.069297836	0.129012935
5	8	8	256	2112	2056	3	423	638.7748425	1.986579954	1.964561209	1.012734655
6	8	8	256	2112	2056	6	250.5	1.212131725	1.167665905	1.154418277	1.235478142
7	8	8	256	2112	2056	9	153	0.444888452	0.404650854	0.422503108	0.468269097
8	8	8	256	2112	2056	12	115.5	0.391305294	1.896158739	1.891573195	0.472246781
9	8	8	256	2112	2056	15	108	0.205082481	0.174535518	0.174524761	0.246797801
10	9	9	512	4689	4617	3	536	19159.12445	7.403944927	6.829137704	3.42470312
11	9	9	512	4689	4617	6	333.5	7119.747438	8.361493413	7.944223719	3.092331576
12	9	9	512	4689	4617	9	206	81.27691594	8.27255646	8.189590138	2.466196243
13	9	9	512	4689	4617	12	148.5	3.639284012	6.059876915	5.99240164	1.717763047
14	9	9	512	4689	4617	15	126	0.651957741	0.71214667	0.559582406	0.718010264

Table 3.5. Computational Times for Different Methods When $|J|=|I|$ and $e_1=10$ (Continued)

#	$\|I\|$	$\|J\|$	$\|S\|$	Number of Variables	Number of Constraints	b_1	Optimal Objective Function Value	CPU Time (Seconds)			
								BNC-SGAP	**BNP-SGAP**	**BNPDST-SGAP**	**BNPDSTKP-SGAP**
15	10	10	1024	10340	10250	3	662.5	**	171.0648676	169.9210398	162.1698781
16	10	10	1024	10340	10250	6	430	**	55.76328097	55.52567532	40.96406434
17	10	10	1024	10340	10250	9	272.5	2487.059066	41.28215467	40.80182668	21.42361864
18	10	10	1024	10340	10250	12	190	61.3776406	25.05624817	25.08938952	5.748306615
19	10	10	1024	10340	10250	15	152.5	3.668648301	18.61068138	18.0652811	6.295753711
20	10	10	1024	10340	10250	18	145	2.23387207	2.852748258	2.112035828	4.435975237
21	11	11	2048	22649	22539	3	802.5	**	935.1152036	941.2678275	916.6493903
22	11	11	2048	22649	22539	6	540	**	309.4125295	289.0901321	255.2931675
23	11	11	2048	22649	22539	9	352.5	94872.71	225.7517435	184.7426181	129.5520914
24	11	11	2048	22649	22539	12	240	1069.265933	85.98563383	85.22994076	22.30054077
25	11	11	2048	22649	22539	15	187.5	23.85355302	66.02377386	67.46818698	24.07186485
26	11	11	2048	22649	22539	18	165	10.01119461	10.9030199	111.172295	15.17426756

**: The corresponding algorithm fails to get an optimal solution due to excessive memory requirements

Table 3.6. Comparative Results for the Four Methods When $|J|=|I|$ and $e_1=10$

BNC-SGAP vs. **BNP-SGAP**		**BNP-SGAP** vs. **BNPDST-SGAP**		**BNPDST-SGAP** vs. **BNPDSTKP-SGAP**	
BNC-SGAP	**BNP-SGAP**	**BNP-SGAP**	**BNPDST-SGAP**	**BNPDST-SGAP**	**BNPDSTKP-SGAP**
8	18	6	20	7	19

Table 3.6 summarizes the results of these comparisons. The number in each cell gives the number of experiments for which each method does better than the other method.

Consider the first comparison. **BNP-SGAP** does better than **BNC-SGAP** in 18 out of the 26 experiments. Furthermore, **BNPDST-SGAP** dominates **BNP-SGAP** in 20 out of the 26 experiments. **BNPDSTKP-SGAP**, on the other hand, is better than the **BNPDST-SGAP** method in 19 of the 26 experiments. Therefore, as noted above, a BNP-based method outperforms the BNC method. Also, the dual stabilization technique and the greedy algorithm, the two features added to the BNP method, help in improving its performance. Another interesting observation that we can make is that **BNPDSTKP-SGAP**, generally, performs several-fold better than **BNC-SGAP** for $b_1=3$ and 6, the instances for which the resource constraint is tighter and **BNC-SGAP** requires relatively large cpu times, or excessive memory requirements.

3.7.2. Experimental Results of Experiments for the BNC-SGAP, BNP-SGAP, BNPDST-SGAP, and BNPDSTKP-SGAP Methods When $|J|=|I|$ and $e_1=100$

Next, we change the overtime penalty value from 10 to 100. Table 3.7 summarizes the results of the experiments performed. For this case, **BNPDSTKP-SGAP** continues to be the best among all the methods. It gives the best results for 19 experiments while **BNC-SGAP**, **BNP-SGAP**, and **BNPDST-SGAP** give the best results for 2, 4, and 1 experiments, respectively.

Table 3.7. Computational Times for Different Methods When $|J|=|I|$ and $e_1=100$

#	$\|I\|$	$\|J\|$	$\|S\|$	Number Of Variables	Number Of Constraints	b_1	Optimal Objective Function Value	CPU Time (Seconds)			
								BNC-SGAP	**BNP-SGAP**	**BNPDST-SGAP**	**BNPDSTKP-SGAP**
1	7	7	128	945	903	3	2416	0.22875985	0.22513114	0.229775707	0.310840355
2	7	7	128	945	903	6	991	0.23712873	0.93208356	0.967443283	0.241597579
3	7	7	128	945	903	9	316	0.01328965	0.01280534	0.013842576	0.012124683
4	7	7	128	945	903	12	91	0.07372141	0.07092703	0.07265731	0.070293529
5	8	8	256	2112	2056	3	3258	628.338254	2.20588596	2.094131636	0.906132572
6	8	8	256	2112	2056	6	1533	1.15471241	1.17000776	1.134553387	0.952370048
7	8	8	256	2112	2056	9	558	0.40967411	0.40780259	0.409301758	0.602322748
8	8	8	256	2112	2056	12	183	0.35893904	2.17765744	2.213687755	0.178166144
9	8	8	256	2112	2056	15	108	0.17823023	0.17173428	0.171945281	0.180612144
10	9	9	512	4689	4617	3	4226	19937.24	7.88427259	7.508068854	3.423187336
11	9	9	512	4689	4617	6	2201	6366.75576	9.95960845	9.730663808	4.692428173
12	9	9	512	4689	4617	9	926	77.4175967	8.39732727	8.164664657	2.464487962
13	9	9	512	4689	4617	12	351	3.6011075	16.2883586	5.8487105	1.817195904
14	9	9	512	4689	4617	15	126	0.60648563	0.57888061	0.55595312	0.648439288

Table 3.7. Computational Times for Different Methods When $|J|=|I|$ and e_1=100 (Continued)

#	$\|I\|$	$\|J\|$	$\|S\|$	Number Of Variables	Number Of Constraints	b_1	Optimal Objective Function Value	CPU Time (Seconds)			
								BNC-SGAP	**BNP-SGAP**	**BNPDST-SGAP**	**BNPDSTKP-SGAP**
15	10	10	1024	10340	10250	3	5320	**	168.881967	170.7826405	162.195921
16	10	10	1024	10340	10250	6	2995	**	56.9208464	52.39056864	40.80159195
17	10	10	1024	10340	10250	9	1420	2445.19153	46.1461082	44.56221646	21.48500881
18	10	10	1024	10340	10250	12	595	61.1962905	20.7951559	21.72684362	6.047604796
19	10	10	1024	10340	10250	15	220	3.63689	21.768956	23.47983524	6.947366594
20	10	10	1024	10340	10250	18	145	2.1731931	2.22123832	3.400302864	2.613912895
21	11	11	2048	22649	22539	3	6540	**	956.910866	938.1709851	920.2956331
22	11	11	2048	22649	22539	6	3915	**	307.482873	290.2851142	255.8170964
23	11	11	2048	22649	22539	9	2040	**	212.588946	175.6258492	129.3052956
24	11	11	2048	22649	22539	12	915	1113.48132	193.621891	198.4127602	24.06000429
25	11	11	2048	22649	22539	15	390	31.1651246	2117.97402	2183.875781	29.71656492
26	11	11	2048	22649	22539	18	165	16.6715958	10.1842459	12.10921285	9.657139926

**: The corresponding algorithm fails to get an optimal solution due to excessive memory requirements

Table 3.8. Comparative Results for the Four Methods when $|J|=|I|$ and $e_1=100$

BNC-SGAP vs. **BNP-SGAP**		**BNP-SGAP** vs. **BNPDST-SGAP**		**BNPDST-SGAP** vs. **BNPDSTKP-SGAP**	
BNC-SGAP	**BNP-SGAP**	**BNP-SGAP**	**BNPDST-SGAP**	**BNPDST-SGAP**	**BNPDSTKP-SGAP**
7	19	14	12	4	22

Table 3.8 gives the comparative results for **BNC-SGAP** and **BNP-SGAP** as well as incremental benefits due to the addition of dual stabilization technique and greedy heuristic. **BNP-SGAP** gives better results over **BNC-SGAP** for 19 of the 26 experiments used. Similarly, **BNPDSTKP-SGAP** does better than **BNPDST-SGAP** for 22 of these 26 experiments. However, incremental benefit due to the dual stabilization technique, i.e., the performance of **BNPDST-SGAP** is not that obvious.

Another point worth making is regarding the stability of each method with respect to the cpu time required with increment in the complexity of the problems. While the other three methods take less than 2,200 seconds for all the experiments, **BNC-SGAP** takes 20,000 seconds for experiment 10, and is unable to solve problem instances for experiments 15, 16, 21, 22, and 23 due to excessive memory requirements.

3.7.3. Experimental Results for the BNC-SGAP, BNP-SGAP, BNPDST-SGAP, and BNPDSTKP-SGAP Methods When $|J|=|I|$ and $e_1=1,000$

Table 3.9 presents results for this case. The experimental data is shown in Table A.3 in Appendix A. Here, the overtime penalty value is increased to 1,000. Note that the **BNC-SGAP**, **BNP-SGAP**, **BNPDST-SGAP**, and **BNPDSTKP-SGAP** methods yield the best result for 3, 5, 2, and 16 experiments while the **BNPDST-SGAP** method does not yield the best result for any instance.

Table 3.9. Computational Times for Different Methods When $|J|=|I|$ and e_1=1,000

#	$\|I\|$	$\|J\|$	$\|S\|$	Number Of Variables	Number Of Constraints	b_1	Optimal Objective Function Value	CPU Time (Seconds)			
								BNC-SGAP	**BNP-SGAP**	**BNPDST-SGAP**	**BNPDSTKP-SGAP**
1	7	7	128	945	903	3	23341	30.7526947	0.77559134	0.719458293	0.307669651
2	7	7	128	945	903	6	9091	0.24086088	0.22948574	0.23589779	0.230200897
3	7	7	128	945	903	9	2341	0.24032195	0.94287482	0.980046098	0.258335312
4	7	7	128	945	903	12	91	0.08239266	0.10314006	0.092973474	0.070538269
5	8	8	256	2112	2056	3	31608	620.373725	1.85620828	1.824238524	0.972538389
6	8	8	256	2112	2056	6	14358	1.19951311	1.16616106	1.1803023	1.183332099
7	8	8	256	2112	2056	9	4608	0.43786794	0.41146528	0.429301293	0.459485856
8	8	8	256	2112	2056	12	858	0.39493648	1.91923008	1.971271375	0.544602832
9	8	8	256	2112	2056	15	108	0.18812726	0.17290174	0.176609092	0.184582843
10	9	9	512	4689	4617	3	41126	18621.8362	7.84078367	7.427151491	3.62901841
11	9	9	512	4689	4617	6	20876	6560.13675	9.69545351	9.392709648	4.822797539
12	9	9	512	4689	4617	9	8126	81.9227524	8.94494749	8.37963481	2.609216947
13	9	9	512	4689	4617	12	2376	3.66214255	15.9439024	5.712523779	1.822648449
14	9	9	512	4689	4617	15	126	0.64607687	0.56286135	0.594470196	0.632984643

Table 3.9. Computational Times for Different Methods When $|J|=|I|$ and e_1=1,000 (Continued)

#	$\|I\|$	$\|J\|$	$\|S\|$	Number Of Variables	Number Of Constraints	b_1	Optimal Objective Function Value	CPU Time (Seconds)			
								BNC-SGAP	**BNP-SGAP**	**BNPDST-SGAP**	**BNPDSTKP-SGAP**
15	10	10	1024	10340	10250	3	51895	**	168.789398	169.1090189	161.2564731
16	10	10	1024	10340	10250	6	28645	**	56.6558274	57.04506987	40.65873484
17	10	10	1024	10340	10250	9	12895	2505.09214	47.0078428	46.27045624	21.53167152
18	10	10	1024	10340	10250	12	4645	64.1181901	20.497772	21.35689362	6.638410521
19	10	10	1024	10340	10250	15	895	3.98606546	19.1096727	19.437534	6.900619132
20	10	10	1024	10340	10250	18	145	2.58752958	2.22458803	2.220077375	3.007964434
21	11	11	2048	22649	22539	3	63915	**	934.127263	916.4541851	912.8344186
22	11	11	2048	22649	22539	6	37665	**	302.203151	284.1568543	253.933224
23	11	11	2048	22649	22539	9	18915	**	203.444627	152.3560833	129.5042056
24	11	11	2048	22649	22539	12	7665	1080.49018	82.0827494	88.59673916	23.92828336
25	11	11	2048	22649	22539	15	2415	29.9710491	106.167444	90.9174264	27.68247299
26	11	11	2048	22649	22539	18	165	30.2552695	10.1619439	9.673347185	20.55780572

**: Tv cbyr6the corresponding algorithm fails to get an optimal solution due to excessive memory requirements

Table 3.10. Comparative Results for the Four Methods When $|J|=|I|$ and e_1=1,000

BNC-SGAP vs. **BNP-SGAP**		**BNP-SGAP** vs. **BNPDST-SGAP**		**BNPDST-SGAP** vs. **BNPDSTKP-SGAP**	
BNC-SGAP	**BNP-SGAP**	**BNP-SGAP**	**BNPDST-SGAP**	**BNPDST-SGAP**	**BNPDSTKP-SGAP**
6	20	12	14	6	20

Referring to Table 3.10, **BNP-SGAP** gives the best result for 20 out of the 26 experiments over that of **BNC-SGAP**. Similarly, **BNPDSTKP-SGAP** dominates **BNPDST-SGAP** for 20 out of 26 of these experiments while the superiority of **BNPDST-SGAP** over **BNP-SGAP** is not that obvious.

BNC-SGAP shows extreme fluctuations in cpu times, which increase up to 18,621 seconds, while the other three BNP-based methods requires less than 1,000 seconds for every experiment. Also, **BNC-SGAP** fails to find an optimal solution for experiments 15, 16, 21, 22, and 23. Notice that these are the same experiments for which it fails to do so in Table 3.7, indicating its invariance in performance to change in the overtime penalty value.

From the experimental results presented in Sections 3.7.1, 3.7.2, and 3.7.3, clearly a BNP-based method, and especially, **BNPDSTKP-SGAP**, performs better than **BNC-SGAP**.

3.7.4. Effect of Changes in the Values of Coefficients for the SGAP

In this section, we study the effect of changes in the values of coefficients for the SGAP on the cpu time required by each method. Table 3.11 depicts the experiment number, the number of machines ($|I|$), the number of jobs ($|J|$), and the machine capacity presented from Tables 3.5, 3.7, and 3.9 and used for this comparison.

We observe the effect of changes in $|I|$, $|J|$, the machine capacity, and the overtime penalty on the performance of the **BNP-SGAP**, **BNPDST-SGAP**, and **BNPDSTKP-SGAP** methods. To that end, Figures 3.7, 3.8, and 3.9 present the plots of the cpu times required for different experiments when e_1=10, e_1=100, and e_1=1,000, respectively.

Table 3.11. Experimental Data

Experiment #	$\|I\|$	$\|J\|$	b_1
1	7	7	3
2	7	7	6
3	7	7	9
4	7	7	12
5	8	8	3
6	8	8	6
7	8	8	9
8	8	8	12
9	8	8	15
10	9	9	3
11	9	9	6
12	9	9	9
13	9	9	12
14	9	9	15

Note that: (1) the cpu time stays relatively low until the value of $|I|$=8, i.e., until Experiment # 9, and it increases rapidly for $|I|$=9; (2) the problem generally becomes more difficult to solve for lower values of capacity, e.g., b_1=3 or 6. This is due to the fact that the problem becomes an MIP in which the values of y_{is} also need to be determined because of the tightness of resource capacity. On the other hand, when b_i is large, no penalty cost is incurred and the values of y_{is}=0. At lower values of b_i, the cpu time can potentially increase though not all the time. For example, **BNP-SGAP** seems to require greater amount of cpu time when b_1=12 than when either b_1=6 or b_1=9, and **BNPDSTKP-SGAP** requires more cpu time for b_1=6 than for b_1=3; (3) **BNPDSTKP-SGAP** provides the best performance as problems become tighter, and also, less fluctuations in cpu time than those for the other two methods. As shown in all three charts, the values for **BNPDSTKP-SGAP** are always smaller than those for the other two methods. It solves problems for all of the experiments in less than 5 seconds; (4) in the first graph, the performances of **BNP-SGAP** and **BNPDST-SGAP** are almost the same. In the second and third graphs, however, **BNP-SGAP** shows a sudden increase in cpu time for Experiment # 13 while the values for **BNPDST-SGAP** stay smooth, that is, **BNPDST-SGAP** shows a more robust performance against changes in machine capacity; (5) when the penalty cost changes from 10 to 1,000, the cpu times do not change significantly, for all the experiments excluding Experiment 13, for which **BNP-SGAP** and **BNPDST-SGAP** require about twice as much cpu time as **BNPDSTKP-SGAP**. Therefore, we can conclude that the penalty cost is not a decisive factor in judging the computational effort required to solve the SGAP.

So far, the ratio of the number of jobs to the number of machines has been kept at one, but in the next two sections, we study the effect of change in this ratio to values of 1.5 and 2.

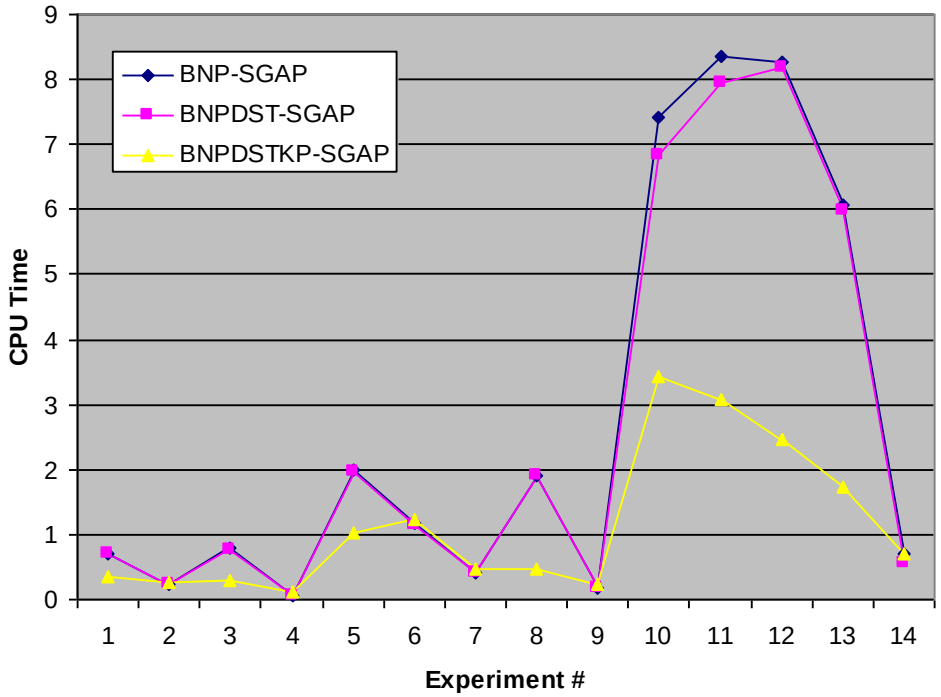

Figure 3.7. Effect of Changes in Machine Capacity When e_1=10

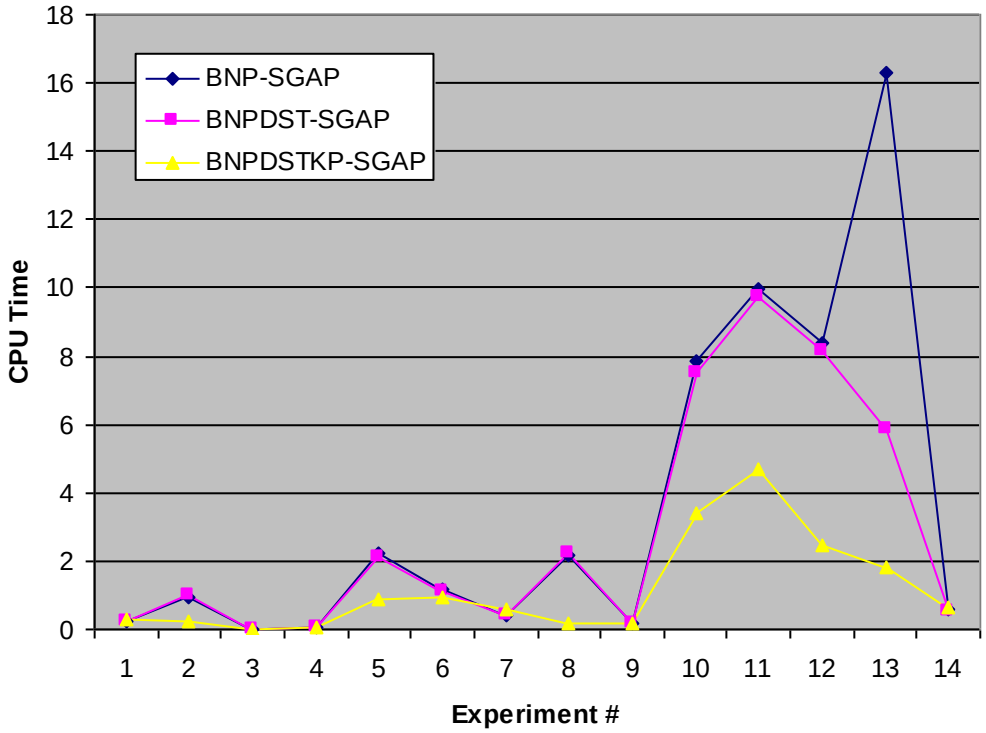

Figure 3.8. Effect of Change in Machine Capacity When e_1=100

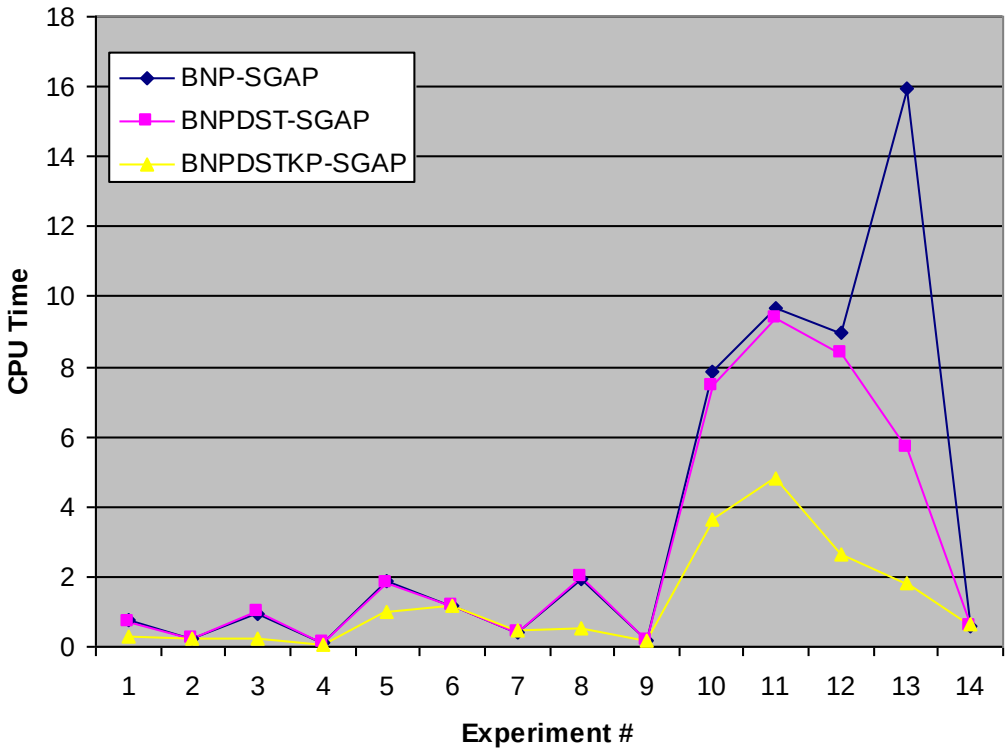

Figure 3.9. Effect of Change in Machine Capacity When e_1=1,000

3.7.5. Experimental Results for the BNC-SGAP, BNP-SGAP, BNPDST-SGAP, and BNPDSTKP-SGAP Methods When $|J|=1.5|I|$ and $e_1=100$

Table 3.12 shows the results for this case. The data for these experiments is shown in Table A.4 in Appendix A, where $|J|=1.5|I|$, i.e., the ratio of the number of jobs to the number of machines is increased by 50%, compared with the instances considered in Sections 3.7.1, 3.7.2, and 3.7.3. As a result, $|J|$ increases up to 17 (when $|I|=11$). If we consider the number of experiments for which each method yields the best result, a claim cannot be made for the **BNP-SGAP** and **BNPDST-SGAP** methods to be better than the **BNC-SGAP** method since both are better for 5 experiments each, while the **BNC-SGAP** method is better for 7 experiments. However, the **BNPDSTKP-SGAP** method is overwhelmingly better than the other three methods since it requires the least cpu times for 21 experiments.

Table 3.12. Computational Times for Different Methods When $|J|=1.5|I|$ and $e_1=100$

#	$\|I\|$	$\|J\|$	$\|S\|$	Number Of Variables	Number Of Constraints	b_1	Optimal Objective Function Value	CPU Time (Seconds)			
								BNC-SGAP	**BNP-SGAP**	**BNPDST-SGAP**	**BNPDSTKP-SGAP**
1	7	11	128	973	907	3	7690	1.66875986	0.13345774	0.135854909	0.152243177
2	7	11	128	973	907	6	5590	1.41155342	0.12963621	0.129325086	0.13100638
3	7	11	128	973	907	9	3490	1.18007931	0.14534362	0.145912251	0.151586917
4	7	11	128	973	907	12	1740	0.93064733	4.99302515	4.891448038	1.616453758
5	7	11	128	973	907	15	690	0.64538566	2.06013801	2.085667027	0.736098362
6	7	11	128	973	907	18	165	0.27258688	1.97779036	1.570245398	0.325367362
7	8	12	256	2144	2060	3	9036	6.40377758	0.52700799	0.541295631	0.527809157
8	8	12	256	2144	2060	6	6636	5.58163146	0.42117094	0.553086485	0.441163921
9	8	12	256	2144	2060	9	4236	4.86714205	0.56132123	36.43106836	0.577993249
10	8	12	256	2144	2060	12	2186	3.96543779	4.90297305	4.894764273	4.696117614
11	8	12	256	2144	2060	15	936	2.61135088	5.28212665	5.441765956	1.776276979
12	8	12	256	2144	2060	18	261	1.30011496	109.160851	113.7549209	0.833063013
13	8	12	256	2144	2060	21	186	0.42440869	0.24543949	0.237315124	0.237293996
14	9	14	512	4734	4622	3	12406	30.7974122	37.1497486	36.29798778	3.930479858
15	9	14	512	4734	4622	6	9706	26.9766057	1.80216785	1.789746512	1.839340853
16	9	14	512	4734	4622	9	7006	23.8007333	1.74102031	1.716803248	1.807372496
17	9	14	512	4734	4622	12	4306	20.6687462	18.226773	17.59036229	12.08704712
18	9	14	512	4734	4622	15	2406	11.8522807	19.2499303	19.12798526	37.3724408
19	9	14	512	4734	4622	18	1056	8.9459057	26.267213	25.80650194	5.48697249
20	9	14	512	4734	4622	21	306	4.34017256	17.3140944	17.10905384	1.950849092
21	9	14	512	4734	4622	24	231	1.77512198	77.6832881	79.47990197	2.426705713

Table 3.12. Computational Times for Different Methods When $|J|=1.5|I|$ and $e_1=100$ (Continued)

#	$\|I\|$	$\|J\|$	$\|S\|$	Number Of Variables	Number Of Constraints	b_1	Optimal Objective Function Value	CPU Time (Seconds)			
								BNC-SGAP	**BNP-SGAP**	**BNPDST-SGAP**	**BNPDSTKP-SGAP**
22	10	15	1024	10390	10255	3	14130	1354.64166	231.733602	231.992229	213.3752008
23	10	15	1024	10390	10255	6	11130	982.686084	286.351721	290.1198739	235.4048755
24	10	15	1024	10390	10255	9	8130	578.891363	178.413005	480.7644094	176.5763764
25	10	15	1024	10390	10255	12	5130	266.472882	966.210147	1043.949108	64.46077878
26	10	15	1024	10390	10255	15	2930	142.964789	50.4881453	50.20724329	35.19749304
27	10	15	1024	10390	10255	18	1380	98.3132761	44.6732624	42.95811747	33.58754888
28	10	15	1024	10390	10255	21	480	24.1686142	1451.79843	1448.964404	7.413161131
29	10	15	1024	10390	10255	24	255	7.43367411	1766.981	1744.442237	6.518222826
30	11	17	2048	22715	22545	3	18256	9602.16888	1806.8537	1834.374584	1701.972604
31	11	17	2048	22715	22545	6	14956	6430.54781	1455.73391	1507.552853	1454.953144
32	11	17	2048	22715	22545	9	11656	3784.3531	1260.79584	1346.76429	1206.279342
33	11	17	2048	22715	22545	12	8356	1589.63293	960.038397	966.334909	1085.814472
34	11	17	2048	22715	22545	15	5106	110443.607	452.679576	441.2692526	416.1403314
35	11	17	2048	22715	22545	18	3156	648.525927	347.460807	340.6742006	590.0306288
36	11	17	2048	22715	22545	21	1506	417.641737	304.645884	291.1673027	108.8136076
37	11	17	2048	22715	22545	24	531	73.0279994	71.7832059	72.06399611	39.2617713
38	11	17	2048	22715	22545	27	306	24.0762569	100.093508	99.57695538	26.07450661

Table 3.13. Comparative Results for the Four Methods When $|J|=1.5|I|$ and $e_1=100$

BNC-SGAP vs. BNP-SGAP		BNP-SGAP vs. BNPDST-SGAP		BNPDST-SGAP vs. BNPDSTKP-SGAP	
BNC-SGAP	BNP-SGAP	BNP-SGAP	BNPDST-SGAP	BNPDST-SGAP	BNPDSTKP-SGAP
12	26	16	22	7	31

Referring to Table 3.13, the usage of the **BNP-SGAP** method can still be justified not only because it does better than the **BNC-SGAP** method in 26 out of the 38 experiments, but also because the **BNP-SGAP** method requires at most about 30 minutes for any one experiment while the **BNC-SGAP** method requires about 30 hours for experiment 34 in the worst case. The **BNPDST-SGAP** method is better than the **BNP-SGAP** method in 22 out of the 38 experiments, and, finally, the **BNPDSTKP-SGAP** method performs better than the **BNPDST-SGAP** method in 31 out of the 38 experiments.

3.7.6. Experimental Results for the BNC-SGAP, BNP-SGAP, BNPDST-SGAP, and BNPDSTKP-SGAP Methods When $|J|=2|I|$ and $e_1=100$

Table 3.14 presents the results for this case. The experimental data is given in Table A.5 in Appendix A, where $|J|=2|I|$, i.e., the number of jobs is twice as many as the number of machines. Note that for this case, the method which gives the best performance is not **BNPDSTKP-SGAP** method, but **BNPDST-SGAP**, which gives the best results in 19 out of 45 experiments. Both **BNC-SGAP** and **BNPDSTKP-SGAP** give the best results for 12 experiments each.

Referring to Table 3.15, **BNP-SGAP** performs better than **BNC-SGAP** for 31 out of 45 experiments. Especially, note that, **BNP-SGAP** performs better than **BNC-SGAP** when the problems are more complicated. Furthermore, **BNPDST-SGAP** performs better than **BNP-SGAP** for 40 out of 45 experiments. Similarly, **BNPDSTKP-SGAP** performs better than **BNPDST-SGAP** for 27 experiments.

Table 3.14. Computational Times for Different Methods When $|J|=2|I|$ and $e_1=100$

#	$\|I\|$	$\|J\|$	$\|S\|$	Number Of Variables	Number Of Constraints	b_1	Optimal Objective Function Value	CPU Time (Seconds)			
								BNC-SGAP	**BNP-SGAP**	**BNPDST-SGAP**	**BNPDSTKP-SGAP**
1	7	14	128	994	910	3	13006	3.7567026	1.4269168	1.377416426	0.182709358
2	7	14	128	994	910	6	10906	3.34803921	0.13067691	0.127343273	0.144721873
3	7	14	128	994	910	9	8806	3.11006287	0.13175503	0.124888582	0.157140484
4	7	14	128	994	910	12	6706	2.88896187	4.53933607	4.468884712	2.80593496
5	7	14	128	994	910	15	4606	2.57257668	18.8817764	18.95921723	7.784425237
6	7	14	128	994	910	18	2856	2.00382144	3.81451098	3.796799674	2.528885123
7	7	14	128	994	910	21	1806	1.5465541	2.26687703	2.161971935	7.297830034
8	7	14	128	994	910	24	756	1.18360602	3.27752468	3.233416837	1.538589835
9	7	14	128	994	910	27	231	0.38696399	78.5991966	77.5725089	0.561098617
10	8	16	256	2176	2064	3	16880	8.73893749	0.55384369	0.50042619	0.591533998
11	8	16	256	2176	2064	6	14480	7.8765818	0.55335897	0.516737788	0.58870392
12	8	16	256	2176	2064	9	12080	7.0365147	0.55114563	0.542995662	0.598714561
13	8	16	256	2176	2064	12	9680	6.00625527	4.93326329	4.60632021	10.11254379
14	8	16	256	2176	2064	15	7280	5.06314015	5.21346856	4.832283909	4.127396266
15	8	16	256	2176	2064	18	4880	3.78098781	5.79782298	5.269809776	11.28077908
16	8	16	256	2176	2064	21	3280	5.74558163	99.2100788	91.61060183	44.71888894
17	8	16	256	2176	2064	24	2080	4.27244552	207.3425	190.7747081	15.10448643
18	8	16	256	2176	2064	27	880	2.82492859	6.31880894	5.532624962	5.703395524

Table 3.14. Computational Times for Different Methods When $|J|=2|I|$ and $e_1=100$ (Continued)

#	$\|I\|$	$\|J\|$	$\|S\|$	Number Of Variables	Number Of Constraints	b_1	Optimal Objective Function Value	CPU Time (Seconds)			
								BNC-SGAP	**BNP-SGAP**	**BNPDST-SGAP**	**BNPDSTKP-SGAP**
19	9	18	512	4770	4626	3	21258	40.7611682	2.52020516	1.975760594	2.443786509
20	9	18	512	4770	4626	6	18558	36.5407511	11.7528469	10.83705407	2.469977992
21	9	18	512	4770	4626	9	15858	32.7701018	2.03991781	1.842520072	2.042716843
22	9	18	512	4770	4626	12	13158	29.2306207	11.3213444	10.70790335	19.13145822
23	9	18	512	4770	4626	15	10458	25.2539479	2.05229711	1.925589667	2.09319726
24	9	18	512	4770	4626	18	7758	21.6643613	12.0331232	11.15580454	177.1789466
25	9	18	512	4770	4626	21	5058	15.4891549	12.3143294	11.35554034	42.66897436
26	9	18	512	4770	4626	24	3708	9.04904199	13.121937	12.15203445	20.51789684
27	9	18	512	4770	4626	27	2358	5.10580451	14.7014717	13.29262615	5.76309196
28	10	20	1024	10440	10260	3	26140	2872.84736	313.896388	300.7069354	276.4761856
29	10	20	1024	10440	10260	6	23140	2409.11573	256.944404	254.5039429	251.9472634
30	10	20	1024	10440	10260	9	20140	1976.09984	265.502434	263.8057436	261.7328318
31	10	20	1024	10440	10260	12	17140	1529.00079	226.522373	226.6891613	224.874655
32	10	20	1024	10440	10260	15	14140	1143.79222	431.391062	429.1424171	417.0508966
33	10	20	1024	10440	10260	18	11140	757.217913	406.837758	401.2938311	455.3764138
34	10	20	1024	10440	10260	21	8140	422.134173	147.681002	144.8953993	645.1260497
35	10	20	1024	10440	10260	24	5640	131.796381	68.0687369	64.8208494	745.0706168
36	10	20	1024	10440	10260	27	4140	56.3151766	61.3401009	60.90970741	482.526745

Table 3.14. Computational Times for Different Methods When $|J|=2|I|$ and $e_1=100$ (Continued)

#	$\|I\|$	$\|J\|$	$\|S\|$	Number Of Variables	Number Of Constraints	b_1	Optimal Objective Function Value	CPU Time of Algorithms (Seconds)			
								BNC-SGAP	**BNP-SGAP**	**BNPDST-SGAP**	**BNPDSTKP-SGAP**
37	11	22	2048	22770	22550	3	31526	22515.7152	1702.76854	1727.467957	1708.615247
38	11	22	2048	22770	22550	6	28226	19462.0492	1645.54193	1676.279165	1652.164825
39	11	22	2048	22770	22550	9	24926	16524.7048	1799.38084	2857.220012	1751.050165
40	11	22	2048	22770	22550	12	21626	13454.9933	1751.28529	1739.05465	1725.036357
41	11	22	2048	22770	22550	15	18326	10389.336	1530.38357	1528.697579	1525.696647
42	11	22	2048	22770	22550	18	15026	7667.49611	1522.47975	1520.065757	2205.795689
43	11	22	2048	22770	22550	21	11726	5034.95295	1017.88223	1000.853273	1263.152497
44	11	22	2048	22770	22550	24	8426	3224.74323	740.855718	720.6505897	1937.111849
45	11	22	2048	22770	22550	27	6226	814.699129	489.860534	483.9349043	3334.4479

Table 3.15. Comparative Results for the Four Methods When $|J|=2|I|$ and $e_1=100$

BNC-SGAP vs. **BNP-SGAP**		**BNP-SGAP** vs. **BNPDST-SGAP**		**BNPDST-SGAP** vs. **BNPDSTKP-SGAP**	
BNC-SGAP	**BNP-SGAP**	**BNP-SGAP**	**BNPDST-SGAP**	**BNPDST-SGAP**	**BNPDSTKP-SGAP**
14	31	5	40	18	27

3.7.7. Effect of Variation in the Number of Jobs per Machine

In this section, we observe the effect of changes made in the number of jobs per machine, i.e., changes in $|J|/|I|$, on the cpu time. To that end, we compare the information depicted in Tables 3.7, 3.11, and 3.13, where $e_1=100$ and $|J|=|I|$, $|J|=1.5|I|$, and $|J|=2|I|$, respectively. Table 3.16 summarizes information on the number of jobs, the number of variables, and the number of constraints. Needless to say, all three of these parameters increase with the ratio of $|J|$ to $|I|$. Note that in our experimentation, the number of scenario does not change with the number of jobs, but with the number of machines. As a result, the number of variables and the number of constraints do not increase when the number of jobs increases by 150% and 200% as much as when the number of machines increases to 11.

Table 3.16. Problem Size with Change in the Ratio of $|J|$ to $|I|$

$\|I\|$	$\|S\|$	$\|J\|$			Number of Variables			Number of Constraints		
		$\|J\|=\|I\|$	$\|J\|=1.5\|I\|$	$\|J\|=2\|I\|$	$\|J\|=\|I\|$	$\|J\|=1.5\|I\|$	$\|J\|=2\|I\|$	$\|J\|=\|I\|$	$\|J\|=1.5\|I\|$	$\|J\|=2\|I\|$
7	128	7	11	14	945	973	994	903	907	910
8	256	8	12	16	2,112	2,144	2,176	2,056	2,060	2,064
9	512	9	14	18	4,689	4,734	4,770	4,617	4,622	4,626
10	1,024	10	15	20	10,340	10,390	10,440	10,250	10,255	10,260
11	2,048	11	17	22	22,649	22,715	22,770	22,539	22,545	22,550

We compare the performances of the best among the BNP-based methods, namely, **BNPDSTKP-SGAP** with those of **BNC-SGAP** with change in the ratio of $|J|$ and $|I|$. Figures 3.10, 3.11, and 3.12 present plots for the cases of $|J|=|I|$, $|J|=1.5|I|$, and $|J|=2|I|$, respectively. The horizontal axis of the plot in each figure represents combinations of number of machines, number of jobs, and machine capacity, i.e., ($|I|$, $|J|$, b_1), while the vertical axis represents cpu time in seconds.

For the cases $|J|=|I|$, $|J|=1.5|I|$, and $|J|=2|I|$, **BNPDSTKP-SGAP** requires 62.5, 196.9, and 472.3 seconds on average, respectively. **BNPDSTKP-SGAP** does not require more than 1,000 seconds when $|J|=|I|$, while it does so for four and nine times when $|J|=1.5|I|$ and $|J|=2|I|$, respectively. Finally, the largest cpu time required by **BNPDSTKP-SGAP** in each case is 920, 1,700 and 3,334 seconds, respectively. Note that, at higher values of $|I|$ (see for instance $|I|=11$), the cpu times required by **BNPDSTKP-SGAP** tend to increase with increment in the ratio of $|J|$ to $|I|$.

Referring to Figure 3.10, 3.11, and 3.12, a similar behavior is observed for **BNC-SGAP** as well albeit more pronounced. Note that when ($|I|$, $|J|$, b_1) is equal to (10,10,3), (10,10,6), (11,11,3), (11,11,6), and (11,11,9) (see Figure 3.10), **BNC-SGAP** fails to find an optimal solution. On the other hand, **BNC-SGAP** is able to find an optimal solution for all the experiments when $|J|=1.5|I|$ and $|J|=2|I|$. It is interesting to note that **BNC-SGAP** requires more cpu time when $|J|=1.5|I|$ than when $|J|=2|I|$ (3,595.5 and 2,459.4 seconds on average, respectively), and also, the largest cpu time required by **BNC-SGAP** for each case is 110,444 and 22,556 seconds, respectively.

In Table 3.17, we have summarized comparative results for all four methods from Tables 3.7, 3.11, and 3.13, where $e_1=100$ and $|J|=|I|$, $|J|=1.5|I|$, and $|J|=2|I|$, respectively. Note that a BNP-based method, either **BNPDST-SGAP** or **BNPDSTKP-SGAP**, performs better than **BNC-SGAP** more often for all ratios of $|J|$ to $|I|$ investigated. Also, note that all three BNP-based methods perform better than **BNC-SGAP** over all ratios of $|J|$ and $|I|$ investigated.

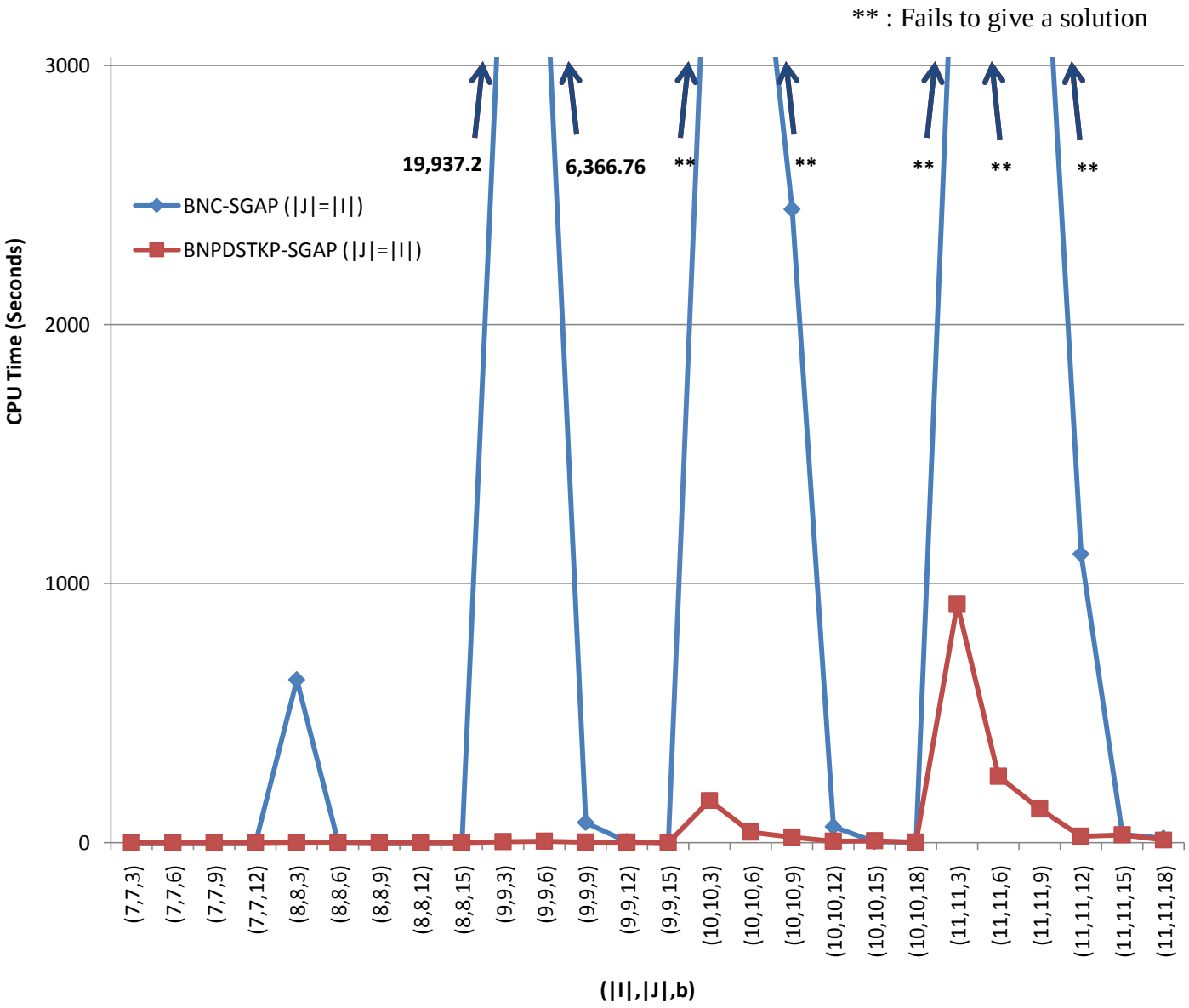

Figure 3.10. **BNC-SGAP** vs. **BNPDSTKP-SGAP** When $|J|=|I|$

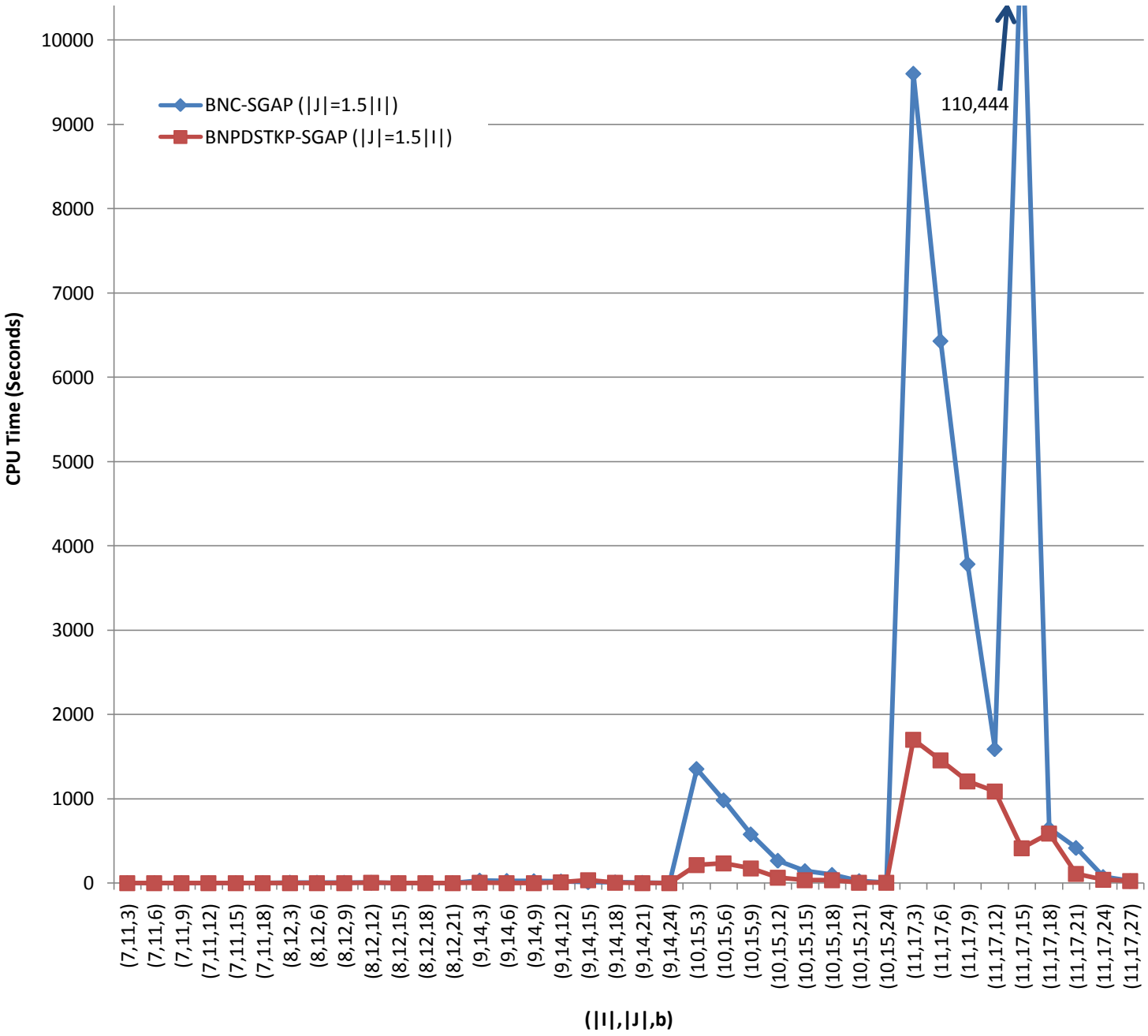

Figure 3.11. **BNC-SGAP** vs. **BNPDSTKP-SGAP** When $|J|=1.5|I|$

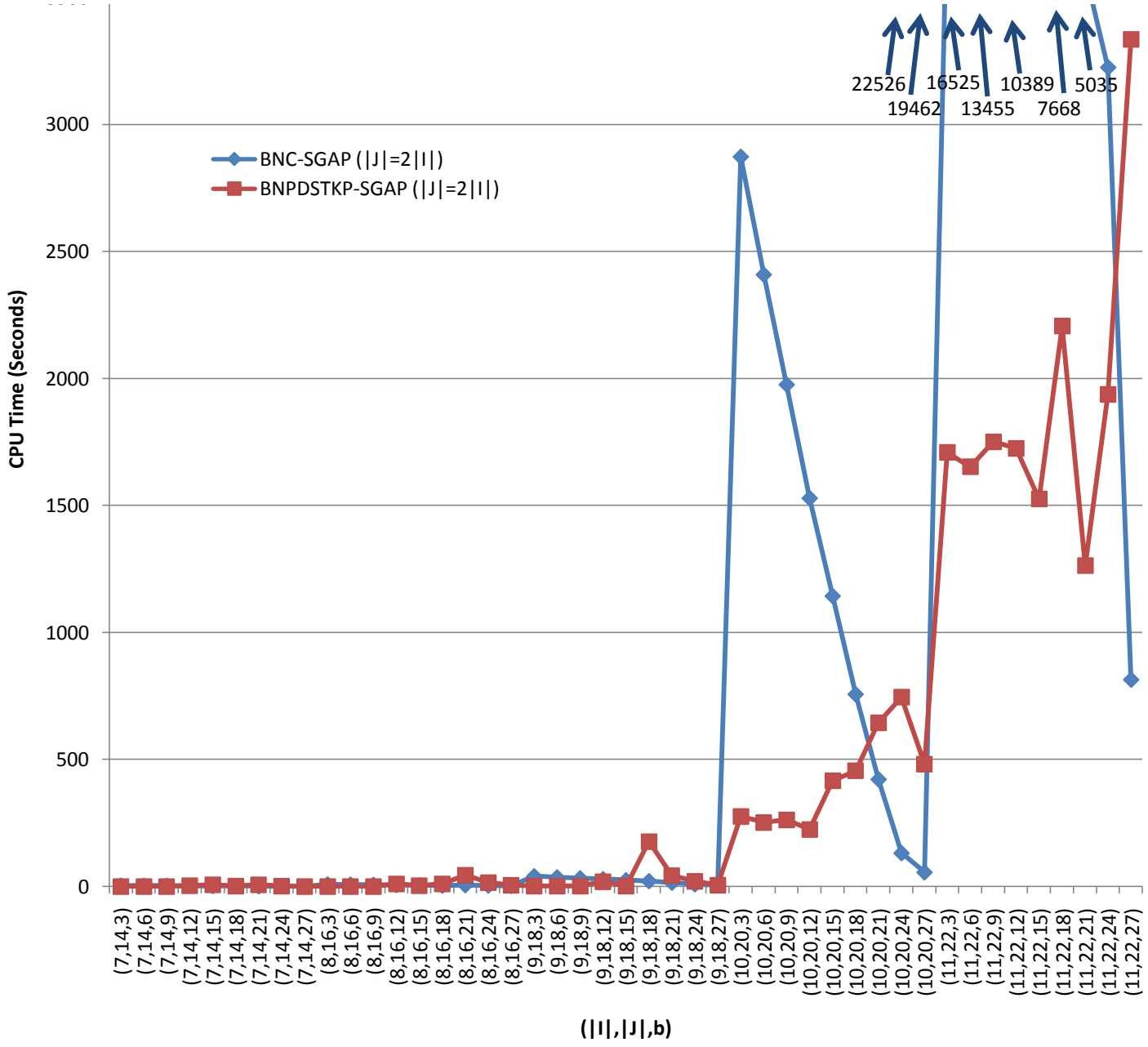

Figure 3.12. **BNC-SGAP** vs. **BNPDSTKP-SGAP** When $|J|=2|I|$

Table 3.17. Number of Experiments with the Best CPU Time and Average CPU Time of Each Method

		$\|J\|=\|I\|$	$\|J\|=1.5\|I\|$	$\|J\|=2\|I\|$		Total
BNC-SGAP	Number of Experiments with the Best CPU Time	2	7	12		21
	Average CPU Time (Seconds)	N/A*	3,595.5	2,459.4		N/A*
BNP-SGAP	Number of Experiments with the Best CPU Time	4	5	2		11
	Average CPU Time (Seconds)	160.2	317.5	331.2		285.7
BNPDST-SGAP	Number of Experiments with the Best CPU Time	1	5	19		25
	Average CPU Time (Seconds)	159.8	331.7	353.2		300.0
BNPDSTKP-SGAP	Number of Experiments with the Best CPU Time	19	21	12		52
	Average CPU Time (Seconds)	62.5	196.9	472.3		278.5

*: **BNC-SGAP** fails to get an optimal solution for five experiments for this case, so the average cpu time cannot be obtained.

3.7.8. Results for the BNPDSTKP-SGAP with MCM Method

In this section, we present results on the performance of the Monte Carlo method for the SGAP. This performance is measured in terms of the optimality gap as well as the cpu time required. We use 10 and 11-machine problems as the results are more discriminating for these cases. As a result, the following experiments are selected from those presented in Sections 3.7.1, 3.7.2, 3.7.3, 3.7.5, and 3.7.6:

- Experiments 15 through 26 from Tables 3.5, 3.7, and 3.9, where $|J|=|I|$
- Experiments 22 through 38 from Table 3.12, where $|J|=1.5|I|$
- Experiments 28 through 45from Table 3.14, where $|J|=2|I|$

In the next five sections, we make observations on the use of **BNPDSTKP-SGAP with MCM** for these problems.

3.7.8.1. $|J|=|I|$ and $e_1=10$

These are presented in Table 3.18. The number in the 1st column corresponds to the same experiment number as depicted in Table 3.5. The 2nd, 3rd, 4th, and 5th columns contain the values of $|I|$, $|J|$, $|S|$, and b_1, respectively. The 6th column contains the optimal objective function value as given in Table 3.5. The 7th column, named as 'Best CPU Time', contains the best cpu time from Table 3.5, obtained by one of the four methods. For example, for experiment 15, **BNPDSTKP-SGAP** gives the best cpu time of 162.17 seconds. The 8th and 9th columns, called Initial (N, $\tilde{N}$, M) and Final (N, $\tilde{N}$, M), represent the initial (final) number of scenarios for problem h_N^m, the initial (final) number of scenarios for problem $h_{\tilde{N}}^m$, and the initial (final) number of replications, respectively. In this case, we only increase M by one every time the stopping criterion is violated, while N and $\tilde{N}$ remain the same. Therefore, if the method fails to meet the stopping criterion at the first iteration, M is increased from 2 to 3. Note that the values of Final (N, $\tilde{N}$, M) in the table are identical to those of Initial (N, $\tilde{N}$, M), which means that the stopping criterion is satisfied after the first iteration for all the experiments.

Table 3.18. Results for **BNPDSTKP-SGAP with MCM** When $|J|=|I|$, $e_1=10$, and $\varepsilon=0.20$

#	$\|I\|$	$\|J\|$	$\|S\|$	b_1	Optimal Objective Function Value	Best CPU Time (Seconds)	Initial $(N,\tilde{N},M)$	Final $(N,\tilde{N},M)$	Best Objective Function Value for **BNPDSTKP-SGAP with MCM** $(h_{\tilde{N}}((\hat{X}_N)^*))$	Average Objective Function Value $(\bar{h}_N)$	Tolerance Ratio $\left(\frac{h_{\tilde{N}}((X_N)^*}{\bar{h}_N}-1\right)$	Near-Optimality $\left(\frac{h_{\tilde{N}}((X_N)^*}{h(X^*)}\right)$	CPU Time for **BNPDSTKP-SGAP with MCM** (Seconds)
15	10	10	1024	3	662.5	162.1699	(128,256,2)	(128,256,2)	658.5742	655.5078	0.0047	0.9941	0.9332
16	10	10	1024	6	430	40.9641	(128,256,2)	(128,256,2)	427.8320	419.7461	0.0193	0.9950	1.5763
17	10	10	1024	9	272.5	21.4236	(128,256,2)	(128,256,2)	268.6523	262.7148	0.0226	0.9859	1.3125
18	10	10	1024	12	190	5.7483	(128,256,2)	(128,256,2)	189.4727	185.0781	0.0237	0.9972	0.6670
19	10	10	1024	15	152.5	3.6686	(128,256,2)	(128,256,2)	152.2656	151.2695	0.0066	0.9985	1.2552
20	10	10	1024	18	145	2.1120	(128,256,2)	(128,256,2)	145	145	0	1	0.2628
21	11	11	2048	3	802.5	916.6494	(256,512,2)	(256,512,2)	803.9648	800.9473	0.0038	1.0018	6.9416
22	11	11	2048	6	540	255.2932	(256,512,2)	(256,512,2)	535.7617	535.4492	0.0006	0.9922	6.4159
23	11	11	2048	9	352.5	129.5521	(256,512,2)	(256,512,2)	350.9473	343.6035	0.0214	0.9956	5.0164
24	11	11	2048	12	240	22.3005	(256,512,2)	(256,512,2)	238.8574	236.3672	0.0105	0.9952	2.4236
25	11	11	2048	15	187.5	23.8536	(256,512,2)	(256,512,2)	187.7344	185.3613	0.0128	1.0013	3.1100
26	11	11	2048	18	165	10.0112	(256,512,2)	(256,512,2)	165	165	0	1	0.8401

The 10th and 11th columns contain the values of $h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)$ and $\bar{h}_N$, respectively. The 12th column contains the tolerance ratio, which is the value of $\frac{h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)}{\bar{h}_N}-1$ when the stopping criterion is met, i.e., when $\frac{h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)}{\bar{h}_N}-1<\varepsilon$. Note that, based on Propositions 3.8 and 3.9, $h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)$ and $\bar{h}_N$ are unbiased estimators of an upper bound and a lower bound, respectively. Therefore, even if $\frac{h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)}{\bar{h}_N}-1<\varepsilon$, we stop the algorithm only when $h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)\geq\bar{h}_N$. As mentioned in Section 3.5.2, the result that violates this condition is considered as a failure whether $\frac{h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)}{\bar{h}_N}-1<\varepsilon$ or not. As a result, note that, in all the tables, the values of $h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)$ are not less than those of the corresponding $\bar{h}_N$, i.e., $\frac{h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)}{\bar{h}_N}-1\geq 0$ for all the experiments. We use the value of ε=0.20, which means that unless $h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)$ is greater than $\bar{h}_N$ by 20%, the algorithm will be stopped. However, as shown in the table, none of the values in the column for Tolerance Ratio is bigger than 0.03. Therefore, we can say that **BNPDSTKP-SGAP with MCM** achieves a good quality of tolerance ratio for each experiment.

Since there are no guarantees that $h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)$ will be equal to $h\left(X^*\right)$, we also evaluate the quality of the solution obtained by **BNPDSTKP-SGAP with MCM** by determining how close the value is to $h\left(X^*\right)$ for each experiment. The 13th column, named 'Near-Optimality', provides the values of $\frac{h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)}{h(X^*)}$, which indicate a better performance when a value is close to 1. The values in the 'Near-Optimality' column also support the high quality of the solutions

obtained by **BNPDSTKP-SGAP with MCM**. None of the values is less than 0.98, and, in fact, except for Experiment 17, all the values are within 1% of $h(X^*)$.

The last column depicts the cpu time required by **BNPDSTKP-SGAP with MCM** for each experiment. As it is expected, managing only a part of the entire set of scenarios makes a positive effect on the cpu time. None of the cpu time values exceeds the corresponding best cpu time among the other methods included in the 7th column of Table 3.18. For example, for experiment 15, it requires less than one second while the other three methods require at least 162 seconds.

Figure 3.13 shows a plot of the ratio of the entries in the 'Best CPU Time' column (the 7th column of Table 3.18) to those in the last column vs. problem parameters ($|I|$, $|J|$, b_1). The number next to each dot is the value of each ratio, and the average value of those ratios is 38.5, which means that **BNPDSTKP-SGAP with MCM** solves about 38.5 times as fast as any of the other methods on average. Thus, in conclusion, at the expense of 1% gap between $h(X^*)$ and $h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)$, we are able to solve the problems given in Table 3.18 faster by a factor of 38.5 times on average.

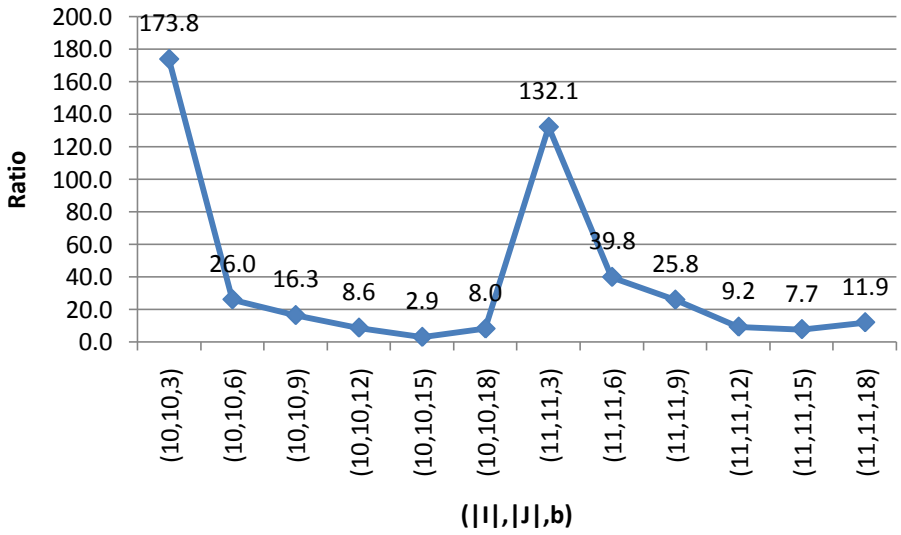

Figure 3.13. Values of Ratio= $\frac{\text{Best CPU Time Value Given in Column 7 of Table 3.16}}{\text{CPU Time Value Given in Column 14 of Table 3.16}}$ vs. ($|I|$,$|J|$,b_1)

3.7.8.2. $|J|=|I|$ and $e_1=100$

Table 3.19 depicts the results for **BNPDSTKP-SGAP with MCM** when $|J|=|I|$ and $e_1=100$. The worst value of tolerance ratio is 0.09 that occurs for Experiment 18. Otherwise, the tolerance ratio values are less than 0.05. Therefore, the estimated upper and lower bounds of $h(X^*)$ for each experiment, i.e., the values in the 10th and 11th columns, respectively, are tight enough to meet stopping criterion even when ε=0.10.

Referring to the 'Near-Optimality' column, the objective function value of each solution obtained by the MCM-based method is within 4% of the "true" optimal objective function value. Note that the values in the 'Final (N, $\tilde{N}$, M)' column are identical to those of Initial (N, $\tilde{N}$, M), a fact which also leads to lower cpu time for the **BNPDSTKP-SGAP with MCM** method.

Figure 3.14 displays the ratios of the values of 'Best CPU Time' in the 7th column to those in the last column of Table 3.19 vs. problem parameters ($|I|$, $|J|$, b_1). The number next to each dot is the value of each ratio. The maximum and minimum values of ratios are about 167 and 2.3 for Experiments 15 and 19, respectively, and the average value of the ratio is 32.6. Thus, a 32.6-fold savings in cpu time is achieved by the MCM-based method for an optimality gap of 4%.

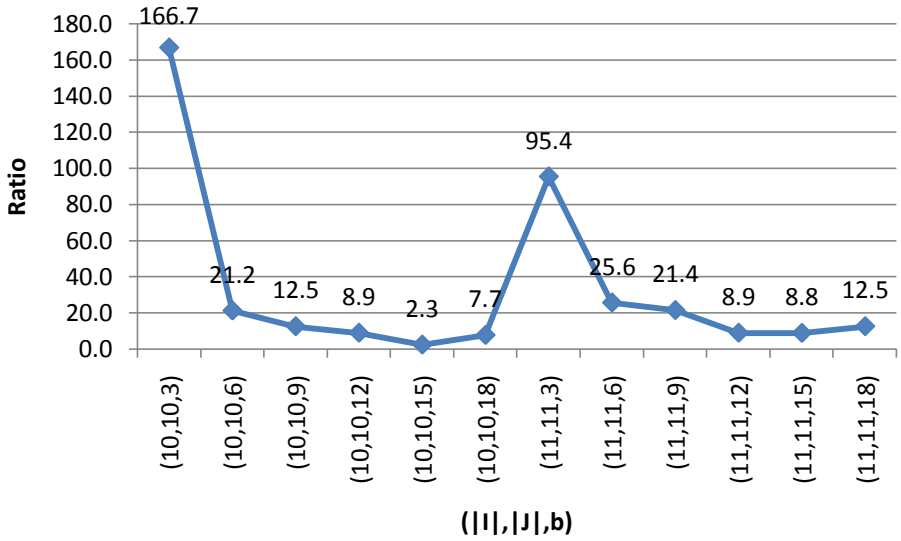

Figure 3.14. Values of Ratio= $\frac{\text{Best CPU Time Value Given in Column 7 of Table 3.17}}{\text{CPU Time Value Given in Column 14 of Table 3.17}}$ vs. ($|I|$,$|J|$,b_1)

Table 3.19. Results for **BNPDSTKP-SGAP with MCM** When $|J|=|I|$, $e_1=100$, and $\varepsilon=0.20$

#	$\|I\|$	$\|J\|$	$\|S\|$	b_1	Optimal Objective Function Value	Best CPU Time (Seconds)	Initial $(N,\tilde{N},M)$	Final $(N,\tilde{N},M)$	Best Objective Function Value for **BNPDSTKP-SGAP with MCM** $(h_{\tilde{N}}((\hat{X}_N)^*))$	Average Objective Function Value $(\bar{h}_N)$	Tolerance Ratio $\left(\frac{h_{\tilde{N}}((X_N)^*}{\bar{h}_N}-1\right)$	Near-Optimality $\left(\frac{h_{\tilde{N}}((X_N)^*}{h(X^*)}\right)$	CPU Time for **BNPDSTKP-SGAP with MCM** (Seconds)
15	10	10	1024	3	5320	162.195	(128,256,2)	(128,256,2)	5334.6484	5307.3047	0.0052	1.0028	0.9728
16	10	10	1024	6	2995	40.8016	(128,256,2)	(128,256,2)	2983.4766	2938.9453	0.0152	0.9962	1.9258
17	10	10	1024	9	1420	21.4850	(128,256,2)	(128,256,2)	1419.4141	1354.1797	0.0482	0.9996	1.7179
18	10	10	1024	12	595	6.0476	(128,256,2)	(128,256,2)	593.8281	544.6094	0.0904	0.9980	0.6794
19	10	10	1024	15	220	3.6369	(128,256,2)	(128,256,2)	220.5859	210.6250	0.0473	1.0027	1.5591
20	10	10	1024	18	145	2.1732	(128,256,2)	(128,256,2)	145	145	0	1	0.2838
21	11	11	2048	3	6540	920.295	(256,512,2)	(256,512,2)	6552.8906	6500.6445	0.0080	1.0020	9.6468
22	11	11	2048	6	3915	255.817	(256,512,2)	(256,512,2)	3920.5664	3822.8125	0.0256	1.0014	9.9915
23	11	11	2048	9	2040	129.305	(256,512,2)	(256,512,2)	2016.5625	1951.1328	0.0335	0.9885	6.0326
24	11	11	2048	12	915	24.0600	(256,512,2)	(256,512,2)	901.2305	861.6797	0.0459	0.9850	2.7071
25	11	11	2048	15	390	29.7166	(256,512,2)	(256,512,2)	375.9375	367.4414	0.0231	0.9639	3.3726
26	11	11	2048	18	165	9.6571	(256,512,2)	(256,512,2)	165	165	0	1	0.7719

3.7.8.3. $|J|=|I|$ and e_1=1,000

The results for this case are shown in Table 3.20. Once again, the stopping criterion is satisfied after the initial set of replications is completed as the column for Final (N, $\tilde{N}$, M) is the same as that for Initial (N, $\tilde{N}$, M). **BNPDSTKP-SGAP with MCM** keeps the tolerance ratios and optimality gaps to within 0.07 and 4%, respectively, except for Experiment 19, for which the values are 0.18 and 5.2%. For Experiments 15, 17, 20, 21, 22, 25, and 26, the values of optimality gaps are less than 1%. Again, the cpu times required by **BNPDSTKP-SGAP with MCM** are very small compared to those required by other methods (given in the 7th column). A plot of their ratios is shown in Figure 3.15. The average value of the ratios is 34.4, implying 34.4-fold savings.

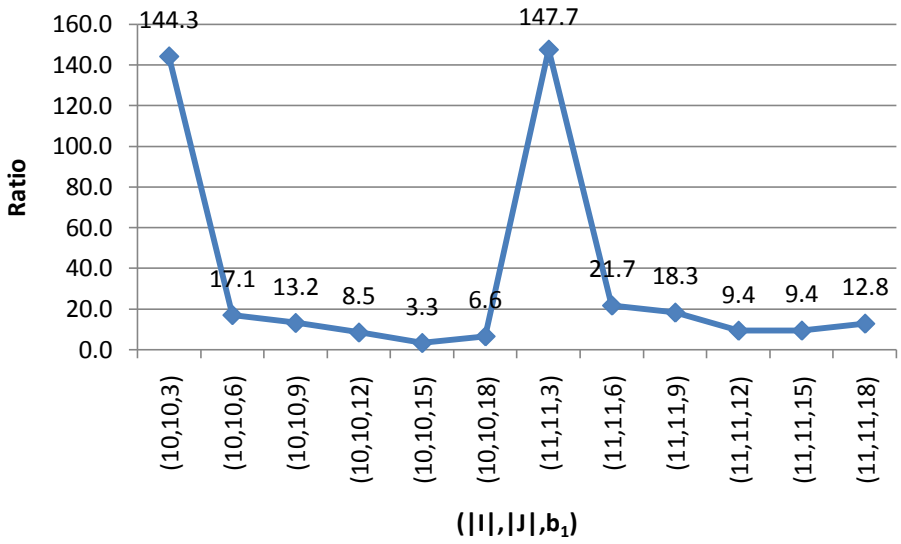

Figure 3.15. Values of Ratio= $\frac{\text{Best CPU Time Value Given in Column 7 of Table 3.18}}{\text{CPU Time Value Given in Column 14 of Table 3.18}}$ vs. ($|I|$,$|J|$,b_1)

Table 3.20. Results for **BNPDSTKP-SGAP with MCM** When $|J|=|I|$, e_1=1,000, and ε=0.20

#	$\|I\|$	$\|J\|$	$\|S\|$	b_1	Optimal Objective Function Value	Best CPU Time (Seconds)	Initial $(N,\tilde{N},M)$	Final $(N,\tilde{N},M)$	Best Objective Function Value for **BNPDSTKP-SGAP with MCM** $(h_{\tilde{N}}((\hat{X}_N)^*))$	Average Objective Function Value $(\bar{h}_N)$	Tolerance Ratio $\left(\frac{h_{\tilde{N}}((X_N)^*}{\bar{h}_N}-1\right)$	Near-Optimality $\left(\frac{h_{\tilde{N}}((X_N)^*}{h(X^*)}\right)$	CPU Time for **BNPDSTKP-SGAP with MCM** (Seconds)
15	10	10	1024	3	51895	161.2565	(128,256,2)	(128,256,2)	52021.9531	51340.3125	0.0133	1.0024	1.1172
16	10	10	1024	6	28645	40.6587	(128,256,2)	(128,256,2)	29033.6719	27838.3594	0.0429	1.0136	2.3843
17	10	10	1024	9	12895	21.5317	(128,256,2)	(128,256,2)	12963.3594	12217.2656	0.0611	1.0053	1.6341
18	10	10	1024	12	4645	6.6384	(128,256,2)	(128,256,2)	4475.0781	4311.0156	0.0381	0.9634	0.7842
19	10	10	1024	15	895	3.9861	(128,256,2)	(128,256,2)	941.8750	795.3906	0.1842	1.0524	1.2020
20	10	10	1024	18	145	2.2201	(128,256,2)	(128,256,2)	145	145	0	1	0.3368
21	11	11	2048	3	63915	912.8344	(256,512,2)	(256,512,2)	64111.2891	63298.7891	0.0128	1.0031	6.1818
22	11	11	2048	6	37665	253.9332	(256,512,2)	(256,512,2)	37758.7500	37411.0938	0.0093	1.0025	11.6762
23	11	11	2048	9	18915	129.5042	(256,512,2)	(256,512,2)	18715.7813	18581.0156	0.0073	0.9895	7.0831
24	11	11	2048	12	7665	23.9283	(256,512,2)	(256,512,2)	7752.8906	7269.4922	0.0665	1.0115	2.5560
25	11	11	2048	15	2415	27.6825	(256,512,2)	(256,512,2)	2394.4922	2192.3438	0.0922	0.9915	2.9398
26	11	11	2048	18	165	9.6733	(256,512,2)	(256,512,2)	165	165	0	1	0.7528

3.7.8.4. $|J|=1.5|I|$ and $e_1=100$

Table 3.21 displays the results for this case. Note that Experiments 29, 32, and 34 do not terminate using the initial values of (N, $\tilde{N}$, M). The final values for the number of replications for these experiments are 3, 4, and 3, respectively. The tolerance ratio for all the experiments is within 0.05, except for experiment 37 for which it is 0.084. Besides, all the solutions obtained by the MCM-based method are within 2% of the "true" objective function value.

The cpu times for **BNPDSTKP-SGAP with MCM** shown in the last column are much smaller than those for the best of the other methods shown in the 7th column. Their ratios are plotted in Figure 3.16. On average, **BNPDSTKP-SGAP with MCM** results in 21.4-fold savings in cpu time over the best of the other methods.

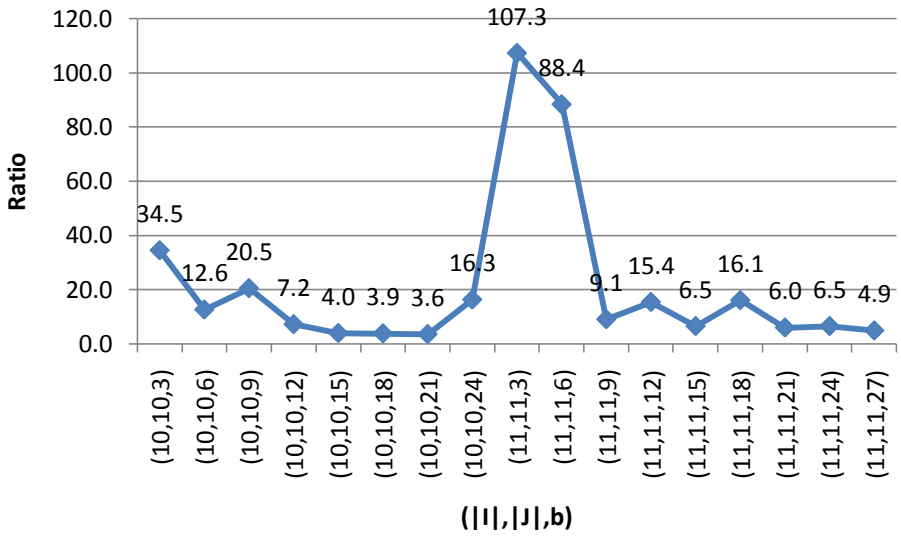

Figure 3.16. Values of Ratio= $\frac{\text{Best CPU Time Value Given in Column 7 of Table 3.19}}{\text{CPU Time Value Given in Column 14 of Table 3.19}}$ vs. ($|I|$,$|J|$,b_1)

Table 3.21. Results for **BNPDSTKP-SGAP with MCM** When $|J|=1.5|I|$, $e_1=100$, and $\varepsilon=0.20$

#	$\|I\|$	$\|J\|$	$\|S\|$	b_1	Optimal Objective Function Value	Best CPU Time (Seconds)	Initial $(N,\tilde{N},M)$	Final $(N,\tilde{N},M)$	Best Objective Function Value for **BNPDSTKP-SGAP with MCM** $(h_{\tilde{N}}((\hat{X}_N)^*))$	Average Objective Function Value $(\bar{h}_N)$	Tolerance Ratio $\left(\frac{h_{\tilde{N}}((X_N)^*}{\bar{h}_N}-1\right)$	Near-Optimality $\left(\frac{h_{\tilde{N}}((X_N)^*}{h(X^*)}\right)$	CPU Time for **BNPDSTKP-SGAP with MCM** (Seconds)
22	10	15	1024	3	14130	213.3752	(128,256,2)	(128,256,2)	14228.2422	13937.2266	0.0209	1.0070	6.1804
23	10	15	1024	6	11130	235.4049	(128,256,2)	(128,256,2)	11069.4531	10907.7344	0.0148	0.9946	18.6389
24	10	15	1024	9	8130	176.5764	(128,256,2)	(128,256,2)	8130.9766	8017.6953	0.0141	1.0001	8.5974
25	10	15	1024	12	5130	64.4608	(128,256,2)	(128,256,2)	5262.2266	5144.6484	0.0229	1.0258	8.8952
26	10	15	1024	15	2930	35.1975	(128,256,2)	(128,256,2)	2888.0078	2832.3438	0.0197	0.9857	8.8853
27	10	15	1024	18	1380	33.5875	(128,256,2)	(128,256,2)	1408.1250	1298.5547	0.0844	1.0204	8.7078
28	10	15	1024	21	480	7.4132	(128,256,2)	(128,256,2)	472.9688	454.2188	0.0413	0.9854	2.0716
29	10	15	1024	24	255	6.5182	(128,256,2)	(128,256,3)	255	255	0	1	0.3992
30	11	17	2048	3	18256	1701.9726	(256,512,2)	(256,512,2)	18193.0117	18001.3125	0.0106	0.9965	15.8547
31	11	17	2048	6	14956	1454.9531	(256,512,2)	(256,512,2)	14843.7930	14650.6289	0.0132	0.9925	16.4600
32	11	17	2048	9	11656	1206.2793	(256,512,2)	(256,512,4)	11656.5859	11503.4609	0.0133	1.0001	133.0044
33	11	17	2048	12	8356	960.0384	(256,512,2)	(256,512,2)	8385.1016	8343.2070	0.0050	1.0035	62.2963
34	11	17	2048	15	5106	416.1403	(256,512,2)	(256,512,3)	5100.3359	5095.6484	0.0009	0.9989	63.5626
35	11	17	2048	18	3156	340.6742	(256,512,2)	(256,512,2)	3134.3203	3025.9219	0.0358	0.9931	21.1462
36	11	17	2048	21	1506	108.8136	(256,512,2)	(256,512,2)	1511.8594	1445.9414	0.0456	1.0039	18.1156
37	11	17	2048	24	531	39.2618	(256,512,2)	(256,512,2)	527.1914	513.7148	0.0262	0.9928	6.0016
38	11	17	2048	27	306	24.0763	(512,1024,2)	(512,1024,2)	306	306	0	1	4.9032

3.7.8.5. $|J|=2|I|$ and $e_1=100$

Table 3.22 displays the results for **BNPDSTKP-SGAP with MCM** for this case. Additional replications are required for Experiments 43 and 44. The tolerance ratios for this case are within 0.03. The optimality gap for all the experiments is within 2%. The ratios of the best cpu time and the cpu time required by **BNPDSTKP-SGAP with MCM** are plotted in Figure 3.17. On average, the **BNPDSTKP-SGAP with MCM** method results in 17-fold savings in cpu times. However, for the first time, the values of ratios are less than 1 for Experiments 34, 35, 36, and 43 (see the experiments circled in red at Figure 3.17). Note that the value of the ratio is high, i.e., **BNPDSTKP-SGAP with MCM** gets relatively more efficient for difficult experiments, when the value of b_1 is 3.

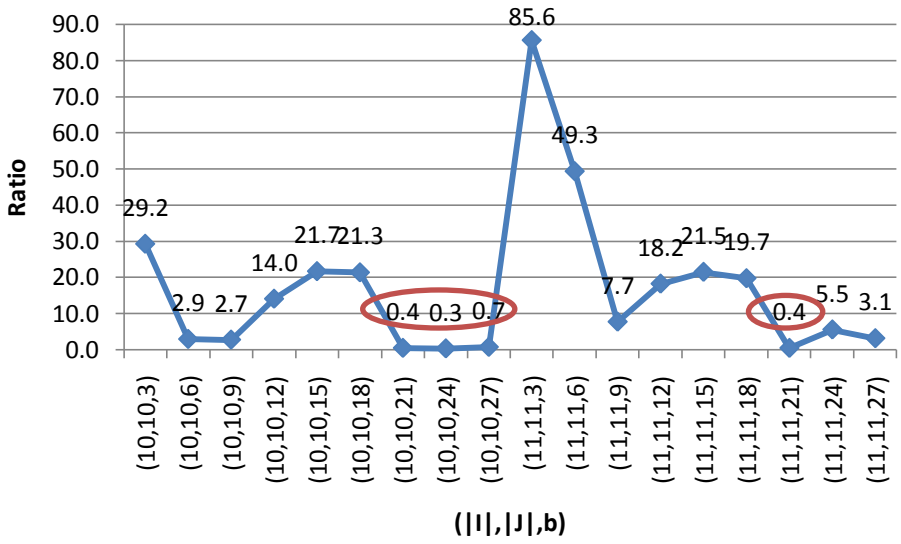

Figure 3.17. Values of Ratio= $\frac{\text{Best CPU Time Value Given in Column 7 of Table 3.20}}{\text{CPU Time Value Given in Column 14 of Table 3.20}}$ vs. $(|I|,|J|,b_1)$

Table 3.22. Results for **BNPDSTKP-SGAP with MCM** When $|J|=2|I|$, $e_1=100$, and $\varepsilon=0.20$

#	$\|I\|$	$\|J\|$	$\|S\|$	b_1	Optimal Objective Function Value	Best CPU Time (Seconds)	Initial $(N,\tilde{N},M)$	Final $(N,\tilde{N},M)$	Best Objective Function Value for **BNPDSTKP-SGAP with MCM** $(h_{\tilde{N}}((\hat{X}_N)^*))$	Average Objective Function Value $(\bar{h}_N)$	Tolerance Ratio $\left(\frac{h_{\tilde{N}}((X_N)^*}{\bar{h}_N}-1\right)$	Near-Optimality $\left(\frac{h_{\tilde{N}}((X_N)^*}{h(X^*)}\right)$	CPU Time for **BNPDSTKP-SGAP with MCM** (Seconds)
28	10	20	1024	3	26140	276.4762	(128,256,2)	(128,256,2)	26261.4844	25497.6172	0.0300	1.0046	9.4561
29	10	20	1024	6	23140	251.9473	(128,256,2)	(128,256,2)	23271.0547	22709.7266	0.0247	1.0057	86.9862
30	10	20	1024	9	20140	261.7328	(128,256,2)	(128,256,2)	19914.2188	19550.7422	0.0186	0.9888	96.0597
31	10	20	1024	12	17140	224.8747	(128,256,2)	(128,256,2)	17046.8359	16844.2969	0.0120	0.9946	16.0169
32	10	20	1024	15	14140	417.0509	(128,256,2)	(128,256,2)	14049.5703	13898.0078	0.0109	0.9936	19.2539
33	10	20	1024	18	11140	401.2938	(128,256,2)	(128,256,2)	11106.7969	10646.2500	0.0433	0.9970	18.8103
34	10	20	1024	21	8140	144.8954	(128,256,2)	(128,256,2)	8218.1250	8045.2734	0.0215	1.0096	333.9101
35	10	20	1024	24	5640	64.8208	(128,256,2)	(128,256,2)	5658.7500	5636.4844	0.0040	1.0033	218.7403
36	10	20	1024	27	4140	56.3152	(128,256,2)	(128,256,2)	3946.0547	3823.5938	0.0320	0.9532	79.9573
37	11	22	2048	3	31526	1702.7685	(256,512,2)	(256,512,2)	31701.3906	31026.5859	0.0217	1.0056	19.8938
38	11	22	2048	6	28226	1645.5419	(256,512,2)	(256,512,2)	28188.6953	27848.7539	0.0122	0.9987	33.3592
39	11	22	2048	9	24926	1751.0502	(256,512,2)	(256,512,2)	24687.8164	24323.0703	0.0150	0.9904	227.7540
40	11	22	2048	12	21626	1725.0364	(256,512,2)	(256,512,2)	21549.1445	21243.8711	0.0144	0.9964	94.7352
41	11	22	2048	15	18326	1525.6966	(256,512,2)	(256,512,2)	18393.1875	18094.7500	0.0165	1.0037	71.0607
42	11	22	2048	18	15026	1520.0658	(256,512,2)	(256,512,2)	15127.9531	14807.8359	0.0216	1.0068	77.1287
43	11	22	2048	21	11726	1000.8533	(256,512,2)	(256,512,3)	11827.2695	11723.1354	0.0089	1.0086	2322.0213
44	11	22	2048	24	8426	720.6506	(256,512,2)	(256,512,3)	8417.1133	8361.8724	0.0066	0.9989	130.2549
45	11	22	2048	27	6226	483.9349	(512,1024,2)	(256,512,2)	6265.5508	6077.4648	0.0309	1.0064	155.3079

3.7.9. Results for the BNPDSTKP-SGAP with MCM Method with Variations in *N*, $\tilde{N}$, and *M*

As mentioned earlier (with the use of 10 and 11-machine problems), the MCM-based method continues to be very effective when a problem becomes difficult to solve. This section deals with the experiments for which an optimal solution is unknown, unlike the experiments in Section 3.7.8 for which an optimal solution is known. We study the effect of changes in the number of scenarios for h_N^m in a replication, the number of scenarios for $h_{\tilde{N}}^m$ in a replication, and the number of replications, denoted as *N*, $\tilde{N}$, and *M*, respectively. We also observe the effect of these changes on the cpu time and on the best objective function value achieved. To study the effect of these changes, we consider 9 experiments where the number of machines and jobs are 13 and 26, respectively, and the machine capacity varies from 3 to 27.

3.7.9.1. Variation in *N*

Table 3.23. Best Objective function values with Variation in *N*

($\|I\|$,$\|J\|$,b_1)	*N*=32	*N*=64	*N*=128	*N*=256	*N*=512
(13,26,3)	43041.25	44254.14	43570.55	43881.39	43784.02
(13,26,6)	39556.88	39729.14	40153.16	40321.04	39838.61
(13,26,9)	34670.16	36001.02	35467.42	36124.55	35704.68
(13,26,12)	32008.44	31956.48	31744.57	32281.00	31914.74
(13,26,15)	27869.38	27424.06	28059.80*	28438.03	28164.88
(13,26,18)	24444.38	24480.70	23931.88	23935.88*	24027.14
(13,26,21)	20399.84	20326.41	20506.88	20485.68	20283.83
(13,26,24)	18345.16	16599.45	18265.47	16404.53	18349.31
(13,26,27)	15338.91*	15512.34	15385.39	15460.29	15403.55*
*: Initial *M*=2, but Final *M*=3					

Table 3.24. Confidence Intervals on Optimality Gap with Variation in N

($\|I\|$,$\|J\|$,b_1)	N=32	N=64	N=128	N=256	N=512
(13,26,3)	4116.90	3481.35	1663.91	1649.94	915.53*
(13,26,6)	2420.18	2207.28	2004.44	1303.71	449.00*
(13,26,9)	4861.65	1873.17	1561.13	1636.57	622.15*
(13,26,12)	4097.49	2871.43	1317.17	1536.83	1007.86*
(13,26,15)	2353.14	1267.64	855.04	1627.27	709.14*
(13,26,18)	2007.98	1809.61	1377.75	1175.54	640.64*
(13,26,21)	1054.70	1763.70	2398.33	433.42	338.12*
(13,26,24)	3059.05	865.02	1841.74	460.14*	4707.30
(13,26,27)	3399.39	4349.28	2492.13*	2700.80	3560.60
AVERAGE	3041.6	2276.5	1723.5	1391.6	1439.0
*: The least confidence interval on each case					

We change N from 32 to 512 while $\tilde{N}$ is fixed at $2N$. Tables 3.23 and 3.24 summarize the results of the best objective function values and the confidence intervals on the optimality gap, denoted by $h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)$ and $\left[0, \mathrm{Max}\left[0, h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right) - \bar{h}_N\right] + z_{1-\alpha}\sqrt{\mathrm{V}\left[h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)\right] + \mathrm{V}\left[\bar{h}_N\right]}\right]$, respectively. The starting value of M is two and is incremented by one as it needs to be. For example, when N=512 and ($|I|$, $|J|$, b_1)=(13,26,27), the first set of replications fails to satisfy the stopping criterion. The final number of replications for this experiment is three.

There are four cases that end up with M=3 (see the yellow cells in Table 3.23) while the other cases end up with M=2. As the machine capacity increases, the "true" optimal objective function values gets smaller or remains the same, as expected. Note that this behavior is maintained for all values of N.

A plot of the confidence intervals is given in Figure 3.18. When N increases, the confidence interval is expected to get smaller. The confidence intervals in Table 3.24 are evaluated with 99% confidence, i.e., $\alpha = 0.01$. Note that N=512 gives the best confidence interval 7 times for experiments (13,26,3) through (13,26,21). The graph for N=32 shows the most volatility. On the other hand, the graphs for N=128, N=256, and N=512 show relatively a great amount of stability by remaining at the bottom of the chart. We can recognize this trend from Table 3.24 in the last row where the average of the values of the confidence interval for each case is presented. For instance, the cases when N=256 and N=512 give lower value of the average than any other cases.

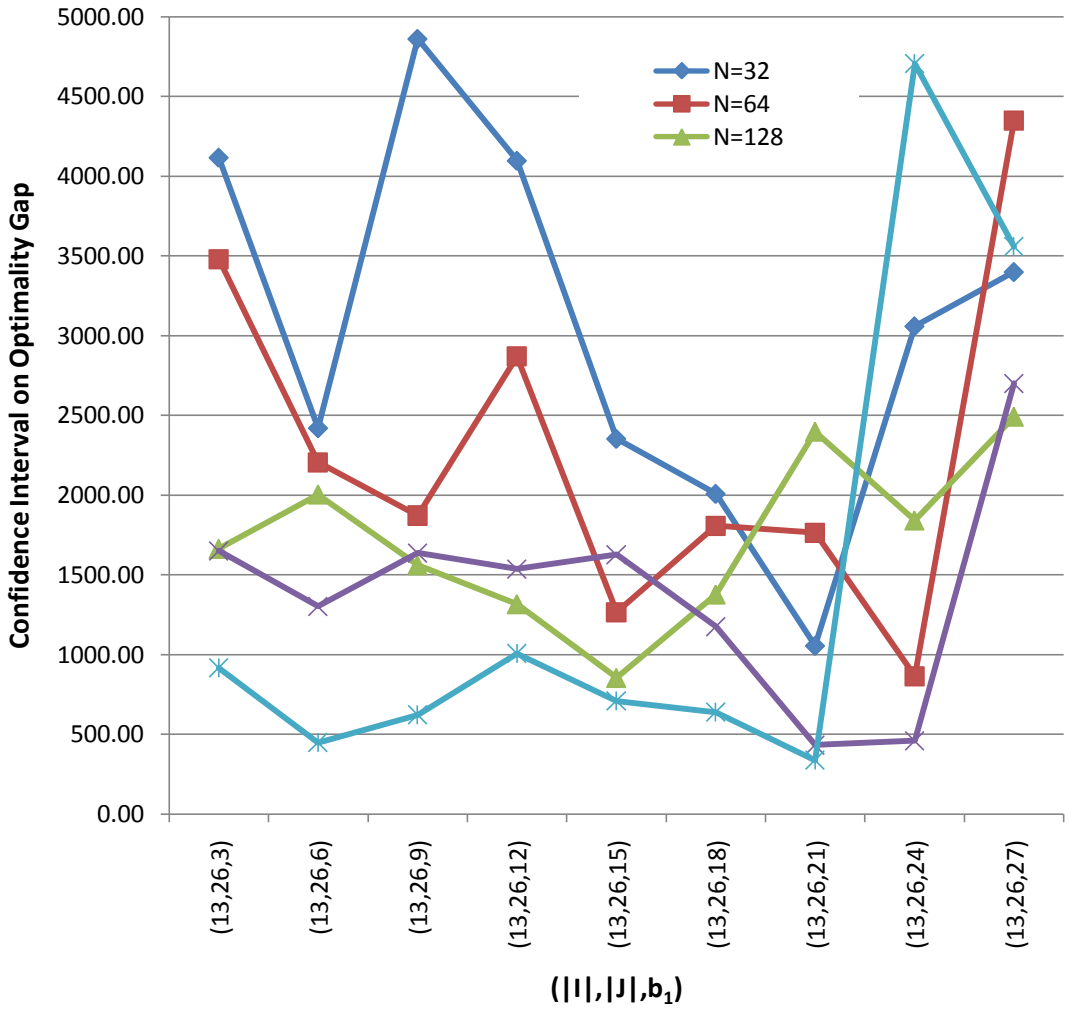

Figure 3.18. Confidence Interval on Optimality Gap with Changes in N

3.7.9.2. Variation in $\tilde{N}$

The value of $\tilde{N}$ is increased from 512 to 8192 while N and initial M are set at 256 and 2, respectively. Tables 3.25 and 3.26 contain the best objective function values and the confidence intervals on the optimality gap with changes in $\tilde{N}$, respectively. Referring to Figure 3.19, $\tilde{N}$=8192 gives the least confidence interval six times, while the use of $\tilde{N}$=1024 results in the least interval three times. However, the case of $\tilde{N}$=4096 seems the most erratic.

Table 3.25. Best Objective function values with Variation in $\tilde{N}$

($\|I\|$,$\|J\|$,b_1)	$\tilde{N}$=512	$\tilde{N}$=1024	$\tilde{N}$=2048	$\tilde{N}$=4096	$\tilde{N}$=8192
(13,26,3)	43881.39	43653.21	43827.68	43824.86	43767.19
(13,26,6)	40321.04	39808.68	39673.21	39912.56	39788.41
(13,26,9)	36124.55	36000.48	36048.65	35978.87	35860.12
(13,26,12)	32281	32028.65	31959.15	32074.75	32137.66
(13,26,15)	28438.03	28072.4	28200.63	28240.66	28190.45
(13,26,18)	23935.88	24245.84	24202.85	24248.26	24282.63
(13,26,21)	20485.68	20544.52	20435.9	20339.16	20361.79
(13,26,24)	16404.53	16459.12*	16396.08	18475.8	16482.59
(13,26,27)	15460.29	15490.62	15532.88**	15434.84	15455.18

* : Initial M=2, but Final M=6
**: Initial M=2, but Final M=3

Table 3.26. Confidence Intervals on Optimality Gap with Variation in $\tilde{N}$

($\|I\|$,$\|J\|$,b_1)	$\tilde{N}$=512	$\tilde{N}$=1024	$\tilde{N}$=2048	$\tilde{N}$=4096	$\tilde{N}$=8192
(13,26,3)	1649.936	848.9528*	884.7091	1210.921	1012.522
(13,26,6)	1303.707	539.2868*	601.9303	855.6278	1004.041
(13,26,9)	1636.567	1389.351	1162.622	1564.571	806.0492*
(13,26,12)	1536.834	538.0084*	971.0265	788.9124	672.4606
(13,26,15)	1627.272	787.756	727.3295	762.5339	522.149*
(13,26,18)	1175.544	424.1035	762.2197	369.386*	608.1701
(13,26,21)	433.4203	982.1563	842.5884	340.383	244.2655*
(13,26,24)	460.1439	964.2913	444.2407	5052.546	382.1437*
(13,26,27)	2700.802	2441.925	3022.532	2877.404	2322.256*
AVERAGE	1391.6	990.6	1046.6	1535.8	841.6

*: The least confidence interval on each case

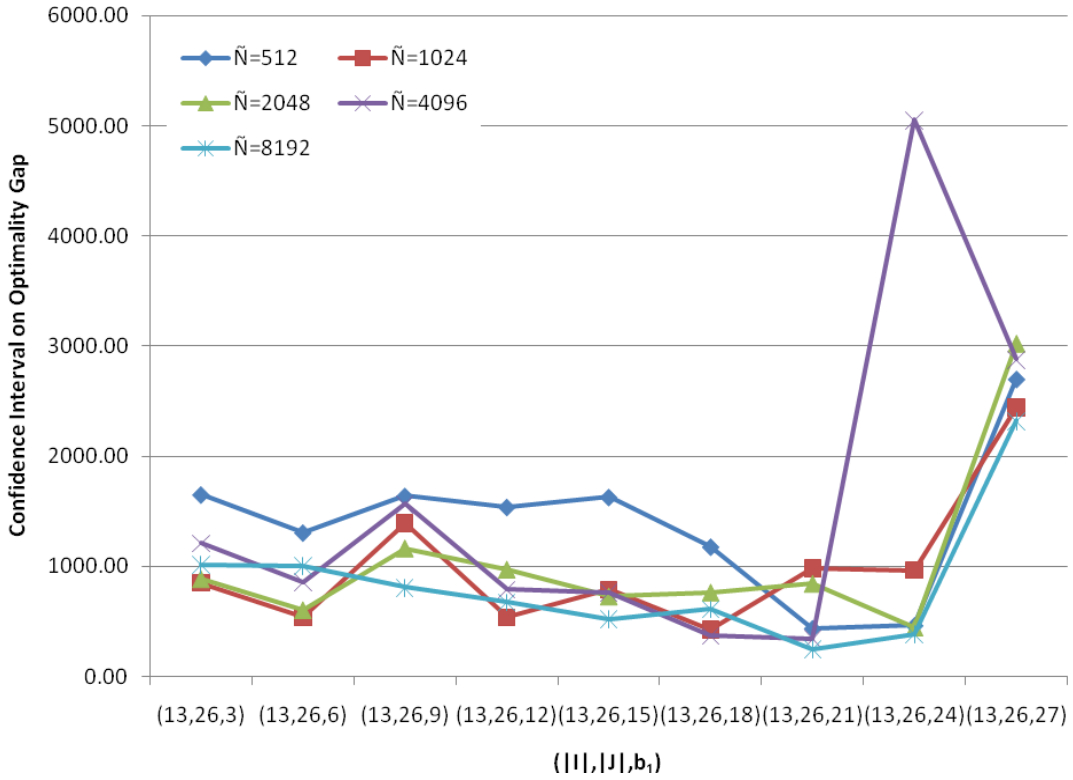

Figure 3.19. Confidence Interval on Optimality Gap with Variation in $\tilde{N}$

3.7.9.3. Variation in *M*

Next, we observe the effect of varying the initial number of replications, i.e., the initial value of *M*, which is varied from 2 to 5. The value of $|I|$=13, $|J|$=26, and b_1 is varied from 3 to 27. Table 3.27 summarizes the best objective function value and the confidential interval on the optimal gap for each case. Note that when initial value of *M* is 3, for the cases of (13,26,24) and (13,26,27), the value of *M* ends up with 6 and 4, respectively, but it still fails to obtain the least confidence interval for those cases. For the cases of (13,26,24) and (13,26,27), we obtain the least confidence interval when initial value of *M* is 4 and 5, respectively. When initial value of *M* is 2, 3, 4, and 5, the best confidence interval value is obtained once, twice, twice, and four times, respectively.

Table 3.27. Results for Variation in Initial M

Initial M	($\|I\|$,$\|J\|$,b_1)	Best Objective Function Value	Confidence Interval on Optimality Gap	Initial M	($\|I\|$,$\|J\|$,b_1)	Best Objective Function Value	Confidence Interval on Optimality Gap
2	(13,26,3)	43767.2	1012.52	4	(13,26,3)	43765.9	978.43
2	(13,26,6)	39788.4	1004.04	4	(13,26,6)	39896.3	924.24
2	(13,26,9)	35860.1	806.05	4	(13,26,9)	35922	901.88
2	(13,26,12)	32137.7	672.46*	4	(13,26,12)	32062.1	724.03
2	(13,26,15)	28190.5	522.15	4	(13,26,15)	28136	621.68
2	(13,26,18)	24282.6	608.17	4	(13,26,18)	24255.9	574.67
2	(13,26,21)	20361.8	244.27	4	(13,26,21)	20374.9	162.56*
2	(13,26,24)	16482.6	382.14	4	(13,26,24)	16499.6	170.34*
2	(13,26,27)	15455.2	2322.26	4	(13,26,27)	15426	2544.49
Average of Confidence Interval			841.56	Average of Confidence Interval			844.70
3	(13,26,3)	43679.1	1206.54	5	(13,26,3)	43752.2	746.75*
3	(13,26,6)	39767.9	1078.47	5	(13,26,6)	39753.8	756.01*
3	(13,26,9)	35892.9	695.10	5	(13,26,9)	35942.9	667.88*
3	(13,26,12)	32072.5	769.52	5	(13,26,12)	32059	878.89
3	(13,26,15)	28148.8	458.25*	5	(13,26,15)	28175.6	553.95
3	(13,26,18)	24186.8	189.53*	5	(13,26,18)	24238.6	460.05
3	(13,26,21)	20388.1	422.35	5	(13,26,21)	20394.3	505.67
3	(13,26,24)**	16475.9	989.06	5	(13,26,24)	16487.5	358.65
3	(13,26,27)***	15401.3	2525.49	5	(13,26,27)	15453.9	2232.60*
Average of Confidence Interval			926.03	Average of Confidence Interval			795.61

* : The least confidence interval on each case
** : Initial M=3, but Final M=6
***: Initial M=3, but Final M=4

Therefore, it appears that an initial *M* does not make a significant impact on the quality of the confidence interval. The average of the confidence intervals is 841.56, 926.03, 844.70, and 795.61 for the cases of initial *M* = 2, 3, 4, and 5, respectively. Moreover, at 95% confidence level, the mean of nine confidential interval value for the case of initial *M* = 5 is 795.61 (see the last row in Table 3.27) plus or minus 368.00, which implies that all the average confidence intervals are not significant with variation in initial *M*. The graphs for the other cases follow a similar pattern.

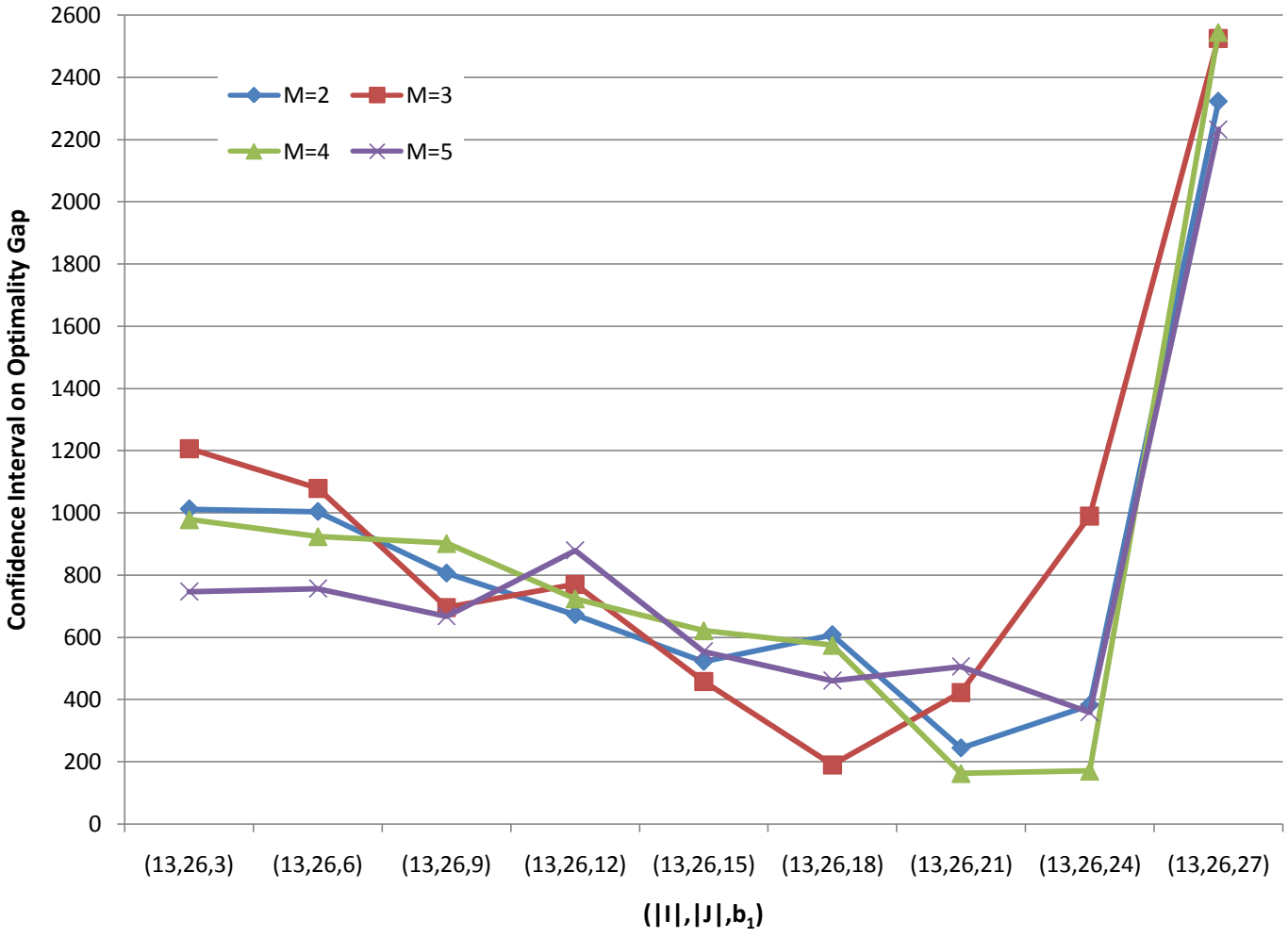

Figure 3.20. Confidence Interval on Optimality Gap with Changes in *M*

3.7.9.4. An Observation on the Relationship between the Quality of Confidential Interval and Size of Feasible Region

Note that in Figures 3.18, 3.19, and 3.20, the confidence interval has a tendency of becoming larger at higher values of b_1, especially when b_1=21, 24, and 27. Table 3.29 also summarizes this tendency. When b_1=27, the average of confidence interval becomes largest for every case in Figures 3.18, 3.19, and 3.20. Note also that, when the machine capacity is tight, (i.e., smaller b_1 values), machine utilizations increase, thereby, requiring extra hours to finish all the jobs. This is likely to make the SGAP have a smaller feasible region. Mathematically speaking, in the formulation **SGAP2**, $\sum_{j \in J} x_{ij}$ is most likely to be one for $\forall\ i \in I$. On the other hand, when the machine capacity is in excess, only a fraction of all the machines is used to finish all the jobs. Therefore, $\sum_{j \in J} x_{ij}$ for $\forall\ i \in I$ in **SGAP2** can be either one or zero. This can lead to a relatively larger solution space, compared with the one when $\sum_{j \in J} x_{ij} = 1$ for $\forall\ i \in I$.

Table 3.28. Effect of Variation in b_1 on Confidence Interval

	Average Value of Confidence Intervals			
($\|I\|$,$\|J\|$,b_1)	Table 3.24	Table 3.26	Table 3.27	Total Average
(13,26,3)	2365.53	1121.41	986.06	1527.07
(13,26,6)	1676.92	860.92	940.69	1175.14
(13,26,9)	2110.93	1311.83	767.73	1441.77
(13,26,12)	2166.16	901.45	761.23	1313.07
(13,26,15)	1362.45	885.41	539.01	956.81
(13,26,18)	1402.30	667.88	458.11	870.24
(13,26,21)	1197.65	568.56	333.71	726.14
(13,26,24)	2186.65	1460.67	475.05	1438.34
(13,26,27)	3300.44	2672.98	2406.21	2820.85

Now, the length of the confidence interval is equal to

$\left[0, \text{Max}\left[0, h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right) - \bar{h}_N\right] + z_{1-\alpha}\sqrt{\text{V}\left[h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)\right] + \text{V}\left[\bar{h}_N\right]}\right]$. Its quality is determined by the following two terms: $h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right) - \bar{h}_N$ and $\text{V}\left[h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)\right] + \text{V}\left[\bar{h}_N\right]$. If the objective function values obtained from each replication are close to each other, then the values of both terms get smaller; otherwise, if they are far from each other, then the values of both terms become larger. The MCM-based method that selects solutions from a smaller pool of feasible solutions has a greater chance of picking solutions that are too close to each other than those in the case of a larger pool. This is why **BNPDSTKP-SGAP with MCM** yields a poor quality of confidence intervals for experiments with larger b_1=27 like 21, 24, and 27.

3.7.10. Impact of Variance of Processing Time Distributions

In this section, we study the impact that the variance of processing time distributions makes on the optimal objective function value obtained. In particular, we want to study the impact that the use of a stochastic program makes over the use of expected values as the variance of a processing time distributions varies. We employ the **BNPDSTKP-SGAP** for this study since it has been shown to be the most effective one from the four optimum-seeking methods.

The machine rates that we have used in all the preceding experiments so far have been 1 and 1.5. In other words, if a machine normally takes 100 seconds for a job, it can take 150 seconds for the same job in its other status. In this section, we study the effect of variation in rates. The results are presented in Table 3.29. The first column represents the maximum rate, which are 1.5, 6, 10, and 20. In other words, for the same job, depending on a scenario, the processing time on a machine can be 1.5, 6, 10, and 20 times of its basic time. The 2nd, 3rd, and 4th columns represent the number of machines, the number of jobs, and machine capacity, respectively.

Table 3.29. Results for **BNPDSTKP-SGAP** with Variation in the Value of Maximum Rate

Max Rate	$\|I\|$	$\|J\|$	b_1	Objective Function Value			$\frac{\text{UpperBound}}{\textbf{SGAP2}}$ (%)	Max Rate	$\|I\|$	$\|J\|$	b_1	Objective Function Value			$\frac{\text{UpperBound}}{\textbf{SGAP2}}$ (%)
				SGAP2	**Expected SGAP2**	Upper Bound						**SGAP2**	**Expected SGAP2**	Upper Bound	
1.5	4	4	3	369.08	369.08	369.08	100.00%	6	4	4	3	897.37	897.37	897.37	100.00%
1.5	4	4	6	46.01	46.00	66.88	145.36%	6	4	4	6	345.54	196.61	495.17	143.30%
1.5	4	4	9	46.00	46.00	46.00	100.00%	6	4	4	9	116.45	46.00	289.15	248.29%
1.5	4	4	12	46.00	46.00	87.74	190.74%	6	4	4	12	48.74	46.00	183.07	375.62%
1.5	4	4	15	46.00	46.00	46.00	100.00%	6	4	4	15	48.60	46.00	49.15	101.13%
1.5	4	4	18	46.00	46.00	46.02	100.03%	6	4	4	18	47.64	46.00	49.01	102.87%
1.5	4	4	21	46.00	46.00	46.00	100.00%	6	4	4	21	47.51	46.00	48.88	102.88%
1.5	4	4	24	46.00	46.00	46.00	100.00%	6	4	4	24	46.27	46.00	48.74	105.33%
1.5	5	5	3	673.62	673.62	673.62	100.00%	6	5	5	3	989.96	900.37	989.96	100.00%
1.5	5	5	6	60.09	60.00	69.70	116.00%	6	5	5	6	289.53	60.02	346.09	119.53%
1.5	5	5	9	60.00	60.00	72.51	120.85%	6	5	5	9	79.07	60.00	278.62	352.37%
1.5	5	5	12	60.00	60.00	60.00	100.00%	6	5	5	12	62.29	60.00	360.42	578.66%
1.5	5	5	15	60.00	60.00	60.00	100.00%	6	5	5	15	60.34	60.00	586.04	971.30%
1.5	5	5	18	60.00	60.00	98.47	164.11%	6	5	5	18	60.05	60.00	527.53	878.43%
1.5	5	5	21	60.00	60.00	60.00	100.00%	6	5	5	21	60.05	60.00	60.34	100.47%
1.5	5	5	24	60.00	60.00	60.00	100.00%	6	5	5	24	60.03	60.00	60.05	100.04%
1.5	6	6	3	1125.93	1125.93	1125.93	100.00%	6	6	6	3	1299.05	1299.05	1299.05	100.00%
1.5	6	6	6	176.97	175.00	205.74	116.25%	6	6	6	6	197.03	175.00	208.63	105.89%
1.5	6	6	9	75.00	75.00	75.00	100.00%	6	6	6	9	75.37	75.00	128.84	170.94%
1.5	6	6	12	75.00	75.00	87.68	116.91%	6	6	6	12	75.05	75.00	116.78	155.60%
1.5	6	6	15	75.00	75.00	75.00	100.00%	6	6	6	15	75.00	75.00	141.70	188.93%
1.5	6	6	18	75.00	75.00	75.00	100.00%	6	6	6	18	75.00	75.00	161.31	215.07%
1.5	6	6	21	75.00	75.00	105.73	140.98%	6	6	6	21	75.00	75.00	156.85	209.13%
1.5	6	6	24	75.00	75.00	75.00	100.00%	6	6	6	24	75.00	75.00	185.68	247.57%

Table 3.29. Results for **BNPDSTKP-SGAP** with Variation in the Value of Maximum Rate (Continued)

Max Rate	$\|I\|$	$\|J\|$	b_1	Objective Function Value			$\frac{\text{UpperBound}}{\textbf{SGAP2}}$ (%)	Max Rate	$\|I\|$	$\|J\|$	b_1	Objective Function Value			$\frac{\text{UpperBound}}{\textbf{SGAP2}}$ (%)
				SGAP2	**Expected SGAP2**	Upper Bound						**SGAP2**	**Expected SGAP2**	Upper Bound	
10	4	4	3	668.42	578.07	668.42	100.00%	20	4	4	3	909.44	909.44	909.44	100.00%
10	4	4	6	193.95	46.00	367.66	189.56%	20	4	4	6	372.84	351.71	450.60	120.85%
10	4	4	9	71.20	46.00	377.98	530.91%	20	4	4	9	147.26	46.00	191.75	130.21%
10	4	4	12	51.83	46.00	459.36	886.32%	20	4	4	12	77.18	46.00	188.42	244.12%
10	4	4	15	47.24	46.00	47.24	100.00%	20	4	4	15	46.04	46.00	46.04	100.00%
10	4	4	18	47.21	46.00	47.21	100.00%	20	4	4	18	46.04	46.00	46.04	100.00%
10	4	4	21	46.98	46.00	299.06	636.60%	20	4	4	21	46.03	46.00	46.03	100.00%
10	4	4	24	46.95	46.00	282.10	600.86%	20	4	4	24	46.03	46.00	46.03	100.00%
10	5	5	3	1236.46	1198.35	1283.80	103.83%	20	5	5	3	1586.37	1560.05	1642.05	103.51%
10	5	5	6	412.63	180.18	533.72	129.34%	20	5	5	6	1112.69	860.04	1324.58	119.04%
10	5	5	9	92.20	60.00	204.47	221.76%	20	5	5	9	639.00	254.78	901.16	141.03%
10	5	5	12	61.08	60.00	776.66	1271.53%	20	5	5	12	388.03	60.00	723.67	186.50%
10	5	5	15	60.56	60.00	724.54	1196.46%	20	5	5	15	248.42	60.00	1177.04	473.81%
10	5	5	18	60.03	60.00	376.45	627.08%	20	5	5	18	108.81	60.00	1143.87	1051.28%
10	5	5	21	60.00	60.00	60.30	100.50%	20	5	5	21	60.06	60.00	60.06	100.00%
10	5	5	24	60.00	60.00	60.25	100.42%	20	5	5	24	60.06	60.00	60.06	100.00%
10	6	6	3	1589.70	1589.70	1589.70	100.00%	20	6	6	3	1935.23	1935.23	1935.23	100.00%
10	6	6	6	403.50	403.50	403.50	100.00%	20	6	6	6	1095.45	780.44	1455.22	132.84%
10	6	6	9	78.99	75.00	106.21	134.47%	20	6	6	9	255.68	75.00	919.92	359.79%
10	6	6	12	75.89	75.00	121.45	160.02%	20	6	6	12	125.73	75.00	1082.95	861.36%
10	6	6	15	75.01	75.00	129.43	172.56%	20	6	6	15	75.01	75.00	912.70	1216.77%
10	6	6	18	75.01	75.00	114.97	153.27%	20	6	6	18	75.01	75.00	1610.11	2146.60%
10	6	6	21	75.00	75.00	113.38	151.17%	20	6	6	21	75.00	75.00	1553.67	2071.44%
10	6	6	24	75.00	75.00	340.43	453.90%	20	6	6	24	75.00	75.00	2256.51	3008.58%

The 5th and 6th columns represent the optimal objective function value of the formulations **SGAP2** and **Expected SGAP2**, respectively. If we denote x_{ij}^{E}, $i \in I, j \in J$, as the optimal solution of x_{ij}, $i \in I, j \in J$, to the formulation **Expected SGAP2**, substituting x_{ij}^{E} for x_{ij}, $i \in I, j \in J$, in constraint set (3.3g), which is the machine capacity constraint set for **SGAP2**, allows us to obtain legitimate values of y_{is}, $i \in I, s \in S$. Let y_{is}^{E} be the solution of $y_{is,}$ $i \in I, s \in S$. In other words, the set of (x_{ij}^{E}, y_{is}^{E}) is feasible to the formulation **SGAP2**. Therefore, the objective function value obtained by (x_{ij}^{E}, y_{is}^{E}) gives us an upper bound for each experiment, which is presented in the 7th column, designated as the "Upper Bound" column. Therefore, the value in the "Upper Bound" column will be the true objective function value of the assignments made by the values of x_{ij}^{E}, $i \in I, j \in J$. The ratio of this solution to the optimal solution of **SGAP2**, depicted in the 5th column, is presented in the 8th column.

Note that, as expected, the optimal objective function value of **SGAP2** increases with the value of the maximum rate. When the maximum rate is 1.5, 6, 10, and 20, the values of the average optimal objective function value is 147.40, 216.91, 236.03, and 400.27, respectively. Therefore, the objective function value worsens with increase in variability of processing time. In other words, to reduce total expected cost, the variability of processing time should be decreased. Comparing the objective function value in the 5th column with that of the expected value solution in the 7th column, the former does better than the latter by 266.45% on average. This shows the advantage of using the stochastic program instead of the expected value solution. Also, note that the gap between the solution of **SGAP2** and that of **Expected SGAP2** gets bigger with increase in the value of the maximum rate. When the maximum rate is 1.5, 6, 10, and 20, **SGAP2** outperforms **Expected SGAP2** by 112.97%, 244.72%, 346.69%, and 548.66% on average, respectively. Therefore, the greater the variability in data, the better it is to use the stochastic program.

3.8. Concluding Remarks

Table 3.30 presents statistics on the number of times each of the methods (**BNC-SGAP**, **BNP-SGAP**, **BNPDST-SGAP**, **BNPDSTKP-SGAP**, **and BNPDSTKP-SGAP with MCM**) gives the least cpu time from among the experiments conducted. We have split the experiments into two groups. One consists of the entire set of experiments where the number of machines ranges from 7 to 11, and the other for which the number of machines ranges from 10 to 11; the latter because **BNPDSTKP-SGAP with MCM** was experimented only on those instances.

Table 3.30. Number of Experiments with the Best CPU Times among the Four Optimization Methods, and Results for the **BNPDSTKP-SGAP with MCM** Method

		BNC -SGAP	**BNP -SGAP**	**BNPDST -SGAP**	**BNPDSTKP -SGAP**	**BNPDSTKP -SGAP with MCM**	Total
$\|J\|=\|I\|$ and $e_1=10$	$\|I\|$=7~11	5	2	5	14	N/A	26
	$\|I\|$=10~11	3	0	1	8	12 (0.9859,1.0018)*	12
$\|J\|=\|I\|$ and $e_1=100$	$\|I\|$=7~11	2	4	1	19	N/A	26
	$\|I\|$=10~11	1	0	0	11	12 (0.9639,1.0028)*	12
$\|J\|=\|I\|$ and $e_1=1000$	$\|I\|$=7~11	3	5	2	16	N/A	26
	$\|I\|$=10~11	1	0	2	9	12 (0.9634,1.0524)*	12
$\|J\|=1.5\|I\|$ and $e_1=100$	$\|I\|$=7~11	7	5	5	21	N/A	38
	$\|I\|$=10~11	1	0	2	14	17 (0.9854,1.0258)*	17
$\|J\|=2\|I\|$ and $e_1=100$	$\|I\|$=7~11	12	2	19	12	N/A	45
	$\|I\|$=10~11	1	2	7	8	14 (0.9532,1.0096)*	18
Total	$\|I\|$=7~11	29	18	32	82	N/A	161
	$\|I\|$=10~11	7	2	12	50	67	71
*: (the lowest value of near-optimality, the highest value of near-optimality) from Tables 3.18 to 3.22							

Referring to Table 3.30, it is clear that, among the four optimum seeking methods, **BNPDSTKP-SGAP** gives the best results more often except for the case where $|J|=2|I|$ and $e_1=100$, for which **BNPDST-SGAP** gives better results more often. However, if we only consider the complicated cases where $|I|$ is 10 and 11, **BNPDSTKP-SGAP** is better in every case. In total, **BNPDSTKP-SGAP** gives superior performance over the other optimization methods in 82 out of 161 experiments.

BNC-SGAP seems quite competitive with **BNP-SGAP** and **BNPDST-SGAP**. However, as observed earlier, **BNC-SGAP** encounters large cpu times, requiring up to tens of thousands of seconds for some instances. Also, it fails to obtain optimal solutions for many experiments. Furthermore, **BNPDST-SGAP** outperforms **BNP-SGAP**, and **BNPDSTKP-SGAP** is overwhelmingly better than **BNPDST-SGAP**. As a result, the two features added to **BNP-SGAP**, namely, the dual stabilization technique and the greedy heuristic, make a positive impact on solving various types of the SGAPs. Table 3.31 provides pair-wise comparisons between **BNC-SGAP** and the other three optimum-seeking methods. As shown in the table, all the BNP-based methods perform better than the BNC-based method.

Table 3.31. **BNC-SGAP** vs. **BNP-SGAP**, **BNPDST-SGAP**, and **BNPDSTKP-SGAP** with respect to CPU Times

	BNC-SGAP vs. **BNP-SGAP**	**BNC-SGAP** vs. **BNPDST-SGAP**	**BNC-SGAP** vs. **BNPDSTKP-SGAP**	Total
$\|J\|=\|I\|$ and $e_1=10$	8 vs. 18	6 vs. 20	12 vs. 14	26
$\|J\|=\|I\|$ and $e_1=100$	7 vs. 19	8 vs. 18	7 vs. 19	26
$\|J\|=\|I\|$ and $e_1=1000$	6 vs. 20	6 vs. 20	5 vs. 21	26
$\|J\|=1.5\|I\|$ and $e_1=100$	15 vs. 23	16 vs. 22	7 vs. 31	38
$\|J\|=2\|I\|$ and $e_1=100$	14 vs. 31	13 vs. 32	18 vs. 27	45

Referring to Table 3.30, the column for **BNPDSTKP-SGAP with MCM** provides not only the largest number of experiments for which it performs better than all the other methods, but also, it gives tighter "near-optimality" range for each case given in Tables 3.18 to 3.22. For example, for the first case where $|J|=|I|$ and $e_1=10$, it performs better in all 12 experiments, and the "near-optimality" value range is (0.9859, 1.0018). That is, the objective function values remain within 2% of the "true" objective function values for all the problems.

In fact, out of 71 experiments, **BNPDSTKP-SGAP with MCM** performs better in 67 experiments. The worst "near-optimality" range obtained is (0.9634, 1.0524), which occurs for the case of $|J|=|I|$ and $e_1=100$. The objective function values that give are within 6% of the "true" optimal objective function values for all the problems.

In conclusion, based on the results and analysis for all the experiments presented in previous sections and summarized in Table 3.31, it is obvious that **BNPDSTKP-SGAP** gives the most dependable performance out of all optimum-seeking methods, and it effectively deals with various changes in parameters of the SGAP. Also, **BNP-SGAP**, **BNPDST-SGAP**, and **BNPDSTKP-SGAP** show relatively reliable and predictable performance compared with that of **BNC-SGAP**.

As regards the Monte Carlo-based method, it requires several order of magnitude smaller cpu times with a very small optimality gap (5%). We also observe positive effects of increasing the value of either N or $\tilde{N}$ on the confidence interval and on the optimality gap for each experiment.

Chapter IV. A Branch-and-Price Method for the Hospital Staff Scheduling Problem (HSSP)

The problem that we address in this chapter can concisely be defined as follows: Given a set of operating rooms, the time taken by an operation and the operation to be performed by each surgeon, determine the start time of each operation and the operating room in which to perform that operation so as to maximize the net revenue earned by the hospital. We call this problem as the hospital staff scheduling problem (HSSP). We use the terms *hospital staff* and *facilities*, specifically, to refer to surgeons and operating rooms, respectively. The hospital earns a profit from each operation that is performed. However, if the operation is performed during overtime, a penalty cost is incurred, leading to the objective of maximizing the net revenue by the hospital. We address the deterministic version of the HSSP in which the time required to perform an operation is assumed to be deterministic. In fact, we consider three types of the deterministic HSSP.

First, we address a general hospital staff scheduling problem which incorporates staff, operations to be performed, facilities, and operation times. A hospital has the operating rooms available for the surgeons to perform operations. The overall objective of most hospitals is to maximize the net revenue given that only a limited number of operating rooms are available for use by the surgeons.

In reality, a hospital divides surgeons into two classes, designated as Type I and Type II surgeons. Type I surgeons are those who bring more patients (and revenue) to the hospital than those in the Type II category. Consequently, Type I surgeons are given a higher priority for scheduling their operations than the Type II surgeons. The HSSP under this classification of surgeons is the second problem that we address. Mutually-agreed-upon time slots are assigned to Type I surgeons, and the operations to be performed by Type II surgeons are scheduled around these time slots. In the third problem, the Type I surgeons pre-select their time slots and an operating room for their operations.

4.1. HSSP (General): Integration of Surgeons, Operations, Operating Rooms and Operation Times

4.1.1. Formulation for the HSSP (General)

There are four elements of the HSSP, namely, surgeons, operations to be performed, operating rooms, and time for each operation. In the general version of the HSSP, there are no restrictions on the scheduling of operations to be performed by a surgeon. Next, we present a formulation for this problem. We use the following notation.

Indices and Sets

i : Surgeon, $i \in I$

j : Operation to be performed by a surgeon, $j \in J$

m : Operating room, $m \in M$

t : Period, t=1,…,TP

Parameters and Variables

TP : Planning period including regular time and overtime, t=1,…,TP

T : Regular time if t=1,..,T; overtime if T+1 $\leq t \leq TP$

r_j : Profit accrued from operation j, $\forall j \in J$

c_i : Extra cost per unit time incurred if surgeon i works overtime, $\forall i \in I$

d_{ij} : Time required to perform operation $j \in J$ by surgeon $i \in I$

$J(i)$: Set of operations that surgeon i can perform, $\forall i \in I$

$I(j)$: Set of surgeons who can perform operation j, $\forall j \in J$

x_{ijmt} : Binary variable, =1 if surgeon $i \in I$ starts operation $j \in J$ in operating room $m \in M$ during period t, t=1,…,TP; =0, otherwise

y_{ijmt} : Binary variable, =1 if surgeon $i \in I$ performs operation $j \in J$ in operating room $m \in M$ during period t, t=1,…,TP; =0, otherwise

HSSP (General)

$$\text{Maximize} \quad \sum_{j\in J}\sum_{i\in I(j)}\sum_{m\in M}\sum_{t=1}^{TP} r_j x_{ijmt} - \sum_{j\in J}\sum_{i\in I(j)}\sum_{m\in M}\sum_{t=T+1}^{TP} c_i(t-T)y_{ijmt} \tag{4.1a}$$

$$\text{Subject to} \quad \sum_{i\in I(j)}\sum_{m\in M}\sum_{t=1}^{TP} x_{ijmt} \le 1 \qquad \forall j \in J \tag{4.1b}$$

$$\sum_{j\in J}\sum_{i\in I(j)} y_{ijmt} \le 1 \qquad \forall m \in M, \forall t = 1,...,TP \tag{4.1c}$$

$$\sum_{j\in J(i)}\sum_{m\in M} y_{ijmt} \le 1 \qquad \forall i \in I, \forall t = 1,...,TP \tag{4.1d}$$

$$x_{ijm1} = y_{ijm1} \qquad \forall j \in J, \forall i \in I(j), \forall m \in M \tag{4.1e}$$

$$x_{ijmt} \ge y_{ijmt} - y_{ijm(t-1)} \qquad \forall j \in J, \forall i \in I(j), \forall m \in M, \forall t = 2,...,TP \tag{4.1f}$$

$$\sum_{m\in M}\sum_{t=1}^{TP} y_{ijmt} = d_{ij}\sum_{m\in M}\sum_{t=1}^{TP} x_{ijmt} \qquad \forall j \in J, \forall i \in I(j) \tag{4.1g}$$

$$x_{ijmt}, y_{ijmt} \in \{0,1\} \qquad \forall j \in J, \forall i \in I(j), \forall m \in M, \forall t = 1,...,TP \tag{4.1h}$$

The first term in the objective function (4.1a) represents the net profit earned by the hospital as a result of the operations performed by surgeons, while the second term captures the total extra cost incurred during overtime. This includes the extra cost incurred by the hospital in having the staff work overtime. Constraint set (4.1b) ensures that an operation starts at most once by a surgeon in an operating room during a period. Constraint set (4.1c) asserts that an operating room must be occupied by at most one surgeon for an operation during a period. Constraint set (4.1d) ensures that a surgeon performs at most one operation in an operating room during a period. Constraint sets (4.1e) and (4.1f) capture the fact that when $y_{ijmt} = 1$ and $y_{ijm(t-1)} = 0$, then $x_{ijmt} = 1$, while if $y_{ijmt} = 1$ and $y_{ijm(t-1)} = 1$, then x_{ijmt} is necessarily equal to o in light of (4.1b). Similarly, if $y_{ijmt} = 0$ and $y_{ijm(t-1)} = 1$, then $x_{ijmt} = 0$. Constraint set (4.1g) ensures the required duration of operation *j* performed by surgeon *i*.

Proposition 4.1. The model **HSSP (General)** is a valid formulation for a hospital staff scheduling problem.

Proof. We show that, in general, all legitimate schedules satisfy all the constraint sets from (4.1b) to (4.1h), and that, an infeasible solution will violate at least one of them.

Suppose that we have a feasible schedule for the planning period, *TP*. Let $A(i)$ be the set of jobs assigned to surgeon i and $A(j)$ be the surgeon to whom operation j is assigned in that schedule. Consider the following conditions which a feasible schedule must satisfy.

Restrictions on Surgeons

i. A surgeon conducts at most one operation in one operating room during a period.

$$\sum_{j\in A(i)}\sum_{m\in M} y_{ijmt} \le 1. \qquad \forall i \in I\ ,\ \forall t = 1,...,TP \qquad (4.2a)$$

ii. Once a surgeon starts an operation, there should not be any interruption until it is completed, that is, there is no restart for an operation.

$$\sum_{m\in M}\sum_{t=1}^{TP} x_{ijmt} = 1. \qquad \forall i \in I\ ,\ \forall j \in A(i) \qquad (4.2b)$$

Restrictions on Operations

iii. An operation is performed only by one surgeon in an operating room once it is assigned.

$$\sum_{i\in I(j)}\sum_{m\in M}\sum_{t=1}^{TP} x_{ijmt} \le 1. \qquad \forall j \in J \qquad (4.2c)$$

iv. Once an operation is started, it is not stopped until it is completed.

$$\sum_{m\in M}\sum_{t=1}^{TP} x_{ijmt} = 1. \qquad \forall j \in J\ ,\ i = A(j) \qquad (4.2d)$$

v. The time required by an operation should equal its pre-specified value.

$$\sum_{m\in M}\sum_{t=1}^{TP} y_{ijmt} = d_{ij}\,. \qquad \forall j \in J\,,\ i = A(j) \qquad (4.2e)$$

Restriction on Operating Rooms

vi. The number of operations performed in an operating room during a period should be no more than one.

$$\sum_{i\in I}\sum_{j\in A(i)} y_{ijmt} \le 1. \qquad \forall m \in M\,,\ \forall t = 1,...,TP \qquad (4.2f)$$

Restriction on Operation Times

vii. Every assigned operation has its own starting time.

$$x_{ijm1} = y_{ijm1}, \qquad \forall j \in J\,,\ i = A(j)\,,\ \forall m \in M \qquad (4.2g)$$

$$x_{ijmt} \ge y_{ijmt} - y_{ijm(t-1)}\,. \qquad \forall j \in J\,,\ i = A(j)\,,\ \forall m \in M\,,\ \forall t = 2,...,TP \qquad (4.2h)$$

Note that $A(i) \subset J(i)$ and $A(j) \subset I(j)$, and all the other pairs of i and j are not assigned for the given schedule. Denote the unassigned pairs by $\{(i, j) : \forall i \in I, \forall j \in J(i) \setminus A(i)\}$. Then, constraint set (4.1d) can be split into constraint set (4.2a) and the following constraint set:

$$\sum_{j\in J(i)\setminus A(i)}\sum_{m\in M} y_{ijmt} = 0\,. \qquad \forall i \in I\,, \forall t = 1,...,TP \qquad (4.2i)$$

Similarly, (4.1b) can be equivalently expressed as a combination of either (4.2b) or (4.2d) and the following constraint set:

$$\sum_{i\in I(j)\setminus A(j)}\sum_{m\in M}\sum_{t=1}^{TP} x_{ijmt} = 0\,. \qquad \forall j \in J \qquad (4.2j)$$

Also, (4.1c) can be expressed as a combination of (4.2f) and the following constraint set:

$$\sum_{i \in I} \sum_{j \in I(i) \setminus A(i)} y_{ijmt} = 0. \qquad \forall m \in M,\ \forall t = 1,...,TP \qquad \text{(4.2k)}$$

Next, (4.1e) and (4.1f) can be equally expressed as the combination of (4.2g) and (4.2h) and the following constraint set:

$$x_{ijmt} = y_{ijmt} = 0. \qquad \forall j \in J,\ i \in I(j) \setminus A(j),\ \forall m \in M,\ \forall t = 1,...,TP \qquad \text{(4.2l)}$$

Finally, (4.1g) is rearranged with (4.2e) and the following constraint set:

$$\sum_{m \in M} \sum_{t=1}^{TP} y_{ijmt} = 0. \qquad \forall j \in J,\ i = I(j) \setminus A(j) \qquad \text{(4.2m)}$$

Note that (4.2a), (4.2b), (4.2d), (4.2e), (4.2f), (4.2g), and (4.2h) are for the pairs (i, j) that are assigned and (4.2i), (4.2j), (4.2k), (4.2l), and (4.2m) are for the pairs that are not assigned in the given schedule. Thus, a feasible schedule satisfies all the constraint sets in **HSSP (General)**.

Next, we show that **HSSP (General)** excludes all infeasible schedules by enumerating all cases that result in infeasible schedules. If **HSSP (General)** is a valid formulation, then it should not yield an infeasible schedule. There are the following cases, which can lead to infeasible examples, where a pair of (i, j) in a cell represents surgeon i conducting operation j.

CASE (1): An operation is assigned twice. This is depicted in the following figure. We have

		Period			
		1	2	3	4
Operating Room	1	(1,1)	(1,1)		
	2			(2,1)	(2,1)

$$y_{1111} = y_{1112} = y_{2123} = y_{2124} = 1$$

$$x_{1111} = y_{1111} \qquad \Rightarrow \qquad x_{1111} = 1$$

$$x_{1112} \geq y_{1112} - y_{1111} \quad \Rightarrow \quad x_{1112} \geq 1-1 \quad \Rightarrow \quad x_{1112} = 0$$

$$x_{2123} \geq y_{2123} - y_{2122} \quad \Rightarrow \quad x_{2123} \geq 1-0 \quad \Rightarrow \quad x_{2123} = 1$$

$$x_{2124} \geq y_{2124} - y_{2123} \quad \Rightarrow \quad x_{2124} \geq 1-1 \quad \Rightarrow \quad x_{2124} = 0$$

This violates constraints (4.1b) since $x_{1111} + x_{2123} = 2$.

CASE (2): More than one operation is scheduled at the same time in an operating room. In the following example, operations 1 and 2 are scheduled in operating room 1 during period 2 at the same time.

		Period			
		1	2	3	4
Operating Room	1	(1,1)	(1,1), (2,2)	(2,2)	
	2				

We have $y_{1112} = 1$ and $y_{2212} = 1$, which gives $y_{1112} + y_{2212} = 2$, thereby violating (4.1c).

CASE (3): A surgeon cannot perform two different operations at the same time. In the illustration below, surgeon 1 is assigned to two operations during period 2.

		Period			
		1	2	3	4
Operating Room	1	(1,1)	(1,1)		
	2		(1,2)	(1,2)	

We have $y_{1111} = y_{1112} = y_{1222} = y_{1223} = 1$, thereby violating (4.1d) since $y_{1112} + y_{1222} = 2$.

CASE (4): An operation is scheduled in more than one operating room. Consider the situation illustrated in the figure below. The operation (1,1) is scheduled in two operating rooms.

		Period			
		1	2	3	4
Operating Room	1	(1,1)			
	2		(1,1)		

We have $x_{1111} = x_{1122} = 1$, which violates (4.1b) since $x_{1111} + x_{1122} = 2$.

CASE (5): An operation, once started, is paused and restarted in a different time slot. Again, consider the illustration below. We have $y_{1111} = y_{1113} = 1$

		Period			
		1	2	3	4
Operating Room	1	(1,1)		(1,1)	
	2				

$$x_{1111} = y_{1111} \quad \Rightarrow \quad x_{1111} = 1$$

$$x_{1112} \geq y_{1112} - y_{1111} \quad \Rightarrow \quad x_{1112} \geq 0 - 1 \quad \Rightarrow \quad x_{1112} = 0$$

$$x_{1113} \geq y_{1113} - y_{1112} \quad \Rightarrow \quad x_{1223} \geq 1 - 0 \quad \Rightarrow \quad x_{1223} = 1$$

This violates constraint (4.1b) since $x_{1111} + x_{1113} = 2$.

CASE (6): An operation cannot be scheduled in two rooms during a period. In accordance with the illustration below, we have $x_{1111} = x_{1121} = 1$, which violates (4.1b) because $x_{1111} + x_{1121} = 2$.

		Period			
		1	2	3	4
Operating Room	1	(1,1)			
	2	(1,1)			

CASE (7): An operation is assigned to more than one surgeon. In the illustration below, operation 1 is assigned to surgeon 1 and surgeon 2. Then, we have $y_{1111} = y_{2112} = 1$

		Period			
		1	2	3	4
Operating Room	1	(1,1)	(2,1)		
	2				

$$x_{1111} = y_{1111} \quad \Rightarrow \quad x_{1111} = 1$$

$$x_{1112} \geq y_{1112} - y_{1111} \quad \Rightarrow \quad x_{1112} \geq 0 - 1 \quad \Rightarrow \quad x_{1112} = 0$$

$$x_{2112} \geq y_{2112} - y_{2111} \quad \Rightarrow \quad x_{2113} \geq 1 - 0 \quad \Rightarrow \quad x_{2112} = 1$$

$$x_{2113} \geq y_{2113} - y_{2112} \quad \Rightarrow \quad x_{2113} \geq 0 - 1 \quad \Rightarrow \quad x_{2113} = 0$$

However, this violates constraints (4.1b) since $x_{1111} + x_{2112} = 2$.

CASE (8): An operation is assigned to more than or less than its predetermined duration. Assume that surgeon 1 takes two periods to finish operation 1, i.e., $d_{ij} = 2$. The schedule shown in the illustration below takes 3 periods instead of 2 periods for the operation. We have $x_{1111} = y_{1111} = y_{1112} = y_{1113} = 1$.

		Period			
		1	2	3	4
Operating Room	1	(1,1)	(1,1)	(1,1)	
	2				

However, this violates (4.1g) since $y_{1111} + y_{1112} + y_{1113} = 2x_{1111}$. □

4.1.2. Size of the Formulation HSSP (General)

In this section, we explore the complexity of **HSSP (General)** with respect to the number of variables and the number of constraints, and also, justify the use of the branch-and-price method for its solution by showing that the number of variables is greater than the number of constraints. We have :

$$\text{Total Number of Binary Variables} = 2 * \sum_{j} |I(j)| * M * TP \tag{4.3a}$$

where $|I(j)|$ is the number of surgeons who can conduct operation j.

Adding the number of all the constraints from (4.1b) to (4.1g), we have :

$$\text{Total Number of Constraints} = \sum_{j} |I(j)| * M * TP + |J| + M * TP + \sum_{j} |I(j)| + |I| * TP \tag{4.3b}$$

Note that both the number of variables and the number of constraints are polynomial functions of order four with respect to the number of surgeons, the number of operations, and/or the number of operating rooms, and the number of time periods. The difference between the number of variables, and the number of constraints, denoted as D, is:

$$\begin{aligned} D &= (\text{Number of Variables} - \text{Number of Constraints}) \\ &= (\sum_{j} |I(j)| * M * TP - |J| - M * TP - \sum_{j} |I(j)| - |I| * TP), \end{aligned} \tag{4.3c}$$

which is a positive number.

To illustrate, consider an instance with three surgeons, ten operations, four operating rooms, and twelve total periods. Suppose that exactly two surgeons can perform each operation. That is, $|I| = 3$, $|J| = 10$, $M = 4$, $TP = 12$, and $|I(j)| = 2$, $j \in J$. Suppose that the sets $I(j)$, $\forall j \in J$ are as follows:

$I(1)=\{1,2\}, I(2)=\{2,3\}, I(3)=\{3,1\}, I(4)=\{1,2\}, I(5)=\{2,3\},$

$I(6)=\{3,1\}, I(7)=\{1,2\}, I(8)=\{2,3\}, I(9)=\{3,1\}, I(10)=\{1,2\}.$

Based on the information above, $J(i)\ \ \forall i \in I$, take on the values shown below:

$J(1)=\{1,3,4,6,7,9,10\}$, $J(2)=\{1,2,4,5,7,8,10\}$, $J(3)=\{2,3,5,6,8,9\}$

Then, the total number of variables is $2*20*4*12=1{,}920$, and the total number of constraints is $20*4*12+10+4*12+20+3*12=1{,}074$, so the former is greater than the latter by 846.

4.1.3. A Branch-and-Price Method for HSSP (General)

We designate constraint sets (4.1b) and (4.1c) as the linking constraints, and constraint sets (4.1d), (4.1e), (4.1f), and (4.1g) as the complicating constraints. That is, constraint sets (4.1b) and (4.1c) remain in the master problem and the other constraint sets are included in the pricing problem. Note that the constraint sets (4.1d), (4.1e), (4.1f), and (4.1g) are separable by each surgeon. As a result, the number of pricing problems is equal to the number of surgeons. We need the following additional notation.

k_i : Order of columns obtained from the pricing problem for surgeon i, $\forall i \in I$, $k_i = 1,...,K_i$

$\left(x_{ijmt}\right)^{k_i}$: Value of x_{ijmt} in the k_i th column obtained from the pricing problem for surgeon i, $\forall i \in I$, $\forall j \in J(i)$, $\forall m \in M$, $\forall t = 1,...,TP$, $k_i = 1,...,K_i$

$\left(y_{ijmt}\right)^{k_i}$: Value of y_{ijmt} in the k_i th column obtained from the pricing problem for surgeon i, $\forall i \in I$, $\forall j \in J(i)$, $\forall m \in M$, $\forall t = 1,...,TP$, $k_i = 1,...,K_i$

$\bar{\pi}_j$: Value of dual variable for constraint set (4.4b), $\forall i \in I$, $\forall j \in J(i)$

$\bar{\pi}_{mt}$: Value of dual variable for constraint set (4.4c), $\forall m \in M$, $\forall t = 1,...,TP$

$\bar{\pi}_i$: Value of dual variable for constraint set (4.4d), $\forall i \in I$

λ^{k_i} : Binary convexity variable for the k_i th column obtained from the pricing problem for surgeon i; =1 if the k_i th column is chosen in a solution; otherwise, =0, $\forall i \in I$, $k_i = 1,...,K_i$

We have the following master and pricing problems.

BNP-HSSP (General)

MP-HSSP (General)

$$\text{Maximize} \quad \sum_{i\in I}\sum_{k_i=1}^{K_i}\lambda^{k_i}\left[\sum_{j\in J(i)}\sum_{m\in M}\left\{\sum_{t=1}^{TP} r_j\left(x_{ijmt}\right)^{k_i} - \sum_{t=T+1}^{TP}\left(c_i(t-T)\left(y_{ijmt}\right)^{k_i}\right)\right\}\right] \tag{4.4a}$$

$$\text{Subject to} \quad \sum_{i\in I(j)}\sum_{k_i=1}^{K_i}\lambda^{k_i}\left[\sum_{m\in M}\sum_{t=1}^{TP}\left(x_{ijmt}\right)^{k_i}\right] \le 1 \qquad \forall j \in J \qquad \left(\bar{\pi}_j\right) \tag{4.4b}$$

$$\sum_{i\in I}\sum_{k_i=1}^{K_i}\lambda^{k_i}\left[\sum_{j\in J(i)}\left(y_{ijmt}\right)^{k_i}\right] \le 1 \qquad \forall m \in M,\ \forall t = 1,...,TP \qquad (\bar{\pi}_{mt}) \tag{4.4c}$$

$$\sum_{k_i=1}^{K_i}\lambda^{k_i} \le 1 \qquad \forall i \in I \qquad \left(\bar{\pi}_i\right) \tag{4.4d}$$

$$\lambda^{k_i} \in \{0,1\} \qquad \forall i \in I,\ \forall k_i = 1,...,K_i \tag{4.4e}$$

PP-HSSP (General) for Surgeon i, $\forall i \in I$

$$\text{Maximize} \quad \begin{cases} -\bar{\pi}_i + \sum_{j\in J(i)}\sum_{m\in M}\sum_{t=1}^{TP}\left(r_j - \bar{\pi}_j\right)x_{ijmt} \\ -\sum_{j\in J(i)}\left[\sum_{m\in M}\sum_{t=1}^{T}\bar{\pi}_{mt}y_{ijmt} + \sum_{m\in M}\sum_{t=T+1}^{TP}\left(c_i(t-T)+\bar{\pi}_{mt}\right)y_{ijmt}\right] \end{cases} \tag{4.4f}$$

$$\text{Subject to} \quad \sum_{j\in J(i)}\sum_{m\in M} y_{ijmt} \le 1 \qquad \forall t = 1,...,TP \tag{4.4g}$$

$$x_{ijm1} = y_{ijm1} \qquad \forall j \in J(i),\ \forall m \in M \tag{4.4h}$$

$$x_{ijmt} \ge y_{ijmt} - y_{ijm(t-1)} \qquad \forall j \in J(i),\ \forall m \in M,\ \forall t = 2,...,TP \tag{4.4i}$$

$$\sum_{m\in M}\sum_{t=1}^{TP} y_{ijmt} = d_{ij}\sum_{m\in M}\sum_{t=1}^{TP} x_{ijmt} \qquad \forall j \in J(i) \qquad (4.4j)$$

$$x_{ijmt}, y_{ijmt} \in \{0,1\} \qquad \forall j \in J(i),\ \forall m \in M,\ \forall t = 1,...,TP \qquad (4.4k)$$

Note that the summation of convexity variables for a surgeon can be zero in constraint set (4.4d) because some surgeons could be idle during the planning period.

4.2. Priority in HSSP (General)

Note that the formulation **HSSP (General)** does not consider priorities among the surgeons, that is, it does not reflect the fact that Type I surgeons have priority over Type II surgeons in scheduling time slots. Also, once Type I surgeons finish their operations, they leave the hospital, and they are not available to perform other operations later in the hospital. We modify model HSSP (General) in view of these additional features.

Figure 4.1 illustrates an example of a schedule considering only Type I surgeons. Both Surgeons 1 and 2 are Type I surgeons, with Operations 1 and 2 assigned to Surgeon 1 and Operations 3 and 4 assigned to Surgeon 2. Operations 1 and 3 take 2 hours each, while Operation 2 and 4 take 3 hours each. Suppose that Surgeon 1 prefers to perform both the operations between periods 1 and 6, and Surgeon 2 prefers time periods from 3 to 8, and that, all operating rooms can accommodate all the operations. Figure 4.1 shows one possible schedule for these Type I surgeons.

Figure 4.1. An Example of Feasible Schedules for Type I Surgeons

		Period											
		1	2	3	4	5	6	7	8	9	10	11	12
Operating Room	1	(1,1)	(1,1)		(1,2)	(1,2)	(1,2)						
	2			(2,3)	(2,3)		(2,4)	(2,4)	(2,4)				
	3												
	4												

4.2.1. Incorporation of Priority into Model HSSP (General)

With the inclusion of the priority previously mentioned, we designate the model as **HSSP (Priority)**. We introduce the following additional notation.

I_1 (I_2) : Set of Type I (Type II) surgeons, $i_1 \in I_1$ ($i_2 \in I_2$)

J_1 (J_2) : Set of operations that are assigned to a Type I (Type II) surgeon

$\underline{t}(i_1)$ ($\bar{t}(i_1)$) : The earliest (latest) period in which a Type I surgeon i_1 can start (must end) the operations

We have the following formulation.

HSSP (Priority)

$$\text{Maximize} \quad \left(\begin{array}{l} \sum_{i_1 \in I_1} \sum_{j_1 \in J(i_1)} \sum_{m \in M} \sum_{t=\underline{t}(i_1)}^{\bar{t}(i_1)} r_{j_1} x_{i_1 j_1 mt} - \sum_{i_1 \in I_1} \sum_{j_1 \in J(i_1)} \sum_{m \in M} \sum_{t=T+1}^{TP} c_{i_1} (t-T) y_{i_1 j_1 mt} + \\ \sum_{j_2 \in J_2} \sum_{i_1 \in I(j_2)} \sum_{m \in M} \sum_{t=1}^{TP} r_{j_2} x_{i_2 j_2 mt} - \sum_{j_2 \in J_2} \sum_{i_2 \in I(j_2)} \sum_{m \in M} \sum_{t=T+1}^{TP} c_{i_2} (t-T) y_{i_2 j_2 mt} \end{array} \right) \tag{4.5a}$$

$$\text{Subject to} \quad \sum_{m \in M} \sum_{t=\underline{t}(i_1)}^{\bar{t}(i_1)} x_{i_1 j_1 mt} = 1 \qquad \forall j_1 \in J_1,\ i_1 = I(j_1) \tag{4.5b}$$

$$\sum_{i_2 \in I(j_2)} \sum_{m \in M} \sum_{t=1}^{TP} x_{i_2 j_2 mt} \le 1 \qquad \forall j_2 \in J_2 \tag{4.5b'}$$

$$\sum_{i_1 \in I_1} \sum_{j_1 \in J(i_1)} y_{i_1 j_1 mt} \le 1 \qquad \forall m \in M,\ \forall t = \underline{t}(i_1),...,\bar{t}(i_1) \tag{4.5c}$$

$$\sum_{i_2 \in I_2} \sum_{j_2 \in J(i_2)} y_{i_2 j_2 mt} \le 1 \qquad \forall m \in M,\ \forall t = 1,...,TP \tag{4.5c'}$$

$$\sum_{j_1 \in J(i_1)} \sum_{m \in M} y_{i_1 j_1 mt} \le 1 \qquad \forall i_1 \in I_1,\ \forall t = \underline{t}(i_1),...,\bar{t}(i_1) \tag{4.5d}$$

$$\sum_{j_2 \in J(i_2)} \sum_{m \in M} y_{i_2 j_2 mt} \le 1 \qquad \forall i_2 \in I_2,\ \forall t = 1,...,TP \tag{4.5d'}$$

$$x_{i_1 j_1 m \underline{t}(i_1)} = y_{i_1 j_1 m \underline{t}(i_1)} \qquad \forall j_1 \in J_1,\ \forall i_1 \in I(j_1),\ \forall m \in M \tag{4.5e}$$

$$x_{i_1 j_1 mt} \geq y_{i_1 j_1 mt} - y_{i_1 j_1 m(t-1)} \qquad \forall j_1 \in J_1, \forall i_1 \in I(j_1), \forall m \in M, \forall t = \underline{t}(i_1)+1,...,\bar{t}(i_1) \quad (4.5f)$$

$$x_{i_2 j_2 m1} = y_{i_2 j_2 m1} \qquad \forall j_2 \in J_2, \forall i_2 \in I(j_2), \forall m \in M \quad (4.5e')$$

$$x_{i_2 j_2 mt} \geq y_{i_2 j_2 mt} - y_{i_2 j_2 m(t-1)} \qquad \forall j_2 \in J_2, \forall i_2 \in I(j_2), \forall m \in M, \forall t = 2,...,TP \quad (4.5f')$$

$$\sum_{m \in M} \sum_{t=\underline{t}(i_1)}^{\bar{t}(i_1)} y_{i_1 j_1 mt} = d_{i_1 j_1} \qquad \forall j_1 \in J_1, \forall i_1 \in I(j_1) \quad (4.5g)$$

$$\sum_{m \in M} \sum_{t=1}^{TP} y_{i_2 j_2 mt} = d_{i_2 j_2} \sum_{m \in M} \sum_{t=1}^{TP} x_{i_2 j_2 mt} \qquad \forall j_2 \in J_2, \forall i_2 \in I(j_2) \quad (4.5g')$$

$$x_{i_1 j_1 mt}, y_{i_1 j_1 mt} \in \{0,1\} \qquad \forall j_1 \in J_1, \forall i_1 \in I(j_1), \forall m \in M, \forall t = \underline{t}(i_1),...,\bar{t}(i_1) \quad (4.5h)$$

$$x_{i_2 j_2 mt}, y_{i_2 j_2 mt} \in \{0,1\} \qquad \forall j_2 \in J_2, \forall i_2 \in I(j_2), \forall m \in M, \forall t = 1,...,TP \quad (4.5h')$$

All Type I surgeons provide their favorite time slots for their operations, and we assume that none of them has an operating room preference. All the operations assigned to the Type I surgeon i_1 must start and end between $\underline{t}(i_1)$ and $\bar{t}(i_1)$. We also assume that a specific Type I surgeon can perform a Type I operation, while multiple Type II surgeons might be available for a Type II operation – a fact that reflects each Type I surgeon's specialty or that a Type I surgeon operates his/her own patients.

Model **HSSP (Priority)** integrates Type II surgeon's priority in model **HSSP (General)**. The objective function (4.5a) divides (4.1a) into two parts: the first two terms represent the profit generated by the operations assigned to Type I surgeons and the cost incurred during their overtime work, and the next two terms represent the same for Type 2 surgeons.

Constraint set (4.1b) is rewritten into two constraint sets (4.5b) and (4.5b'). While (4.5b') is identical to (4.1b) for Type I surgeons; (4.5b) is different because it keeps the operations assigned to Type I surgeons from being dropped out of the schedule.

Meanwhile, (4.5c), (4.5c'), (4.5d), (4.5d'), (4.5e), (4.5e'), (4.5f), (4.5f'), (4.5g), (4.5g'), (4.5h), and (4.5h') correspond to (4.1c), (4.1d), (4.1e), (4.1f), (4.1g), and (4.1h), respectively, in the same way that (4.5b) and (4.5b') represent (4.1b). To summarize, when formulating model **HSSP (Priority)**, each constraint set from model **HSSP (General)** is divided into two constraint sets, one serving for Type I surgeons and the other for Type II surgeons. Note that the right-hand

side of constraint set (4.5g) is not $d_{i_2 j_2} \sum_{m \in M} \sum_{t=\underline{t}(i_1)}^{\bar{t}(i_1)} x_{i_2 j_2 m t}$ due to constraint set (4.5b). To make this constraint set feasible, we assume that for all $i_1 \in I_1$, the length of $[\underline{t}(i_1), \bar{t}(i_1)]$ is at least as great as $\sum_{j_1 \in J(i_1)} d_{i_1 j_1}$, i.e., the length of the time slot for every Type I surgeon is at least as long as the sum of operation times for all the operations assigned to that surgeon. Also, note that variables for Type I surgeons in (4.5h) are valid only when their period index is $\underline{t}(i_1)$ through $\bar{t}(i_1)$.

Observation 4.1. Because of constraint set (4.5b), the objective function (4.5a) can be reduced to:

$$\text{Maximize} \quad \left(\begin{array}{c} \sum_{j_1 \in J(i_1)} r_{j_1} - \sum_{i_1 \in I_1} \sum_{j_1 \in J(i_1)} \sum_{m \in M} \sum_{t=T+1}^{TP} c_{i_1} (t-T) y_{i_1 j_1 m t} + \\ \sum_{j_2 \in J_2} \sum_{i_1 \in I(j_2)} \sum_{m \in M} \sum_{t=1}^{TP} r_{j_2} x_{i_2 j_2 m t} - \sum_{j_2 \in J_2} \sum_{i_2 \in I(j_2)} \sum_{m \in M} \sum_{t=T+1}^{TP} c_{i_2} (t-T) y_{i_2 j_2 m t} \end{array} \right) \qquad (4.5a')$$

Since $\sum_{j_1 \in J(i_1)} r_{j_1}$ is a constant term, it can be removed from the objective function.

Observation 4.2. It is reasonable to assume that none of Type I surgeons want to assign themselves to overtime periods. Even if some do, it is by their choice, and, therefore, the hospital may not pay them overtime. As a result, the second term in (4.5a') will be zero.

Observation 4.3. We can assume that for all $i_2 \in I_2$, $\underline{t}(i_2) = 1$, and $\bar{t}(i_2) = TP$ for the sake of simplicity.

As a result, the preceding **HSSP (Priority)** can be reduced to the following model.

HSSP (Priority)

Maximize $$\sum_{j_2 \in J_2} \sum_{i_2 \in I(j_2)} \sum_{m \in M} \sum_{t=1}^{TP} r_{j_2} x_{i_2 j_2 mt} - \sum_{j_2 \in J_2} \sum_{i_2 \in I(j_2)} \sum_{m \in M} \sum_{t=T+1}^{TP} c_{i_2} (t-T) y_{i_2 j_2 mt} \quad (4.5a'')$$

Subject to $$\sum_{m \in M} \sum_{t=\underline{t}(i_1)}^{\bar{t}(i_1)} x_{i_1 j_1 mt} = 1 \qquad \forall j_1 \in J_1,\ \forall i_1 = I(j_1) \quad (4.5b)$$

$$\sum_{i_2 \in I(j_2)} \sum_{m \in M} \sum_{t=1}^{TP} x_{i_2 j_2 mt} \le 1 \qquad \forall j_2 \in J_2 \quad (4.5b')$$

$$\sum_{j \in J} \sum_{i \in I(j)} y_{ijmt} \le 1 \qquad \forall m \in M,\ \forall t = 1,...,TP \quad (4.5c)$$

$$\sum_{j \in J(i)} \sum_{m \in M} y_{ijmt} \le 1 \qquad \forall i \in I,\ \forall t = \underline{t}(i),...,\bar{t}(i) \quad (4.5d)$$

$$x_{ijm\underline{t}(i)} = y_{ijm\underline{t}(i)} \qquad \forall j \in J,\ \forall i \in I(j),\ \forall m \in M \quad (4.5e)$$

$$x_{ijmt} \ge y_{ijmt} - y_{ijm(t-1)} \qquad \forall j \in J,\ \forall i \in I(j),\ \forall m \in M,\ \forall t = \underline{t}(i)+1,...,\bar{t}(i) \quad (4.5f)$$

$$\sum_{m \in M} \sum_{t=\underline{t}(i_1)}^{\bar{t}(i_1)} y_{i_1 j_1 mt} = d_{i_1 j_1} \qquad \forall j_1 \in J_1, \forall i_1 \in I(j_1) \quad (4.5g)$$

$$\sum_{m \in M} \sum_{t=1}^{TP} y_{i_2 j_2 mt} = d_{i_2 j_2} \sum_{m \in M} \sum_{t=1}^{TP} x_{i_2 j_2 mt} \qquad \forall j_2 \in J_2, \forall i_2 \in I(j_2) \quad (4.5g')$$

$$x_{ijmt}, y_{ijmt} \in \{0,1\} \qquad \forall j \in J,\ \forall i \in I(j),\ \forall m \in M,\ \forall t = \underline{t}(i),...,\bar{t}(i) \quad (4.5h)$$

Compared to model **HSSP (General)**, model **HSSP (Priority)** is different in the following ways:

- Objective function (4.5a'') only considers Type II surgeons,
- Constraint sets (4.5b) and (4.5b') guarantee that all the operations assigned to Type I surgeons will be performed during pre-specified operation times, and
- Each variable in (4.5h) is restricted within its own earliest start time and latest finish time.

4.2.2. Differences between HSSP (Priority) and Trainee Scheduling Problems

The Trainee Scheduling Problem was introduced in Beliën [10], which we formulated in Section 2.6.1 and designated as **Trainee Scheduling Problem**. Table 4.1 summarizes the difference between the formulations **HSSP (Priority)** and **Trainee Scheduling Problem**.

First, the formulation **Trainee Scheduling Problem** considers the minimization of the cost of assignment of trainees to operations, while the formulation **HSSP (Priority)** considers the maximization of net profit considering the revenue received from the operations and the extra cost incurred during overtime. We assume that the revenue from an operation is the same irrespective of the surgeon performing it as long as it occurs during regular work hours, the hospital incurs the same amount of expenses for managing surgeons, regardless of the number patients treated. However, for operations assigned to overtime, the hospital incurs extra cost. Before the hospital allows an operation to be performed, it has to estimate the net profit to be

Table 4.1. Comparison of the formulations **HSSP (Priority)** and **Trainee Scheduling Problem**

	Trainee Scheduling Problem	**HSSP (Priority)**
Objective Function	Minimization of assignment cost	Maximization of net profit considering profit obtained by operations and overtime
Integrity	Staffs (Trainees) Operations Periods	Staffs (Surgeons) Operations Operating rooms Operation times Periods
Additional Feature	Maximum and minimum training hours	Priority of Type I surgeons over Type II surgeons

derived from it and would accept only if it is profitable.

Second, the formulation **Trainee Scheduling Problem** incorporates trainees, operations, and periods, and likewise, the formulation **HSSP (Priority)** considers surgeons, operating rooms, and operation times. However, the way these two models assign operations to periods is different. In the formulation **Trainee Scheduling Problem**, trainees are assigned to operations in single periods, and schedules for trainees are issued solely on the basis of the operations to which they are assigned, and without consideration of operation times. In contrast, in the formulation **HSSP (Priority)**, a schedule includes not only the assignments of operations to rooms, but also, the length of time for which the assignment will last.

Third, as additional feature, the formulation **HSSP (Priority)** reflects the priority that some surgeons may have over others, while the formulation **Trainee Scheduling Problem** considers the minimum and maximum training hours of a trainee for an operation.

4.2.3. A Branch-and-Price Method for HSSP (Priority)

In the formulation **HSSP (Priority)**, we designate constraint sets (4.5b') and (4.5c) as linking constraints and constraint sets (4.5b), (4.5d), (4.5e), (4.5f), (4.5g), and (4.5g') as complicating constraints. Therefore, constraint sets (4.5b'), and (4.5c) remain in the master problem, while the other constraints are contained in the pricing problems.

Note that the complicating constraints for the formulation **HSSP (Priority)** are separable by surgeons just like that for the formulation **HSSP (General)**. Therefore, it results in $|I|$ pricing problems. Below, we introduce some additional notation for solution by the branch-and-price (BNP) method.

k_{i_1} : Order of columns generated from pricing problem for Type I surgeon i_1, $\forall i_1 \in I_1$, $k_{i_1} = 1,...,K_{i_1}$

k_{i_2} : Order of columns generated from pricing problem for Type II surgeon i_2, $\forall i_2 \in I_2$, $k_{i_2} = 1,...,K_{i_2}$

$\left(x_{i_1 j_1 mt}\right)^{k_{i_1}}$: Value of $x_{i_1 j_1 mt}$ in the k_{i_1} th column obtained from Pricing Problem for Type I

surgeon i_1, $\forall i_1 \in I_1$, $\forall j_1 \in J(i_1)$, $\forall m \in M$, $\forall t = 1,...,TP$, $k_{i_1} = 1,...,K_{i_1}$

$\left(x_{i_2 j_2 mt}\right)^{k_{i_2}}$: Value of $x_{i_2 j_2 mt}$ in the k_{i_2} th column obtained from Pricing Problem for Type II surgeon i_2, $\forall i_2 \in I_2$, $\forall j_2 \in J(i_2)$, $\forall m \in M$, $\forall t = 1,...,TP$, $k_{i_2} = 1,...,K_{i_2}$

$\left(y_{i_1 j_1 mt}\right)^{k_{i_1}}$: Value of $y_{i_1 j_1 mt}$ in the k_{i_1} th column obtained from Pricing Problem for Type I surgeon i_1, $\forall i_1 \in I_1$, $\forall j_1 \in J(i_1)$, $\forall m \in M$, $\forall t = 1,...,TP$, $k_{i_1} = 1,...,K_{i_1}$

$\left(y_{i_2 j_2 mt}\right)^{k_{i_2}}$: Value of $y_{i_2 j_2 mt}$ in the k_{i_2} th column obtained from Pricing Problem for Type II surgeon i_2, $\forall i_2 \in I_2$, $\forall j_2 \in J(i_2)$, $\forall m \in M$, $\forall t = 1,...,TP$, $k_{i_2} = 1,...,K_{i_2}$

$\overline{\pi}_{j_2}$: Value of dual variable for constraint set (4.6b'), $\forall j_2 \in J_2$

$\overline{\pi}_{mt}$: Value of dual variable for constraint set (4.6c), $\forall m \in M$, $\forall t = 1,...,TP$

$\overline{\pi}_{i_1}$: Value of dual variable for constraint set (4.6d), $\forall i_1 \in I_1$

$\overline{\pi}_{i_2}$: Value of dual variable for constraint set (4.6d'), $\forall i_2 \in I_2$

$\lambda^{k_{i_1}}$: Convexity binary variable for the k_{i_1} th column obtained from the Pricing Problem for Type I surgeon i_1; =1 if the k_{i_1} th column is chosen in a solution; otherwise, =0,

$\forall i_1 \in I_1$, $k_{i_1} = 1,...,K_{i_1}$

$\lambda^{k_{i_2}}$: Convexity binary variable for the k_{i_2} th column obtained from the Pricing Problem for Type II surgeon i_2; =1 if the k_{i_2} th column is chosen in a solution; otherwise, =0, $\forall i_2 \in I_2$, $k_{i_2} = 1,...,K_{i_2}$

<u>BNP-HSSP (Priority)</u>

MP-HSSP (Priority)

Maximize $$\sum_{i_2 \in I_2} \sum_{k_{i_2}=1}^{K_{i_2}} \lambda^{k_{i_2}} \left[\sum_{j_2 \in J(i_2)} \left\{ \sum_{m \in M} \sum_{t=1}^{TP} r_{j_2} \left(x_{i_2 j_2 mt}\right)^{k_{i_2}} - \sum_{m \in M} \sum_{t=T+1}^{TP} \left(c_{i_2} (t-T) \left(y_{i_2 j_2 mt}\right)^{k_{i_2}} \right) \right\} \right] \quad (4.6a)$$

Subject to $$\sum_{i_2 \in I(j_2)} \sum_{k_{i_2}=1}^{K_{i_2}} \lambda^{k_{i_2}} \left[\sum_{m \in M} \sum_{t=1}^{TP} \left(x_{i_2 j_2 mt}\right)^{k_{i_2}} \right] \le 1 \qquad \forall j_2 \in J_2 \qquad \left(\overline{\pi}_{j_2}\right) \quad (4.6b)$$

$$\sum_{i\in I}\sum_{k_i=1}^{K_i}\lambda^{k_i}\left[\sum_{j\in J(i)}\left(y_{ijmt}\right)^{k_i}\right]\le 1 \qquad \forall m\in M,\ \forall t=1,...,TP \qquad (\bar{\pi}_{mt}) \qquad (4.6c)$$

$$\sum_{k_{i_1}=1}^{K_{i_1}}\lambda^{k_{i_1}}=1 \qquad \forall i_1\in I_1 \qquad \left(\bar{\pi}_{i_1}\right) \qquad (4.6d)$$

$$\sum_{k_{i_2}=1}^{K_{i_2}}\lambda^{k_{i_2}}\le 1 \qquad \forall i_2\in I_2 \qquad \left(\bar{\pi}_{i_2}\right) \qquad (4.6e)$$

$$\lambda^{k_{i_1}},\lambda^{k_{i_2}}\in\{0,1\} \qquad \forall i_1\in I_1,\forall i_2\in I_2,\forall k_{i_1}=1,...,K_{i_1},\forall k_{i_2}=1,...,K_{i_2} \qquad (4.6f)$$

PP-HSSP (Priority) for Type I Surgeon i_1, $\forall i_1\in I_1$

Maximize $$-\bar{\pi}_{i_1}-\sum_{j_1\in J(i_1)}\sum_{m\in M}\sum_{t=\underline{t}(i_1)}^{\bar{t}(i_1)}\bar{\pi}_{mt}y_{i_1j_1mt} \qquad (4.6g)$$

Subject to $$\sum_{m\in M}\sum_{t=\underline{t}(i_1)}^{\bar{t}(i_1)}x_{i_1j_1mt}=1 \qquad \forall j_1\in J_1 \qquad (4.6h)$$

$$\sum_{j\in J(i_1)}\sum_{m\in M}y_{i_1j_1mt}\le 1 \qquad \forall t=\underline{t}(i_1),...,\bar{t}(i_1) \qquad (4.6i)$$

$$x_{i_1j_1m\underline{t}(i_1)}=y_{i_1j_1m\underline{t}(i_1)} \qquad \forall j_1\in J_1,\ \forall m\in M \qquad (4.6j)$$

$$x_{i_1j_1mt}\ge y_{i_1j_1mt}-y_{i_1j_1m(t-1)} \qquad \forall j_1\in J_1,\ \forall m\in M,\ \forall t=\underline{t}(i_1)+1,...,\bar{t}(i_1) \qquad (4.6k)$$

$$\sum_{m\in M}\sum_{t=\underline{t}(i_1)}^{\bar{t}(i_1)}y_{i_1j_1mt}=d_{i_1j_1} \qquad \forall j_1\in J_1 \qquad (4.6l)$$

$$x_{i_1j_1mt},y_{i_1j_1mt}\in\{0,1\} \qquad \forall j_1\in J_1,\ \forall m\in M,\ \forall t=\underline{t}(i_1),...,\bar{t}(i_1) \qquad (4.6m)$$

PP-HSSP (Priority) for Type II Surgeon i_2, $\forall i_2\in I_2$

Maximize $$\begin{cases} -\bar{\pi}_{i_2}+\sum_{j_2\in J(i_2)}\sum_{m\in M}\sum_{t=1}^{TP}\left(r_{j_2}-\bar{\pi}_{j_2}\right)x_{i_2j_2mt} \\ -\sum_{j_2\in J(i_2)}\left[\sum_{m\in M}\sum_{t=1}^{T}\bar{\pi}_{mt}y_{i_2j_2mt}+\sum_{m\in M}\sum_{t=T+1}^{TP}\left(c_{i_2}(t-T)+\bar{\pi}_{mt}\right)y_{i_2j_2mt}\right] \end{cases} \qquad (4.6n)$$

Subject to $\sum_{j_2 \in J(i_2)} \sum_{m \in M} y_{i_2 j_2 mt} \leq 1$ $\forall t = 1,...,TP$ (4.6o)

$x_{i_2 j_2 m1} = y_{i_2 j_2 m1}$ $\forall j_2 \in J_2,\ \forall m \in M$ (4.6p)

$x_{i_2 j_2 mt} \geq y_{i_2 j_2 mt} - y_{i_2 j_2 m(t-1)}$ $\forall j_2 \in J_2,\ \forall m \in M,\ \forall t = 2,...,TP$ (4.6q)

$\sum_{m \in M} \sum_{t=1}^{TP} y_{i_2 j_2 mt} = d_{i_2 j_2} \sum_{m \in M} \sum_{t=1}^{TP} x_{i_2 j_2 mt}$ $\forall j_2 \in J_2$ (4.6r)

$x_{i_2 j_2 mt}, y_{i_2 j_2 mt} \in \{0,1\}$ $\forall j_2 \in J_2,\ \forall m \in M,\ \forall t = 1,...,TP$ (4.6s)

BNP-HSSP (Priority) applies the BNP method to **HSSP-Priority**, and it consists of the master problem and the pricing problems, **MP-HSSP (Priority)** and **PP-HSSP (Priority)**, respectively. Both the master problem and the pricing problems of **BNP-HSSP (Priority)** have different types of constraint sets than those for **BNP-HSSP (General)** due to consideration of priorities between surgeons.

Note that (4.5b') in **HSSP (Priority)** is implemented as (4.6b) in **MP-HSSP (Priority)**, but (4.5b) instead belongs to the pricing problems for Type I surgeons. We are able to do this for **HSSP (Priority)** in contrast to **HSSP (General)** because every Type I operation, $\forall j_1 \in J_1$, can be performed by a specific surgeon *i*. Consequently, it can be considered in the pricing problem **PP-HSSP (Priority) for Type I Surgeon i_1**, which is defined for each type of Type I surgeon. The formulation **MP-HSSP (Priority)** has $|J_1|$ fewer constraints in the master problem than the formulation **MP-HSSP (General)**. Constraint set (4.5b) is also different from (4.5b') because of the two different kinds of convexity constraint sets in (4.6d) and (4.6e). Since every Type I surgeon is assigned to at least one operation, the equality in (4.6d) holds. On the other hand, since a Type II surgeon can be idle, (4.6e) might be non-binding. On the other hand, if any Type II surgeon is assigned to at least one operation, then (4.6e) will be binding.

As noted earlier, the operation assignment constraint set (4.6h) exists only in **PP-HSSP (Priority) for Type I Surgeon**. In addition, the objective functions in these pricing problems are different from the objective functions of the corresponding master problem, which are simplified by removing the terms corresponding to Type II surgeons in the objective function of **MP-HSSP (Priority)**. On the other hand, **PP-HSSP (Priority) for Type II Surgeon** has the same structure as **PP-HSSP (General)**.

4.3. A Special Case of HSSP (Priority)

4.3.1. Completely Pre-arranged Schedule for Type I Surgeons

Recall that in model HSSP (Priority), the time slot for each Type I surgeon can be greater than the total operation times for the operations assigned to that surgeon, i.e., $\bar{t}(i_1) - \underline{t}(i_1) + 1 \geq \sum_{j_1 \in J(i_1)} d_{i_1 j_1}$ for all $i_1 \in I_1$. This can lead to multiple feasible schedules for a surgeon. In this section, we consider the situation in which each Type I surgeon pre-specifies the exact starting time for each operation.

Figure 4.2 illustrates an example of such a schedule that only considers Type I surgeons, Surgeons 1 and 2. Operations 1 and 2 are assigned to Surgeon 1, and operations 3 and 4 are assigned to Surgeon 2. Surgeon 1 schedules operations 1 and 2 to start in periods 1 and 5, and Surgeon 2 schedules operations 3 and 4 to start in periods 1 and 4, respectively.

The yellow and green cells denote Surgeons 1 and 2, respectively. Once the schedule for Type I surgeons has been established, the hospital schedules the operations to be performed by Type II surgeons in the remaining time slots, which are denoted by red, blue, grey, and pink cells.

Figure 4.2. Example of Type I Surgeons' Schedule

		Period											
		1	2	3	4	5	6	7	8	9	10	11	12
Operating Room	1	(1,1)	(1,1)	(1,1)	(1,1)								
	2					(1,2)	(1,2)	(1,2)					
	3	(2,3)	(2,3)	(2,3)									
	4				(2,4)	(2,4)	(2,4)	(2,4)					

One way to assign operations to these time slots is to find the sets of operations that fit in each block while optimizing profit. For example, the operations that take no more than three periods might be proper candidates for the blue time slots in Figure 4.2. However, they are candidates for all the other slots as well since all of them contain more than three empty time slots. Furthermore, if we put an operation taking three hours into the red time slot, which has four empty periods, we may waste one time slot if we do not have an operation that requires exactly one period. We would rather choose an operation that takes exactly four hours, which maximizes the profit for that time slot if the profit increases with the number of pre-specified operation times. However, it is not guaranteed that a solution that is optimal for a single time slot will be free of conflicts with the other solutions. Therefore, we need to resolve the entire problem from the beginning after having removed any conflicting solutions. However, a better way is to fix both time slots and room assignments for Type I surgeons, and then, to solve the remaining problem. Considering the schedule shown in Figure 4.2, the variables that will be fixed to one in the formulation **HSSP (General)** are as follows:

$$x_{1111} = x_{1225} = x_{2331} = x_{2444} = 1$$

$$y_{1111} = y_{1112} = y_{1113} = y_{1114} = y_{1225} = y_{1226} = y_{1227} = 1$$

$$y_{2331} = y_{2332} = y_{2333} = y_{2444} = y_{2445} = y_{2446} = y_{2447} = 1$$

With this modification, we designate the corresponding model as **HSSP (Pre-arranged)**.

4.3.2. Formulation for the HSSP (Pre-arranged)

The formulation below, **HSSP (Pre-arranged)**, is a variation of **HSSP (General)** obtained by fixing all variables related to Type I surgeons to either one or zero. Let

$m(i, j)$: Operating room where Type I surgeon i conducts operation j

$t(i, j)$: Period when Type I surgeon i starts operation j in operating room $m(i, j)$

HSSP (Pre-arranged)

Maximize $$\sum_{j_2 \in J_2} \sum_{i_1 \in I(j_2)} \sum_{m \in M} \sum_{t=1}^{TP} r_{j_2} x_{i_2 j_2 m t} - \sum_{j_2 \in J_2} \sum_{i_2 \in I(j_2)} \sum_{m \in M} \sum_{t=T+1}^{TP} c_{i_2} (t-T) y_{i_2 j_2 m t} \quad (4.7a)$$

Subject to $$\sum_{i_2 \in I(j_2)} \sum_{m \in M} \sum_{t=1}^{TP} x_{i_2 j_2 m t} \le 1 \qquad \forall j_2 \in J_2 \quad (4.7b)$$

$$\sum_{i_2 \in I_2} \sum_{j_2 \in J(i_2)} y_{i_2 j_2 m t} \le 1 \qquad \forall m \in M, \forall t = 1,...,TP \quad (4.7c)$$

$$\sum_{j_2 \in J(i_2)} \sum_{m \in M} y_{i_2 j_2 m t} \le 1 \qquad \forall i_2 \in I_2, \forall t = 1,...,TP \quad (4.7d)$$

$$x_{i_2 j_2 m 1} = y_{i_2 j_2 m 1} \qquad \forall j_2 \in J_2, \forall i_2 \in I(j_2), \forall m \in M \quad (4.7e)$$

$$x_{i_2 j_2 m t} \ge y_{i_2 j_2 m t} - y_{i_2 j_2 m (t-1)} \qquad \forall j_2 \in J_2, \forall i_2 \in I(j_2), \forall m \in M, \forall t = 2,...,TP \quad (4.7f)$$

$$\sum_{m \in M} \sum_{t=1}^{TP} y_{i_2 j_2 m t} = d_{i_2 j_2} \sum_{m \in M} \sum_{t=1}^{TP} x_{i_2 j_2 m t} \qquad \forall j_2 \in J_2, \forall i_2 \in I(j_2) \quad (4.7g)$$

$$x_{i_2 j_2 m t}, y_{i_2 j_2 m t} \in \{0,1\} \qquad \forall j_2 \in J_2, \forall i_2 \in I(j_2), \forall m \in M, \forall t = 1,...,TP \quad (4.7h)$$

$$x_{i_1 j_1 m t} = 1 \qquad \forall i_1 \in I_1, \forall j_1 \in J(i_1), m = m(i_1, j_1), t = t(i_1, j_1) \quad (4.7i)$$

$$y_{i_1 j_1 m t} = 1 \quad \forall i_1 \in I_1, \forall j_1 \in J(i_1), m = m(i_1, j_1), \forall t \in \{t(i_1, j_1) \le t \le t(i_1, j_1) + d_{i_1 j_1} - 1\} \quad (4.7j)$$

$$x_{i_1 j_1 m t} = 0 \qquad \forall i_1 \in I_1, \forall j_1 \in J(i_1), m = m(i_1, j_1), \forall t \ne t(i_1, j_1) \quad (4.7k)$$

$$x_{i_1 j_1 m t} = 0 \qquad \forall i_1 \in I_1, \forall j_1 \in J(i_1), m \ne m(i_1, j_1), \forall t = 1,...,TP \quad (4.7l)$$

$$y_{i_1 j_1 mt} = 0 \quad \forall i_1 \in I_1, \forall j_1 \in J(i_1), m = m(i_1, j_1), \forall t \notin \{t(i_1, j_1) \le t \le t(i_1, j_1) + d_{i_1 j_1} - 1\} \quad \text{(4.7m)}$$

$$y_{i_1 j_1 mt} = 0 \quad \forall i_1 \in I_1, \forall j_1 \in J(i_1), m \neq m(i_1, j_1), \forall t = 1,\ldots,TP \quad \text{(4.7n)}$$

The objective function (4.7a) only consists of the terms related to Type II surgeons, which is similar to the objective function of **HSSP (Priority)**, under the assumption that all operations assigned to Type I surgeons will be performed. All constraint sets from (4.7b) to (4.7h) serve the same function as (4.5b'), (4.5c'), (4.5d'), (4.5e'), (4.5f'), (4.5g'), and (4.5h') in **HSSP (Priority)**. All variables related to Type I surgeons are fixed by constraint sets (4.7i) through (4.7n). For example, if Type I surgeon i starts to work on operation j in operating room $m(i, j)$ in period $t(i, j)$, that is, if $x_{ijm(i,j)t(i,j)} = 1$ and the operation takes d_{ij}, then $y_{ijm(i,j)t} = 1$ for $t(i, j) \le t \le t(i, j) + d_{ij} - 1$. All the other variables related to this assignment should be zero. Constraint sets (4.7i) through (4.7n) represent the following restrictions, respectively:

(4.7i): Type I surgeon i_1 starts operation j_1 in operating room $m(i_1, j_1)$ in period $t(i_1, j_1)$

(4.7j): Type I surgeon i_1 performs operation j_1 in operating room $m(i_1, j_1)$ during periods $t(i_1, j_1)$ through $t(i_1, j_1) + d_{i_1 j_1} - 1$

(4.7k): Operation j_1 does not start in a period other than $t(i_1, j_1)$

(4.7l): Operation j_1 does not start in a room other than $m(i_1, j_1)$

(4.7m): Operation j_1 is not performed at any period other than from $t(i_1, j_1)$ to $t(i_1, j_1) + d_{i_1 j_1} - 1$

(4.7n): Operation j_1 is not performed in any operating room other than $m(i_1, j_1)$

4.3.3. Examples of the HSSP (Pre-arranged)

In this section, we illustrate model **HSSP (Pre-arranged)** through three examples. In the figures, the yellow cells, the green cells, the blue cells, and the orange cells will denote Type I Surgeons 1 and 2 and Type II Surgeons 3 and 4, respectively. The regular work periods are from 1 to 8, while the overtime periods are from 9 to 12.

EXAMPLE (1): In Figure 4.3, Surgeon 1 performs operations 1 and 2, Surgeon 2 performs operations 3 and 4, and Surgeon 3 performs operation 7, and Surgeon 4 performs operations 5, 6, and 8. All operations are successfully scheduled during regular work hours. Therefore, no overtime cost will be incurred.

Figure 4.3. 2 Type I Surgeons, 2 Type II Surgeons, 8 Operations, and 4 Operating Rooms

		Period (T=8)											
		1	2	3	4	5	6	7	8	9	10	11	12
Operating Room	1	(1,1)					(4,6)						
	2	(4,5)				(1,2)							
	3	(2,3)			(4,8)		(3,7)						
	4				(2,4)								

EXAMPLE (2): In Figure 4.4, Type II Surgeons 3 and 4 perform 8 operations. Surgeon 3 performs operations 6, 7 and 10, and Surgeon 4 performs operations 5, 8, 9, 11, and 12. In Figure 4.3, both Type II surgeons encounter some idle time. However, in Figure 4.4, Surgeon 3 does not encounter any idle time and all his operations are scheduled during regular work hours, while the operations of Surgeon 4 are scheduled throughout the day.

Figure 4.4. 2 Type I Surgeons, 2 Type II Surgeons, 12 Operations, and 4 Operating Rooms

		Period (T=8)											
		1	2	3	4	5	6	7	8	9	10	11	12
Operating Room	1	(1,1)				(4,8)		(3,6)			(4,9)		
	2	(4,5)		(4,11)		(1,2)							
	3	(2,3)			(3,10)			(4,12)					
	4	(3,7)			(2,4)								

EXAMPLE (3): In Figure 4.5, Type II Surgeons 3 and 4 are scheduled for the entire day while Surgeon 3 performs operations 7, 9, 10, 15, and 16, and Surgeon 4 performs operations 5, 6, 8, 11, and 12. Nonetheless, operations 13 and 14 cannot be assigned to any surgeon because of either unavailability of resources or relatively low profitability.

Figure 4.5. 2 Type I Surgeons, 2 Type II Surgeons, 16 Operations, and 5 Operating Rooms

		Period (T=8)											
		1	2	3	4	5	6	7	8	9	10	11	12
Operating Room	1	(1,1)					(3,7)						
	2	(4,12)				(1,2)				(3,10)			(3,16)
	3	(2,3)			(4,8)								
	4				(2,4)				(4,6)			(4,11)	
	5	(3,15)			(3,9)		(4,5)						

4.3.4. Modification and Reduction of HSSP (Pre-arranged)

Note that constraint sets (4.7i) and (4.7j) not only define the value of the variables regarding the rest of the Type I surgeons, but also, restrict the values of the variables regarding some Type II surgeons. If $\sum_{i_1 \in I_1} \sum_{j_1 \in J(i_1)} y_{i_1 j_1 mt} = 1$ for $m = m(i_1, j_1)$ and $t = t(i_1, j_1)$, i.e., if operating room $m(i_1, j_1)$ is occupied in period $t(i_1, j_1)$ by a Type I Surgeon i_1, none of Type II surgeons can use $m(i_1, j_1)$ at $t(i_1, j_1)$. We can express this restriction as $\sum_{i_2 \in I_2} \sum_{j_2 \in J(i_2)} y_{i_2 j_2 mt} = 0$ for $m = m(i_1, j_1)$ and $t = t(i_1, j_1)$.

This leads to the constraint set:

$$y_{i_2 j_2 mt} = 0 \quad \forall i_2 \in I_2,\ \forall j_2 \in J(i_2),\ m = m(i_1, j_1) \text{ and } t = t(i_1, j_1) \tag{4.8a}$$

Thus, the variables of Type II surgeons in constraint set (4.8a) as well as the variables of all Type I surgeons become fixed, i.e., only the variables of Type II surgeons that are not involved with Type I surgeons' operating rooms and operation periods are free. Based on this observation, we can add the following two constraint sets to **HSSP (Priority)** :

$$x_{i_2 j_2 mt} = 0 \qquad \forall i_2 \in I_2, \forall j_2 \in J(i_2),\ \forall m \in M, \forall t = 1,...,TP \text{ such that } \sum_{i \in I_1} \sum_{j \in J(i)} y_{ijmt} = 1 \tag{4.8b}$$

$$y_{i_2 j_2 mt} = 0 \qquad \forall i_2 \in I_2, \forall j_2 \in J(i_2),\ \forall m \in M, \forall t = 1,...,TP \text{ such that } \sum_{i \in I_1} \sum_{j \in J(i)} y_{ijmt} = 1 \tag{4.8c}$$

Constraint sets (4.8b) and (4.8c) require that if a pair of (m,t) is already taken by one of Type I surgeons, i.e., if $\sum_{i \in I_1} \sum_{j \in J(i)} y_{ijmt} = 1$, then all Type II surgeons' variables related to (m,t) must be zero. Only Type II surgeons' variables other than the ones in (4.8b) and (4.8c) are free. Denote the set of (m, t) that are not assigned to any Type I surgeon as $\underline{(m,t)}$, i.e., $\underline{(m,t)} \in \underline{(M,T)} \equiv \left\{ (m,t) : \sum_{i \in I_1} \sum_{j \in J(i)} y_{ijmt} = 0 \right\}$.

As a result, we can convert **HSSP (Pre-arranged)** to the following equivalent formulation where all Type I surgeons' variables in constraint sets (4.7i) to (4.7n) and those in constraints sets (4.8b) and (4.8c) are removed:

HSSP (Pre-arranged)

$$\text{Maximize} \quad \sum_{i_2 \in I_2} \sum_{j_2 \in J(i_2)} \sum_{(m,t) \in \underline{(M,T)}} r_{j_2} x_{i_2 j_2 mt} - \sum_{i_2 \in I_2} \sum_{j_2 \in J(i_2)} \sum_{(m,t) \in \underline{(M,T)}: t \geq T+1} c_{i_2} (t-T) y_{i_2 j_2 mt} \tag{4.8a}$$

Subject to

$$\sum_{i_2 \in I(j_2)} \sum_{(m,t) \in \underline{(M,T)}} x_{i_2 j_2 mt} \leq 1 \qquad \forall j_2 \in J_2 \tag{4.8b}$$

$$\sum_{i_2 \in I_2} \sum_{j_2 \in J(i_2)} y_{i_2 j_2 mt} \leq 1 \qquad \forall m \in M,\ \forall t = 1,...,TP,\ \underline{(m,t)} \in \underline{(M,T)} \tag{4.8c}$$

$$\sum_{j_2 \in J(i_2)} \sum_{(m,t) \in \underline{(M,T)}} y_{i_2 j_2 mt} \leq 1 \qquad \forall i_2 \in I_2,\ \forall t = 1,...,TP \tag{4.8d}$$

$$x_{i_2 j_2 m1} = y_{i_2 j_2 m1} \qquad \forall i_2 \in I_2,\ \forall j_2 \in J(i_2),\ \forall m \in M,\ \underline{(m,1)} \in \underline{(M,T)} \tag{4.8e}$$

$$x_{i_2 j_2 mt} \geq y_{i_2 j_2 mt} - y_{i_2 j_2 m(t-1)} \qquad \forall i_2 \in I_2,\ \forall j_2 \in J(i_2),\ \forall m \in M,\ \forall t = 2,...,TP,\ \underline{(m,t)} \in \underline{(M,T)} \tag{4.8f}$$

$$\sum_{(m,t) \in \underline{(M,T)}} y_{i_2 j_2 mt} = d_{i_2 j_2} \sum_{(m,t) \in \underline{(M,T)}} x_{i_2 j_2 mt} \qquad \forall i_2 \in I_2,\ \forall j_2 \in J(i_2) \tag{4.8g}$$

$$x_{i_2 j_2 mt}, y_{i_2 j_2 mt} \in \{0,1\} \qquad \forall i_2 \in I_2,\ \forall j_2 \in J(i_2),\ \forall m \in M,\ \forall t = 1,...,TP,\ \underline{(m,t)} \in \underline{(M,T)} \tag{4.8h}$$

Note that not only does it exclude all Type I surgeons' variables, but also, it only allows Type II surgeons' variables in which operating rooms and periods are not occupied by any Type I surgeon in constraint set (4.8h). As a result, the only pairs of (m,t) satisfying $\sum_{i_1 \in I_1} \sum_{j_1 \in J(i_1)} y_{i_1 j_1 mt} = 0$ are allowed in **HSSP (Pre-arranged)**.

4.3.5. A Branch-and-Price Method for HSSP (Pre-arranged)

In the formulation **HSSP (Pre-arranged)**, we classify constraint sets (4.5b) and (4.5c) as linking constraints and constraint sets (4.5d), (4.5e), (4.5f), and (4.5g) as complicating constraints. Constraint sets (4.5b) and (4.5c) remain in the master problem and the other constraint sets are contained in the pricing problems. The constraint sets (4.5d), (4.5e), (4.5f), and (4.5g) are completely separable by each Type II surgeon, which means that the number of pricing problems is equal to the number of Type II surgeons. Based on this structure, we suggest the following master problem and pricing problems. As before, we need the following additional notation.

k_{i_2} : Order of columns generated from pricing problem for Type II surgeon i_2, $i_2 \in I_2$, $k_{i_2} = 1,...,K_{i_2}$

$\left(x_{i_2 j_2 mt}\right)^{k_{i_2}}$: Value of $x_{i_2 j_2 mt}$ in the k_{i_2} th column obtained from Pricing Problem for Type II surgeon i_2, $i_2 \in I_2$, $j_2 \in J(i_2)$, $\forall m \in M$, $\forall t = 1,...,TP$, $k_{i_2} = 1,...,K_{i_2}$

$\left(y_{i_2 j_2 mt}\right)^{k_{i_2}}$: Value of $y_{i_2 j_2 mt}$ in the k_{i_2} th column obtained from Pricing Problem for Type II surgeon i_2, $i_2 \in I_2$, $j_2 \in J(i_2)$, $\forall m \in M$, $\forall t = 1,...,TP$, $k_{i_2} = 1,...,K_{i_2}$

$\bar{\pi}_{j_2}$: Value of dual variable for constraint set (4.6b), $j_2 \in J(i_2)$

$\bar{\pi}_{mt}$: Value of dual variable for constraint set (4.6c), $\forall m \in M$, $\forall t = 1,...,TP$

$\bar{\pi}_{i_2}$: Value of dual variable for constraint set (4.6d), $\forall i_2 \in I_2$

$\lambda^{k_{i_2}}$: Convexity binary variable for the k_{i_2} th column obtained from Pricing Problem for Type II surgeon i_2; =1 if the k_{i_2} th column is chosen in a solution; =0, otherwise, $i_2 \in I_2$, $k_{i_2} = 1,...,K_{i_2}$

BNP-HSSP (Pre-arranged)

MP-HSSP (Pre-arranged)

$$\text{Maximize} \quad \sum_{i_2 \in I_2} \sum_{k_{i_2}=1}^{K_{i_2}} \lambda^{k_{i_2}} \left[\sum_{j_2 \in J(i_2)} \left\{ \sum_{(m,t) \in \underline{(M,T)}} r_{j_2} \left(x_{i_2 j_2 mt} \right)^{k_{i_2}} - \sum_{(m,t) \in \underline{(M,T)}: t \geq T+1} \left(c_{i_2} (t-T) \left(y_{i_2 j_2 mt} \right)^{k_{i_2}} \right) \right\} \right] \tag{4.6a}$$

$$\text{Subject to} \quad \sum_{i_2 \in I(j_2)} \sum_{k_{i_2}=1}^{K_{i_2}} \lambda^{k_{i_2}} \left[\sum_{(m,t) \in \underline{(M,T)}} \left(x_{i_2 j_2 mt} \right)^{k_{i_2}} \right] \leq 1 \qquad \forall j_2 \in J_2 \qquad \left(\overline{\pi}_{j_2} \right) \tag{4.6b}$$

$$\sum_{i_2 \in I_2} \sum_{k_{i_2}=1}^{K_{i_2}} \lambda^{k_{i_2}} \left[\sum_{j_2 \in J(i_2)} \left(y_{i_2 j_2 mt} \right)^{k_{i_2}} \right] \leq 1 \quad \forall m \in M, \ \forall t = 1,...,TP, \ \underline{(m,t)} \in \underline{(M,T)} \qquad \left(\overline{\pi}_{mt} \right) \tag{4.6c}$$

$$\sum_{k_{i_2}=1}^{K_{i_2}} \lambda^{k_{i_2}} \leq 1 \qquad \forall i_2 \in I_2 \qquad \left(\overline{\pi}_{i_2} \right) \tag{4.6d}$$

$$\lambda^{k_{i_2}} \in \{0,1\} \qquad \forall i_2 \in I_2, \ \forall k_{i_2} = 1,..., K_{i_2} \tag{4.6e}$$

PP-HSSP (Pre-arranged) for Type II Surgeon i_2, $\forall i_2 \in I_2$

$$\text{Maximize} \quad \begin{cases} -\overline{\pi}_{i_2} + \sum_{j_2 \in J(i_2)} \sum_{(m,t) \in \underline{(M,T)}} \left(r_{j_2} - \overline{\pi}_{j_2} \right) x_{i_2 j_2 mt} \\ - \sum_{j_2 \in J(i_2)} \left[\sum_{(m,t) \in \underline{(M,T)}: 1 \leq t \leq T} \overline{\pi}_{mt} y_{i_2 j_2 mt} + \sum_{(m,t) \in \underline{(M,T)}: t \geq T+1} \left(c_{i_2} (t-T) + \overline{\pi}_{mt} \right) y_{i_2 j_2 mt} \right] \end{cases} \tag{4.6f}$$

$$\text{Subject to} \quad \sum_{j_2 \in J(i_2)} \sum_{(m,t) \in \underline{(M,T)}} y_{i_2 j_2 mt} \leq 1 \qquad \forall t = 1,...,TP \tag{4.6g}$$

$$x_{i_2 j_2 m1} = y_{i_2 j_2 m1} \qquad \forall j_2 \in J(i_2) \ \ \forall m, \ \underline{(m,1)} \in \underline{(M,T)} \tag{4.6h}$$

$$x_{i_2 j_2 mt} \geq y_{i_2 j_2 mt} - y_{i_2 j_2 m(t-1)} \qquad \forall j_2 \in J(i_2), \ \forall m \in M, \ \forall t = 2,...,TP, \ \underline{(m,t)} \in \underline{(M,T)} \tag{4.6i}$$

$$\sum_{(m,t) \in \underline{(M,T)}} y_{i_2 j_2 mt} = d_{i_2 j_2} \sum_{(m,t) \in \underline{(M,T)}} x_{i_2 j_2 mt} \qquad \forall j_2 \in J(i_2) \tag{4.6j}$$

$$x_{i_2 j_2 mt}, y_{i_2 j_2 mt} \in \{0,1\} \qquad \forall j_2 \in J(i_2), \ \forall m \in M, \ \forall t = 1,...,TP, \ \underline{(m,t)} \in \underline{(M,T)} \tag{4.6k}$$

BNP-HSSP (Pre-arranged) applies the BNP method to **HSSP-Pre-arranged**, and consists of the master problem and the pricing problems, which we call **MP-HSSP (Pre-arranged)** and **PP-HSSP (Pre-arranged)** with one pricing problem for each Type II surgeon, respectively.

4.4. A Branch-and-Price Method for HSSP Models

Since we have already explained the general procedure of the Branch-and-Price (BNP) method, which is expressed as a combination of the column generation and branch-and-bound methods, in previous chapters, we will not repeat the details here. In this section, we discuss the distinct properties of the BNP method for the three HSSP Models: HSSP (General), HSSP (Priority), and HSSP (Prearranged).

4.4.1. Branching Variable Selection Strategy

4.4.1.1. Setting Values of Variables to 0 or 1

First, in the branch-and-bound method for the HSSP models, we use a branching variable selection strategy that is different from that used for the SGAP. Recall that the branching variable selection strategy that we use for the SGAP is the most infeasibility strategy, where we choose the fractional variable that is closest in value to 0.5. For the HSSP models, we combine the most infeasibility strategy with a strategy that considers the effect caused by a variable in solving the HSSP models. We will measure the effectiveness of this impact by the number of variables that can be set to either zero or one. Clearly, the higher the number of variables that are set to zero or one, the smaller will be the size of the problem for solution at a given node. To that end, consider the following observation and proposition.

Observation 4.4. Assume that we have a solution to the formulation **HSSP2 (General)** at a root node, and there is no priority among the surgeons. Also, assume that x_{ijmt} and y_{ijmt}, $i \in I$, $j \in J(i)$, $m \in M$, are fractional. Furthermore, assume that either $x_{ijmt} = 1$ or $y_{ijmt} = 1$ does not make the children nodes infeasible. Then, if t is the first period, i.e., t=1, fixing y_{ijm1} to 1 sets x_{ijm1} to one, and if t is the last period, i.e., $t=TP$, fixing $y_{ijm(TP)}$ to 1 sets $x_{ijm(TP-d_{ij}+1)}$ to one in order to maintain a schedule feasible. Consequently, setting $y_{ijmt}=1$, in which t is either 1 or TP, gives us the same information obtained by setting $x_{ijmt}=1$.

Proposition 4.2. Under the same assumptions made in Observation 4.4, assume x_{ijmt} and y_{ijmt}, $i \in I$, $j \in J(i)$, $m \in M$, are fractional, but t is not 1 or TP, i.e., $2 \leq t \leq TP\text{-}1$. Then, at least d_{ij} additional variables are fixed to values of one or zero if we set $x_{ijmt} = 1$ than if we set $y_{ijmt} = 1$. In other words, we obtain more information from setting $x_{ijmt} = 1$.

Proof.

a. Branching on y_{ijmt}, $2 \leq t \leq TP\text{-}1$

Suppose that y_{ijmt} is fractional, and it is set to one. Then, we can set the following variables to zero:

- Operation j can start only during one of periods from $t - d_{ij} + 1$ to t. Therefore,

$$x_{i'jm't'} = 0, \qquad \forall i' \in I,\ \forall m' \in M,\ \forall t' \in \{t' = 1,...,TP : t' \leq t - d_{ij} \text{ or } t' \geq t + 1\}. \quad (4.7a)$$

- Operation j starts in operating room m. Therefore,

$$x_{i'jm't'} = 0, \qquad \forall i' \in I,\ \forall m' \in M \setminus m,\ \forall t' = 1,...,TP. \quad (4.7b)$$

Once the variables already set to zero by (4.7a) are excluded, then (4.7b) is equal to

$$x_{i'jm't'} = 0, \qquad \forall i' \in I,\ \forall m' \in M \setminus m,\ \forall t' \in \{t' = 1,...,TP : t - d_{ij} + 1 \leq t' \leq t\}. \quad (4.7c)$$

- Surgeon i performs operation j. Therefore,

$$x_{i'jm't'} = 0, \qquad \forall i' \in I \setminus i,\ \forall m' \in M,\ \forall t' = 1,...,TP. \quad (4.7d)$$

Once the variables already set to zero by (4.7a) and (4.7c) are excluded, then (4.7d) is equal to

$$x_{i'jmt'} = 0, \qquad \forall i' \in I \setminus i,\ \forall t' \in \{t' = 1,...,TP : t - d_{ij} + 1 \le t' \le t\}. \tag{4.7e}$$

- Surgeon i performs only operation j during period t. Therefore,

$$x_{ij'm't} = 0, \qquad \forall j' \in J(i) \setminus j,\ \forall m' \in M. \tag{4.7f}$$

- Surgeon i starts operation j in operating room m during one of periods from $t - d_{ij} + 1$ to t. Therefore,

$$x_{ijm't'} = 0, \qquad \forall (m', t') \ne (m, t''), \forall t'' \in \{t'' = 1,...,TP : t - d_{ij} + 1 \le t'' \le t\}. \tag{4.7g}$$

There is no additional variable already set to zero after excluding the variables that are set to zero by (4.7a) and (4.7c).

- During period t, operating room m cannot have any other pair of surgeon and operation than (i, j). Therefore,

$$x_{i'j'mt} = 0, \qquad \forall (i', j') \ne (i, j). \tag{4.7h}$$

Once the variables already set to zero by (4.7e) and (4.7f) are excluded, then (4.7h) is equal to

$$x_{i'j'mt} = 0, \qquad \forall i' \in I \setminus i,\ \forall j' \in J(i) \setminus j. \tag{4.7i}$$

- Operation j is not scheduled in a room other than m. Therefore,

$$y_{i'jm't'} = 0, \qquad \forall i' \in I,\ \forall m' \in M \setminus m,\ \forall t' = 1,...,TP. \tag{4.7j}$$

- Operation j is not processed by any other surgeon than surgeon i. Therefore,

$$y_{i'jm't'} = 0, \qquad \forall i' \in I \setminus i,\ \forall m' \in M,\ \forall t' = 1,...,TP. \tag{4.7k}$$

Once the variables already set to zero by (4.7j) are excluded, then (4.7k) is equal to

$$y_{i'jmt'} = 0, \qquad \forall i' \in I \setminus i,\ \forall t' = 1,...,TP. \tag{4.7l}$$

- The only assignment possible in operating room m during period t is for surgeon i to perform operation j. Therefore,

$$y_{i'j'mt} = 0, \qquad \forall (i', j') \neq (i, j). \tag{4.7m}$$

Once the variables already set to zero by (4.7l) are excluded, then (4.7m) is equal to

$$y_{ij'mt} = 0, \qquad \forall j' \in J(i) \setminus j. \tag{4.7n}$$

- If period t is the starting period for operation j, it ends during period $t + d_{ij} - 1$. If period t is its ending period, it should start during period $t - d_{ij} + 1$. In other words, operation j cannot be performed beyond these two periods. Therefore,

$$y_{ijm't'} = 0, \qquad \forall m' \in M,\ \forall t' \in \{t' = 1,...,TP : t' \leq t - d_{ij} \text{ or } t' \geq t + d_{ij}\}. \tag{4.7o}$$

Once the variables already set to zero by (4.7j) are excluded, then (4.7o) is equal to

$$y_{ijmt'} = 0 \quad , \qquad \forall t' \in \{t' = 1,...,TP : t' \leq t - d_{ij} \text{ or } t' \geq t + d_{ij}\}. \tag{4.7p}$$

If we branch on a variable at the root node by setting y_{ijmt} to 1, then the number of variables that we can set to either 0 or 1 is as follows:

$$
\begin{aligned}
&1+|I|*|M|*(\mathrm{Max}(t-d_{ij},0)+TP-t) \quad \text{(by 4.14a)}\\
&+|I|*(|M|-1)*(d_{ij}-\mathrm{Max}(d_{ij}-t,0)) \quad \text{(by 4.14c)}\\
&+(|I|-1)*(d_{ij}-\mathrm{Max}(d_{ij}-t,0)) \quad \text{(by 4.14e)} \quad + \quad (|J(i)|-1)*|M| \quad \text{(by 4.14f)}\\
&+(|I|-1)*(|J(i)|-1) \quad \text{(by 4.14i)} \quad + \quad |I|*(|M|-1)*TP \quad \text{(by 4.14j)}\\
&+(|I|-1)*TP \quad \text{(by 4.14l)} \quad + \quad (|J(i)|-1) \quad \text{(by 4.14n)}\\
&+\mathrm{Max}(t-d_{ij},0) \quad + \quad \mathrm{Max}(TP-t-d_{ij}+1,0) \quad \text{(by 4.14p)}
\end{aligned}
\tag{4.7q}
$$

b. Branching on x_{ijmt}, 2≤ t ≤TP-1

Setting x_{ijmt} to one allows us to set the following variables to either one or zero:

- Operation j starts only once during period t. Therefore,

$$x_{i'jm't'}=0, \qquad \forall i'\in I,\ \forall m'\in M,\ \forall t'\neq t. \tag{4.8a}$$

- Operation j starts at operating room m. Therefore,

$$x_{i'jm't'}=0, \qquad \forall i'\in I,\ \forall m'\in M\setminus m,\ \forall t'=1,...,TP. \tag{4.8b}$$

Once the variables already set to zero by (4.8a) are excluded, then (4.8b) is equal to

$$x_{i'jm't}=0, \qquad \forall i'\in I,\ \forall m'\in M\setminus m. \tag{4.8c}$$

- Surgeon i performs operation j. Therefore,

$$x_{i'jm't'}=0, \qquad \forall i'\in I\setminus i,\ \forall m'\in M,\ \forall t'=1,...,TP. \tag{4.8d}$$

Once the variables already set to zero by (4.8a) and (4.8c) are excluded, then (4.8d) is equal to

$$x_{i'jmt} = 0 \quad , \qquad \forall i' \in I \setminus i \, . \tag{4.8e}$$

- Surgeon i performs no other operation than j during periods between t and $(t + d_{ij} - 1)$.

$$x_{ij'm't'} = 0 , \qquad \forall j' \in J(i) \setminus j , \ \forall m' \in M , \ \forall t' \in \{t \le t' \le t + d_{ij} - 1\} . \tag{4.8f}$$

- Surgeon i starts operation j in operating room m during period t. Therefore,

$$x_{ijm't'} = 0 \quad , \qquad \forall (m', t') \ne (m, t) , \ \forall m' \in M , \ \forall t' = 1, ..., TP . \tag{4.8g}$$

There is no additional variable already set to zero after excluding the variables set to zero by (4.8a) and (4.8c).

- During periods between t and $(t + d_{ij} - 1)$, operating room m cannot accommodate any other pair of surgeon and operation than (i, j). We have

$$x_{i'j'mt'} = 0 , \qquad \forall (i', j') \ne (i, j) , \ \forall t' \in \{t \le t' \le t + d_{ij} - 1\} . \tag{4.8h}$$

Once the variables are set to zero by (4.8a), (4.8e), and (4.8f), then (4.8h) is equal to

$$x_{i'j'mt'} = 0 , \qquad \forall i' \in I \setminus i , \ \forall j' \in J \setminus j , \ \forall t' \in \{t \le t' \le t + d_{ij} - 1\} . \tag{4.8i}$$

- Operation j continues up to period $(t + d_{ij} - 1)$ once it starts in period t. Therefore,

$$y_{ijmt'} = 1, \qquad \forall t' \in \{t \le t' \le t + d_{ij} - 1\}. \tag{4.8j}$$

Note that the value of $(t + d_{ij} - 1)$ must be guaranteed to be no greater than TP; because, otherwise, setting this x_{ijmt} to one would lead to an infeasible node, thereby pruning the node.

- Operation j is not scheduled in any other operating room than m. Therefore,

$$y_{i'jm't'} = 0, \qquad \forall i' \in I,\ \forall m' \in M \setminus m,\ \forall t' = 1,...,TP. \tag{4.8k}$$

- Operation j is not processed by any surgeon other than surgeon i. Therefore,

$$y_{i'jm't'} = 0, \qquad \forall i' \in I \setminus i,\ \forall m' \in M,\ \forall t' = 1,...,TP. \tag{4.8l}$$

Once the variables already set to zero by (4.8k) are excluded, then (4.8l) is equal to

$$y_{i'jmt'} = 0, \qquad \forall i' \in I \setminus i,\ \forall t' = 1,...,TP. \tag{4.8m}$$

- The only assignment possible in operating room *m* during periods specified in (4.8j) is surgeon i for operation j. Therefore,

$$y_{i'j'mt'} = 0, \qquad \forall (i', j') \ne (i, j),\ \forall t' \in \{t \le t' \le t + d_{ij} - 1\}. \tag{4.8n}$$

Once the variables already set to zero by (4.8m) are excluded, then (4.8n) is equal to

$$y_{ij'mt} = 0, \qquad \forall j' \in J(i) \setminus j,\ \forall t' \in \{t \le t' \le t + d_{ij} - 1\}. \tag{4.8o}$$

- Operation j is scheduled during periods specified in (4.8j). Therefore,

$$y_{ijm't'} = 0, \qquad \forall m' \in M,\ \forall t' \in \{t' = 1,...,TP : t' \le t-1 \text{ or } t' \ge t + d_{ij}\}. \tag{4.8p}$$

Once the variables already set to zero by (4.8k) are excluded, then (4.8p) is equal to

$$y_{ijmt'} = 0, \qquad \forall t' \in \{t' = 1,...,TP : t' \le t-1 \text{ or } t' \ge t + d_{ij}\}. \tag{4.8q}$$

If the setting x_{ijmt} to 1 does not make the children nodes infeasible, then the number of variables that are set to either 0 or 1 when x_{ijmt} is set to 1 is as follows:

$$\begin{aligned} &1 + |I|*|M|*(TP-1) \quad (\text{by 4.15a}) \quad + \quad |I|*(|M|-1) \quad (\text{by 4.8c}) \quad + \quad (|I|-1) \quad (\text{by 4.15e}) \\ &+ (|J(i)|-1)*M*d_{ij} \quad (\text{by 4.15f}) \quad + \quad (|I|\text{-}1)*(|J(i)|\text{-}1)*d_{ij} \quad (\text{by 4.15i}) \\ &+ d_{ij} \quad (\text{by 4.15j}) \quad + \quad |I|*(|M|-1)*TP \quad (\text{by 4.15k}) \quad + \quad (|I|-1)*TP \quad (\text{by 4.15m}) \\ &+ (|J(i)|-1)*d_{ij} \quad (\text{by 4.15o}) \quad + \quad (t-1) \quad + (TP - t - d_{ij} + 1) \quad (\text{by 4.15q}) \\ &= 2*|I|*|M|*TP \quad + \quad (|J(i)|\text{-}1)*(|M|+|I|)*d_{ij} \end{aligned} \tag{4.8r}$$

When $t - d_{ij} \ge 0$,

$$\begin{aligned} (4.15r) - (4.14q) = {} & d_{ij} - 1 + (|J(i)|-1)*(M+I)(d_{ij}-1) + d_{ij} \\ & + (t-1) - (t-d_{ij}) + (TP - t - d_{ij} + 1) - Max(TP - t - d_{ij} + 1, 0) \end{aligned} \tag{4.9a}$$

It is guaranteed that $TP - t - d_{ij} + 1 \ge 0$ due to the assumption that the setting of $x_{ijmt} = 1$ does not make the problem infeasible. Then, the R.H.S. of (4.9a) is equal to $\{3d_{ij}\text{-}2+(|J(i)|\text{-}1)*(M+1)(d_{ij}\text{-}1)\}$. When $d_{ij}=1$, it is equal to d_{ij}.

When $t - d_{ij} < 0$,

$$\begin{aligned} (4.15r) - (4.14q) = {} & t - 1 + (|J(i)|-1)*(M+I)(d_{ij}-1) + d_{ij} \\ & + (t-1) + (TP - t - d_{ij} + 1) - Max(TP - t - d_{ij} + 1, 0) \end{aligned} \tag{4.9b}$$

Since $TP - t - d_{ij} + 1 \geq 0$, the R.H.S. of (4.9b) is equal to $\{2(t\text{-}1)+d_{ij}+(|J(i)|\text{-}1)*(M+1)(d_{ij}\text{-}1)\}$. When d_{ij}=1, it is equal to (2t-1), which is within the range of [3, 2TP-3].

Therefore, whether $t - d_{ij} \geq 0$ or $t - d_{ij} < 0$, the number of (4.8r) is always bigger than that of (4.7q) by at least d_{ij} and 3, respectively. □

Consider an example to illustrate constraint sets (4.7g) and (4.7h). Refer to Figure 4.6, where TP=8, $d_{11} = d_{22} = d_{33} = 4$, and $y_{1114} = y_{2223} = y_{3336} = 1$. Then, the values of y_{ijmt} corresponding to all gray cells are not determined. For example, if surgeon 1 starts operation 1 in period 4, the operation will continue until period 7, but if it ends in period 4, it should start in period 1. Since we do not know when it starts or ends, we cannot set any variable with respect to these cells. However, the blank cells are irrelevant to three assignments in Figure 4.6. Therefore, we have $y_{1118} = y_{2227} = y_{2228} = y_{3331} = y_{3332} = 0$. Also, operations 1, 2, and 3 cannot start after periods 4, 3, and 6, respectively, since they are already processed during these periods. In addition, operation 3 can start neither in period 1 nor in 2. Therefore,

(1) $x_{1115} = x_{1116} = x_{1117} = x_{1118} = 0$ for operation 1,

(2) $x_{2224} = x_{2225} = x_{2226} = x_{2227} = x_{2228} = 0$ for operation 2, and

(3) $x_{3331} = x_{3332} = x_{3337} = x_{3338} = 0$ for operation 3.

Figure 4.6. An Example of Branching on y_{ijmt}

		Period (TP=8)							
		1	2	3	4	5	6	7	8
Operating Room	1				(1,1)				
	2			(2,2)					
	3						(3,3)		

4.4.1.2. A Numerical Example to Illustrate Proposition 4.2

Figure 4.7. A Numerical Example for Branching Variable Selection Strategy

		Period			
		1	2	3	4
Operating Room	1		(1,1)		
	2				

In this section, we provide a numerical example for the above branching variable selection strategy. Figure 4.7 gives an example of the formulation **HSSP (General)**, where we consider two surgeons, two operations, two operating rooms, and four periods. Suppose that y_{1112} and x_{1112} have fractional values after solving **HSSP (General)** at its root node. Furthermore, suppose that $d_{11}=2$ and Surgeons 1 and 2 can perform operation 1, i.e., $I(1)=\{1,2\}$. We compare the number of variables that are set to either one or zero when $y_{1112}=1$ or $x_{1112}=1$.

a. When y_{1112} is set to one

17 variables among x_{ijmt}, $\forall i \in I$, $\forall j \in J$, $\forall m \in M$, $\forall t = 1,...,TP$, are set as follows:

$x_{1113}=x_{1114}=x_{2113}=x_{2114}=x_{1123}=x_{1124}=x_{2123}=x_{2124}=0$ by (4.7a)

$x_{1121}=x_{1122}=x_{2121}=x_{2122}=0$ by (4.7c)

$x_{2111}=x_{2112}=0$ by (4.7e)

$x_{1212}=x_{1222}=0$ by (4.7f)

$x_{2212}=0$ by (4.7i)

15 variables among y_{ijmt}, $\forall i \in I$, $\forall j \in J$, $\forall m \in M$, $\forall t = 1,...,TP$, are set as follows:

$y_{1112}=1$

$y_{1121}=y_{1122}=y_{1123}=y_{1124}=y_{2121}=y_{2122}=y_{2123}=y_{2124}=0$ by (4.7j)

$y_{2111}=y_{2112}=y_{2113}=y_{2114}=0$ by (4.7k)

$y_{1212}=0$ by (4.7l)

$y_{1114}=0$ by (4.7p)

All 32 variables above are set in the children nodes of the root node when $y_{1112}=1$.

b. When $x_{1112}=1$ is set to one

22 variables among x_{ijmt}, $\forall i \in I$, $\forall j \in J$, $\forall m \in M$, $\forall t = 1,...,TP$, are set as follows:

$x_{1112}=1$

$x_{1111}=x_{1113}=x_{1114}=x_{1121}=x_{1123}=x_{1124}=0$
$x_{2111}=x_{2113}=x_{2114}=x_{2121}=x_{2123}=x_{2124}=0$ by (4.8a)

$x_{1122}=x_{2122}=0$ by (4.8c)

$x_{2112}=0$ by (4.8e)

$x_{1212}=x_{1213}=x_{1222}=x_{1223}=0$ by (4.8f)

$x_{2212}=x_{2213}=0$ by (4.8i)

18 variables among y_{ijmt}, $\forall i \in I$, $\forall j \in J$, $\forall m \in M$, $\forall t = 1,...,TP$, are set as follows:

$y_{1112}=y_{1113}=1$ by (4.8j)

$y_{1121}=y_{1122}=y_{1123}=y_{1124}=y_{2121}=y_{2122}=y_{2123}=y_{2124}=0$ by (4.8k)

$y_{2111}=y_{2112}=y_{2113}=y_{2114}=0$ by (4.8m)

$y_{1212}=y_{1213}=0$ by (4.8o)

$y_{1111}=y_{1114}=0$ by (4.8p)

All 40 variables above are set in the children nodes of the root node when x_{1111}=1.

The difference in the numbers given by previous calculations is 8, which corresponds to (4.9a). If we increase the values of |*I*|, |*J*|, |*M*|, and/or *TP*, the difference will increase as well. Therefore, it is better to choose x_{ijmt} than y_{ijmt} when both variables are fractional.

4.4.1.3. Implementation of the New Branching Variable Selection Strategy

The first rule to find a branching variable for a node is based on the number of variables that will be set to either zero or one at its children nodes. However, it is possible that some of those variables are, in fact, redundant, i.e., some of them have already been set to either zero or one in the preceding nodes of the branch-and-bound tree. To illustrate, consider the following example.

Consider Figure 4.7 again, and suppose that we branch on x_{1111} at the root node. We investigate a child node when x_{1111} is set to one. Assume that x_{1213} and x_{2221} have fractional values after solving one of its children nodes, i.e., $\tilde{X} = \{x_{1213}, x_{2221}\}$, and $d_{12} = d_{22} = 2$.

a. When we choose x_{1213} as a branching variable

$x_{1213} = 1$

$\underline{x_{1211}} = \underline{x_{1212}} = x_{1214} = \underline{x_{1221}} = \underline{x_{1222}} = x_{1224} = 0$
$\underline{x_{2211}} = \underline{x_{2212}} = x_{2214} = x_{2221} = x_{2222} = x_{2224} = 0$ by (4.8a)

$x_{1223} = x_{2223} = 0$ by (4.8c)

$x_{2213} = 0$ by (4.8e)

$\underline{x_{1113}} = \underline{x_{1114}} = \underline{x_{1123}} = \underline{x_{1124}} = 0$ by (4.8f)

$\underline{x_{2113}} = \underline{x_{2114}} = 0$ by (4.8i)

$y_{1213} = y_{1214} = 1$ by (4.8j)

$y_{1221} = y_{1222} = y_{1223} = y_{1224} = y_{2221} = y_{2222} = y_{2223} = y_{2224} = 0$ by (4.8k)

$y_{2211} = y_{2212} = y_{2213} = y_{2214} = 0$ by (4.8m)

$\underline{y_{1113}} = \underline{y_{1114}} = 0$ by (4.8o)

$\underline{y_{1211}} = \underline{y_{1212}} = 0$ by (4.8q)

b. When we choose x_{2221} as a branching variable

$x_{2221} = 1$

$x_{2222} = x_{2223} = x_{2224} = \underline{x_{2212}} = x_{2213} = x_{2214} = 0$
$x_{1222} = x_{1223} = x_{1224} = x_{1212} = x_{1213} = x_{1214} = 0$ by (4.8a)

$\underline{x_{1211}} = \underline{x_{2211}} = 0$ by (4.8c)

$x_{1221} = 0$ by (4.8e)

$x_{2111} = \underline{x_{2112}} = x_{2121} = \underline{x_{2122}} = 0$ by (4.8f)

$\underline{x_{1121}} = x_{1122} = 0$ by (4.8i)

$y_{2221} = y_{2222} = 1$ by (4.8j)

$\underline{y_{1211}} = \underline{y_{1212}} = y_{1213} = y_{1214} = \underline{y_{2111}} = \underline{y_{2112}} = \underline{y_{2113}} = \underline{y_{2114}} = 0$ by (4.8k)

$y_{1221} = y_{1222} = y_{1223} = y_{1224} = 0$ by (4.8m)

$\underline{y_{2121}} = \underline{y_{2122}} = 0$ by (4.8o)

$y_{2223} = y_{2224} = 0$ by (4.8q)

Note that both cases seem to have 40 variables that we can fix. However, the variables that are underlined, which are 16 and 14 variables for the first and second cases, respectively, are already set to zero at a root node. In other words, 24 and 26 new variables are added to the list of variables set to one or zero, for the first and second cases, respectively. Therefore, it is better to choose x_{2221} as a branching variable for the child node.

Suppose $\widetilde{X}$ and $\widetilde{Y}$ are the sets of x_{ijmt} and y_{ijmt} variables with fractional values, respectively, as follows:

$$\widetilde{X} = \{x_{ijmt}, \forall i \in I, \forall j \in J(i), \forall m \in M, \forall t = 1,...,TP : 0 < x^*_{\text{ijmt}} < 1\} \quad (4.9a)$$

$$\widetilde{Y} = \{y_{ijmt}, \forall i \in I, \forall j \in J(i), \forall m \in M, \forall t = 1,...,TP : 0 < y^*_{\text{ijmt}} < 1\} \quad (4.9b)$$

where x^*_{ijmt} and y^*_{ijmt} represent the solution given at the current node.

Then, a new branching variable selection strategy, which considers both the size of problems in the children nodes and the most infeasibility strategy, is as follows:

Step 1. If $\tilde{X} \neq \phi$, go to step 3; otherwise, go to step 2.

Step 2. If $\tilde{Y} \neq \phi$, go to step 4; otherwise, fathom the node since the node yields an integer feasible solution.

Step 3. Find $x_{ijmt} \in \tilde{X}$ for which we can set the most number of new variables to either one or zero by setting it to one. If there is a tie, go to step 5; otherwise, choose it as the branching variable for the current node and stop the branching variable selection process.

Step 4. Find $y_{ijmt} \in \tilde{Y}$ for which we can set the most number of new variables to either one or zero by setting it to one. If there is a tie, go to step 5; otherwise, choose it as the branching variable for the current node and stop the branching variable selection process.

Step 5. Find a variable with the most infeasibility among the variables transferred from either step 3 or step 4 and choose it as the branching variable for the current node. If there is a tie, choose arbitrarily.

4.4.2. A Procedure to Find Initial Feasible Solutions at Each Node

Recall that for the SGAP, we employ the Phase II method (**BNP-SGAP**) and develop a greedy algorithm (**BNPDSTKP-SGAP**) to find an initial feasible solution at each node. Both of them guarantee a feasible solution at the node. For the three HSSP models, we also provide a way to find an initial feasible solution.

We set to zero all the variables except for the ones that are already set to either zero or one at the current node of the branch-and-bound tree. That is, we start with no additional operation assigned to any surgeon unless it is pre-fixed in the branch-and-bound tree. We use this solution as an initial feasible solution at each node.

4.4.3. A BNP Method for the HSSP Models

Figure 4.8 presents the flow diagram of the BNP method for the HSSP models presented in this chapter. It depicts the new strategies to find branching variable and an initial feasible solution for each node mentioned in sections 4.5.1 and 4.5.2, respectively, as well as the general procedure of the BNP method. Steps 1 to 6 represent the column generation method (CGM) and steps 7 to 16 represent the branch-and-bound (BNB) process. Referring to Figure 3.2, which describes a procedure of the BNP method for the **SGAP** presented in Chapter III, all the steps belonging to Phase I are not needed, and hence, are deleted.

- **Step 1 (BNB):** Solve the relaxed HSSP model after updating the relevant node information from the branch-and-bound tree. Since it is a maximization problem, the objective function value provides an upper bound to the corresponding HSSP model.
- **Step 2 (BNB, Finding an initial feasible solution):** As mentioned in section 4.4.2, set all the variables to either zero or one based on the history of the current BNB tree, and set all the rest of variables to zero.
- **Step 3 (CGM):** Relax the convexity variables and solve the relaxed RMP and set the objective function value as a lower bound to the optimal objective function value of the relaxed RMP. Pass the dual variables to the corresponding pricing problems.
- **Step 4 (CGM):** With the dual prices from the relaxed RMP, construct the pricing problems corresponding to each surgeon. Solve all the pricing problems.
- **Step 5 (CGM, Check for the availability of a column with a negative reduced cost):** If a positive reduced cost exists, i.e., if there is a legitimate solution to improve the objective function value of the next RMP, go to step 3; otherwise, if all objective function values of all pricing problems are zero, go to step 6.
- **Step 6 (BNB):** Obtain the values of the original variables by combining the columns and the values of convexity variables. If the objective function value of the current node is better than the incumbent objective function value, go to step 7; otherwise, go to step 12.

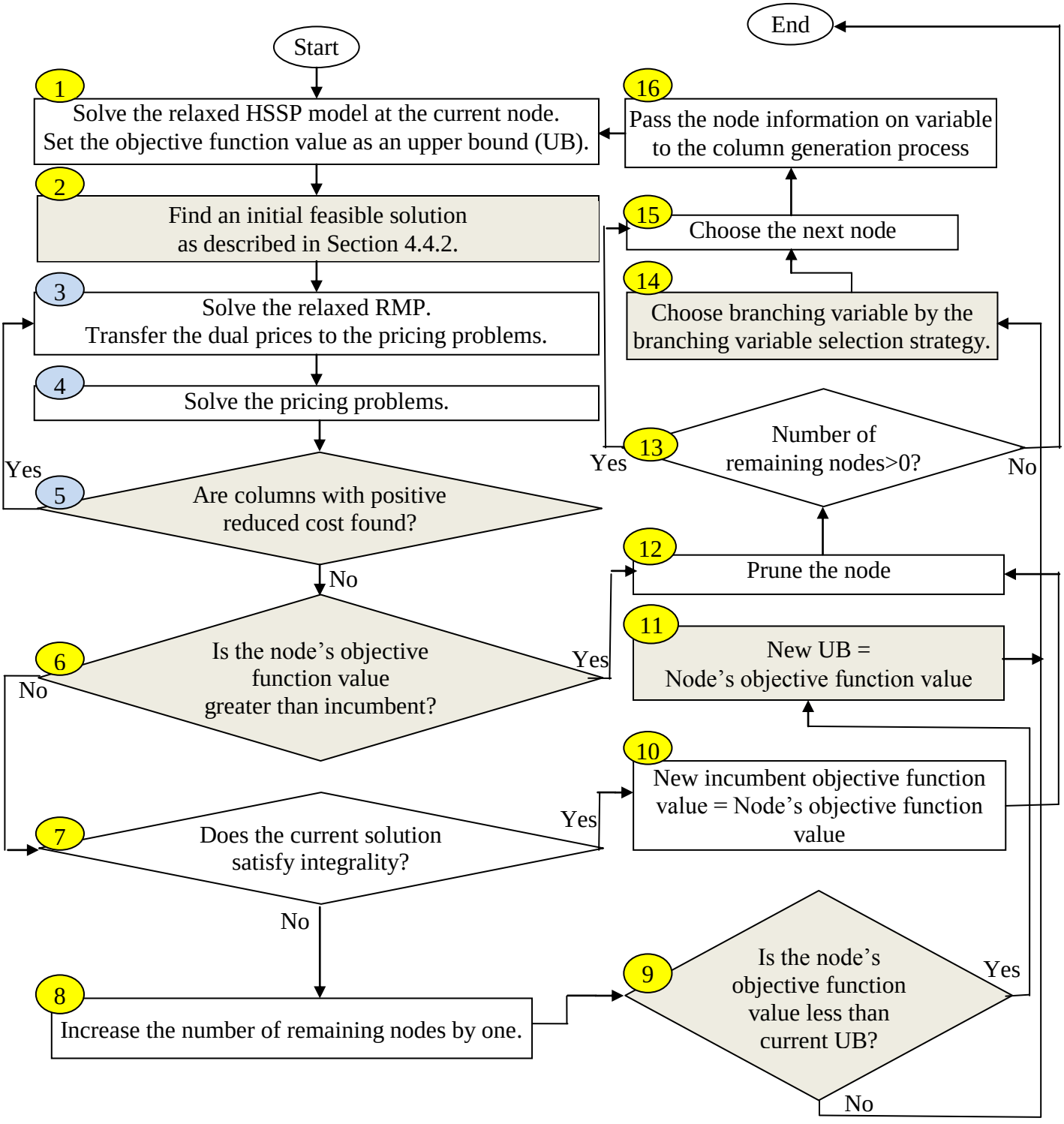

Figure 4.12. Branch-and-Price Method for HSSP Models

- **Step 7 (BNB, Check for integrality):** Check for integrality of the original variables. If integrality is satisfied, go to step 10; otherwise, go to step 8.
- **Step 8 (BNB, Unsatisfied integrality):** Increase the number of remaining nodes by one.
- **Step 9 (BNB):** If the objective function value is less than the current upper bound, go to step 11; otherwise, go to step 14.
- **Step 10 (BNB, New incumbent solution and objective function value):** Claim the solution and the objective function value as new incumbent solution and objective function value, and go to step 12.
- **Step 11 (BNB, New upper bound):** Claim the objective function value as the new upper bound. Go to step 14.
- **Step 12 (BNB):** Prune the node. Decrease the number of the remaining nodes by 1, and go to step 13.
- **Step 13 (BNB, Check for the number of the remaining nodes):** If the number of remaining nodes is zero, generate the incumbent solution and the objective function value as the optimal solution and the optimal objective function value and stop the algorithm; otherwise, go to step 15.
- **Step 14 (BNB, Branching variable selection strategy):** Choose the branching variable for the current node by the strategy devised in Section 4.4.1. Find a variable that allows us to set the most variables to either one or zero, and if there are ties, choose the one that was nearest to 0.5. Go to step 15.
- **Step 15 (BNB, Node selection strategy):** Choose the next node out of the set of the remaining nodes by the node selection strategy under which the next node is the one that has the lowest upper bound and go to step 16.
- **Step 16 (BNB, Node information):** Collect the node information, such as variables which are fixed at either one or zero, the current upper bound, and the incumbent objective function value as a lower bound, and go to step 1 to start a new column generation process.

Compared to the BNP procedure for the SGAP, which is described in Figure 3.2, the above procedure differs in steps 2, 5, 6, 9, 11 and 14, which are in gray in Figure 4.8. In step 2, it uses the method addressed in Section 4.4.2. In step 5, the CGM checks if there exists a column with positive reduced costs (in contrast to negative reduced cost earlier). In step 6, if the objective

function value at a node is no greater than the incumbent objective function value, the node is pruned. In steps 9 and 11, the procedure limits the upper bound when we have a fractional solution. In step 14, a branching variable is determined by the branching variable section strategy presented in Section 4.4.1.

4.5. Computational Experimentation

4.5.1. Data Generation for the HSSP (Prearranged)

First, we show an example for data or coefficients for one of the HSSP models, which is **HSSP (Prearranged)**, where Type I surgeons perform their operations with a prearranged schedule.

4.5.1.1. Data on Type I Surgeons

In this section, we introduce a procedure to generate the required data by introducing one of the examples that we employ in the experimentation. Recall that I represents the set of surgeons, i.e., $|I| = |I_1| + |I_2|$ and S_{I_1} and S_{I_2} are the sets of Type I surgeons and Type II surgeons, i.e., $S_{I_1} = \{1,...,I_1\}$ and $S_{I_2} = \{1,...,I_2\}$. Then, we determine the data concerning Type I surgeons as follows:

- Number of Type I surgeons : $|I_1| = \left\lfloor \frac{M}{2} \right\rfloor$.
- Number of operations that a Type I surgeon conducts : $|J(i_1)| = 2 \quad \forall i_1 \in S_{I_1}$.
- Set of operations that Type I surgeon i_1 conducts : $J(i_1) = \{2i_1 - 1, 2i_1\}$ where $1 \le i_1 \le \left\lfloor \frac{M}{2} \right\rfloor$.
- Number of operations assigned to Type I surgeons : $\sum_{i_1} J(i_1) = 2I_1 = 2\left\lfloor \frac{M}{2} \right\rfloor$.

- Variables set to one :

  $y_{i_1,2i_1-1,2i_1-1,1}=1,\ x_{i_1,2i_1-1,2i_1-1,t}=1$ where $1\le t\le i_1\%2+3$, and

  $y_{i_1,2i_1-1,2i_1-1,i_1\%2+4}=1,\ x_{i_1,2i_1,2i_1,t}=1$ where $i_1\%2+4\le t\le 7$.

4.5.1.2. Data on Type II Surgeons

The data related to Type II surgeons is as follows:

- Number of Type II surgeons : $|I_2|=k_1|I_1|=k_1\left\lfloor\frac{M}{2}\right\rfloor$, where k_1 is a constant.
- Number of operations assigned to Type II surgeons : $|J(i_2)|=k_2\sum_{i_1}|J(i_1)|=2k_2\left\lfloor\frac{M}{2}\right\rfloor$

  where k_2 is a constant.
- Set of Type II surgeons : $S_{I_2}=\left\{\left\lfloor\frac{M}{2}\right\rfloor+1,...,|I_2|\right\}$, where $|I_2|=(k_1+1)|I_1|$.
- Set of operations assigned to Type II surgeons : $J_2=\left(2\left\lfloor\frac{M}{2}\right\rfloor+1,...,2(k_2+1)\left\lfloor\frac{M}{2}\right\rfloor\right)$.
- Sets of Type II surgeons who can conduct each $j_2\in J_2$:

  $I(M+1)=\left\{\left\lfloor\frac{M}{2}\right\rfloor+1,\left\lfloor\frac{M}{2}\right\rfloor+2,\left\lfloor\frac{M}{2}\right\rfloor+3\right\}$,

  $I(M+2)=\left\{\left\lfloor\frac{M}{2}\right\rfloor+2,\left\lfloor\frac{M}{2}\right\rfloor+3,\left\lfloor\frac{M}{2}\right\rfloor+4\right\}$, and

  if the set of Type II surgeons for operation $M+k$, $I(M+k)$, says $\left\{i_2^1,i_2^2\right\}$, then

$$I(M+k+1)=\begin{bmatrix}\left\{i_2^1+1,i_2^2+1\right\}, & \text{if } i_2^1+1\le 2(k_2+1)\left\lfloor\frac{M}{2}\right\rfloor \text{ and } i_2^2+1\le 2(k_2+1)\left\lfloor\frac{M}{2}\right\rfloor\\ \left\{\left\lfloor\frac{M}{2}\right\rfloor+1,i_2^2+1\right\}, & \text{if } i_2^1+1>2(k_2+1)\left\lfloor\frac{M}{2}\right\rfloor \text{ and } i_2^2+1\le 2(k_2+1)\left\lfloor\frac{M}{2}\right\rfloor\\ \left\{i_2^1+1,\left\lfloor\frac{M}{2}\right\rfloor+1\right\}, & \text{if } i_2^1+1\le 2(k_2+1)\left\lfloor\frac{M}{2}\right\rfloor \text{ and } i_2^2+1>2(k_2+1)\left\lfloor\frac{M}{2}\right\rfloor\end{bmatrix}.$$

4.5.1.3. Data on Coefficients

Coefficients for profits (r_j), costs (c_i), and operation times (d_{ij})

- Operation time for each operation

  $d_{i_1 j} = (i_1 + j) \bmod (2) + 3 \qquad \forall i_1 \in S_{I_1} \qquad j = 1,\ldots,M$

  $d_{i_2 j} = (i_2 + j) \bmod (3) + 1 \qquad \forall i_2 \in S_{I_2} \qquad j = M+1,\ldots,(k_2+1)M$

  where ($a \bmod b$) is the remainder when a is divided by b.

- Profit gained from each operation :

  $p_j = 4{,}000 d_{i_1 j} \qquad \forall i_1 \in S_{I_1}, \forall j \in J(i_1)$

  $p_j = 2{,}000 d_{i_2 j} \qquad \forall i_2 \in S_{I_2}, \forall j \in J(i_2)$

- Cost incurred by each operation during overtime

  $c_{i_1} = 1{,}000 \qquad \forall i_1 \in S_{I_1}$

  $c_{i_2} = 300 + 200 * (i_2 \bmod 2) \qquad \forall i_2 \in S_{I_2}$

4.5.2. Experimental Designs

The formulations we have presented so far are:

- **HSSP (General)** : Implementation of schedules incorporating surgeons, operations, operating rooms, and operation times
- **HSSP (Priority)** : Inclusion of priority for Type I surgeons over Type II surgeons in **HSSP (General)**
- **HSSP (Pre-arranged)** : Implementation of completely pre-fixed schedule for Type II surgeons

We apply the Branch-and-Cut (BNC) and Branch-and-Price (BNP) methods to each case. We designate the BNC method for **HSSP (General)**, **HSSP (Priority)**, and **HSSP (Pre-arranged)** as **BNC-HSSP (General)**, **BNC-HSSP (Priority)**, and **BNC-HSSP (Pre-arranged)**, respectively. The corresponding BNP methods for these models are denoted by **BNP-HSSP (General)**, **BNP-HSSP (Priority)**, and **BNP-HSSP (Pre-arranged)**, respectively. The results of the BNC methods for a HSSP model are compared with those of the BNP methods. In our experimentation, we study the following aspects.

- Comparative performances of the algorithms:
  - **BNC-HSSP (General)** vs. **BNP-HSSP (General)**
  - **BNC-HSSP (Priority)** vs. **BNP-HSSP (Priority)**
  - **BNC-HSSP (Pre-arranged)** vs. **BNP-HSSP (Pre-arranged)**
- Effect of variation in the number of surgeons for **HSSP (General)**
  - Number of Type I and Type II surgeons for **HSSP (Priority)** and **HSSP (Pre-arranged)**.
- Effect of variation in the number of operations
  - Number of operations of Type I and Type II surgeons for **HSSP (Priority)** and **HSSP (Pre-arranged)**.
- Effect of variation in the number of operating rooms
- Effect of variation in the operation times

The following three sections give further details on the experimental designs for each of the above aspects.

4.5.2.1. BNC-HSSP (General) vs. BNP-HSSP (General)

The data used for this experimentation is included in Table C.1 in Appendix C. It shows information on the experiment number, the number of operating rooms, the number of surgeons, and the number of operations to be planned. We ran 10 different instances for each case of the number of operating rooms leading to 30 experiments in total. The values of $|M|$ and $|I|$ are varied from 2 to 4, and from 2 to 16, respectively. Then, the ratio of $|J|$ to $|I|$ is varied from 1 to 4. As a result, $|J|$ increases up to 16.

4.5.2.2. BNC-HSSP (Priority) vs. BNP-HSSP (Priority)

The data used for this experimentation is included in Table C.2 in Appendix C. For this case, the surgeons are split into two groups, Type I and Type II surgeons. Type I surgeons have advantage over Type II surgeons in scheduling their operations. Also, time slots are pre-specified for Type I surgeons. However, operating rooms for their operations are still to be determined. Therefore, the time index in the variables for Type I surgeons is restricted by the range of those durations.

The number of operating rooms is varied from 2 to 5, and the number of Type I surgeons is varied from 1 to 2. The number of Type II surgeons is at least twice as many as that of Type I surgeons and increases up to 12 times as many as that of Type I surgeons. As a result, the largest value of $|I_1|$ is 16. For the sake of simplicity, we assume that each Type I surgeon performs 2 operations. Therefore, $|J_1| = 2|I_1|$ for all the experiments. The ratio of $|J_2|$ to $|J_1|$, that is, $\frac{|J_2|}{|J_1|}\left(= \frac{\sum_{i_2 \in I_2} |J_2(i_2)|}{2|I_1|}\right)$, ranges from 2 to 7.5. As a result, the value of $|J_2|\left(= \sum_{i_2 \in I_2} |J_2(i_2)|\right)$ increases up to 20.

4.5.2.3. BNC-HSSP (Pre-arranged) vs. BNP-HSSP (Pre-arranged)

The data used for this experimentation is depicted in Table C.3 in Appendix C. Type I surgeons have pre-specified schedule for their operations including operating rooms and the duration of

each operation. Therefore, all the variables in these formulations belong to Type II surgeons. For each operating room, we use 16 experiments as shown in Table 4.4, and in total, we ran 32 experiments. The values of $|M|$ and $|I_1|$ vary from 3 to 4 and from 1 to 2, respectively. The value of $\frac{|I_2|}{|I_1|}$ varies from 1 to 2 with the change of $|I_2|$ from 1 to 8. The value of $|J_1|$ is equal to $2|I_1|$, and the value of $\frac{|J_2|}{|J_1|}$ varies from 1 to 4. As a result, the value of $|J_2|$ increases up to 16.

4.6. Results and Analysis

4.6.1. Results for BNC-HSSP (General) and BNP-HSSP (General)

Table 4.2 depicts the results for **BNC-HSSP (General)** and **BNP-HSSP (General)**. The experiment number, the number of operating rooms, the number of surgeons, and the number of operations are presented in the 1st, 2nd, 3rd, and 4th columns, respectively. The 5th, 6th, and 7th columns contain the number of binary variables, the number of constraints, and the optimal objective function value, respectively. The cpu times for **BNC-HSSP (General)** and **BNP-HSSP (General)** are presented in the 8th and 9th columns, respectively.

As shown in Table 4.2, the BNP method performs better than the BNC method in 22 out of 30 experiments (a method is highlighted when it performs better). In addition, the BNC method fails to attain an optimal solution for Experiments 24, 27, 29, and 30 within 24 hours of cpu time. On the other hand, all the problems are successfully solved by the BNP method. The maximum size of a problem solved by the BNC method consists of 3,456 binary variables and 1,968 constraints for Experiment 28, while the BNP method still can solve a problem involving 4,608 binary variables and 2,608 constraints for Experiment 30.

Table 4.2. Results for **BNC-HSSP (General)** and **BNP-HSSP (General)**

Experiment #	$\|M\|$	$\|I\|$	$\|J\|$	Number of Binary Variables	Number of Constraints	Optimal Objective Function Value	CPU Time (Seconds)	
							BNC-HSSP (General)	**BNP-HSSP (General)**
1	2	2	2	192	150	16000	0.1922	0.1563
2	2	2	4	384	252	50000	0.0313	0.0185
3	2	2	6	576	354	115400	0.2168	0.2288
4	2	2	8	768	456	186000	25.8250	1.3601
5	2	4	4	576	376	52000	0.1124	0.0919
6	2	4	6	864	528	117800	1.1473	0.3997
7	2	4	8	1152	680	204800	125.6726	14.0883
8	2	6	6	864	552	125600	0.6220	1.3735
9	2	6	8	1152	704	212000	261.3154	1.7444
10	2	8	8	1152	728	198800	189.8812	4.6797
11	3	3	3	648	408	36000	0.0425	0.0475
12	3	3	6	1296	744	126000	0.2219	0.2065
13	3	3	9	1944	1080	269400	0.3180	0.2916
14	3	3	12	2592	1416	462000	1162.7899	299.6061

Table 4.2. Results for **BNC-HSSP (General)** and **BNP-HSSP (General)** (Continued)

Experiment #	$\|M\|$	$\|I\|$	$\|J\|$	Number of Binary Variables	Number of Constraints	Optimal Objective Function Value	CPU Time (Seconds)	
							BNC-HSSP (General)	**BNP-HSSP (General)**
15	3	6	6	1296	780	126000	0.1391	0.1239
16	3	6	9	1944	1116	269400	378.4302	3.0668
17	3	6	12	2592	1452	462000	946.5295	504.3430
18	3	9	9	1944	1152	269400	730.6724	3.5628
19	3	9	12	2592	1488	462000	1596.7929	23.0229
20	3	12	12	2592	1524	462000	0.9224	0.8944
21	4	4	4	1152	688	52000	0.0485	0.0659
22	4	4	8	2304	1280	206800	1.7947	17.3450
23	4	4	12	3456	1872	436600	15.5108	7.0655
24	4	4	16	4608	2464	714400	**	4249.5659
25	4	8	8	2304	1328	202000	0.4737	7.7491
26	4	8	12	3456	1920	453400	45.4257	2611.0511
27	4	8	16	4608	2512	762800	**	5533.7637
28	4	12	12	3456	1968	467200	11.0941	811.2395
29	4	12	16	4608	2560	784400	**	5150.6845
30	4	16	16	4608	2608	731200	**	7291.6951

: The **BNC-HSSP (General) method cannot yield an optimal solution to the corresponding experiment within 24 hours of cpu time.

4.6.2. Results for BNC-HSSP (Priority) and BNP-HSSP (Priority)

Table 4.3 summarizes the results for **BNC-HSSP (Priority)** and **BNP-HSSP (Priority)**. The columns of this table are identical to those of Table 4.2 except for the following columns. The 3rd, 4th, 5th, and 6th columns contain information on the number of Type I surgeons, the number of Type II surgeons, the number of operations assigned to Type I surgeons, and the number of operations assigned to Type II surgeons, respectively. The cpu times of **BNC-HSSP (Priority)** and **BNP-HSSP (Priority)**,are presented in the 10th and 11th columns, respectively. The cells with "**" in Table 4.3 indicates that the BNC method cannot attain an optimal solution for Experiments 26, 27, 29, 30, and 31 within 24 hours of cpu time. The BNP method obtains an optimal solution for all the experiments. The BNP method performs better than the BNC method for **HSSP (Priority)** for 29 out of 34 experiments (a method is highlighted when it performs better).

4.6.3. Results for BNC-HSSP (Pre-arranged) and BNP-HSSP (Pre-arranged)

Table 4.4 presents the results for **BNC-HSSP (Pre-arranged)** and **BNP-HSSP (Pre-arranged)**. All the other columns are identical to those of Table 4.3. Note that an optimal solution for **HSSP (Pre-arranged)** cannot be found for 6 experiments by the BNC method within the pre-specified cpu time limit of 24 hours, while all the instances can be solved by the BNP method. The maximum size of the problem solved by the BNC method is for Experiment 30, which has 2,304 binary variables and 1,328 constraints, while the BNP method can solve a larger problem for Experiment 32, which has 4,608 binary variables and 2,512 constraints.

As shown in Table 4.4, the performance of the BNC method deteriorates after Experiment 18. The BNC method fails to produce an optimal solution for Experiments 23, 24, 27, 28, 31, and 32 when the number of Type II surgeons' operations is either 12 or 16. The problem size impacts the performance of both BNC and BNP methods, however, more so for the BNC method than the BNP method. The performance of the BNP method is almost two-fold better than that of the BNC method for large-size problems.

Table 4.3. Results for **BNC-HSSP (Priority)** and **BNP-HSSP (Priority)**

Experiment #	$\|M\|$	$\|I_1\|$	$\|I_2\|$	$\|J_1\|$	$\|J_2\|$	Number of Binary Variables	Number of Constraints	Optimal Objective Function Value	CPU Time (Seconds)	
									BNC-HSSP (Priority)	**BNP-HSSP (Priority)**
1	2	1	2	2	4	440	291	204000	0.275463	0.270739
2	2	1	2	2	6	632	393	282000	0.150833	0.043238
3	2	1	2	2	8	824	495	354000	0.340341	12.82922
4	2	1	2	2	10	1016	597	432000	0.414406	0.045939
5	2	1	4	2	6	920	567	286000	0.289351	0.244912
6	2	1	4	2	8	1208	719	366000	0.211284	0.171553
7	2	1	6	2	8	1208	743	318000	0.461813	4.315312
8	2	1	6	2	10	1496	895	438000	0.405673	0.18225
9	2	1	8	2	10	1496	919	392000	1590.669	1.332128
10	3	1	3	2	6	1380	797	368200	1.559512	0.251106
11	3	1	3	2	9	2028	1133	564000	8.1263	0.061173
12	3	1	3	2	12	2676	1469	750000	4.670578	0.13817
13	3	1	3	2	15	3324	1805	936000	6.374635	0.092481
14	3	1	6	2	9	2028	1169	500000	5.616553	1.7792
15	3	1	6	2	12	2676	1505	750000	23.96261	8.465811

Table 4.3. Results for **BNC-HSSP (Priority)** and **BNP-HSSP (Priority)** (Continued)

Experiment #	$\|M\|$	$\|I_1\|$	$\|I_2\|$	$\|J_1\|$	$\|J_2\|$	Number of Binary Variables	Number of Constraints	Optimal Objective Function Value	CPU Time (Seconds)	
									BNC-HSSP (Priority)	**BNP-HSSP (Priority)**
16	3	1	9	2	12	2676	1541	686000	127.5594	5.832567
17	3	1	9	2	15	3324	1877	802000	198.4657	2.390887
18	3	1	12	2	15	3324	1913	872000	23.4063	3117.296
19	4	2	4	4	8	2528	1414	796000	2.455984	4.956785
20	4	2	4	4	12	3680	2006	1120000	22.02154	6.461505
21	4	2	4	4	16	4832	2598	1420000	22294.2	9.618318
22	4	2	4	4	20	5984	3190	1708000	16.6276	14.5576
23	4	2	8	4	12	3680	2054	942000	1.136048	1.016485
24	4	2	8	4	16	4832	2646	1416000	11.02344	4.689388
25	4	2	12	4	16	4832	2694	1246000	258.3213	15.70642
26	4	2	12	4	20	5984	3286	1708000	**	2953.759
27	4	2	16	4	20	5984	3334	942000	**	13.02692
28	5	2	5	4	10	3880	2122	1087600	258.0406	5.975523
29	5	2	5	4	15	5680	3042	1478400	**	12.79423
30	5	2	5	4	20	7480	3962	1791200	**	15460.51
31	5	2	5	4	25	9280	4882	2290000	**	26.30478
32	5	2	10	4	15	5680	3102	1418000	1265.22	24.09636
33	5	2	10	4	20	7480	4022	2108000	13165.46	18.47846
34	5	2	15	4	20	7480	4082	1914000	29.01138	46.77946

: The **BNC-HSSP (Priority) method cannot yield an optimal solution to the corresponding experiment within 24 hours.

Table 4.4. Results for **BNC-HSSP (Pre-arranged)** and **BNP-HSSP (Pre-arranged)**

Experiment #	$\|M\|$	$\|I_1\|$	$\|I_2\|$	$\|J_1\|$	$\|J_2\|$	Number of Binary Variables	Number of Constraints	Optimal Objective Function Value	CPU Time (Seconds)	
									BNC-HSSP (Pre-arranged)	**BNP-HSSP (Pre-arranged)**
1	3	1	1	2	2	144	124	50000	0.0104	0.0101
2	3	1	1	2	4	288	200	76000	0.0129	0.0146
3	3	1	1	2	6	432	276	130000	0.0494	0.0510
4	3	1	1	2	8	576	352	166000	0.0775	0.0755
5	3	1	2	2	2	288	210	56000	0.0171	0.0180
6	3	1	2	2	4	576	360	90000	0.0906	7.2147
7	3	1	2	2	6	864	510	155400	0.1183	0.1000
8	3	1	2	2	8	1152	660	226000	4.5616	101.7347
9	3	1	3	2	2	432	296	58000	0.0186	0.0163
10	3	1	3	2	4	864	520	100000	0.0943	0.1242
11	3	1	3	2	6	1296	744	165200	14.5197	86.7186
12	3	1	3	2	8	1728	968	252800	1066.8200	163.5428
13	3	1	4	2	2	432	308	58000	0.0184	0.0163
14	3	1	4	2	4	864	532	92000	0.0487	0.0715
15	3	1	4	2	6	1296	756	157600	20.2979	56.3632
16	3	1	4	2	8	1728	980	245600	78.8438	2.2726

Table 4.4. Results for **BNC-HSSP (Pre-arranged)** and **BNP-HSSP (Pre-arranged)** (Continued)

Experiment #	$\|M\|$	$\|I_1\|$	$\|I_2\|$	$\|J_1\|$	$\|J_2\|$	Number of Binary Variables	Number of Constraints	Optimal Objective Function Value	CPU Time (Seconds)	
									BNC-HSSP (Pre-arranged)	**BNP-HSSP (Pre-arranged)**
17	4	2	2	4	4	768	468	134000	0.2136	0.2296
18	4	2	2	4	8	1536	864	270000	2.8786	124.9201
19	4	2	2	4	12	2304	1260	462000	5210.9501	151.9375
20	4	2	2	4	16	3072	1656	648000	20.9835	108.0589
21	4	2	4	4	4	1152	688	136000	0.2291	0.0688
22	4	2	4	4	8	2304	1280	289600	69.1063	30.0705
23	4	2	4	4	12	3456	1872	496000	**	939.9493
24	4	2	4	4	16	4608	2464	756000	**	311.4741
25	4	2	6	4	4	1152	712	144000	0.2310	0.0724
26	4	2	6	4	8	2304	1304	298000	229.8048	33.2847
27	4	2	6	4	12	3456	1896	484200	**	1097.2207
28	4	2	6	4	16	4608	2488	784000	**	662.4164
29	4	2	8	4	4	1152	736	144000	0.0791	0.0733
30	4	2	8	4	8	2304	1328	284400	41.2013	4.0334
31	4	2	8	4	12	3456	1920	515400	**	254.5011
32	4	2	8	4	16	4608	2512	762000	**	2572.0084

** : The **BNC-HSSP (Pre-arranged)** method cannot yield an optimal solution to the corresponding experiment within 24 hours.

4.7. Concluding Remarks

In this chapter, we have addressed three different types of the hospital staff scheduling problem (HSSP): **HSSP (General)**, **HSSP (Priority)**, and **HSSP (Pre-arranged)**. We have introduced and developed the BNC and BNP methods for each of these problems. These methods are designated as **BNC-HSSP (General)**, **BNP-HSSP (General)**, **BNC-HSSP (Priority)**, **BNP-HSSP (Priority)**, **BNC-HSSP (Pre-arranged)**, and **BNP-HSSP (Pre-arranged)**, respectively. We have compared the performances of each pair of these methods. Table 4.5 summarizes the results of these comparisons. The numbers in the cells represent the number of times each method performs better than the other method from the viewpoint of cpu time. For example, for the formulation **HSSP (General)**, the BNC and BNP methods outperform the other method in 8 and 22 experiments out of 30 experiments, respectively. Furthermore, the number in the parenthesis represents the number of experiments for which the BNC method fails to obtain an optimal solution.

Overall, the BNP method performs better than the BNC method in 70 % of experiments. Another important aspect is the ability of a method to find an optimal solution. The BNC method fails to do so for 4, 5, and 6 experiments for **HSSP (General)**, **HSSP (Priority)**, and **HSSP (Pre-arranged)**, respectively. In total, it fails to obtain an optimal solution for 15% of the experiments. Clearly, we can conclude that the BNP method is far better than the BNC method in solving the HSSP problems that we have addressed.

Table 4.5. BNC vs. BNP Methods for HSSP Models

	BNC Method	BNP Method
HSSP (General)	8 (4)	22
HSSP (Priority)	5 (5)	29
HSSP (Pre-arranged)	12 (6)	20
Total	25 (15)	71

Chapter V. Stochastic Hospital Staff Scheduling Problem (SHSSP)

5.1. Model Description

In this chapter, we add another feature to the general HSSP discussed in Chapter IV by making the operation times stochastic. In other words, the time required by surgeon i to perform operation j, d_{ij}, which is deterministic in the HSSP (General), is now assumed to be stochastic. Let d_{ij} follow a discrete probability function, i.e., $p(d_{ijs})=p_s$, $s=1,\ldots,|S|$ for a given (i, j), where S is the set of scenarios, $|S|$ is the total number of scenarios, and p_s is the probability of occurrence of scenario s. We designate this problem as SHSSP (General).

5.2. Formulation for the Stochastic Hospital Staff Scheduling Problem (General)

We use the following additional notation.

d_{ijs} : Length of operation times required by surgeon i to perform operation j under scenario s, $\forall i \in I$, $\forall j \in J(i)$, $\forall s \in S$

e^M : Penalty for assigning more than one operation in an operating room at a time

e^I : Penalty for assigning more than one operation to a surgeon at a time

e^D : Penalty for violating pre-specified operation times

y_{ijmts} : Binary variable; =1 if surgeon i performs operation j in operating room m during period t under scenario s; otherwise, =0, $\forall i \in I$, $\forall j \in J(i)$, $\forall m \in M$, $\forall t = 1,...,TP$, $\forall s \in S$

w^{M}_{mts} : Continuous variable representing the number of operations assigned to operating room *m* during period *t* under scenario *s*

w^{I}_{its} : Continuous variable representing the number of operations assigned to surgeon *i* during period *t* under scenario *s*

w^{-}_{ijs} $\left(w^{+}_{ijs}\right)$: Continuous slack (surplus) variable representing over-assigned (under-assigned) operation times to surgeon *i* to perform operation j under scenario *s*

$\vec{x}$: Vector for the 1st stage variables, x_{ijmt} , $\forall i \in I$, $\forall j \in J(i)$, $\forall m \in M$, $\forall t = 1,...,TP$

$f_s(\vec{x})$: Recourse model for scenario s , $\forall s \in S$

$\text{Exp}[\tilde{f}(\vec{x})]$: Expected cost for operations performed during overtime

We use a two-stage approach for the SHSSP (General), where the variables are split into the first and second stage variables. This separation of variables depends on which decisions should be executed sooner than others. If a decision contains urgent or important information, the variables regarding that decision should constitute the first stage variables, and the others should constitute the second stage variables.

The SHSSP (General) consists of two sets of decision variables, x_{ijmt} and y_{ijmts}, as in the HSSP (General). The x_{ijmt} variables determine the starting time and the operating room for an operation, and the y_{ijmts} variables determine the length of operation times required by a surgeon to perform an operation. Apparently, x_{ijmt} are the key decisions, and y_{ijmts} are subservient to them because a surgeon cannot stop an operation prematurely. These decision variables are related to the variables in **HSSP (General)**, which are x_{ijmt} and y_{ijmt}. Besides, the value of y_{ijmt} is subject to that of x_{ijmt} and d_{ij}. Therefore, x_{ijmt} and y_{ijmt} are the first and second stage variables, respectively.

SHSSP1 (General)

Maximize $\sum_{i\in I}\sum_{j\in J(i)}\sum_{m\in M}\sum_{t=1}^{TP} r_j x_{ijmt} + \text{Exp}[\tilde{f}(\vec{x})]$ (5.1a)

Subject to $\sum_{i\in I(j)}\sum_{m\in M}\sum_{t=1}^{TP} x_{ijmt} \le 1 \qquad \forall j \in J$ (5.1b)

$x_{ijmt} \in \{0,1\} \qquad \forall i \in I,\ \forall j \in J(i),\ \forall m \in M,\ \forall t = 1,...,TP$ (5.1c)

The recourse model for each scenario s, $\forall s \in S$, $f_s(\vec{x})$, is given by :

Recourse Model for SHSSP1 (General)

Maximize $-\sum_{i\in I}\sum_{j\in J(i)}\left[\sum_{m\in M}\sum_{t=T+1}^{TP}\left(c_i(t-T)y_{ijmts}\right)\right] - \sum_{t=1}^{TP}\left(e^M\sum_{m\in M} w_{mts}^M + e^I\sum_{i\in I} w_{its}^I\right) - e^D\sum_{i\in I}\left(w_{ijs}^- + w_{ijs}^+\right)$ (5.2a)

Subject to $\sum_{i\in I}\sum_{j\in J(i)} y_{ijmts} - w_{mts}^M \le 1 \qquad \forall m \in M,\ \forall t = 1,...,TP$ (5.2b)

$\sum_{j\in J(i)}\sum_{m\in M} y_{ijmts} - w_{its}^I \le 1 \qquad \forall i \in I,\ \forall t = 1,...,TP$ (5.2c)

$y_{ijm1s} = x_{ijm1} \qquad \forall i \in I,\ \forall j \in J(i),\ \forall m \in M$ (5.2d)

$x_{ijm1} \ge y_{ijmts} - y_{ijm(t-1)s} \qquad \forall i \in I,\ \forall j \in J(i),\ \forall m \in M,\ \forall t = 2,...,TP$ (5.2e)

$\sum_{m\in M}\sum_{t=1}^{TP} y_{ijmts} - w_{ijs}^- + w_{ijs}^+ = d_{ijs}\sum_{m\in M}\sum_{t=1}^{TP} x_{ijmt} \qquad \forall i \in I,\ \forall j \in J(i)$ (5.2f)

$y_{ijmts} \in \{0,1\} \qquad \forall i \in I,\ \forall j \in J(i),\ \forall m \in M,\ \forall t = 1,...,TP$ (5.2g)

$w_{mts}^M \ge 0 \qquad \forall m \in M,\ \forall t = 1,...,TP$ (5.2h)

$w_{its}^I \ge 0 \qquad \forall i \in I,\ \forall t = 1,...,TP$ (5.2i)

$w_{ijs}^-,\ w_{ijs}^+ \ge 0 \qquad \forall i \in I,\ \forall j \in J(i)$ (5.2j)

Note that w_{mts}^M, $\forall m \in M$, $\forall t = 1,...,TP$, $\forall s \in S$ and w_{its}^I, $\forall i \in I$, $\forall t = 1,...,TP$, $\forall s \in S$ are equal to $\text{Max}\left\{0, \sum_{i\in I}\sum_{j\in J(i)} y_{ijmts} - 1\right\}$ and $\text{Max}\left\{0, \sum_{j\in J(i)}\sum_{m\in M} y_{ijmts} - 1\right\}$, respectively. In other words, w_{mts}^M and w_{its}^I become positive once multiple operations are assigned to an operating room and a

surgeon, respectively, at a time, in a given scenario. Note that w_{ijs}^{-} and w_{ijs}^{+}, $\forall i \in I$, $\forall j \in J(i)$, $\forall s \in S$ are equal to $\text{Max}\left\{0, \sum_{m\in M}\sum_{t=1}^{TP} y_{ijmts} - d_{ijs}\sum_{m\in M}\sum_{t=1}^{TP} x_{ijmt}\right\}$ and $\text{Max}\left\{d_{ijs}\sum_{m\in M}\sum_{t=1}^{TP} x_{ijmt} - \sum_{m\in M}\sum_{t=1}^{TP} y_{ijmts}, 0\right\}$, respectively. When $\sum_{m\in M}\sum_{t=1}^{TP} y_{ijmts} > d_{ijs}\sum_{m\in M}\sum_{t=1}^{TP} x_{ijmt}$, i.e., the operation times assigned to a surgeon *i* to perform operation *j* under scenario *s* is greater than d_{ijs}, then w_{ijs}^{-} becomes positive, and when $\sum_{m\in M}\sum_{t=1}^{TP} y_{ijmts} < d_{ijs}\sum_{m\in M}\sum_{t=1}^{TP} x_{ijmt}$, i.e., the operation times assigned to surgeon *i* to perform operation *j* under scenario *s* is less than d_{ijs}, then w_{ijs}^{+} becomes positive. However, in view of the objective function, we will never over-assign operation times to any (*i*, *j*, *s*). Therefore, we can remove w_{ijs}^{-}, and the objective function (5.2a) and the constraint set (5.2f) are reduced to

$$-\sum_{i\in I}\sum_{j\in J(i)}\left[\sum_{m\in M}\sum_{t=T+1}^{TP}\left(c_i(t-T)y_{ijmts}\right)\right] - \sum_{t=1}^{TP}\left(e^M\sum_{m\in M} w_{mts}^M + e^I\sum_{i\in I} w_{its}^I\right) - e^D\sum_{i\in I} w_{ijs}^{+}, \qquad (5.2a')$$

$$\sum_{m\in M}\sum_{t=1}^{TP} y_{ijmts} + w_{ijs}^{+} = d_{ijs}\sum_{m\in M}\sum_{t=1}^{TP} x_{ijmt} \qquad \forall i \in I,\ \forall j \in J(i). \qquad (5.2f')$$

Each scenario is assumed to follow a discrete probability density function. Then, $\text{Exp}[\tilde{f}(\vec{x})]$ can be expressed as follows.

$$\text{Exp}\left[\tilde{f}(\vec{x})\right] = \sum_{s\in S} p_s f_s(\vec{x}) = \begin{cases} -\sum_{i\in I}\sum_{j\in J(i)}\left[\sum_{m\in M}\sum_{t=T+1}^{TP}\left(c_i(t-T)\sum_{s\in S} p_s y_{ijmts}\right)\right] \\ -\sum_{s\in S} p_s\left[\sum_{t=1}^{TP}\left(e^M\sum_{m\in M} w_{mts}^M + e^I\sum_{i\in I} w_{its}^I\right) + e^D\sum_{i\in I}\sum_{j\in J(i)} w_{ijs}^{+}\right] \end{cases} \qquad (5.2h)$$

where y_{ijmts} satisfies constraint sets (5.2b) through (5.2g) in each corresponding recourse model for scenario s.

Considering all of these adjustments, the formulation **SHSSP2 (General)** is as follows:

SHSSP2 (General)

Maximize
$$\sum_{i\in I}\sum_{j\in J(i)}\left(\sum_{m\in M}\sum_{t=1}^{TP} r_j x_{ijmt}\right)-\sum_{i\in I}\sum_{j\in J(i)}\left[\sum_{m\in M}\sum_{t=T+1}^{TP}\left(c_i(t-T)\sum_{s\in S} p_s y_{ijmts}\right)\right]$$
$$-\sum_{s\in S} p_s\left[\sum_{t=1}^{TP}\left(e^M\sum_{m\in M} w^M_{mts}+e^I\sum_{i\in I} w^I_{its}\right)+e^D\sum_{i\in I}\sum_{j\in J(i)} w^+_{ijs}\right] \quad (5.3a)$$

Subject to
$$\sum_{i\in I(j)}\sum_{m\in M}\sum_{t=1}^{TP} x_{ijmt}\le 1 \qquad \forall j\in J \quad (5.3b)$$

$$\sum_{i\in I}\sum_{j\in J(i)} y_{ijmts}-w^M_{mts}\le 1 \qquad \forall m\in M,\ \forall t=1,...,TP,\ \forall s\in S \quad (5.3c)$$

$$\sum_{j\in J(i)}\sum_{m\in M} y_{ijmts}-w^I_{its}\le 1 \qquad \forall i\in I,\ \forall t=1,...,TP,\ \forall s\in S \quad (5.3d)$$

$$y_{ijm1s}=x_{ijm1} \qquad \forall i\in I,\ \forall j\in J(i),\ \forall m\in M,\ \forall s\in S \quad (5.3e)$$

$$x_{ijmt}\ge y_{ijmts}-y_{ijm(t-1)s} \quad \forall i\in I,\forall j\in J(i),\ \forall m\in M,\ \forall t=2,...,TP,\ \forall s\in S \quad (5.3f)$$

$$\sum_{m\in M}\sum_{t=1}^{TP} y_{ijmts}+w^+_{ijs}=d_{ijs}\sum_{m\in M}\sum_{t=1}^{TP} x_{ijmt} \qquad \forall i\in I,\ \forall j\in J(i),\forall s\in S \quad (5.3g)$$

$$x_{ijmt}\in\{0,1\} \qquad \forall i\in I,\ \forall j\in J(i),\ \forall m\in M,\ \forall t=1,...,TP \quad (5.3h)$$

$$y_{ijmts}\in\{0,1\} \qquad \forall i\in I,\ \forall j\in J(i),\ \forall m\in M,\ \forall t=1,...,TP,\ \forall s\in S \quad (5.3i)$$

$$w^M_{mts}\ge 0 \qquad \forall m\in M,\ \forall t=1,...,TP,\ \forall s\in S \quad (5.3j)$$

$$w^I_{its}\ge 0 \qquad \forall i\in I,\ \forall t=1,...,TP,\ \forall s\in S \quad (5.3k)$$

$$w^+_{ijs}\ge 0 \qquad \forall i\in I,\ \forall j\in J(i),\ \forall s\in S \quad (5.3l)$$

Like the formulation **SGAP2**, x_{ijmt} and y_{ijmts} in **SHSSP2 (General)** are the first and second stage variables. However, deciding y_{ijmts} is not the only decision to be made later. The values of the w^M_{mts}, w^I_{its}, and w^+_{ijs} variables are also determined later based on the values of x_{ijmt}, $\forall i\in I$, $\forall j\in J(i)$, $\forall m\in M$, $\forall t=1,...,TP$.

5.3. Classification of Recourse Model for SHSSP1 (General)

The recourse model of **SHSSP1 (General)**, designated as **Recourse Model for SHSSP1 (General)**, is a two-stage recourse model because all of the variables are divided into the first stage and the second stage variables. It is also a random recourse model since the coefficients of the second stage variables in the constraint sets change randomly with scenarios.

Proposition 5.1. The formulation **Recourse Model for SHSSP1 (General)** is not a complete recourse model but is a relatively complete recourse model.

Proof. To be a complete recourse model, **Recourse Model for SHSSP1 (General)** should be able to maintain feasibility for any solution of the first stage variables, x_{ijmt}, $\forall i \in I$, $\forall j \in J(i)$, $\forall m \in M$, $\forall t = 1,...,TP$. However, any solution violating constraint set (5.1b), i.e., any schedule for which an operation starts more than once, violates constraint sets (5.2d) and (5.2e). That solution makes the recourse model infeasible. Therefore, it is not a complete recourse model. On the other hand, as long as the solution of x_{ijmt} satisfies (5.3b), w^M_{mts}, w^I_{its}, and w^+_{ijs} will continue to maintain feasibility with respect to assignment of operations to an operating room and a surgeon and under-assigned operation times to a pair of a surgeon and an operation. In other words, all the constraints in the recourse model will be satisfied. Therefore, the recourse model is a relatively complete recourse model. □

5.4. A Special Case of SHSSP (General)

The parameters, e^M, e^I, and e^D represent penalty costs for assigning multiple operations and surgeons to an operating room during a timeslot and for under-assigned operation times, respectively, and therefore, different values of e^M, e^I, and e^D can give rise to different types of the **SHSSP2 (General)** problem. If the values of e^M, e^I, and e^D are very large, then the overlapping of multiple operations as well as of surgeons during a timeslot and under-assignment of operation times will not be permitted. This would lead to a conservative schedule in which the

patients will have enough time for their operations, and it would provide the least chance for the delay of operations. However, if the penalty costs are not large enough, then the situation would amount to achieving a tradeoff between the costs incurred and the revenue earned because of overlapping operations, surgeon assignments, and under-assigned operation times. In this section, we consider the first case where the penalty costs are assumed to be so large that a schedule with overlapping operations and under-assigned operation times is not allowed.

5.4.1. Formulation for a Special Case of HSSP (General)

We designate the model with no overlapping schedule as SHSSP (Special), and implement the formulation for the model as follows:

SHSSP1 (Special)

Maximize $\sum_{i\in I}\sum_{j\in J(i)}\sum_{m\in M}\sum_{t=1}^{TP} r_j x_{ijmt} + \mathrm{Exp}[\tilde{f}(\vec{x})]$ (5.4a)

Subject to $\sum_{i\in I(j)}\sum_{m\in M}\sum_{t=1}^{TP} x_{ijmt} \le 1 \qquad \forall j \in J$ (5.4b)

$x_{ijmt} \in \{0,1\} \qquad \forall i \in I,\ \forall j \in J(i),\ \forall m \in M,\ \forall t = 1,...,TP$ (5.4c)

The recourse model for each scenario s $\forall s \in S$, $f_s(\vec{x})$, is given by :

Recourse Model for SHSSP1 (Special)

Maximize $-\sum_{i\in I}\sum_{j\in J(i)}\left[\sum_{m\in M}\sum_{t=T+1}^{TP}\left(c_i(t-T)y_{ijmts}\right)\right]$

(5.5a)

Subject to $\sum_{i\in I}\sum_{j\in J(i)} y_{ijmts} \le 1 \qquad \forall m \in M,\ \forall t = 1,...,TP$ (5.5b)

$\sum_{j\in J(i)}\sum_{m\in M} y_{ijmts} \le 1 \qquad \forall i \in I,\ \forall t = 1,...,TP$ (5.5c)

$$y_{ijm1s} = x_{ijm1} \qquad \forall i \in I,\ \forall j \in J(i),\ \forall m \in M \tag{5.5d}$$

$$x_{ijm1} \geq y_{ijmts} - y_{ijm(t-1)s} \qquad \forall i \in I,\ \forall j \in J(i),\ \forall m \in M,\ \forall t = 2,...,TP \tag{5.5e}$$

$$\sum_{m \in M} \sum_{t=1}^{TP} y_{ijmts} = d_{ijs} \sum_{m \in M} \sum_{t=1}^{TP} x_{ijmt} \quad \forall i \in I,\ \forall j \in J(i) \tag{5.5f}$$

$$y_{ijmts} \in \{0,1\} \qquad \forall i \in I,\ \forall j \in J(i),\ \forall m \in M,\ \forall t = 1,...,TP \tag{5.5g}$$

Note that the above formulation does not contain the penalty parameters (e^M , e^I and e^D) or the continuous variables (w^M_{mts} , w^I_{its} and w^+_{ijs}) shown in the formulation **SHSSP2 (General)**.

Each scenario is assumed to follow a discrete probability distribution function. Then $\mathrm{Exp}[\tilde{f}(\vec{x})]$ can be calculated as follows.

$$\sum_{s \in S} p_s f_s(\vec{x}) = -\sum_{i \in I} \sum_{j \in J(i)} \left[\sum_{m \in M} \sum_{t=T+1}^{TP} \left(c_i (t-T) \sum_{s \in S} p_s y_{ijmts} \right) \right], \tag{5.5h}$$

where y_{ijmts} satisfies constraint sets (5.5b) through (5.5g) in each corresponding recourse model for scenario s .

The formulation **SHSSP2 (Special)** is as follows:

SHSSP2 (Special)

$$\text{Maximize} \quad \sum_{i \in I} \sum_{j \in J(i)} \left(\sum_{m \in M} \sum_{t=1}^{TP} r_j x_{ijmt} \right) - \sum_{i \in I} \sum_{j \in J(i)} \left[\sum_{m \in M} \sum_{t=T+1}^{TP} \left(c_i (t-T) \sum_{s \in S} p_s y_{ijmts} \right) \right] \tag{5.6a}$$

$$\text{Subject to} \quad \sum_{i \in I(j)} \sum_{m \in M} \sum_{t=1}^{TP} x_{ijmt} \leq 1 \qquad \forall j \in J \tag{5.6b}$$

$$\sum_{i \in I} \sum_{j \in J(i)} y_{ijmts} \leq 1 \qquad \forall m \in M,\ \forall t = 1,...,TP,\ \forall s \in S \tag{5.6c}$$

$$\sum_{j \in J(i)} \sum_{m \in M} y_{ijmts} \leq 1 \qquad \forall i \in I,\ \forall t = 1,...,TP,\ \forall s \in S \tag{5.6d}$$

$$y_{ijm1s} = x_{ijm1} \qquad \forall i \in I,\ \forall j \in J(i),\ \forall m \in M,\ \forall s \in S \tag{5.6e}$$

$$x_{ijm1} \geq y_{ijmts} - y_{ijm(t-1)s} \qquad \forall i \in I,\ \forall j \in J(i),\ \forall m \in M,\ \forall t = 2,...,TP,\ \forall s \in S \tag{5.6f}$$

$$\sum_{m\in M}\sum_{t=1}^{TP} y_{ijmts} = d_{ijs}\sum_{m\in M}\sum_{t=1}^{TP} x_{ijmt} \qquad \forall i \in I,\ \forall j \in J(i), \forall s \in S \tag{5.6g}$$

$$x_{ijmt} \in \{0,1\} \qquad \forall i \in I,\ \forall j \in J(i),\ \forall m \in M,\ \forall t = 1,...,TP \tag{5.6h}$$

$$y_{ijmts} \in \{0,1\} \qquad \forall i \in I,\ \forall j \in J(i),\ \forall m \in M,\ \forall t = 1,...,TP,\ \forall s \in S \tag{5.6i}$$

5.4.2. Classification of Recourse Model for SHSSP1 (Special)

Recourse Model for SHSSP1 (Special), which is the recourse model of **SHSSP1 (Special)**, is a two-stage recourse model because the variables are divided into the first stage and the second stage variables. It is a random recourse model since the coefficients of the second stage variables in the constraint sets change randomly with scenarios.

Remark 5.1. Unlike the formulation **Recourse Model for SHSSP1 (General)**, **Recourse Model for SHSSP2 (Special)** is neither a complete recourse model nor a relatively complete recourse model because (5.6c), (5.6d), and (5.6g) are strictly enforced and do not contain w_{mts}^{M}, w_{its}^{I} and w_{ijs}^{+} variables.

5.4.3. An Example of the SHSSP (Special)

Suppose that we have one Type I surgeon and two Type II surgeons. Two operations are to be performed by Type I surgeon and four operations by Type II surgeons. Figures 5.1, 5.2, and 5.3 depict the schedules of these Type I and II surgeons. In Figure 5.1, Surgeon 1 performs operation 1 in operating rooms 1 during periods 1 through 4 and operation 2 in operating room 2 during periods 5 through 7. Figure 5.2 shows the values of the first stage variables. Surgeon 3 starts operations 3 and 5 in operating room 3 during periods 7 and 1, respectively. Surgeon 4 starts operation 4 in operating room 1 during period 5 and starts operation 6 in operating room 2 during period 1.

Figure 5.1. Schedule of Type I Surgeons' Operations

		Period (TP=12)											
		1	2	3	4	5	6	7	8	9	10	11	12
Operating Room	1	(1,1)											
	2					(1,2)							
	3												

Figure 5.2. Schedule of Type I and Type II Surgeons' Operations

		Period (TP=12)											
		1	2	3	4	5	6	7	8	9	10	11	12
Operation Room	1	(1,1)				(3,4)							
	2	(3,6)				(1,2)							
	3	(2,5)						(2,3)					

Figure 5.3. Full Schedule Considering Stochastic Operation Times

		Period (TP=12)											
		1	2	3	4	5	6	7	8	9	10	11	12
Operating Room	1	(1,1)				(3,4,1), (3,4,3)							
						(3,4,2), (3,4,4)							
	2	(3,6,1), (3,6,3)				(1,2)							
		(3,6,2), (3,6,4)											
	3	(2,5,1), (2,5,2)						(2,3,1), (2,3,2)					
		(2,5,3), (2,5,4)						(2,3,3), (2,3,4)					

Finally, Figure 5.3 shows the complete schedule covering all the scenarios that might occur. The set (i, j, s) in the cells of Figure 5.3 stands for surgeon i conducting operation j under scenario s. For example, Type II surgeon 3 starts operation 4 in operating room 1 during period 5. Note that operation 4 finishes in period 7 under scenarios 1 and 3, but it does so in period 10 under scenarios 2 and 4. Figure 5.2 provides the information on surgeons and operations scheduled on a given day and the staring time of each operation, while Figure 5.3 provides detailed information on all the possibilities that may be encountered on that day.

Figure 5.3 also provides information for tactical planning. Suppose an emergency operation requires a surgeon in period 8. Let p_s be the probability that scenario s occurs, and $p(i)$ and $1-p(i)$ are the probabilities that surgeon i will or will not be available during period 8, respectively. If all 4 scenarios have the same probability of occurrence, i.e., $p_1 = p_2 = p_3 = p_4 = 0.25$, then the probability that either surgeon 2 or surgeon 3 is available during period 8 is 0.75 as follows:

$$p(2)(1-p(3)) + (1-p(2))p(3) + p(2)p(3) = 0.5*0.5 + 0.5*0.5 + 0.5*0.5 = 0.75$$

If both Type II surgeons can perform that emergency operation, then the hospital can accept the patient with 0.75 probability of having either of the surgeons being available.

5.4.4. Another Formulation for the SHSSP (Special)

5.4.4.1. Model Description

Note that the first stage variables are only subject to the second stage variables that are involved with the longest operation times for each assignment in Figure 5.3. Assume that, in addition to the assignments shown in Figure 5.3, we need to pair Surgeon 2 with operation 7. It requires 1 period under scenarios 1 and 2, and 2 periods under scenarios 3 and 4. Suppose we start operation 7 in operating room 3 during period 8 under scenarios 1 and 2, giving us $x_{2738} = y_{27381} = y_{27382} = 1$. By constraint sets (5.6e) and (5.6f), we also have: $y_{27383} = y_{27384} = y_{27393} = y_{27394} = 1$ since it takes two periods under scenario 3 and 4. However, as

Surgeon 2 still works on operation 3 during period 8 under scenarios 3 and 4, this will conflict with $y_{23383} = y_{23384} = 1$. These equations violate constraint set (5.6c) since $y_{23383} + y_{27383} = 2$ and $y_{23384} + y_{27384} = 2$, i.e., operating room 3 has both operations 3 and 4 during period 8 under scenario 3 and 4. Furthermore, the equations also violate (5.6d) because Surgeon 2 has to perform two operations 3 and 4 at the same time. Similarly, in Figure 5.3, operating rooms 1, 2, and 3 cannot have another operation until periods 11, 8, and 9, respectively.

Observation 5.1. In case, no potential overlap of operations is sought, in a room or a time slot, then it is necessary to consider scenarios with largest operation times for each assignment of a surgeon to a room to perform an operation.

To enforce this observation, we use a formulation that relies only on those second stage variables that are associated with the longest operation times for each assignment. We designate this model as **SHSSP (Longest)**. Let

$\bar{S}$: Set of scenarios containing the longest operation times for each assignment of a surgeon to an operation

$\bar{s}_{ij}$: Element of $\bar{S}$ for a pair of surgeon *i* and operation *j*, $\forall i \in I$, $\forall j \in J$

SHSSP (Longest)

Maximize $$\sum_{i \in I} \sum_{j \in J(i)} \left[\sum_{m \in M} \sum_{t=1}^{TP} r_j x_{ijmt} \right] - \sum_{i \in I} \sum_{j \in J(i)} \left[\sum_{m \in M} \sum_{t=T+1}^{TP} \left(c_i (t - T) \sum_{\bar{s} \in \bar{S}} p_{\bar{s}} y_{ijmt\bar{s}} \right) \right] \quad (5.7a)$$

Subject to $$\sum_{i \in I(j)} \sum_{m \in M} \sum_{t=1}^{TP} x_{ijmt} \le 1 \qquad \forall j \in J \quad (5.7b)$$

$$\sum_{i \in I} \sum_{j \in J(i)} y_{ijmt\bar{s}} \le 1 \qquad \forall m,\ \forall t,\ \forall \bar{s} \in \bar{S} \quad (5.7c)$$

$$\sum_{j \in J(i)} \sum_{m \in M} y_{ijmt\bar{s}} \le 1 \qquad \forall i \in I, \forall t, \forall \bar{s} \in \bar{S} \quad (5.7d)$$

$$x_{ijm1} = y_{ijm1\bar{s}} \qquad \forall i \in I, \forall j \in J(i), \forall m, \forall \bar{s} \in \bar{S} \quad (5.7e)$$

$$x_{ijmt} \ge y_{ijmt\bar{s}} - y_{ijm(t-1)\bar{s}} \qquad \forall i \in I, \forall j \in J(i), \forall m, \forall t \ge 2, \forall \bar{s} \in \bar{S} \quad (5.7f)$$

$$\sum_{m\in M}\sum_{t=1}^{TP} y_{ijmt\bar{s}} = d_{ij\bar{s}} \sum_{m\in M}\sum_{t=1}^{TP} x_{ijmt} \qquad \forall i \in I, \forall j \in J(i), \forall \bar{s} \in \bar{S} \qquad (5.7g)$$

$$x_{ijmt} \in \{0,1\} \qquad \forall i \in I, \forall j \in J(i), \forall m, \forall t \qquad (5.7h)$$

$$y_{ijmt\bar{s}} \in \{0,1\} \qquad \forall i \in I, \forall j \in J(i), \forall m, \forall t, \forall \bar{s} \in \bar{S} \qquad (5.7i)$$

The formulation above considers only the scenarios under which each assignment pairing a surgeon with an operation takes the longest operation times. We denote each corresponding scenario as $\bar{s}_{ij}$ for surgeon *i* and operation *j*. If there are ties for $\bar{s}_{ij}$, then they may be arbitrarily broken. Figure 5.4 illustrates $\bar{s}_{ij}$ for the example in Figure 5.3. We assume that the operation times do not change with operating rooms and periods, so $\bar{s}_{ij}$ is identical for the set $\{(i, j, m, t), \forall m, \forall t\}$.

Figure 5.4. $\bar{s}_{ij}$ for Example in Figure 5.3

Assignment	$\bar{s}_{ij}$
$(2,3,m,t)\ \ \forall m, \forall t$	either scenario 3 or scenario 4
$(2,5,m,t)\ \ \forall m, \forall t$	either scenario 3 or scenario 4
$(3,4,m,t)\ \ \forall m, \forall t$	either scenario 2 or scenario 4
$(3,6,m,t)\ \ \forall m, \forall t$	either scenario 2 or scenario 4

According to the first row in Figure 5.4, for the example presented in Figure 5.3, we can choose either scenario 3 or scenario 4 for the assignment of operation 3 to surgeon 2, regardless of m and t. Suppose that the pairs of (i, j) in Figure 5.4 are all the pairs in the problem. Then, either $\bar{S}_1 = \{2\,(\text{for } \bar{s}_{34} \text{ and } \bar{s}_{36}), 3\,(\text{for } \bar{s}_{23} \text{ and } \bar{s}_{25})\}$ or $\bar{S}_2 = \{4\,(\text{for } \bar{s}_{23}, \bar{s}_{25}, \bar{s}_{34}, \text{and } \bar{s}_{36})\}$ could be the candidates for $\bar{S}$. The number of scenarios in each case would be $|\bar{S}_1| = 2$ and $|\bar{S}_2| = 1$, in that order. Note that, $p_{\bar{s}}$ could be different from p_s because the scenarios considered in **SHSSP (Longest)** is a subset of the entire set of scenarios. Suppose that we choose only one set of scenarios, and thus, the value of $p_{\bar{s}}$ is equal to 1 because it is the only scenario. Then, the objective function (5.7a) is equivalent to

$$\sum_{i\in I}\sum_{j\in J(i)}\left[\sum_{m\in M}\sum_{t=1}^{TP} r_j x_{ijmt}\right]-\sum_{i\in I}\sum_{j\in J(i)}\left[\sum_{m\in M}\sum_{t=T+1}^{TP}\left(c_i(t-T)\sum_{\bar{s}\in\bar{S}} y_{ijmt\bar{s}}\right)\right] \qquad (5.7a')$$

Finally, $y_{ijmt\bar{s}}$ $\forall m$, $\forall t$ represent the second stage variables such that the operation length for a pair of (i, j) under $\bar{s}$ is at least as long as the operation length under the other scenarios, i.e., $\{\bar{s} : d_{ij\bar{s}} \geq d_{ijs}\ \forall s \in S\}$.

Proposition 5.2. A feasible solution of the first stage variables of the formulation **SHSSP2 (Special)** is feasible to the formulation **SHSSP (Longest)**, and vice versa, and the optimal objective function value of **SHSSP (Longest)** provides a lower bound for **SHSSP2 (Special)**.

Proof. First, we show that a feasible solution x_{ijmt} $\forall i \in I$, $\forall j \in J(i)$, $\forall m$, $\forall t$ to the formulation **SHSSP2 (Special)** is feasible to **SHSSP (Longest)**, and vice-versa. This is obviously true because the values of the first stage variables in both the formulations are determined by those of the second stage variables for the scenarios associated with the longest operation times for each assignment, $y_{ijmt\bar{s}}$ $\forall i \in I$, $\forall j \in J(i)$, $\forall m$, $\forall t$, $\forall \bar{s} \in \bar{S}$. Therefore, both formulations have the same feasible solutions of the first stage variables.

To prove the second part, if we show that the objective function value of **SHSSP (Longest)** is not more than that of **SHSSP2 (Special)** for the values of the first stage variables, we can say that the optimal objective function value of **SHSSP (Longest)** gives a lower bound for **SHSSP2 (Special)**. In the objective functions (5.6a) and (5.7a), the first and second terms are associated with the first and second stage variables, respectively. The values of the first stage variables are given, and hence, the value obtained from the first term is the same for both formulations. Consider the following expression for the second terms of both the formulations:

The second term in **SHSSP (Longest)** - The second term in **SHSSP2 (Special)**

$$=-\sum_{i\in I}\sum_{j\in J(i)}\left[\sum_{m\in M}\sum_{t=T+1}^{TP}\left(c_i(t-T)\sum_{\bar{s}\in\bar{S}} p_{\bar{s}} y_{ijmt\bar{s}}\right)\right]+\sum_{i\in I}\sum_{j\in J(i)}\left[\sum_{m\in M}\sum_{t=T+1}^{TP}\left(c_i(t-T)\sum_{s\in S} p_s y_{ijmts}\right)\right] \qquad (5.8a)$$

$$= -\sum_{i\in I}\sum_{j\in J(i)}\left[\sum_{m\in M}\sum_{t=T+1}^{TP}\left(c_i(t-T)\sum_{s'\in\bar{S}}\frac{p_{s'}}{\sum_{s'\in\bar{S}}p_{s'}}y_{ijmts'}\right)\right]+\sum_{i\in I}\sum_{j\in J(i)}\left[\sum_{m\in M}\sum_{t=T+1}^{TP}\left(c_i(t-T)\sum_{s\in S}p_s y_{ijmts}\right)\right] \quad (5.8b)$$

$$= -\sum_{i\in I}\sum_{j\in J(i)}\left[\sum_{m\in M}\sum_{t=T+1}^{TP}c_i(t-T)\left(\sum_{s'\in\bar{S}}\frac{p_{s'}}{\sum_{s'\in\bar{S}}p_{s'}}y_{ijmts'}-\sum_{s\in S}p_s y_{ijmts}\right)\right]\le 0 \quad (5.8c)$$

Therefore, the optimal objective function of **SHSSP (Longest)** is a lower bound of that of **SHSSP2 (Special)**. □

Consider the example in Figure 5.3 where scenario 2 and scenario 4 are the scenarios with the longest operation times for (*i*, *j*)=(3,4), i.e., $\bar{s}_{34}=2$ or $\bar{s}_{34}=4$. When $x_{34mt}=1$, neither y_{34mt1} nor y_{34mt3} can be one for the cell where either y_{34mt2} or y_{34mt4} is zero. On the other hand, they might be zero even if both y_{34mt2} and y_{34mt4} are one. The term in (5.8c), $\sum_{s'\in\bar{S}}\frac{p_{s'}}{\sum_{s'\in\bar{S}}p_{s'}}y_{ijmts'}-\sum_{s\in S}p_s y_{ijmts}$, is as follows:

$$\begin{aligned}&\sum_{s'\in\bar{S}}\frac{p_{s'}}{\sum_{s'\in\bar{S}}p_{s'}}y_{ijmts'}-\sum_{s\in S}p_s y_{ijmts}\\&=\frac{p_2}{p_2+p_4}y_{34mt2}+\frac{p_4}{p_2+p_4}y_{34mt4}-\left(p_1y_{34mt1}+p_2y_{34mt2}+p_3y_{34mt3}+p_4y_{34mt4}\right)\end{aligned} \quad (5.8d)$$

When $y_{34mt2}=y_{34mt4}=1$ and $y_{34mt1}=y_{34mt3}=0$, the R.H.S. of (5.8d) is equal to one. When all of them are zero or all of them are one, the R.H.S. of (5.8d) is equal to zero. As a result, the R.H.S. of (5.8d) becomes nonnegative. Therefore, the proposition above holds for the example in Figure 5.3.

By Proposition 5.2, the optimal objective function value of **SHSSP (Longest)** might be different from the "true" optimal objective function value. However, we can adjust it to obtain "true" optimal objective function value. The following formulation **HSSP (Equivalent)**

guarantees the same optimal objective function value as that of **SHSSP2 (Special)**. Consider the following:

HSSP (Equivalent)

$$\text{Maximize} \quad \sum_{i\in I}\sum_{j\in J(i)}\sum_{m\in M}\sum_{t=1}^{TP} r_j x_{ijmt} - \sum_{i\in I}\sum_{j\in J(i)}\sum_{s\in S} p_s \left(\sum_{m\in M}\sum_{t\geq T-d_{ijs}+2}^{TP} x_{ijmt} \right) \sum_{t'=T+1}^{t+d_{ijs}-1} \left(c_i(t'-T)\right) \tag{5.9}$$

Subject to Constraint sets (5.7b) ~ (5.7i) in **SHSSP (Longest)**

Proposition 5.3. Given the first stage variable values to be identical to those for the formulation **SHSSP2 (Special)**, the formulation **HSSP (Equivalent)** leads to the same objective function value as that for the formulation **SHSSP2 (Special)**.

Proof. In **SHSSP2 (Special)** (see (5.6a)), the second term in the objective function consists of the second stage variables. We can express the objective function as a function only of the first stage variables since the values of the second stage variables rely on the associated first stage variables. Therefore, if $x_{ijmt}=1$, we can determine the values of y_{ijmts}, $\forall s$, which are determined by the corresponding operation times, d_{ijs}. Only if $y_{ijmts}=1$, $\forall t=T+1,...,TP$, it is accounted for in the second term; otherwise, if $t\in\{1,...,T\}$, it does not matter whether its value is zero or one because the extra cost is encountered only during the overtime.

We can make the same argument with the first stage variables. Assume that $x_{ijmt}=1$ for some (i,j,m,t). Then, if $t+d_{ijs}-1\leq T$, there will be no penalty, that is, this assignment will not result in any extra cost. That is, $y_{ijm't's}=0$, $\forall m'\in M$, $T+1\leq t'\leq TP$, $\forall s\in S$. However, if $t+d_{ijs}-1\geq T+1$ and $t+d_{ijs}-1\leq TP$, i.e., $t\in\{T-d_{ijs}+2,...,TP-d_{ijs}+1\}$, the penalty will be incurred from period $T+1$ to period $t+d_{ijs}-1$. That is, $y_{ijmt's}=1$, $T+1\leq t'\leq t+d_{ijs}-1$, $\forall s\in S$. Hence, the objective function (5.6a) is equivalent to

$$\sum_{i\in I_2}\sum_{j\in J(i)}\sum_{m\in M}\sum_{t=1}^{TP} r_j x_{ijmt} - \sum_{i\in I_2}\sum_{j\in J(i)}\sum_{s\in S} p_s \left(\sum_{m\in M}\sum_{t=T-d_{ijs}+2}^{TP-d_{ijs}+1} x_{ijmt} \right) \sum_{t'=T+1}^{t+d_{ijs}-1} \left(c_i(t'-T)\right), \tag{5.10}$$

which is the second term in (5.9).

Therefore, the second term in the objective function of **HSSP (Equivalent)** is equivalent to that of **SHSSP2 (Special)**. □

Remark 5.2. Moreover, by Propositions 5.2 and 5.3, the optimal solution for both **SHSSP2 (Special)** and **HSSP (Equivalent)** will be the same.

5.4.4.2. Comparison of HSSP (Equivalent) and HSSP (General)

As it is easily seen, the formulation **HSSP (Equivalent)** is much more efficient than **SHSSP2 (Special)** in terms of the number of constraints and the number of variables, especially in the number of the second stage variables. The difference results from considering only one scenario for each assignment under which it takes the longest operation times. Table 5.1 compares the size of both the models, **SHSSP2 (Special)** and **HSSP (Equivalent)**, for the stochastic hospital staff scheduling problem. Most importantly, the number of the second stage variables and the number of constraints do not change with the number of scenarios for **HSSP (Equivalent)**. That

Table 5.1. Comparison of **SHSSP2 (Special)** and **HSSP (Equivalent)** in Size of Problems

	SHSSP2 (Special)	**HSSP (Equivalent)**
1st stage variables	$\sum_{i\in I_2} J(i)*M*TP$	$\sum_{i\in I_2} J(i)*M*TP$
2nd stage variables	$\sum_{i\in I_2} J(i)*M*TP*\|S\|$	$\sum_{i\in I_2} J(i)*M*TP$
Total # of variables	$\sum_{i\in I_2} J(i)*M*TP*(\|S\|+1)$	$2\sum_{i\in I_2} J(i)*M*TP$
Constraints	$\|J\|+\left(\begin{array}{l} M*TP+\|I\|*TP+ \\ \sum_{i\in I_2} J(i)*TP+\sum_{i\in I_2} J(i) \end{array}\right)*\|S\|$	$\begin{array}{l} \|J\|+M*TP+\|I\|*TP \\ +\sum_{i\in I_2} J(i)*TP+\sum_{i\in I_2} J(i)\| \end{array}$

is, even though we consider a stochastic situation, our model is of the same size as the deterministic model, **HSSP (General)**. As a result, the second formulation **HSSP (Equivalent)** has the same number of variables and constraints identical to those for **HSSP (General)**.

In view of the above discussion, we can conclude that we only need to consider an equivalent implementation of **SHSSP2 (Special)**, namely **HSSP (Equivalent)**, involving second stage variables that correspond to a pair of a surgeon and an operation with the longest operation times.

Remark 5.3. Note that **HSSP (Equivalent)** has been developed under the condition that the hospital does not permit any potential overlap among the operations performed in a room due to lack of available resources, where an operation cannot be performed if a previous operation scheduled in that room takes longer than expected. Also, it always assigns the exact operation times required for surgeon i to finish operating j under scenario s, $\forall i \in I$, $\forall j \in J$, $\forall s \in S$. Consequently, it is likely that such a schedule may end up with voids due to unused resources in case an operation in a room is completed earlier. However, in that case, the schedule can be left-shifted, if possible, as the scenarios are revealed. Moreover, such voids could be used to perform other procedures (emergency, for example) that are not considered within the scope of the SHSSP addressed. Besides, such a schedule can be used as a guideline to plan resources ahead of time.

5.5. SHSSP (General) with Expected Values

In this section, we modify the formulation **SHSSP2 (General)**, formulated in Section 5.2, into a linear program by considering the expected values of operation times (d_{ijs}). That is, the SHSSP (General) is solved using the expected values of operation times. First, we provide the underlying formulation of the SHSSP (General) for this case, and then, compare the solutions obtained by solving the SHSSP (General) as a stochastic program with those obtained by using expected values of operation times.

5.5.1. Formulation for the SHSSP (General) with Expected Values of Operation Times

Consider the following notation. These are now defined corresponding to the expected operation times.

$\hat{y}_{ijmt}$: Binary variables; =1 if surgeon *i* performs operation *j* at operating room *m* during period *t*; =0 otherwise, $\forall\, i \in I, \forall\, j \in J, \forall\, m \in M, t = 1, ..., TP$

$\hat{w}^{M}_{mt}$: Continuous variable representing the number of overlapping operations in operating room *m* during period *t*, $\forall\, m \in M, t = 1, ..., TP$

$\hat{w}^{I}_{it}$: Continuous variable representing the number of overlapping operations for surgeon *i* during period *t*, $\forall\, i \in I, t = 1, ..., TP$

$\hat{w}^{+}_{ij}$: Continuous variable to representing the under-assigned operation times for surgeon *i* to perform operation *j*, $\forall\, i \in I, \forall\, j \in J$

$\hat{d}_{ij}$: Expected value of operation times required by surgeon *i* for operation *j*, $\forall\, i \in I, \forall\, j \in J$

Note that $\hat{d}_{ij} = \sum_{s \in S} p_s d_{ijs}$ in the SGAP. Also, note that the profit and cost in the HSSP is calculated by the time required for a surgery measured in discrete units. For instance, it takes 2 hours and 30 minutes to perform an operation and the unit period is 1 hour, then it is recorded as either a 2- or 3-hour surgery. Assuming that a hospital always rounds up in calculating operation times, e.g., the hospital will record the surgery above as a 3-hour operation. Then, in the HSSP, $\hat{d}_{ij} = \left\lceil \sum_{s \in S} p_s d_{ijs} \right\rceil$, where $\lceil a \rceil$ is the smallest integer larger than or equal to a. We have the following formulation for the SHSSP with expected values of processing times, designated as **Expected-SHSSP2 (General)**.

Expected-SHSSP2 (General)

Maximize
$$\sum_{i\in I}\sum_{j\in J(i)}\left(\sum_{m\in M}\sum_{t=1}^{TP} r_j x_{ijmt}\right)-\sum_{i\in I}\sum_{j\in J(i)}\left[\sum_{m\in M}\sum_{t=T+1}^{TP}\left(c_i(t-T)\hat{y}_{ijmt}\right)\right] - \sum_{t=1}^{TP}\left(e^M\sum_{m\in M}\hat{w}_{mt}^M + e^I\sum_{i\in I}\hat{w}_{it}^I\right)+e^S\sum_{i\in I}\sum_{j\in J(i)}\hat{w}_{ij}^+ \quad (5.11a)$$

Subject to
$$\sum_{i\in I(j)}\sum_{m\in M}\sum_{t=1}^{TP} x_{ijmt}\le 1 \qquad \forall j\in J \quad (5.11b)$$

$$\sum_{i\in I}\sum_{j\in(i)}\hat{y}_{ijmt}-\hat{w}_{mt}^M\le 1 \qquad \forall m\in M,\ \forall t=1,...,TP \quad (5.11c)$$

$$\sum_{j\in J(i)}\sum_{m\in M}\hat{y}_{ijmt}-\hat{w}_{it}^I\le 1 \qquad \forall i\in I,\ \forall t=1,...,TP \quad (5.11d)$$

$$\hat{y}_{ijm1}=x_{ijm1} \qquad \forall i\in I,\ \forall j\in J(i),\ \forall m\in M \quad (5.11e)$$

$$x_{ijm1}\ge \hat{y}_{ijmt}-\hat{y}_{ijm(t-1)} \qquad \forall i\in I,\ \forall j\in J(i),\ \forall m\in M,\ \forall t=2,...,TP \quad (5.11f)$$

$$\sum_{m\in M}\sum_{t=1}^{TP}\hat{y}_{ijmt}+\hat{w}_{ij}^+=\hat{d}_{ij}\sum_{m\in M}\sum_{t=1}^{TP}x_{ijmt} \qquad \forall i\in I,\ \forall j\in J(i) \quad (5.11g)$$

$$x_{ijmt}\in\{0,1\} \qquad \forall i\in I,\ \forall j\in J(i),\ \forall m\in M,\ \forall t=1,...,TP \quad (5.11h)$$

$$\hat{y}_{ijmt}\in\{0,1\} \qquad \forall i\in I,\ \forall j\in J(i),\ \forall m\in M,\ \forall t=1,...,TP \quad (5.11i)$$

$$\hat{w}_{mt}^M\ge 0 \qquad \forall m\in M,\ \forall t=1,...,TP \quad (5.11j)$$

$$\hat{w}_{it}^I\ge 0 \qquad \forall i\in I,\ \forall t=1,...,TP \quad (5.11k)$$

$$\hat{w}_{ij}^+\ge 0 \qquad \forall i\in I,\ \forall j\in J(i) \quad (5.11l)$$

Note that the difference in the number of variables between the formulations **Expected SHSSP2 (General)** and **SHSSP2 (General)** and the difference in the number of constraints between them are the same, given by

$$\left(TP\left(|M|+|I|+|M|\sum_{i\in I}|J(i)|\right)+\sum_{i\in I}|J(i)|\right)(|S|-1). \quad (5.12)$$

5.5.2. Comparison between SHSSP2 (General) and Expected-SHSSP2 (General)

Note that the value of the first stage variables obtained from the solution of **Expected-SHSSP2 (General)** satisfies (5.11b), which is the same as (5.3b) in **SHSSP2 (General)**, and the recourse model for **SHSSP2 (General)** is a relatively complete recourse model where the feasible second stage variables can be found as long as the first stage variables satisfies (5.3b). Therefore, we can calculate the value of the second stage variables from the values of the first stage variables.

We solve both **SHSSP2 (General)** and **Expected-SHSSP2 (General)** using different values of $|I|$, $|J|$, $|M|$, TP, and $|S|$. The values used for these parameters are presented in Table 5.2. In Table 5.2, the 1st, 2nd, 3rd, 4th, and 5th columns represent the experiment number, the number of surgeons, the number of jobs, the number of operating rooms, and the number scenarios in **SHSSP2 (General)**, respectively. The 6th, 7th, and 8th columns represent the number of variables, the number of constraints, and the optimal objective function value for **SHSSP2 (General)**, respectively. The 9th, 10th, and 11th columns represent the number of variables, the number of constraints, and the optimal objective function value for **Expected-SHSSP2 (General)**, respectively.

Let x_{ijmt}^{E}, $i \in I$, $j \in J$, $m \in M$, $t = 1,\ldots,TP$, be the optimal solution of x_{ijmt}, $i \in I$, $j \in J$, $m \in M$, $t = 1,\ldots,TP$, for **Expected-SHSSP2 (General)**. Since the constraint sets (5.3b) and (5.11b) are the same, x_{ijmt}^{E}, $i \in I$, $j \in J$, $m \in M$, $t = 1,\ldots,TP$, satisfy (5.3b) as well. After correspondingly setting x_{ijmt} to the values of x_{ijmt}^{E}, $i \in I$, $j \in J$, $m \in M$, $t = 1,\ldots,TP$, and solving **SHSSP2 (General)**, we can obtain the objective function value of **SHSSP2 (General)**, and since the solution just obtained is also feasible to **SHSSP2 (General)**, it is a lower bound of the optimal objective function value of **SHSSP2 (General)**, which is presented in the last column, named as the "Lower Bound" column. Accordingly, the values in the last column represent the "true" objective function values obtained by implementing x_{ijmt}^{E}, $i \in I$, $j \in J$, $m \in M$, $t = 1,\ldots,TP$.

The cells highlighted in yellow under the "**SHSSP2 (General)**" column represent the experiments for which **Expected-SHSSP2 (General)** fails to obtain the "true" optimal solutions. This is encountered for 22 out of 35 experiments in total. Therefore, we cannot apply **Expected-SHSSP2** in order to obtain "true" optimal solutions.

Table 5.2. Comparison between **SHSSP2 (General)** and **Expected-SHSSP2 (General)**

Experiment #	$\|I\|$	$\|J\|$	$\|M\|$	**SHSSP2 (General)**				**Expected-SHSSP2 (General)**			
				$\|S\|$	Number of Variables	Number of Constraints	Objective Function Value	Number of Variables	Number of Constraints	Objective Function Value	Lower Bound
1	1	1	1	2	86	75	2000.00	49	38	2000.00	2000.00
2	1	1	1	2	86	75	2000.00	49	38	2000.00	2000.00
3	1	2	1	2	124	102	14000.00	74	52	14000.00	13801.52
4	1	2	1	2	124	102	14000.00	74	52	14000.00	14000.00
5	1	3	1	2	162	129	25225.45	99	66	25800.00	-50798.38
6	2	1	1	4	272	249	8000.00	86	63	8000.00	8000.00
7	2	2	1	4	400	354	19976.65	136	90	20000.00	13774.59
8	2	2	1	4	400	354	19715.00	136	90	19800.00	19715.00
9	2	2	1	4	400	354	19730.54	136	90	19800.00	19730.54
10	2	2	1	4	400	354	19761.55	136	90	19800.00	19761.55
11	2	3	1	4	528	459	37215.28	186	117	42800.00	-13178.64
12	2	3	1	4	528	459	36496.57	186	117	42800.00	-97096.21
13	2	4	1	4	656	564	39275.52	236	144	49400.00	-121966.15
14	2	4	1	4	656	564	38552.60	236	144	49400.00	-65583.73
15	2	5	1	4	784	669	38803.28	286	171	38800.00	38803.28
16	3	2	1	8	1,080	1,010	33705.30	198	128	34000.00	33705.30
17	3	2	1	8	1,080	1,010	33766.67	198	128	34000.00	33766.67
18	3	3	1	8	1,428	1,323	47044.94	273	168	55600.00	23354.20
19	3	4	1	8	1,776	1,636	52517.97	348	208	60800.00	-257820.55
20	3	5	1	8	2,124	1,949	53788.95	423	248	63400.00	-100380.22
21	3	7	1	8	2,820	2,575	44162.41	573	328	74000.00	-50440.98
22	4	2	1	16	2,312	2,210	46213.66	212	140	47400.00	-16728.82
23	2	2	2	4	688	594	20000.00	244	150	20000.00	19999.26
24	2	2	2	4	688	594	20000.00	244	150	20000.00	20000.00
25	2	3	2	4	936	834	43863.00	342	201	44000.00	-10801.67

Table 5.2. Comparison between **SHSSP2 (General)** and **Expected-SHSSP2 (General)** (Continued)

Experiment #	$\|I\|$	$\|J\|$	$\|M\|$	**SHSSP2 (General)**				**Expected-SHSSP2 (General)**			
				$\|S\|$	Number of Variables	Number of Constraints	Optimal Objective Function Value	Number of Variables	Number of Constraints	Objective Function Value	Lower Bound
26	2	4	2	4	1,184	1,074	57714.83	440	252	62800.00	-51440.87
27	2	4	2	4	1,184	1,074	57253.60	440	252	57600.00	-91528.21
28	2	5	2	4	1,432	1,314	79684.03	538	303	87000.00	-185312.76
29	4	2	2	16	3,728	3,554	48000.00	368	224	48000.00	48000.00
30	4	4	2	16	6,304	5,956	91091.55	664	376	92000.00	-125905.39
31	4	4	2	16	6,304	5,956	91802.38	664	376	92000.00	86492.25
32	6	4	2	64	26,400	25,348	143983.73	696	400	144000.00	140773.47
33	3	3	3	8	3,564	3,243	56404.37	729	408	56200.00	56404.37
34	3	6	3	8	6,552	5,910	157613.63	1,386	744	160200.00	-233485.71
35	4	4	4	16	11,584	10,948	91649.23	1,264	688	92000.00	24161.81
				Average Objective Function Value			45857.51	Average Objective Function Value			-23834.98

Furthermore, x_{ijmt}^{E}, $i \in I$, $j \in J$, $m \in M$, $t = 1,\ldots,TP$, yield a negative objective function value, which results from penalties incurred by either overlapping schedule or under-assigned operation times. The cells highlighted in gray under the "**Expected-SHSSP2 (General)**" column represent the experiments for which **Expected-SHSSP2 (General)** produces those x_{ijmt}^{E}, $i \in I$, $j \in J$, $m \in M$, $t = 1,\ldots,TP$. Accordingly, 15 experiments are the ones that end up with a negative lower bound. The last row presents the average of the optimal objective function values over all the experiments for **SHSSP2 (General)** and **Expected-SHSSP2 (General)**, which are 45857.51 and -23834.98, respectively. Therefore, there is a substantial chance that **Expected-SHSSP2 (General)** would end up with a schedule with negative revenue to the hospital.

A disadvantage for the use of the expected value solution that, even if it is feasible to **SHSSP2 (General)**, the expected value of operation times may never be realized. For instance, if an operation requires either 1 or 3 hours, its expected value (2 hours) will never be realized.

5.6. SHSSP2 (General) with the Monte Carlo Method (SHSSP2-MCM (General)).

5.6.1. Monte Carlo Method (MCM) for the SHSSP (General)

As addressed in Section 3.5.1, the MCM is known for its effectiveness for the solution of the stochastic programming problems (SPs) as it manages only a subset of a large number of scenarios. Compared to the SGAP, the situation for the SHSSP (General) is worse because of a faster increment in the size of the problem with an increment in the number of scenarios. For instance, one additional scenario in the formulation **SGAP2** leads to $|I|$ variables and $|I|$ constraints. However, doing so in **SHSSP2 (General)** leads to

$\left(\sum_{i\in I}|J(i)|*|M|*TP+|M|*TP+|I|*TP+\sum_{i\in I}|J(i)|\right)$ variables and

$\left(|M|*TP+|I|*TP+\sum_{i\in I}|J(i)|*|M|*TP+\sum_{i\in I}|J(i)|\right)$ constraints. Therefore, we need to pay a greater attention to deal with scenarios in **SHSSP2 (General)** since we may end up with an unsolvable problem very quickly. The number of scenarios for **SHSSP2 (General)** is

$$|S|=(\text{Number of stochastic instances})^{\text{Number of surgeons}}.$$

For example, if an operation needs either 1 or 3 hours, i.e., the number of stochastic instances is 2, and the number of surgeons is 3, then the number of scenarios is $8(=2^3)$. The numbers of variables and constraints in **SHSSP2 (General)** are, respectively,

$$\sum_{i\in I}|J(i)|*|M|*TP+|S|*\left(\sum_{i\in I}|J(i)|*|M|*TP+|M|*TP+|I|*TP+\sum_{i\in I}|J(i)|\right), \text{ and} \qquad (5.13a)$$

$$|J|+|S|*\left(\sum_{i\in I}|J(i)|*|M|*TP+|M|*TP+|I|*TP+\sum_{i\in I}|J(i)|\right). \quad (5.13b)$$

The problem sizes for different values of $|I|$, $|J|$, $|M|$, and $|S|$ are shown in Table 5.3. For the values depicted in the table, the problem size increases from (86 variables × 75 constraints) to (26,400 variables × 25,348 constraints).

Considering the dramatic changes in the size of **SHSSP2 (General)** with the increment in $|S|$ makes the use of the MCM more meaningful. To formulate the scheme for the MCM, let K and N denote the number of replications and the number of independent random scenarios in each replication, respectively. We can express the k^{th} replication of **SHSSP2 (General)** as follows.

k^{th} Replication of SHSSP2 (General)

Maximize
$$\sum_{i\in I}\sum_{j\in J(i)}\left(\sum_{m\in M}\sum_{t=1}^{TP} r_j x_{ijmt}\right)-\frac{1}{N}\sum_{i\in I}\sum_{j\in J(i)}\left[\sum_{m\in M}\sum_{t=T+1}^{TP}\left(c_i(t-T)\sum_{s\in S^k} y_{ijmts}\right)\right]$$
$$-\frac{1}{N}\sum_{s\in S^k}\left[\sum_{t=1}^{TP}\left(e^M\sum_{m\in M} w^M_{mts}+e^I\sum_{i\in I} w^I_{its}\right)+e^S\sum_{i\in I}\sum_{j\in J(i)} w^+_{ijs}\right] \quad (5.14a)$$

Subject to
$$\sum_{i\in I(j)}\sum_{m\in M}\sum_{t=1}^{TP} x_{ijmt}\le 1 \qquad \forall j\in J \quad (5.14b)$$

$$\sum_{i\in I}\sum_{j\in J(i)} y_{ijmts}-w^M_{mts}\le 1 \qquad \forall m\in M,\ \forall t=1,...,TP,\ \forall s\in S^k \quad (5.14c)$$

$$\sum_{j\in J(i)}\sum_{m\in M} y_{ijmts}-w^I_{its}\le 1 \qquad \forall i\in I,\ \forall t=1,...,TP,\ \forall s\in S^k \quad (5.14d)$$

$$y_{ijm1s}=x_{ijm1} \qquad \forall i\in I,\ \forall j\in J(i),\ \forall m\in M,\ \forall s\in S^k \quad (5.14e)$$

$$x_{ijmt}\ge y_{ijmts}-y_{ijm(t-1)s} \quad \forall i\in I,\forall j\in J(i),\ \forall m\in M,\ \forall t=2,...,TP,\ \forall s\in S^k \quad (5.14f)$$

$$\sum_{m\in M}\sum_{t=1}^{TP} y_{ijmts}+w^+_{ijs}=d_{ijs}\sum_{m\in M}\sum_{t=1}^{TP} x_{ijmt} \qquad \forall i\in I,\ \forall j\in J(i),\ \forall s\in S^k \quad (5.14g)$$

$$x_{ijmt}\in\{0,1\} \qquad \forall i\in I,\ \forall j\in J(i),\ \forall m\in M,\ \forall t=1,...,TP \quad (5.14h)$$

$$y_{ijmts}\in\{0,1\} \qquad \forall i\in I,\ \forall j\in J(i),\ \forall m\in M,\ \forall t=1,...,TP,\ \forall s\in S^k \quad (5.14i)$$

$$w^M_{mts}\ge 0 \qquad \forall m\in M,\ \forall t=1,...,TP,\ \forall s\in S^k \quad (5.14j)$$

Table 5.3. Problem Size of **SHSSP2 (General)** for Different $|I|$, $|J|$, $|M|$, and $|S|$ Values

$\|I\|$	$\|J\|$	$\|M\|$	$\|S\|$	Number of Variables	Number of Constraints	$\|I\|$	$\|J\|$	$\|M\|$	$\|S\|$	Number of Variables	Number of Constraints
1	1	1	2	86	75	3	2	1	8	1,080	1,010
1	3	1	2	162	129	3	3	1	8	1,428	1,323
2	1	1	4	272	249	3	3	3	8	3,564	3,243
2	2	1	4	400	354	3	3	4	8	4,560	4,132
2	2	2	4	688	594	3	3	5	8	5,556	5,021
2	3	1	4	528	459	3	3	6	8	6,552	5,910
2	3	2	4	936	834	3	4	1	8	1,776	1,636
2	3	3	4	2,496	2,283	3	4	4	8	8,880	8,452
2	3	4	4	3,168	2,884	3	4	5	8	10,764	10,229
2	3	5	4	3,840	3,485	3	4	6	8	12,648	12,006
2	3	6	4	4,512	4,086	3	5	1	8	2,124	1,949
2	3	7	4	5,184	4,687	3	5	5	8	21,372	20,837
2	3	8	4	5,856	5,288	3	6	3	8	6,552	5,910
2	4	1	4	656	564	3	7	1	8	2,820	2,575
2	4	2	4	1,184	1,074	4	2	1	16	2,312	2,210
2	4	4	4	6,240	5,956	4	2	2	16	3,728	3,554
2	4	5	4	7,512	7,157	4	4	2	16	6,304	5,956
2	5	1	4	784	669	4	4	4	16	11,584	10,948
2	5	2	4	1,432	1,314	6	4	2	64	26,400	25,348
2	5	5	4	15,048	14,693						

$$w_{its}^{I} \geq 0 \qquad \forall i \in I,\ \forall t = 1,...,TP,\ \forall s \in S^k \qquad (5.14k)$$

$$w_{ijs}^{+} \geq 0 \qquad \forall i \in I,\ \forall j \in J(i),\ \forall s \in S^k \qquad (5.14l)$$

where $S^k = \left\{s_1^k, s_2^k, ..., s_N^k\right\}$ is the set of scenarios randomly generated for the k[th] replication.

Proposition 5.4. The formulation ***k*[th] Replication of SHSSP2 (General)** guarantees a feasible solution to the original formulation **SHSSP2 (General)**.

Proof. Let $\left\{\left(x_{ijmt}^{*}\right)_N^k, \left(y_{ijmts}^{*}\right)_N^k, \left(w_{mts}^{M*}\right)_N^k, \left(w_{its}^{I*}\right)_N^k, \left(w_{ijs}^{+*}\right)_N^k\right\}$ designate an optimal (and feasible) solution to the formulation ***k*[th] Replication of SHSSP2 (General)**. Since (5.14b) and (5.3b) are identical, $\left(x_{ijmt}^{*}\right)_N^k$ satisfies (5.3b) $\forall s \in S$. Note that, by setting $x_{ijmt} = \left(x_{ijmt}^{*}\right)_N^k$, we make **SHSSP2 (General)** separable for each scenario *s* as follows.

<u>Separation of SHSSP2 (General) by Scenario *s*, *s*∈*S*</u>

Maximize
$$\sum_{i\in I}\sum_{j\in J(i)}\left(\sum_{m\in M}\sum_{t=1}^{TP} r_j\left(x_{ijmt}^{*}\right)_N^k\right) - p_s\sum_{i\in I}\sum_{j\in J(i)}\left[\sum_{m\in M}\sum_{t=T+1}^{TP}\left(c_i(t-T)y_{ijmts}\right)\right] - p_s\left[\sum_{t=1}^{TP}\left(e^M\sum_{m\in M} w_{mts}^{M} + e^I\sum_{i\in I} w_{its}^{I}\right) + e^S\sum_{i\in I}\sum_{j\in J(i)} w_{ijs}^{+}\right] \qquad (5.15a)$$

Subject to
$$\sum_{i\in I}\sum_{j\in J(i)} y_{ijmts} - w_{mts}^{M} \leq 1 \qquad \forall m \in M,\ \forall t = 1,...,TP \qquad (5.15b)$$

$$\sum_{j\in J(i)}\sum_{m\in M} y_{ijmts} - w_{its}^{I} \leq 1 \qquad \forall i \in I,\ \forall t = 1,...,TP \qquad (5.15c)$$

$$y_{ijm1s} = \left(x_{ijm1}^{*}\right)_N^k \qquad \forall i \in I,\ \forall j \in J(i),\ \forall m \in M \qquad (5.15d)$$

$$\left(x_{ijmt}^{*}\right)_N^k \geq y_{ijmts} - y_{ijm(t-1)s} \qquad \forall i \in I, \forall j \in J(i),\ \forall m \in M,\ \forall t = 2,...,TP \qquad (5.15e)$$

$$\sum_{m\in M}\sum_{t=1}^{TP} y_{ijmts} + w_{ijs}^{+} = d_{ijs}\sum_{m\in M}\sum_{t=1}^{TP}\left(x_{ijmt}^{*}\right)_N^k \qquad \forall i \in I,\ \forall j \in J(i) \qquad (5.15f)$$

$$y_{ijmts} \in \{0,1\} \qquad \forall i \in I,\ \forall j \in J(i),\ \forall m \in M,\ \forall t = 1,...,TP \qquad (5.15g)$$

$$w_{mts}^{M} \geq 0 \qquad \forall m \in M,\ \forall t = 1,...,TP \qquad (5.15h)$$

$$w_{its}^{I} \geq 0 \qquad \forall i \in I,\ \forall t = 1,...,TP \qquad (5.15i)$$

$$w_{ijs}^{+} \geq 0 \qquad \forall i \in I,\ \forall j \in J(i) \qquad (5.15j)$$

Note that (5.3b) and (5.3h) from **SHSSP2 (General)** are not included and the objective function (5.3a) and constraint sets (5.3e), (5.3f), and (5.3g) are changed as a result of fixing $x_{ijmt} = \left(x^*_{ijmt}\right)^k_N$. It is obvious that $\left\{\left(y^*_{ijmts}\right)^k_N, \left(w^{M*}_{mts}\right)^k_N, \left(w^{I*}_{its}\right)^k_N, \left(w^{+*}_{ijs}\right)^k_N\right\}$, which is obtained from **k^{th} Replication of SHSSP2 (General)**, satisfies the formulation above. Let, $\left\{\left(y^*_{ijmt\bar{s}}\right)^k_N, \left(w^{M*}_{mt\bar{s}}\right)^k_N, \left(w^{I*}_{it\bar{s}}\right)^k_N, \left(w^{+*}_{ij\bar{s}}\right)^k_N\right\}$, $\forall \bar{s} \in S \setminus S^m$, be the solution of the formulation above for each scenario $\bar{s}$, $\bar{s} \in S \setminus S^m$. Then, $\left\{\left(x^*_{ijmt}\right)^k_N, \left(y^*_{ijmts}\right)^k_N, \left(w^{M*}_{mts}\right)^k_N, \left(w^{I*}_{its}\right)^k_N, \left(w^{+*}_{ijs}\right)^k_N, \left(y^*_{ijmt\bar{s}}\right)^k_N, \left(w^{M*}_{mt\bar{s}}\right)^k_N, \left(w^{I*}_{it\bar{s}}\right)^k_N, \left(w^{+*}_{ij\bar{s}}\right)^k_N\right\}$ is a feasible solution to **SHSSP2 (General)**. □

5.6.2. Statistical Properties of the MCM for the SHSSP (General)

In order to build the MCM scheme for the SHSSP (General), we use the notation similar to that for the SGAP and defined in Section 3.5.2. Consider the following.

$\hat{X}^k_N$: Vector of x_{ijmt} that is the optimal solution to the formulation **k^{th} Replication of SHSSP2 (General)**

$\hat{Y}^k_N$: Vector of y_{ijmts} that is the optimal solution to the formulation **k^{th} Replication of SHSSP (General)**

$\left(\hat{W}^I\right)^k_N$: Vector of w^I_{its} that is the optimal solution to the formulation **k^{th} Replication of SHSSP2 (General)**

$\left(\hat{W}^M\right)^k_N$: Vector of w^M_{mts} that is the optimal solution to the formulation **k^{th} Replication of SHSSP2 (General)**

$\left(\hat{W}^+\right)^k_N$: Vector of w^+_{ijs} that is the optimal solution to the formulation **k^{th} Replication of SHSSP2 (General)**

$\left(\hat{x}_{ijmt}\right)^k_N$: An element of $\hat{X}^k_N$, $\forall i \in I$, $\forall j \in J$, $\forall m \in M$, $t = 1,...,TP$

$\left(\hat{y}_{ijmts}\right)_N^k$: An element of $\hat{Y}_N^k$, $\forall i \in I$, $\forall j \in J$, $\forall m \in M$, $t = 1,...,TP$, $\forall s \in S^k$

$\left(\hat{w}_{its}^I\right)_N^k$: An element of $\left(\hat{W}^I\right)_N^k$, $\forall i \in I$, $t = 1,...,TP$, $\forall s \in S^k$

$\left(\hat{w}_{mts}^M\right)_N^k$: An element of $\left(\hat{W}^M\right)_N^k$, $\forall m \in M$, $t = 1,...,TP$, $\forall s \in S^k$

$\left(\hat{w}_{ijs}^+\right)_N^k$: An element of $\left(\hat{W}^+\right)_N^k$, $\forall i \in I$, $\forall j \in J$, $\forall s \in S^k$

X^* : Vector of x_{ij} that is optimal to the formulation **SHSSP2 (General)**

Y^* : Vector of y_{is} that is optimal to the formulation **SHSSP2 (General)**

W^{I*} : Vector of w_{its}^I that is optimal to the formulation **SHSSP2 (General)**

W^{M*} : Vector of w_{mts}^M that is optimal to the formulation **SHSSP2 (General)**

W^{+*} : Vector of w_{ijs}^+ that is optimal to the formulation **SHSSP2 (General)**

h_N^k : Objective function value for ***k*th Replication of SHSSP2 (General)**

h : Objective function value for the formulation **SHSSP2 (General)**

$h_N^k(\hat{X}_N^k)$: Optimal objective function value for ***k*th Replication of SHSSP2 (General)**

$h\left(\hat{X}_N^k\right)$: "True" objective function value corresponding to $\hat{X}_N^k$ for **SHSSP2 (General)**

$h\left(X^*\right)$: "True" optimal objective function value for **SHSSP2 (General)**

$\bar{h}_N$: Average of $h_N^1\left(\hat{X}_N^1\right), h_N^2\left(\hat{X}_N^2\right), \cdots, h_N^K\left(\hat{X}_N^K\right)$ determined by (2.16d)

$\mathrm{V}\left(\bar{h}_N\right)$: Variance of $h_N^1\left(\hat{X}_N^1\right), h_N^2\left(\hat{X}_N^2\right), \cdots, h_N^K\left(\hat{X}_N^K\right)$ determined by (2.16e)

$\tilde{N}$: Number of scenarios used to determine an unbiased estimate of the objective function value once a solution is found using N scenarios, $\tilde{N} >> N$

$h_{\tilde{N}}\left(\hat{X}_N^k\right)$: Unbiased estimate of $h\left(\hat{X}_N^k\right)$ determined by (2.16g)

$\mathrm{V}\left(h_{\tilde{N}}\left(\hat{X}_N^k\right)\right)$: Variance of $h_{\tilde{n}}^k\left(\hat{X}_N^k\right)$, $\tilde{n} = 1,..., \tilde{N}$, determined by (2.16h)

The statistical properties and propositions addressed in the following sections are adapted from those for the formulation **SGAP2**, contained in Sections 3.5.2.1 through 3.5.2.9, to apply them to the formulation **SHSSP2 (General)**.

5.6.2.1. Optimal Objective Function Value of Each Replication

The optimal objective function value of the formulation ***k*th Replication of SHSSP2 (General)**, $h_N^k(\hat{X}_N^k)$, is as follows.

$$h_N^m\left(\hat{X}_N^m\right)=\sum_{i\in I}\sum_{j\in J(i)}\left(\sum_{m\in M}\sum_{t=1}^{TP} r_j\left(\hat{x}_{ijmt}\right)_N^k\right)-\frac{1}{N}\sum_{i\in I}\sum_{j\in J(i)}\left[\sum_{m\in M}\sum_{t=T+1}^{TP}\left(c_i(t-T)\sum_{s\in S^k}\left(\hat{y}_{ijmts}\right)_N^k\right)\right]$$
$$-\frac{1}{N}\sum_{s\in S^k}\left[\sum_{t=1}^{TP}\left(e^M\sum_{m\in M}\left(\hat{w}_{mts}^M\right)_N^k+e^I\sum_{i\in I}\left(\hat{w}_{its}^I\right)_N^k\right)+e^S\sum_{i\in I}\sum_{j\in J(i)}\left(\hat{w}_{ijs}^+\right)_N^k\right] \quad (5.16a)$$

Remark 5.4. (Refer to Proposition 3.5) While **SGAP2** is a minimization problem, **SHSSP2 (General)** is a maximization problem. As a result, the optimal objective function value of the formulation ***k*th Replication of SHSSP2 (General)** , $h_N^k(\hat{X}_N^k)$, provides an unbiased estimate of an upper bound of the "true" optimal objective function value of **SHSSP2 (General)** , $h\left(X^*\right)$.

Proof. Let X be the set of $\vec{x}$, the vector of x_{ijmt} satisfying constraints (5.3b) and (5.3h) in **SHSSP2 (General)**, and also, $N=|S^k|$. We have

$$h\left(X^*\right) = \operatorname*{Max}_{\vec{x}\in X}\left(\vec{r}\vec{x}+\operatorname{Exp}[\tilde{f}(\vec{x})]\right) \quad (5.16b)$$

$$= \operatorname*{Max}_{\vec{x}\in X}\left(\vec{r}\vec{x}+\operatorname{Exp}[\frac{1}{N}\sum_{s\in S^k} f_s(\vec{x})]\right) \quad (5.16c)$$

$$= \operatorname*{Max}_{\vec{x}\in X}\left(\vec{r}\vec{x}\right)+\operatorname*{Max}_{\vec{x}\in X}\left(\operatorname{Exp}[\frac{1}{N}\sum_{s\in S^k} f_s(\vec{x})]\right) \quad (5.16d)$$

$$\le \operatorname*{Max}_{\vec{x}\in X}\left(\vec{r}\vec{x}\right)+\operatorname{Exp}\left(\operatorname*{Max}_{\vec{x}\in X}[\frac{1}{N}\sum_{s\in S^k} f_s(\vec{x})]\right) \quad (5.16e)$$

$$= \text{Exp}\left[\underset{\vec{x}\in X}{\text{Max}}(\vec{r}\vec{x}) + \underset{\vec{x}\in X}{\text{Max}}[\frac{1}{N}\sum_{s\in S^k} f_s(\vec{x})] \right] \quad (5.16f)$$

$$= \text{Exp}\left[\underset{\vec{x}\in X}{\text{Max}}\left(\vec{r}\vec{x} + \frac{1}{N}\sum_{s\in S^k} f_s(\vec{x}) \right) \right] \quad (5.16g)$$

$$= \text{E}[h_N^k(\hat{X}_N^k)] \quad (5.16h)$$

Note that (5.16d) holds since the first term in the objective function is a constant, not a random variable, and we can separate its contribution from the objective function value, and (5.16e) holds because $\text{Max}[\text{Exp}(f_1(x)+\cdots+f_N(x))] \le \text{Exp}(\text{Max}[f_1(x)+\cdots+f_N(x)])$. □

5.6.2.2. Lower and Upper Bounds for the "True" Optimal Objective Function Value, Estimates of the "True" Optimal Solution and Optimality Gap, and Confidence Interval on Optimality Gap for the SHSSP2 (General)

In this section, we present propositions and remarks for **SHSSP2 (General)** and ***k*th Replication of SHSSP2 (General)** similar to those that have been provided for **SGAP2** in Section 3.5.2, especially addressing: (1) lower and upper bounds of the "true" optimal objective function value, (2) estimates of the "true" optimal solution and optimality gap, and (3) confidence interval on the optimality gap. We can easily prove them by exchanging "Min" with "Max" and properly changing the direction of inequalities in them. Please refer to Propositions 3.6, 3.7, 3.8, and 3.9 and Remarks 3.4, 3.5, 3.6, 3.7, 3.8, 3.9, and 3.10 for more details.

Remark 5.5. (Refer to Proposition 3.6) As N increases, $E[h_N^k(\hat{X}_N^k)]$, the expected value of the optimal objective function value of replications, approaches $h(X^*)$ from above.

Remark 5.6. (Refer to Proposition 3.7) As N approaches infinity, the following statements are true with probability one:

(1) The optimal objective function value of the formulation ***k*th Replication of SHSSP2 (General)**, $h_N^k(\hat{X}_N^k)$, converges to the "true" optimal objective function value of **SHSSP2 (General)**, $h(X^*)$.

(2) The objective function value of any solution, say X_N^k, to the formulation ***k*th Replication of SHSSP2 (General)** converges to the objective function value of **SHSSP2 (General)**, i.e., $h_N^k(X_N^k)$ converges to $h(X_N^k)$.

(3) The optimal solution to the formulation ***k*th Replication of SHSSP2 (General)**, $\hat{X}_N^k$, converges to the optimal solution to **SHSSP2 (General)**, X^*, if it is unique.

Remark 5.7. (Refer to Remark 3.4) By Remark 5.6, as $N \to \infty$, $h_N^k(\hat{X}_N^k)$, $\forall k = 1,...,K$, converge to $h(X^*)$. Therefore, regardless of the number of replications, $\bar{h}_N \to h(X^*)$ as $N \to \infty$.

Remark 5.8. (Refer to Proposition 3.8) As *K* approaches infinity, $\bar{h}_N$ provides an unbiased estimate of an upper bound of the "true" optimal objective function value of **SHSSP2 (General)**.

Remark 5.9. (Refer to Proposition 3.9) As $\tilde{N}$ approaches infinity, $h_N^k(\hat{X}_N^k)$ converges to $h(\hat{X}_N^k)$, the objective function value of $\hat{X}_N^k$ in **SHSSP2 (General)**, from above with probability one.

Remark 5.10. (Refer to Remark 3.5) As $N \to \infty$, $\hat{X}_N^k$, $\forall k = 1,...,K$, converge to X^* and $h_{\tilde{N}}^k(\hat{X}_N^k)$ converges to $h(X^*)$ since $\tilde{N} >> N$.

Similarly to (3.16b), we choose a candidate for the "true" optimal solution from the set of optimal solutions to each replication, as follows.

$$\left(\hat{X}_N\right)^* = \left[\left(\hat{X}_N\right)^* \in \left\{\hat{X}_N^k, k = 1,...,K\right\} : h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right) \geq h_{\tilde{N}}\left(\hat{X}_N^k\right), \forall k = 1,...,K\right] \tag{5.17}$$

Remark 5.11. (Refer to Remark 3.6) As $N \to \infty$, $\hat{X}_N^k$, $\forall k = 1,...,K$, converge to optimal solution. Therefore, as $N \to \infty$, $\left(\hat{X}_N\right)^* \to X^*$, and, accordingly, $h\left(\left(\hat{X}_N\right)^*\right) \to h\left(X^*\right)$.

Remark 5.12. (Refer to Remark 3.7) By Remarks 5.8 and 5.9, $\bar{h}_N - h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)$ is an unbiased estimate of the gap between lower and upper bounds. Furthermore, by Remarks 5.7 and 5.10, as $N \to \infty$, $h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right) - \bar{h}_N \to 0$.

The average of the objective function value of each replication (refer to (3.14a)) is given by

$$\bar{h}_N = \frac{1}{K}\sum_{k=1}^{K} h_N^k\left(\hat{X}_N^k\right). \tag{5.18a}$$

The variance of an unbiased estimate of a lower bound (refer to (3.16a)) is given by

$$\mathrm{V}\left[h_{\tilde{N}}^m\left(\hat{X}_N^m\right)\right] = \frac{1}{\tilde{N}(\tilde{N}-1)}\sum_{\tilde{n}=1}^{\tilde{N}}\left[\begin{array}{l} -\sum_{i\in I}\sum_{j\in J(i)}\sum_{m\in M}\sum_{t=T+1}^{TP}\left(c_i(t-T)\left(\hat{y}_{ijmt\tilde{s}_{\tilde{n}}^k}\right)_{\tilde{N}}^k\right) \\ -\sum_{t=1}^{TP}\left(e^M\sum_{m\in M}\left(\hat{w}^M_{mt\tilde{s}_{\tilde{n}}^k}\right)_{\tilde{N}}^k + e^I\sum_{i\in I}\left(\hat{w}^I_{it\tilde{s}_{\tilde{n}}^k}\right)_{\tilde{N}}^k\right) - e^S\sum_{i\in I}\sum_{j\in J(i)}\left(\hat{w}^+_{ij\tilde{s}_{\tilde{n}}^k}\right)_{\tilde{N}}^k + \\ \frac{1}{\tilde{N}}\sum_{s\in\tilde{S}^k}\left\{\begin{array}{l}\sum_{i\in I}\sum_{m\in M}\sum_{t=T+1}^{TP} c_i(t-T)\left(\hat{y}_{ijmt\tilde{s}_{\tilde{n}}^k}\right)_{\tilde{N}}^k + \\ \sum_{t=1}^{TP}\left(e^M\sum_{m\in M}\left(\hat{w}^M_{mt\tilde{s}_{\tilde{n}}^k}\right)_{\tilde{N}}^k + e^I\sum_{i\in I}\left(\hat{w}^I_{it\tilde{s}_{\tilde{n}}^k}\right)_{\tilde{N}}^k\right) + e^S\sum_{i\in I}\sum_{j\in J(i)}\left(\hat{w}^+_{ij\tilde{s}_{\tilde{n}}^k}\right)_{\tilde{N}}^k\end{array}\right\} \end{array}\right] \tag{5.18b}$$

An unbiased estimate of the variance of the optimality gap (refer to (3.18a) and (3.18b)) is

$$\mathrm{V}[\bar{h}_N - h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)] = \mathrm{V}[\bar{h}_N] + \mathrm{V}[h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)]. \tag{5.18c}$$

The 100(1-α) percent confidence interval (refer to (3.19b)) is given by

$$\left[0, \text{Max}\left[0, \bar{h}_N - h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)\right] + z_{1-\alpha}\sqrt{\text{V}[\bar{h}_N] + \text{V}[h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)]}\right]. \tag{5.18d}$$

The ratio of the optimality gap over the "true" optimal objective function value (refer to (3.20)) is given by

$$\text{E}\left[\frac{h - h\left(\left(\hat{X}_N\right)^*\right)}{h}\right] = \text{E}\left[1 - \frac{h\left(\left(\hat{X}_N\right)^*\right)}{h}\right] = 1 - \frac{\text{E}\left[h\left(\left(\hat{X}_N\right)^*\right)\right]}{\text{E}[h]} = 1 - \frac{h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)}{\bar{h}_N}. \tag{5.18e}$$

5.6.3. An Exterior Sampling Method for the SHSSP (General)

Like we have done for SGAP2, an exterior sampling method (ESM) is employed to build a subset of scenarios for each replication in the MCM for **SHSSP2 (General)**. It is a modification of the method suggested by Shapiro ([84] and [85]) and Verweij et al. [94], and it is presented next. Note that steps 10 and 11 are different from those presented in Section 3.5.3 for **SGAP2** because **SHSSP2 (General)** is a maximization problem.

Step 1. Specify the values of N, Ñ, and K.

Step 2. Generate $S^k = \{s_1^k, s_2^k, ..., s_N^k\}$ with a random number generator.

Step 3. Build the formulation ***k*th Replication of SHSSP2 (General)** with S^k.

Step 4. Solve the formulation ***k*th Replication of SHSSP2 (General)**.

Step 5. Store $\hat{X}_N^k$ and h_N^k.

Step 6. Generate $\tilde{S}^k = \{\tilde{s}_1^k, \tilde{s}_2^k, ..., \tilde{s}_{\tilde{N}}^k\}$.

Step 7. Calculate $h_{\tilde{N}}^k(\hat{X}_N^k)$ and $\text{V}\left(h_{\tilde{N}}^k(\hat{X}_N^k)\right)$.

Step 8. If k=K, go to step 9; otherwise, set k=k+1, and go to step 2.

Step 9. Estimate $\bar{h}_N$ and $\mathrm{V}(\bar{h}_N)$.

Step 10. Determine $(\hat{X}_N)^*$, and estimate $\mathrm{E}\left[h - h\left((\hat{X}_N)^*\right)\right]$ and $\mathrm{V}\left[\bar{h}_N - h_{\tilde{N}}\left((\hat{X}_N)^*\right)\right]$.

Step 11. If $1 - \dfrac{h_{\tilde{N}}\left((\hat{X}_N)^*\right)}{\bar{h}_N} \leq \varepsilon$, go to step 12; otherwise, increase the values of N, Ñ, and K, and go to step 2.

Step 12. Estimate $100(1-\alpha)$ percent confidence interval, and stop.

Through steps 1, 2, and 3, we prepare and build the formulation, and in step 4, we solve the formulation. Through the remaining steps, we store the solution and the objective function value, calculate the statistical properties discussed in Section 5.6.2, and evaluate the current solution with them.

5.7. Computational Experimentation

Like for the MCM applied to the SGAP, we split experiments into two groups. In the first group, we justify the use of the MCM for the SHSSP (General) by obtaining near-optimal solutions. In the second group, after having justified the use of the MCM for the SHSSP (General), we demonstrate the use of the MCM further by finding solutions that give a small gap between the unbiased estimates of lower and upper bounds on the "true" optimal objective function values. Moreover, we estimate 100(1-α) percent confidence interval of the objective function values obtained from these solutions. Therefore, for the experiments in the first group, we can measure the near-optimality of the solutions obtained by using the MCM, but for the experiments in the second group, we draw upon the confidence interval as a measure to evaluate the quality of the solutions obtained.

5.7.1. Experimental Data to Justify the Use of the MCM

Table 5.4 presents 24 experiments that we have conducted as the first group of experiments. We will compare an optimum-seeking method, which is the branch-and-cut-based method, with the Monte Carlo method (MCM) in terms of the cpu times required and near-optimality, which measures how close the solution obtained by the MCM is to the "true" optimal solution found by the optimum-seeking method. Table 5.4 also presents various problem sizes for different values of the number of operating rooms, the number of surgeons, the number of operations, and the number of scenarios. The last column contains the values of the product of the number of variables in the 6th column and the number of constraints in the 7th column. Compared to the 1st experiment, the 16th experiment consists of only one more operating room, two more surgeons, four more operations, and six more scenarios. However, the size of the problem in the last column increases from less than 10,000 to more than 1,000,000. Note that, the last experiment has only one more operating room and one more surgeon than for the 16th experiment; nonetheless, the size of the problem becomes more than 10,000,000. Therefore, it is clear that the problem size increases exponentially with increment in $|M|$, $|I|$, $|J|$, and $|S|$.

Along with the optimal objective function values obtained by the optimum-seeking algorithm, like we have done for the SGAP, we also need to keep track of the following parameters.

- Initial values of N, $\tilde{N}$, and K
- Final values of N, $\tilde{N}$, and K
- $\hat{X}_N^k$: The optimal solution to each replication, k=1,…,K
- h_N^k : The objective function value of each replication, k=1,…,K, where average value is used for the unbiased estimator of the lower bound of the "true" optimal objective function value, h
- $h_{\tilde{N}}^k\left(\hat{X}_N^k\right)$: The unbiased estimators of h for each replication, k=1,…,K

Table 5.4. Experimental Data for the First Group

Experiment #	$\|M\|$	$\|I\|$	$\|J\|$	$\|S\|$	Number of Variables	Number of Constraints	Number of Variables × Number of Constraints
1	1	1	1	2	86	75	6,450
2	1	1	2	2	124	102	12,648
3	1	1	3	2	162	129	20,898
4	1	2	2	4	400	354	141,600
5	1	2	3	4	528	459	242,352
6	1	2	4	4	656	564	369,984
7	1	2	5	4	784	669	524,496
8	2	2	2	4	688	594	408,672
9	2	2	3	4	936	795	744,120
10	2	2	4	4	1,184	996	1,179,264
11	2	2	5	4	1,432	1,197	1,714,104
12	2	2	6	4	1,680	1,398	2,348,640
13	2	2	7	4	1,928	1,599	3,082,872
14	2	3	3	8	2,496	2,283	5,698,368
15	2	3	4	8	3,168	2,884	9,136,512
16	2	3	5	8	3,840	3,485	13,382,400
17	2	3	6	8	4,512	4,086	18,436,032
18	2	3	7	8	5,184	4,687	24,297,408
19	2	4	4	16	6,240	5,956	37,165,440
20	3	3	3	8	3,564	3,243	11,558,052
21	3	3	4	8	4,560	4,132	18,841,920
22	3	3	5	8	5,556	5,021	27,896,676
23	3	4	4	16	8,880	8,452	75,053,760
24	3	4	5	16	10,764	10,229	110,104,956

- $h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)$: The best objective function value found from among all replications and unbiased estimator of the upper bound of *h*, where $\left(\hat{X}_N\right)^*$ is the solution such that

$$\left(\hat{X}_N\right)^* \equiv \left[\left(\hat{X}_N\right)^* \in \left\{\hat{X}_N^k, k=1,...,K\right\}: h_{\tilde{N}}\left(\hat{X}_N\right)^* \geq h_{\tilde{N}}^k\left(\hat{X}_N^k\right), \forall k=1,...,K\right]$$

- $\mathrm{E}\left[\dfrac{h-h\left(\left(\hat{X}_N\right)^*\right)}{h}\right]=1-\dfrac{h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)}{\bar{h}_N}$: The tolerance ratio, where $\bar{h}_N$ is the average of $h_N^1, h_N^2, \cdots, h_N^K$

Note that we accept the results of an iteration when $1-\dfrac{h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)}{\bar{h}_N} \leq \varepsilon$, where ε is a positive number. In this study, ε is set to 0.2. If the result of an iteration passes the stopping criterion, we evaluate the solution found after the iteration using the near-optimality criterion, $\dfrac{h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)}{h\left(X^*\right)}$, (recall that $h(X^*)$ is known for the experiments in the first group). Note that, we reject the result of an iteration if $h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)-\bar{h}_N > 0$, that is, we reject all results if we end up with an unbiased estimate of the lower bound of the "true" optimal objective function value that is greater than that of the upper bound. Therefore, the stopping criterion, $1-\dfrac{h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)}{\bar{h}_N}$, will always be nonnegative.

Also, note that $h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)$ is an unbiased estimate of the lower bound of the "true" optimal objective function value, but there is still a possibility that we may end up with a biased

subset of scenarios even after the stopping criterion is satisfied, and in that case, $\frac{h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)}{h\left(X^*\right)}$ can be greater than one.

5.7.2. Experimental Data for the Experiments in the Second Group

Table 5.5 presents the experiments for which optimal solutions are unknown. The size of the problems, shown in the last column of the table, increases from tens of millions to hundreds of millions. The initial value of (N, $\tilde{N}$, K) is (0.125|S|, 0.25|S|, 2). For example, (N, $\tilde{N}$, K) for

Table 5.5. Experiments for the Second Group

Experiment #	$\|M\|$	$\|I\|$	$\|J\|$	$\|S\|$	Number of Variables	Number of Constraints	Number of Variables × Number of Constraints
1	2	3	8	8	5,856	5,288	30,966,528
2	2	3	9	8	6,528	5,889	38,443,392
3	2	3	10	8	7,200	6,490	46,728,000
4	2	4	5	16	9,632	9,157	88,200,224
5	2	4	6	16	11,328	10,758	121,866,624
6	2	4	7	16	13,024	12,359	160,963,616
7	2	4	8	16	14,720	13,960	205,491,200
8	2	5	5	32	23,288	22,693	528,474,584
9	2	5	6	32	27,408	26,694	731,629,152
10	3	3	3	8	3,564	3,243	11,558,052
11	3	3	4	8	4,560	4,132	18,841,920
12	3	3	5	8	5,556	5,021	27,896,676
13	3	4	4	16	11,392	10,820	123,261,440
14	3	4	5	16	13,904	13,189	183,379,856

experiment 8 is (4, 8, 2). The total number of scenarios observed in an iteration is equal to N^*K, and if an iteration fails to satisfy the stopping criterion, the number of replication increases by one until it is satisfied, but the size of the subsets of scenarios is not increased since by doing so would make the problem unsolvable. For these experiments, the 100(1-α) confidence interval provided by the sample average approximation (SAA) method (refer to Sections 2.4.5 and 3.5.2) is observed so as to end with a reliable range for the best solution that the Monte Carlo method produces, instead of near-optimality.

5.8. Results and Analysis

In this section, we present the results for the experimental data for the SHSSP (General) provided in Tables 5.4 and 5.5.

5.8.1. Results on the Performances of the MCM and Optimum-Seeking Method

Table 5.6 presents results for the experiments described in Table 5.4. The first column presents the experiment number as that presented in Table 5.4. The second and third columns contain the optimal objective function values and the cpu times required by the BNC method for each experiment. The results of the MCM are presented in the 4th to the 9th columns. The 4th and 5th columns include the initial value of (N, $\tilde{N}$, K) for problem h_N^k, and the final value of (N, $\tilde{N}$, K), respectively. If the value of the Final (N, $\tilde{N}$, K) column for an experiment in the table is identical to that of the Initial (N, $\tilde{N}$, K) column, it means that the stopping criterion is satisfied after the first iteration for that experiment.

Table 5.6. Results for the Experimental Data Presented in Table 5.4

Experiment #	Branch-and-Cut Method		Monte Carlo Method						Near-Optimality
	Optimal Objective Function Value $h(X^*)$	CPU Time for the Optimum-Seeking Method	Initial (N,Ñ,K)	Final (N,Ñ,K)	Best Objective Function Value $h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)$	Average Objective Function Value $\bar{h}_N$	Tolerance Ratio $1-\frac{h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)}{\bar{h}_N}$	CPU Time for the MCM	$\frac{h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)}{h(X^*)}$
1	2000	0.061	(1,2,2)	(1,2,2)	2000	2000	0	0.003	1
2	14000	0.015	(1,2,2)	(1,2,2)	14000	14000	0	0.004	1
3	24900	0.107	(1,2,2)	(1,2,2)	25000	26000	0.04	0.023	1.004
4	19700	0.305	(2,4,2)	(2,4,2)	19850	19850	0	0.020	1.008
5	35800	1.265	(2,4,2)	(2,4,3)	35400	35733.3	0.009	0.484	0.989
6	38700	4.071	(2,4,2)	(2,4,3)	39400	45133.3	0.146	10.781	1.018
7	38700	4.306	(2,4,2)	(2,4,5)	39400	44880	0.139	91.437	1.018
8	20000	0.072	(2,4,2)	(2,4,2)	20000	20000	0	0.021	1
9	43400	1.024	(2,4,2)	(2,4,2)	43400	43400	0	0.194	1
10	57400	3.231	(2,4,2)	(2,4,2)	57400	57400	0	2.246	1
11	79800	158.440	(2,4,2)	(2,4,2)	78900	81600	0.034	12.783	0.989
12	108700	509.840	(2,4,2)	(2,4,2)	108500	121300	0.118	45.394	0.998
13	105100	900.100	(2,4,2)	(2,4,4)	105950	118900	0.122	6827.8	1.008
14	57100	2.461	(1,2,2)	(1,2,2)	57100	57100	0	0.055	1
15	85600	24.914	(1,2,2)	(1,2,3)	85600	87200	0.019	0.544	1
16	103900	714.913	(1,2,2)	(1,2,2)	104300	111500	0.069	1.608	1.004
17	119700	900.154	(1,2,2)	(1,2,5)	122800	145200	0.182	37.848	1.026
18	144700	900.202	(1,2,2)	(1,2,2)	165500	189500	0.145	9.633	1.144
19	90200	265.162	(2,4,2)	(2,4,2)	89300	91100	0.020	4.426	0.990
20	57100	4.676	(1,2,2)	(1,2,2)	57100	57100	0	0.315	1
21	86700	282.538	(1,2,2)	(1,2,2)	84000	89000	0.060	0.594	0.969
22	115600	900.162	(1,2,2)	(1,2,2)	124000	124000	0	0.508	1.073
23	90800	41.604	(2,4,2)	(2,4,2)	90800	90800	0	0.171	1
24	119700	900.452	(2,4,2)	(2,4,2)	119600	120200	0.005	2.182	0.999

The 6th and 7th columns contain the values of $h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)$ and $\bar{h}_N$, respectively. The 8th column contains the tolerance ratio, which is the value of $1-\frac{h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)}{\bar{h}_N}$ when the stopping criterion is met, i.e., when $1-\frac{h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)}{\bar{h}_N}<\varepsilon$ .In order to maintain inherent properties of $h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)$ (unbiased estimate of the lower bound of $h\left(X^*\right)$) and $\bar{h}_N$ (unbiased estimate of the upper bound of $h\left(X^*\right)$), a result of an iteration is rejected when $h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)\le\bar{h}_N$. The 9th column includes the cpu times required by the MCM until the stopping criterion is satisfied. The 10th column provides the values of $\frac{h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)}{h\left(X^*\right)}$, and when its value is equal to one, the MCM ends up with a good quality of solution. Next, we make some observations on the results presented in the table.

First, the experiments highlighted in gray color in the table (Experiments 5, 6, 7, 13, 15, 17) require extra replications. In particular, Experiments 5, 6, and 15 require one more iteration, Experiment 13 requires two more iterations, and Experiments 7 and 17 require two more iterations. Second, in terms of the tolerance ratio, the worst value is 0.182 for Experiment 17. The average value of the tolerance ratio is only 0.04618, which means that the difference between the lower and upper bounds is just 4.6% on average. Third, in terms of near-optimality, the worse value is 1.144 for Experiment 18, and their average value is only 1.009825, which means that the difference between the optimal objective function value and the best objective function value obtained by the MCM is about 1% on average.

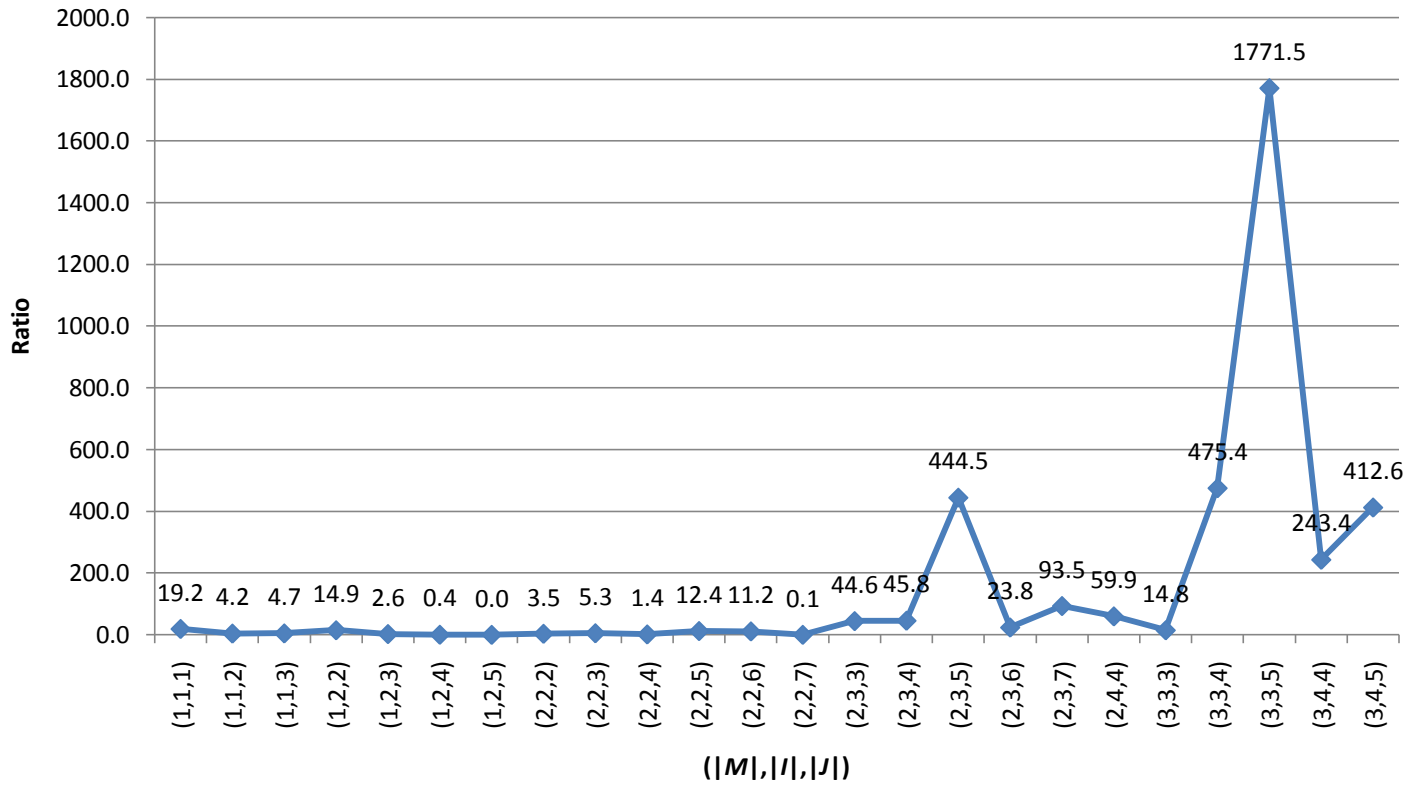

Figure 5.5. Value of Ratio= $\frac{\text{CPU Time Value Given in Column 3 of Table 5.6}}{\text{CPU Time Value Given in Column 9 of Table 5.6}}$ vs. ($|M|$, $|I|$, $|J|$)

Figure 5.5 shows a plot of the ratio of the entries in the 'CPU Time for Optimum-Seeking Method' column (the 3rd column in Table 5.6) to those in the 'CPU Time for the MCM' column (the 9th column in Table 5.6) vs. problem parameters ($|M|$, $|I|$, $|J|$). The number next to each dot is the value of each ratio, and the average value of those ratios is 154.577, which means that the MCM solves about 154.577 times as fast as the BNC method on average. Thus, in conclusion, at the expense of 1% gap between $h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)$ and $h\left(X^*\right)$, we are able to solve the problems given in Table 5.6 faster by a factor of 154.577 on average.

5.8.2. Results for Experiments in the Second Group

Table 5.7 presents the results for the experimental data presented in Table 5.5. The 95% confidence interval on the optimality gap, i.e., the confidence interval of the length of

$\left[h\left(\left(\hat{X}_N\right)^*\right), h\left(X^*\right)\right]$ is presented in the 11th column. The confidence interval for each experiment is between 0 and the value in that column. Eight experiments (2, 3, 4, 5, 6, 7, 10, and 11) highlighted in gray color in Table 5.7 need extra iterations. The descriptions of all other columns are identical to those in Table 5.6. Experiments 3, 4, 6, and 7 need one more iteration, Experiments 5 and 10 need two more iterations, Experiment 11 needs three more iterations, and Experiment 2 needs four more iterations. In terms of tolerance ratio, the worst value is 0.166 for Experiment 2, and the average of these values is 0.055, which means that the difference of unbiased estimates of lower and upper bounds is only 5.5% on average.

Table 5.7. Results for the Experimental Data Presented in Table 5.5

Experiment #	$\|M\|$	$\|I\|$	$\|J\|$	$\|S\|$	Initial (N,Ñ,K)	Final (N,Ñ,K)	Best Objective Function Value $h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)$	Average Objective Function Value $\bar{h}_N$	Tolerance Ratio $1-\frac{h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)}{\bar{h}_N}$	Confidence Interval on Optimality Gap	CPU Time for the MCM
1	2	3	8	8	(1,2,2)	(1,2,2)	197800	222700	0.126	31679.722	10.618
2	2	3	8	8	(1,2,2)	(1,2,6)	197800	230633.3	0.166	35562.323	111.687
3	2	3	9	8	(1,2,2)	(1,2,3)	249400	259066.7	0.039	16348.651	37.766
4	2	3	9	8	(1,2,2)	(1,2,3)	250800	259142.9	0.033	13018.607	194.900
5	2	3	10	8	(1,2,2)	(1,2,4)	245900	282100	0.147	43901.585	25026.880
6	2	4	5	16	(2,4,2)	(2,4,3)	111325	115383.3	0.036	4878.422	1071.765
7	2	4	5	16	(2,4,2)	(2,4,3)	110100	116366.7	0.057	7312.608	253.679
8	2	4	6	16	(2,4,2)	(2,4,2)	134150	146850	0.095	15968.016	590.617
9	2	4	6	16	(2,4,2)	(2,4,2)	133600	138600	0.037	7112.374	2515.015
10	2	4	7	16	(2,4,2)	(2,4,4)	159400	166237.5	0.043	8923.409	7960.481
11	2	4	8	16	(2,4,2)	(2,4,5)	162420	174250	0.073	14691.719	12599.962
12	2	5	5	32	(4,8,2)	(4,8,2)	129875	140800	0.084	13101.800	408.553
13	2	5	6	32	(4,8,2)	(4,8,2)	142600	154900	0.086	14957.317	27123.813
14	3	3	3	8	(1,2,2)	(1,2,2)	57100	57100	0.000	11718.447	0.315
15	3	3	4	8	(1,2,2)	(1,2,2)	84000	89000	0.060	9829.379	0.594
16	3	3	5	8	(1,2,2)	(1,2,2)	124000	124000	0.000	11626.433	0.508
17	3	4	4	16	(2,4,2)	(2,4,2)	90800	91400	0.007	8682.929	1.084
18	3	4	4	16	(2,4,2)	(2,4,2)	90800	90800	0.000	13337.155	0.171
19	3	4	5	16	(2,4,2)	(2,4,2)	119600	120200	0.005	1851.720	2.182
20	3	4	5	16	(2,4,2)	(2,4,2)	120350	120350	0.000	1709.780	27.762

5.9 Concluding Remarks

In this chapter, we have extended the HSSP (General), that we discussed in Chapter IV, by considering operation times as stochastic variables, and we have designated it as the SHSSP (General). We have presented a general stochastic programming formulation for the SHSSP, designated as **SHSSP2 (General)**. We have proven that the recourse model of SHSSP2 (General) is a two-stage relatively complete recourse model. We have also considered a special case of **SHSSP2 (General)**, and its stochastic programming formulation has been provided, designated as **SHSSP2 (Special)**. **SHSSP2 (Special)** allows no overlapping schedules where a surgeon cannot have multiple operations in an operating room at the same time. We have proven that we can obtain an optimal solution for this problem by considering another formulation that contains the longest operation time for each assignment of a surgeon and an operation (designated as **SHSSP (Longest)**), and that has identical objective function value, obtained by properly modifying its objective function (designated as **HSSP (Equivalent)**). We have shown that the degree of complexity of **HSSP (Equivalent)** is identical to that of **HSSP (General)**, formulated in Section 4.1.1.

The formulation **SHSSP2 (General)** allows multiple operations assigned to a surgeon and operating rooms at the same time , and also allows under-assigned operation time for a surgeon to perform an operation, by incorporating appropriate penalties for each case. The problem size can grow rapidly for this case. We have developed the use of the MCM in conjunction with the BNC method for **SHSSP2 (General)**, and we have shown that it can save cpu times of the order of 150 folds. Also, through our experimentation, we have shown that the confidence interval obtained is quite reliable. In conclusion, the use of the MCM is effective in reducing the complexity of solving the formulation **SHSSP2 (General)**.

Chapter VI. Branch-and-Price Method for a Stochastic Short-Term Personnel Planning Problem

As we mentioned in Section 2.6, we can divide personnel planning problems into two groups: the short-term personnel planning (STPP) and the long-term personnel planning (LTPP) problems. The STPP problem has received relatively less attention in the literature than the LTPP problem. However, the need for the STPP problem has grown over the years because of dynamic market conditions and need for organizations, both big and small, to trim cost due to competition and worsening economic conditions. Besides, there are several service agencies that only work with short-term employees in placing them on temporary (or freelance) jobs. An example is Kelly Services that offer staffing solutions for diverse needs of various companies, which arises in the areas of finance and accounting, education, engineering, information technology, law, science, healthcare, art and design, and home care, among others. We can also think of contractors (or sub-contractors) hired by a manufacturing or a construction company for a short-term employment as well.

In this chapter, we address such a short-term personnel planning problem. We develop a model for our STPP problem by adapting the SGAP presented in Chapter III. Our short-term personnel planning problem can be distinguished from the previous short-term personnel planning problems with respect to the following two main aspects. First, we consider the objective of minimizing the total cost incurred due to the training (or setup) of employees, payroll, and overtime. Each contractor is, typically, multi-functional. Therefore, we need to find the right set of contractors so that the required jobs are accomplished at minimum cost. It is reasonable to assume that less experienced contractors will cost less but will require a relatively high setup cost, and the reverse will be true for the experienced contractors. Second, we assume uncertainty in processing times or the amount of resource required to perform a job. We designate our stochastic short-term personnel planning problem as SSTPP.

6.1. Stochastic Short-Term Personnel Planning Problem (SSTPP)

6.1.1. Formulation for SSTPP

We formulate our stochastic short-term personnel planning (SSTPP) problem by adapting the formulation **SGAP2** presented in Section 3.1.1.1 for the stochastic generalized assignment problem. Consider the following notation.

Parameters

- i : Contractor i, $i \in I$
- j : Job j, $j \in J$
- s : Scenario s, $s \in S$
- $I(j)$: Set of contractors who have the required skill for job j, $\forall j \in J$
- $J(i)$: Set of jobs for which contractor i has the necessary skill, $\forall i \in I$
- h_i : Total cost including hiring cost, training cost, setup cost, and payroll for contractor i, $\forall i \in I$
- c_{ij} : Processing cost incurred by contractor i for job j, $\forall i \in I$, $\forall j \in J(i)$
- p_s : Probability of scenario s, $\forall s \in S$
- e_i : Overtime cost for contractor i, $\forall i \in I$
- d_{ijs} : Processing time or amount of resource required by contractor i to complete job j under scenario s, $\forall i \in I$, $\forall j \in J(i)$, $\forall s \in S$
- T : Regular work hours in a given planning horizon
- TP : Maximum overtime permitted in a given planning horizon
- M : A large positive number

Variables

- w_i : Binary variable, =1 if contractor i is hired; =0, otherwise, $\forall i \in I$
- x_{ij} : Binary variable, =1 if contractor i works on job j; =0, otherwise, $\forall i \in I$, $\forall j \in J(i)$

y_{is} : Continuous variable representing the amount of overtime for contractor i under scenario s, $\forall i \in I$, $\forall s \in S$

SSTPP1

Minimize $$\sum_{i \in I} h_i w_i + \sum_{i \in I} \sum_{j \in J(i)} c_{ij} x_{ij} + \mathrm{E}[\tilde{f}(\vec{x}, \vec{w})] \qquad (6.1a)$$

Subject to $$\sum_{i \in I(j)} x_{ij} = 1 \qquad \forall j \in J \qquad (6.1b)$$

$$\sum_{j \in J(i)} x_{ij} - M w_i \leq 0 \qquad \forall i \in I \qquad (6.1c)$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in I, \forall j \in J(i) \qquad (6.1d)$$

$$w_i \in \{0,1\} \qquad \forall i \in I \qquad (6.1e)$$

where

$(\vec{x}, \vec{w})$: Vector of (x_{ij}, w_i) $\forall i \in I$, $\forall j \in J(i)$, that satisfy constraint sets (6.1b), (6.1c), (6.1d) and (6.1e)

$\mathrm{E}[\tilde{f}(\vec{x}, \vec{w})]$: Expected penalty cost for overtime

The formulation considers minimization of the total cost incurred in a given planning horizon. We assume that contractors will be released after the planning horizon is over. Therefore, we do not consider the layoff cost for the contractors. The first term in the objective function (6.1a) consists of the total cost incurred as a result of hiring, training, setup, and payroll by each contractor. The second term in (6.1a) represents the sum of processing costs incurred by the contractors. The third term pertains to the expected total overtime cost considering all the scenarios.

Constraint set (6.1b) represents job assignment constraints like constraint set (3.3f) in Section 3.1.1.1. A contractor $i \in \mathrm{I}$ can work on a job only if he is hired. This is captured by constraint set (6.1c). Then, the recourse model for scenario s, $f_s(X)$, is as follows.

Recourse Model of SSTPP1 for Scenario *s*

Minimize $\sum_{i \in I} e_i y_{is}$ (6.2a)

Subject to $y_{is} = \text{Max}\left(0, \sum_{j \in J(i)} d_{ijs} x_{ij} - T\right)$ $\forall i \in I$ (6.2b)

$0 \le y_{is} \le TP$ $\forall i \in I$ (6.2c)

The objective function in (6.2a) is the penalty cost under scenario *s*. Constraint set (6.2b) determines the amount of overtime, and constraint set (6.2c) limits the amount of overtime to *TP*. Constraint set (6.2b) can be expressed as $\sum_{j \in J(i)} d_{ijs} x_{ij} - y_{is} \le T$. If scenario *s* follows a discrete probability distribution function, then $\text{E}[\tilde{f}(X)]$ in (6.1a) can be expressed as $\sum_s p_s f_s(X)$. We have the following full description for the formulation **SSTPP1**.

SSTPP2

Minimize $\sum_{i \in I} h_i w_i + \sum_{i \in I} \sum_{j \in J(i)} c_{ij} x_{ij} + \sum_{s \in S} p_s \sum_{i \in I} e_i y_{is}$ (6.3a)

Subject to $\sum_{i \in I(j)} x_{ij} = 1$ $\forall j \in J$ (6.3b)

$\sum_{j \in J(i)} d_{ijs} x_{ij} - T \le y_{is} \le TP$ $\forall i \in I, \forall s \in S$ (6.3c)

$\sum_{j \in J(i)} x_{ij} - M w_i \le 0$ $\forall i \in I$ (6.3d)

$x_{ij} \in \{0,1\}$ $\forall i \in I, \forall j \in J(i)$ (6.3e)

$w_i \in \{0,1\}$ $\forall i \in I$ (6.3f)

$y_{is} \ge 0$ $\forall i \in I, \forall s \in S$ (6.3g)

In the formulation above, x_{ij} and w_i, $\forall i \in I, \forall j \in J(i)$ are the first stage variables and y_{is}, $\forall i \in I, \forall s \in S$ are the second stage variables. This separation is logical since in the first stage, we determine the contractors who are hired and the jobs they are assigned to, and in the second stage, we determine the overtime required by each contractor *i*, $i \in I$.

In contrast to the formulation **SGAP2**, we add variables w_i, $i \in I$, to consider the number of contractors, and constraint set (6.3d) to determine whether a contractor is hired. The amount of resource is not unrestricted, and hence, y_{is} $\forall i \in I$, $\forall s \in S$, are bounded from above. Also, it is reasonable to assume that each contractor has the same amount of available work hours, e.g., T=8 hours and TP=4 hours, in which case every contractor has 8 hours of regular work and at most 4 hours of overtime per day.

6.1.2. Stochastic Properties of the Recourse Function of SSTPP1

We can categorize the formulation **Recourse Model of SSTPP1 for Scenario s**, as follows:

- It is a two-stage recourse model since we have two different stage variables to determine.
- It is a fixed recourse model since all the coefficients of the second stage variables, y_{is}, being equal to [-1], remain the same irrespective of i and s, but it is not a simple recourse model.
- It is not a relatively complete recourse model unless the upper bounds for y_{is} $\forall i \in I$, $\forall s \in S$ are infinite. We show this next.

Proposition 6.1. The formulation **Recourse Model of SSTPP1 for Scenario s** is not a relatively complete recourse model if the values of any of y_{is} are limited. Therefore, it is not a complete recourse model, either. However, if y_{is}, $\forall i \in I$, $\forall s \in S$, can take any value, it is a complete recourse model as well as a relatively complete recourse model.

Proof. The proof simply follows from the fact that there can exist x_{ij} and w_i satisfying constraint sets (6.1b), (6.1c), (6.1d), and (6.1e), for which $y_{is} = \sum_{j \in J(i)} d_{ijs} x_{ij} - T > TP$, thereby, violating constraint set (6.2c). However, in case TP is unlimited, then (6.2c) will always hold true. □

Because the formulation **Recourse Model of SSTPP1 for Scenario s** is not a relatively complete recourse model, the solution given by the MCM may be infeasible to the formulation **SSTPP2** on for the selected scenarios. Therefore, we cannot employ the Monte Carlo method (MCM) to the formulation **SSTPP2**.

6.1.3. Size of Formulation SSTPP2

In this section, we calculate the number of variables and the number of constraints of **SSTPP2,** and we show that the branch-and-price method is justified at least in terms of the number of variables being larger than or equal to the number of constraints.

Proposition 6.2. In the formulation **SSTPP2**, the number of variables is larger than or equal to the number of constraints.

Proof. The number of variables in **SSTPP2** is as follows:

$$\text{Total Number of Variables} = \sum_{j \in J} |I(j)| + |I| + |I| * |S|, \tag{6.4a}$$

where $|I(j)|$ is the number of contractors who can conduct job j.

The number of all the constraints from (6.3b) to (6.3g) is equal to ($|J|+|I|*|S|+|I|$). (6.4b)

Then, the difference between (6.4a) and (6.4b) is $= \sum_{j} |I(j)| - |J|$. (6.4c)

The R.H.S. of (6.4c) is always nonnegative, and the equality holds when for every job there is only one contractor. □

6.1.4. Stochastic Short-Term Personnel Planning Problem (SSTPP) with Expected Values

In practice, expected values are, occasionally, employed to calculate solution of a stochastic program (SP). However, there is no guarantee that it results in a feasible solution. In this section, we employ expected values for stochastic parameters in the formulation **SSTPP2** and show that it yields an infeasible solution to the formulation **SSTPP2**.

6.1.4.1. Formulation for the SSTPP with Expected Values

Consider the following notation.

$\hat{y}_i$: Slack variable to capture the expected value of excessive amount of the resource utilized by contractor i, $\forall i \in I$

$\hat{d}_{ij}$: Average processing time of job j of contractor i under scenario s, $\forall i \in I, \forall j \in J$

Note that $\hat{d}_{ij} = \sum_{s \in S} p_s d_{ijs}$. We have the following formulation for the SSTPP with Expected Values, called **Expected-SSTPP2**.

Expected-SSTPP2

Minimize $\sum_{i \in I} h_i w_i + \sum_{i \in I} \sum_{j \in J(i)} c_{ij} x_{ij} + \sum_{i \in I} e_i \hat{y}_i$ (6.5a)

Subject to $\sum_{i \in I(j)} x_{ij} = 1$ $\forall j \in J$ (6.5b)

$\sum_{j \in J(i)} \hat{d}_{ij} x_{ij} - T \le \hat{y}_i \le TP$ $\forall i \in I$ (6.5c)

$\sum_{j \in J(i)} x_{ij} - M w_i \le 0$ $\forall i \in I$ (6.5d)

$x_{ij} \in \{0,1\}$ $\forall i \in I, \forall j \in J(i)$ (6.5e)

$w_i \in \{0,1\}$ $\forall i \in I$ (6.5f)

$\hat{y}_i \ge 0$ $\forall i \in I$ (6.5g)

Note that **Expected-SSTPP2** has $|I|$ ($|S|$-1) less number of constraints (see (6.3c) and (6.5c)) and $|I|$ ($|S|$-1) less number of continuous variables (see y_{is} in **SSTPP2** and $\hat{y}_i$ in **Expected SGAP2 (Stochastic)**) than **SSTPP2**.

6.1.4.2. An Example of Expected-SSTPP2 Producing an Infeasible Solution to SSTPP2

In this section, we show that **Expected-SSTPP2** needs not guarantee a feasible solution to **SSTPP2**. Consider an example consisting of 3 contractors and 8 jobs. The "original" stochastic STPP formulation is as follows:

An Example of SSTPP2

a. Objective Function

Minimize

37.5 {y(0,0) + y(0,1) + y(0,2) + y(0,3) + y(0,4) + y(0,5) + y(0,6) + y(0,7)} +
25 {y(1,0) + y(1,1) + y(1,2) + y(1,3) + y(1,4) + y(1,5) + y(1,6) + y(1,7)} +
12.5 {y(2,0) + y(2,1) + y(2,2) + y(2,3) + y(2,4) + y(2,5) + y(2,6) + y(2,7)} +
110 x(0,0) + 100 x(0,1) + 90 x(0,2) + 80 x(0,3) + 70 x(0,4) + 60 x(0,5) + 50 x(0,6) + 40 x(0,7) +
100 x(1,0) + 90 x(1,1) + 80 x(1,2) + 70 x(1,3) + 60 x(1,4) + 50 x(1,5) + 40 x(1,6) + 30 x(1,7) +
90 x(2,0) + 80 x(2,1) + 70 x(2,2) + 60 x(2,3) + 50 x(2,4) + 40 x(2,5) + 30 x(2,6) + 20 x(2,7) +
200 w(0) + 400 w(1) + 600 w(2)

Subject To:

b. Job Assignment Constraint Set

JOB(0): x(0,0) + x(1,0) + x(2,0) = 1
JOB(1): x(0,1) + x(1,1) + x(2,1) = 1
JOB(2): x(0,2) + x(1,2) + x(2,2) = 1
JOB(3): x(0,3) + x(1,3) + x(2,3) = 1
JOB(4): x(0,4) + x(1,4) + x(2,4) = 1
JOB(5): x(0,5) + x(1,5) + x(2,5) = 1
JOB(6): x(0,6) + x(1,6) + x(2,6) = 1
JOB(7): x(0,7) + x(1,7) + x(2,7) = 1

c. Maximum Contractor Capacity Constraint Set

CAPA(0,0): - y(0,0) + x(0,0) + 1.5 x(0,1) + 2 x(0,2) + 2.5 x(0,3) + 3 x(0,4) + 3.5 x(0,5) + 4 x(0,6) + 4.5 x(0,7) ≤ 8

CAPA(0,1): - y(0,1) + x(0,0) + 1.5 x(0,1) + 2 x(0,2) + 2.5 x(0,3) + 3 x(0,4)+ 3.5 x(0,5) + 4 x(0,6) + 4.5 x(0,7) ≤ 8

CAPA(0,2): - y(0,2) + x(0,0) + 1.5 x(0,1) + 2 x(0,2) + 2.5 x(0,3) + 3 x(0,4) + 3.5 x(0,5) + 4 x(0,6) + 4.5 x(0,7) ≤ 8

CAPA(0,3): - y(0,3) + x(0,0) + 1.5 x(0,1) + 2 x(0,2) + 2.5 x(0,3) + 3 x(0,4) + 3.5 x(0,5) + 4 x(0,6) + 4.5 x(0,7) ≤ 8

CAPA(0,4): - y(0,4)+1.5 x(0,0)+2.25 x(0,1)+3 x(0,2)+3.75 x(0,3)+4.5 x(0,4)+5.25 x(0,5)+6 x(0,6)+6.75 x(0,7) ≤ 8

CAPA(0,5): - y(0,5)+1.5 x(0,0)+2.25 x(0,1)+3 x(0,2)+3.75 x(0,3)+4.5 x(0,4)+5.25 x(0,5)+6 x(0,6)+6.75 x(0,7) ≤ 8

CAPA(0,6): - y(0,6)+1.5 x(0,0)+2.25 x(0,1)+3 x(0,2)+3.75 x(0,3)+4.5 x(0,4)+5.25 x(0,5)+6 x(0,6)+6.75 x(0,7) ≤ 8

CAPA(0,7): - y(0,7)+1.5 x(0,0)+2.25 x(0,1)+3 x(0,2)+3.75 x(0,3)+4.5 x(0,4)+5.25 x(0,5)+6 x(0,6)+6.75 x(0,7) ≤ 8

CAPA(1,0): - y(1,0)+1.5 x(1,0)+2 x(1,1)+2.5 x(1,2)+3 x(1,3)+3.5 x(1,4)+4 x(1,5)+4.5 x(1,6)+5 x(1,7) ≤ 8

CAPA(1,1): - y(1,1)+1.5 x(1,0)+2 x(1,1)+2.5 x(1,2)+3 x(1,3)+3.5 x(1,4)+4 x(1,5)+4.5 x(1,6)+5 x(1,7) ≤ 8

CAPA(1,2): - y(1,2)+2.25 x(1,0)+3 x(1,1)+3.75 x(1,2)+4.5 x(1,3)+5.25 x(1,4)+6 x(1,5)+6.75 x(1,6)+7.5 x(1,7) ≤ 8

CAPA(1,3): - y(1,3)+2.25 x(1,0)+3 x(1,1)+3.75 x(1,2)+4.5 x(1,3)+5.25 x(1,4)+6 x(1,5)+6.75 x(1,6)+7.5 x(1,7) ≤ 8

CAPA(1,4): - y(1,4)+1.5 x(1,0)+2 x(1,1)+2.5 x(1,2)+3 x(1,3)+3.5 x(1,4)+4 x(1,5)+4.5 x(1,6)+5 x(1,7) ≤ 8

CAPA(1,5): - y(1,5)+1.5 x(1,0)+2 x(1,1)+2.5 x(1,2)+3 x(1,3)+3.5 x(1,4)+4 x(1,5)+4.5 x(1,6)+5 x(1,7) ≤ 8

CAPA(1,6): - y(1,6)+2.25 x(1,0)+3 x(1,1)+3.75 x(1,2)+4.5 x(1,3)+5.25 x(1,4)+6 x(1,5)+6.75 x(1,6)+7.5 x(1,7) ≤ 8

CAPA(1,7): - y(1,7)+2.25 x(1,0)+3 x(1,1)+3.75 x(1,2)+4.5 x(1,3)+5.25 x(1,4)+6 x(1,5)+6.75 x(1,6)+7.5 x(1,7) ≤ 8

CAPA(2,0): - y(2,0)+2 x(2,0)+2.5 x(2,1)+3 x(2,2)+3.5 x(2,3)+4 x(2,4)+4.5 x(2,5)+5 x(2,6)+5.5 x(2,7) ≤ 8

CAPA(2,1): - y(2,1)+3 x(2,0)+3.75 x(2,1)+4.5 x(2,2)+5.25 x(2,3)+6 x(2,4)+6.75 x(2,5)+7.5 x(2,6)+8.25 x(2,7) ≤ 8

CAPA(2,2): - y(2,2)+2 x(2,0)+2.5 x(2,1)+3 x(2,2)+3.5 x(2,3)+4 x(2,4)+4.5 x(2,5)+5 x(2,6)+5.5 x(2,7) <= 8

CAPA(2,3): - y(2,3)+3 x(2,0)+3.75 x(2,1)+4.5 x(2,2)+5.25 x(2,3)+6 x(2,4)+6.75 x(2,5)+7.5 x(2,6)+8.25 x(2,7) ≤ 8

CAPA(2,4): - y(2,4)+2 x(2,0)+2.5 x(2,1)+3 x(2,2)+3.5 x(2,3)+4 x(2,4)+4.5 x(2,5)+5 x(2,6)+5.5 x(2,7) <= 8

CAPA(2,5): - y(2,5)+3 x(2,0)+3.75 x(2,1)+4.5 x(2,2)+5.25 x(2,3)+6 x(2,4)+6.75 x(2,5)+7.5 x(2,6)+8.25 x(2,7) ≤ 8

CAPA(2,6): - y(2,6)+2 x(2,0)+2.5 x(2,1)+3 x(2,2)+3.5 x(2,3)+4 x(2,4)+4.5 x(2,5)+5 x(2,6)+5.5 x(2,7) <= 8

CAPA(2,7): - y(2,7)+3 x(2,0)+3.75 x(2,1)+4.5 x(2,2)+5.25 x(2,3)+6 x(2,4)+6.75 x(2,5)+7.5 x(2,6)+8.25 x(2,7) ≤ 8

d. Contractor Configuration Constraint Set

CONFIG(0): x(0,0)+x(0,1)+x(0,2)+x(0,3)+x(0,4)+x(0,5)+x(0,6)+x(0,7)-2147483647 w(0) ≤ 0

CONFIG(1): x(1,0)+x(1,1)+x(1,2)+x(1,3)+x(1,4)+x(1,5)+x(1,6)+x(1,7)-2147483647 w(1) ≤ 0

CONFIG(2): x(2,0)+x(2,1)+x(2,2)+x(2,3)+x(2,4)+x(2,5)+x(2,6)+x(2,7)-2147483647 w(2) ≤ 0

e. Upper Bound and Logical Constraint Set

0 ≤ y(i,s) ≤ 8	∀ i ∈ {0,1,2}, ∀ s ∈ {0,1,2,3,4,5,6,7}
x(i,j) ∈ {0,1}	∀ i ∈ {0,1,2}, ∀ j ∈ {0,1,2,3,4,5,6,7}
w(i) ∈ {0,1}	∀ i ∈ {0,1,2}

f. Optimal Solution

$w(i)^*=\begin{bmatrix}1\\1\\1\end{bmatrix}$, $x(i,j)^*=\begin{bmatrix}1&1&1&1&0&0&0&0\\0&0&0&0&1&1&0&0\\0&0&0&0&0&0&1&1\end{bmatrix}$, $y(i,s)^*=\begin{bmatrix}0&0&0&0&2.5&2.5&2.5&2.5\\0&0&3.25&3.25&0&0&3.25&3.25\\2.5&7.75&2.5&7.75&2.5&7.75&2.5&7.75\end{bmatrix}$, $z^*=2952.5$

Following is the formulation **Expected-SSTPP2** corresponding to the preceding example.

Expected-SSTPP2 for the Above Example

a. Objective Function

Minimize

300 y(0) + 200 y(1) + 100 y(2) + 110 x(0,0) + 100 x(0,1) + 90 x(0,2) +
80 x(0,3) + 70 x(0,4) + 60 x(0,5) + 50 x(0,6) + 40 x(0,7) + 100 x(1,0) +
90 x(1,1) + 80 x(1,2) + 70 x(1,3) + 60 x(1,4) + 50 x(1,5) + 40 x(1,6)+
30 x(1,7) + 90 x(2,0) + 80 x(2,1) + 70 x(2,2) + 60 x(2,3) + 50 x(2,4)+
40 x(2,5) + 30 x(2,6) + 20 x(2,7) + 200 w(0) + 400 w(1) + 600 w(2)

Subject To:

b. Job Assignment Constraint Set

JOB(0): x(0,0) + x(1,0) + x(2,0) = 1
JOB(1): x(0,1) + x(1,1) + x(2,1) = 1
JOB(2): x(0,2) + x(1,2) + x(2,2) = 1
JOB(3): x(0,3) + x(1,3) + x(2,3) = 1
JOB(4): x(0,4) + x(1,4) + x(2,4) = 1
JOB(5): x(0,5) + x(1,5) + x(2,5) = 1
JOB(6): x(0,6) + x(1,6) + x(2,6) = 1
JOB(7): x(0,7) + x(1,7) + x(2,7) = 1

c. Maximum Contractor Capacity Constraint Set

CAPA(0): -y(0)+1.25x(0,0)+1.875x(0,1)+2.5x(0,2)+3.125x(0,3)+3.75x(0,4)+4.375x(0,5)+5x(0,6)+5.625x(0,7)<=8
CAPA(1): -y(1)+1.875x(1,0)+2.5x(1,1)+3.125x(1,2)+3.75x(1,3)+4.375x(1,4)+5x(1,5)+5.625x(1,6)+6.25x(1,7)<=8
CAPA(2): -y(2)+2.5x(2,0)+3.125x(2,1)+3.75x(2,2)+4.375x(2,3)+5x(2,4)+5.625x(2,5)+6.25x(2,6)+6.875x(2,7)<=8

d. Contractor Configuration Constraint Set

CONFIG(0): x(0,0) + x(0,1) + x(0,2) + x(0,3) + x(0,4) + x(0,5) + x(0,6) + x(0,7) - 2147483647 w(0) <= 0

CONFIG(1): x(1,0) + x(1,1) + x(1,2) + x(1,3) + x(1,4) + x(1,5) + x(1,6) + x(1,7) - 2147483647 w(1) <= 0

CONFIG(2): x(2,0) + x(2,1) + x(2,2) + x(2,3) + x(2,4) + x(2,5) + x(2,6) + x(2,7) - 2147483647 w(2) <= 0

e. Upper Bound and Logical Constraint Set

$0 \le y(i) \le 8$ $\quad \forall i \in \{0,1,2\}$

$x(i,j) \in \{0,1\}$ $\quad \forall i \in \{0,1,2\}, \forall j \in \{0,1,2,3,4,5,6,7\}$

$w(i) \in \{0,1\}$ $\quad \forall i \in \{0,1,2\}$

f. Optimal Solution

$$w(i)^*=\begin{bmatrix}1\\1\\1\end{bmatrix},\ x(i,j)^*=\begin{bmatrix}10011000\\01000001\\00100110\end{bmatrix},\ y(i)^*=\begin{bmatrix}0.125\\0.750\\7.625\end{bmatrix},\ z^*=2670$$

Comparing the above formulations, the optimal objective function value of the "original" formulation **SSTPP2** is larger than that of the formulation **Expected-SSTPP2** by 282.5 (=2952.5-2670). However, the solution to the latter is not feasible to **the former**. For instance, x(2,2)=x(2,5)=x(2,6)=1 for **Expected-SSTPP2**. However, if we apply this solution to constraint CAPA(2,1) of **SSTPP2**, it violates the upper bound of 8. Therefore, in general, we cannot use the formulation, **Expected-SSTPP2**.

Table 6.1 gives a summary of the solutions obtained for **Expected-SSTPP2** using 35 examples. Note that 24 out of 35 experiments yield optimal solutions of **Expected-SSTPP2** that are infeasible to the corresponding **SSTPP2**. Furthermore, note that when the number of contractors ($|I|$) and the number of jobs ($|J|$) are increased, the chances of obtaining an infeasible solution by **Expected-SSTPP2** increase.

Table 6.1. Feasibility of a Solution to **Expected-SSTPP2**

Experiment #	$\|I\|$	$\|J\|$	$\|S\|$	Optimal Objective Function Value		Feasibility of Solution for **Expected-SSTPP2**
				Expected-SSTPP2	**SSTPP2**	
1	7	7	128	2670	2862.5	Feasible
2	7	8	128	3680	4122.5	Feasible
3	7	9	128	4887.5	5757.5	Feasible
4	7	10	128	6332.5	7530	Infeasible
5	7	11	128	8187.5	9872.5	Infeasible
6	7	12	128	11322.5	12942.5	Infeasible
7	7	13	128	15690	17187.5	Infeasible
8	8	8	256	3860	4265	Feasible
9	8	9	256	5165	6072.5	Feasible
10	8	10	256	6837.5	8105	Infeasible
11	8	11	256	8965	10212.5	Infeasible
12	8	12	256	11280	13125	Infeasible
13	8	13	256	15645	16915	Infeasible
14	8	14	256	21107.5	22292.5	Infeasible
15	9	9	512	5280	6387.5	Feasible
16	9	10	512	7045	8667.5	Feasible
17	9	11	512	9415	11037.5	Feasible
18	9	12	512	11967.5	13615	Feasible
19	9	13	512	15210	17235	Infeasible
20	9	14	512	20525	21995	Infeasible
21	9	15	512	27297.5	28772.5	Infeasible
22	10	10	1024	7192.5	9230	Feasible
23	10	11	1024	9825	11862.5	Feasible
24	10	12	1024	12660	14827.5	Infeasible
25	10	13	1024	15697.5	18125	Infeasible
26	10	14	1024	19802.5	22765	Infeasible
27	10	15	1024	26387.5	28895	Infeasible
28	10	16	1024	34407.5	37205	Infeasible
29	11	11	2048	10225	12947.5	Infeasible
30	11	12	2048	13195	16435	Infeasible
31	11	13	2048	16557.5	20255	Infeasible
32	11	14	2048	20252.5	24455	Infeasible
33	11	15	2048	25502.5	30340	Infeasible
34	11	16	2048	33607.5	37915	Infeasible
35	11	17	2048	43022.5	47907.5	Infeasible

6.2. Branch-and-Price Method for SSTPP2

In this section, we present the BNP method for the solution of **SSTPP2**. We address the formulations for the master problem and the pricing problems of **SSTPP2**. Subsequently, we present the use of the dual stabilization technique for the BNP method like we have done for the formulation **SGAP2**. We also introduce a greedy heuristic to obtain a feasible solution.

6.2.1. Decomposition for SSTPP2

Recall that the pricing problems for the formulation **SGAP2** are multiple knapsack problems and the number of pricing problems is equal to the number of machines ($|I|$). In spite of constraint set (6.3d) that does not exist in **SGAP2**, note that the formulation **SSTPP2** is identical in structure with **SGAP2** because (6.3d) as well as (6.3c) are separable in i, $i \in I$. In other words, it is reasonable to include (6.3d) in pricing problems. Then, the number of pricing problems in **SSTPP2** is equal to the number of contractors ($|I|$), which is equivalent to the number of machines, $|I|$, in **SGAP2**. Besides, the number of constraints in each pricing problem for the SGAP is equal to the number of scenarios ($|S|$). The formulation **SSTPP2** consists of $|S|$ machine capacity constraints and a contractor configuration constraint for each contractor. Each pricing problem of **SSTPP2** will have only one more constraint than that in **SGAP2**. Therefore, the formulation of the BNP method for **SSTPP2** consists of the job assignment constraint set (6.3b) as the linking one, and the machine capacity (6.3c) and the contractor configuration (6.3d) constraint sets as the complicating ones.

The formulation below, **BNP-SSTPP**, addresses the formulation of the BNP method for the model SSTPP, and consists of the master problem and the pricing problems, respectively, denoted as **MP-SSTPP** and **PP-SSTPP**. We introduce the following additional notation.

k_i : Order of columns obtained from pricing problem for contractor i, $\forall i \in I$, $k_i = 1,...,K_i$

$\left(x_{ij}\right)^{k_i}$: Value of x_{ij} in the k_i th column obtained from pricing problem for contractor i, $\forall i \in I$, $\forall j \in J(i)$, $k_i = 1,...,K_i$

$\left(y_{is}\right)^{k_i}$: Value of y_{is} in the k_i th column obtained from pricing problem for contractor i, $\forall i \in I$, $\forall s \in S$, $k_i = 1,...,K_i$

$\left(w_i\right)^{k_i}$: Value of w_i in the k_i th column obtained from pricing problem for contractor i, $\forall i \in I$, $k_i = 1,...,K_i$

$\bar{\pi}_j$: Value of dual variables for constraint set (6.5b), $\forall i \in I$, $\forall j \in J(i)$

$\bar{\pi}_i$: Value of dual variables for constraint set (6.5c), $\forall i \in I$

λ^{k_i} : Binary convexity variable for the k_i th column obtained from pricing problem for contractor i; =1 if the k_i th column is chosen in a solution; =0, otherwise $\forall i \in I$, $k_i = 1,...,K_i$

BNP-SSTPP

MP-SSTPP

Minimize $$\sum_{i \in I} \sum_{k=1}^{k_i} \left(h_i\left(w_i^k\right) + \sum_{j \in J(i)} c_{ij}\left(x_{ij}^k\right) + e_i \sum_{s \in S} p_s\left(y_{is}^k\right) \right) \lambda_i^k \qquad (6.5a)$$

Subject to $$\sum_{i \in I(j)} \sum_{k=1}^{k_i} \left(x_{ij}^k\right) \lambda_i^k = 1 \qquad \forall j \in J \qquad \left(\bar{\pi}_j\right) \qquad (6.5b)$$

$$\sum_{k=1}^{k_i} \lambda_i^k \le 1 \qquad \forall i \in I \qquad \left(\bar{\pi}_i\right) \qquad (6.5c)$$

$$\lambda_i^k \in \{0,1\} \qquad \forall i \in I,\ k = 1,...,\mathrm{K_i} \qquad (6.5d)$$

PP-SSTPP for Contractor i, $\forall i \in I$

Minimize $$-\bar{\pi}_i + h_i w_i + \sum_{j \in J(i)} \left(c_{ij} - \bar{\pi}_j\right) x_{ij} + \sum_{s \in S} p_s e_i y_{is} \qquad (6.5e)$$

Subject to $\sum_{j\in J(i)} d_{ijs}x_{ij} - T \le y_{is} \le TP \qquad \forall s \in S$ (6.5f)

$\sum_{j\in J(i)} x_{ij} - Mw_i \le 0$ (6.5g)

$x_{ij} \in \{0,1\} \qquad \forall j \in J(i)$ (6.5h)

$w_i \in \{0,1\}$ (6.5i)

$y_{is} \ge 0 \qquad \forall s \in S$ (6.5j)

Note that not every contractor needs to be hired or assigned to a job. Therefore, the L.H.S. of (6.5c) can be zero.

6.2.2. Dual Stabilization Technique (DST) for SSTPP2

6.2.2.1. Modified SSTPP2 with the DST

The formulation **BNPDST-SSTPP** introduces additional variables in the linking constraints to employ the dual stabilization technique (DST) for the formulation **BNP-SSTPP**.

BNPDST-SSTPP

MPDST-SSTPP

Minimize $\sum_{i\in I}\sum_{k=1}^{k_i}\left(h_i\left(w_i^k\right)+\sum_{j\in J(i)}c_{ij}\left(x_{ij}^k\right)+e_i\sum_{s\in S}p_s\left(y_{is}^k\right)\right)\lambda_i^k+\sum_j\delta_j^{(-)K}z_j^{(-)}+\sum_j\delta_j^{(+)K}z_j^{(+)}$ (6.6a)

Subject to $\sum_{i\in I}\sum_{k}^{k_i}\left(x_{ij}^k\right)\lambda_i^k - z_j^{(-)} + z_j^{(+)} = 1 \qquad \forall j \in J \qquad \left(\bar{\pi}_j\right)$ (6.6b)

$\sum_{k}^{k_i}\lambda_i^k \le 1 \qquad \forall i \in I \qquad \left(\bar{\pi}_i\right)$ (6.6c)

$\lambda_i^k \in \{0,1\} \qquad \forall i \in I,\ \forall k = 1,\ldots,k_i$ (6.6d)

$z_j^{(-)},\ z_j^{(+)} \ge 0 \qquad \forall j \in J$ (6.6e)

Note that **MPDST-SSTPP** is the restricted master problem at the K^{th} iteration, and $k_i \le K$, $\forall i \in I$. Also, $\delta_j^{(-)K}$ and $\delta_j^{(+)K}$, $\forall j \in J$, the coefficients of the $z_j^{(-)}$ and $z_j^{(+)}$ variables in the objective function, are the bound parameters for dual price $\bar{\pi}_j$ at the K^{th} iteration.

Remark 6.1. The dual prices in the relaxed **MPDST-SSTPP**, $\bar{\pi}_j$ and $\bar{\pi}_i$, are restricted by $z_j^{(-)}$, $z_j^{(+)}$, $\delta_j^{(-)K}$, and $\delta_j^{(+)K}$ like in the relaxed **MPDST-SGAP2**.

The dual of the relaxed **MPDST-SSTPP** is as follows:

Dual of MPDST-SSTPP

Maximize $\sum_{j\in J}\pi_j - \sum_{i\in I}\pi_i$ (6.7a)

Subject to $\sum_{j\in J}\left(x_{ij}^k\right)\pi_j - \pi_i \le h_i\left(w_i^k\right) + \sum_{j\in J} c_{ij}\left(x_{ij}^k\right) + e_i \sum_{s\in S} p_s\left(y_{is}^k\right)$ $\forall i \in I$, $\forall k = 1,\ldots,k_i$ $\left(\bar{\lambda}_i^k\right)$ (6.7b)

$-\pi_j \le \delta_j^{(-)K}$ $\forall j \in J$ $\left(\bar{z}_j^{(-)}\right)$ (6.7c)

$\pi_j \le \delta_j^{(+)K}$ $\forall j \in J$ $\left(\bar{z}_j^{(+)}\right)$ (6.7d)

π_j unrestricted $\forall j \in J$ (6.7e)

$\pi_i \ge 0$ $\forall i \in I$ (6.7f)

In view of the constraints (6.7c) and (6.7d), we have the following range for each dual price.

$-\delta_j^{(-)K} \le \bar{\pi}_j \le \delta_j^{(+)K}$ $\forall j \in J$ (6.7g)

Hence, the dual prices for the constraints (6.6b) are limited by the values of $-\delta_j^{(-)K}$ and $\delta_j^{(+)K}$, which are the coefficients of the $z_j^{(-)}$ and $z_j^{(+)}$ variables in the objective function. Furthermore, by constraints (6.7b) and (6.7g), the dual prices for convexity constraints (6.6c) are also restricted as follows:

$$\begin{aligned}\bar{\pi}_i &\geq \sum_j \left(x_{ij}^k\right)\bar{\pi}_j - h_i\left(w_i^k\right) - \sum_j c_{ij}\left(x_{ij}^k\right) - e_i \sum_s p_s\left(y_{is}^k\right) \\ &\geq -\sum_j \left(x_{ij}^k\right)\delta_j^{(-)K} - h_i\left(w_i^k\right) - \sum_j c_{ij}\left(x_{ij}^k\right) - e_i \sum_s p_s\left(y_{is}^k\right).\end{aligned} \tag{6.7h}$$

6.2.2.2. Use of the DST for BNPDST-SSTPP

In Section 3.3.1, we have introduced a procedure to update $\delta_j^{(-)K}$ and $\delta_j^{(+)K}$ based on the values of $\bar{\pi}_j^{K-1}$ and ε^K, the dual prices from the previous iteration and a positive predetermined value for the K^{th} iteration, respectively, as follows:

$$\delta_j^{(-)K} = -\bar{\pi}_j^{K-1} + \varepsilon^K \qquad \forall j \in J \tag{6.8a}$$

$$\delta_j^{(+)K} = \bar{\pi}_j^{K-1} + \varepsilon^K \qquad \forall j \in J \tag{6.8b}$$

With (6.7g), (6.8a), and (6.8b), the dual prices at the K^{th} iteration are restricted as follows:

$$-\bar{\pi}_j^{K-1} + \varepsilon^K \leq \bar{\pi}_j^K \leq \bar{\pi}_j^{K-1} + \varepsilon^K \qquad \forall j \in J\,. \tag{6.8c}$$

where $\bar{\pi}_j^K$ is the dual price from the master problem after the K^{th} iteration.

Recall that, we have developed a warm-up period policy to determine ε^K in Section 3.3.2, and a policy to update $z_j^{(-)}$, $z_j^{(+)}$, $\delta_j^{(-)K}$, and $\delta_j^{(+)K}$ $\forall j \in J$ in Section 3.3.3. These policies were shown to be effective for the SGAP. Therefore, we will employ them to the SSTPP as well.

6.2.3. Branching Variable and Node Selection Strategies for the BNPDST-SSTPP Method

The general procedure for the Branch-and-Price (BNP) method has been explained in previous chapters. Hence, we address additional features that we incorporate in the BNP method for **SSTPP**. The BNP method consists of the branch-and-bound (BNB) and the column generation methods. Our additional features pertain to the BNB method with regard to the branching variable selection and node selection strategies with the aim of reducing the depth of the BNB tree or the number of nodes that are investigated in search for the optimal solution.

6.2.3.1. Setting Variables to 0 or 1

The formulation **SSTPP** has two types of binary variables: x_{ij} and w_i. Next, we investigate the information that we can obtain by setting one of these variables to either one or zero. Suppose that we have fractional values $x_{i'j'}$ and w_i for some i' and j' after solving the problem at the root node. Suppose that they can take values of either of 0 or 1. We examine the following four cases: $x_{i'j'}=1$, $x_{i'j'}=0$, $w_{i'}=1$, and $w_{i'}=0$.

(a) When $x_{i'j'}=1$

Since $\sum_{i\in I(j')} x_{ij'} = 1$, i.e., job j' cannot be assigned to any other contractors, we can fix

$$x_{ij'} = 0 \qquad \forall i \in I(j') \setminus i' \qquad (6.9a)$$

Since the total overtime work of contractor i cannot be greater than *TP*, we can fix

$$x_{i'j} = 0 \qquad \forall j \in \{ j \in J(i') \setminus j' : \mathrm{d}_{\mathrm{i'j's}} + \mathrm{d}_{\mathrm{i'js}} - \mathrm{T} > \mathrm{TP}, \exists \mathrm{s} \in \mathrm{S} \} \qquad (6.9b)$$

(b) When $x_{i'j'}$=0

We cannot fix any other variable, so the number of variables we can set is one.

(c) When $w_{i'}$=1

No other variable can be fixed, but we can simplify (6.5a) and (6.5e), the objective function of the master problem and the pricing problem for contractor i', respectively, as follows:

$$\sum_{k=1}^{k_{i'}}\left(h_{i'}+\sum_{j\in J(i')}c_{i'j}\left(x_{i'j}^{k}\right)+e_{i'}\sum_{s\in S}p_s\left(y_{i's}^{k}\right)\right)\lambda_{i'}^{k}+\sum_{i\in I\setminus i'}\sum_{k=1}^{k_i}\left(h_i w_i^{k}+\sum_{j\in J(i)}c_{ij}\left(x_{ij}^{k}\right)+e_i\sum_{s\in S}p_s\left(y_{is}^{k}\right)\right)\lambda_i^{k}\,, \tag{6.9c}$$

$$-\bar{\pi}_{i'}+h_{i'}+\sum_{j\in J(i')}\left(c_{i'j}-\bar{\pi}_j\right)x_{i'j}+\sum_{s\in S}p_s e_{i'}y_{i's}\,. \tag{6.9d}$$

Furthermore, we can make constraint sets (6.5c) and (6.5g) tighter as follows:

$$\sum_{k=1}^{k_{i'}}\lambda_{i'}^{k}=1 \tag{6.9e}$$

$$\sum_{j\in J(i')}x_{i'j}\geq 1 \tag{6.9f}$$

Constraint (6.9e) becomes tight since contractor i' is employed. Constraint (6.9f) asserts that at least one job is assigned to contractor i'.

(d) When $w_{i'}$=0

It means that contractor i' is not hired, i.e., no job is assigned to contractor i', and the following variables can be fixed:

$$x_{i'j}=0 \qquad \forall j\in J(i') \tag{6.9g}$$

$$y_{i's}=0 \qquad \forall s\in S \tag{6.9h}$$

Note that setting $w_{i'}$ to zero allows us to fix the maximum number of variables as long as $|S| \geq |I(j')|$. Also, note that when $w_{i'}$=0, we can remove the following |*S*| constraints:

$$\sum_{j \in J(i)} d_{i'js} x_{i'j} - T \leq y_{i's} \leq TP \qquad \forall s \in S \tag{6.9i}$$

Furthermore, the objective function value of the pricing problem for contractor i' will be zero, and eventually, we can remove both the pricing problem for contractor i' and the convexity variable for contractor i' in the master problem.

6.2.3.2. Branching Variable Selection Strategy

Let x_{ij}^* and w_i^*, $\forall i \in I$, $\forall j \in J(i)$, represent the solution at the current node. Let $\tilde{X}$ and $\tilde{W}$ denote the sets of x_{ij} and w_i with fractional values, respectively, and we can define them as follows:

$$\tilde{X} = \{x_{ij}, \forall i \in I, \forall j \in J(i) : 0 < x_{ij}^* < 1\} \tag{6.10a}$$

$$\tilde{W} = \{w_i, \forall i \in I : 0 < w_i^* < 1\} \tag{6.10b}$$

Combining the most infeasibility strategy and the strategy described in Section 6.2.3.1, we implement a new branching variable selection strategy for the **SSTPP**. It is similar to the branching variable selection strategy described in Section 4.4.1.3.

Step 1. If $\tilde{W} \neq \phi$, go to step 3; otherwise, go to step 2.

Step 2. If $\tilde{X} \neq \phi$, go to step 4; otherwise, fathom the node since the node yields an integer feasible solution.

Step 3. Find $w_i \in \tilde{W}$ such that we can set the most number of new variables to either one or zero by setting it to zero on the basis of the history of the path that the current node belongs to in its branch-and-bound tree. If there is a tie, go to step 5 with tied

variables; otherwise, choose it as the branching variable for the current node, and stop the branching variable selection process.

Step 4. Find $x_{ij} \in \tilde{X}$ for which we can set the most number of new variables to either one or zero by setting it to one on a basis of the history of the path that the current node belongs to in its branch-and-bound tree. If there is a tie, go to step 5 with tied variables; otherwise, choose it as the branching variable for the current node, and stop the branching variable selection process.

Step 5. Find a variable with the most infeasibility among the variables transferred from either step 3 or step 4 and choose it as the branching variable for the current node. If there is a tie, choose a variable arbitrarily. Stop branching variable selection process.

6.2.3.3. Node Selection Strategy

As mentioned in Section 6.3.1.1, setting w_i to zero allows us to ignore the pricing problem and the convexity for contractor *i*. That is, it leads to much less computational effort. Therefore, another measure that we employ for a node selection strategy is the number of variables already fixed before solving the current node. Let $\hat{x}_{ij}^0$ and $\hat{x}_{ij}^1$ represent variables x_{ij}, $\forall i \in I$, $\forall j \in J(i)$, fixed to zero or one before entering the current node, respectively. Similarly, let $\hat{w}_i^0$, and $\hat{w}_i^1$, $i \in I$, represent variables, w_i, fixed to zero or one before entering the current node, respectively. Furthermore, let $\hat{X}^0$, $\hat{X}^1$, $\hat{W}^0$, and $\hat{W}^1$ be the sets of $\hat{x}_{ij}^0$, $\hat{x}_{ij}^1$, $\hat{w}_i^0$, and $\hat{w}_i^1$, respectively, and we can define them as follows:

$$\hat{X}^0 = \{x_{ij}, \forall i \in I, \forall j \in J(i) : x_{ij} \text{ that are fixed to zero in the path of the current node}\} \quad (6.11a)$$

$$\hat{X}^1 = \{x_{ij}, \forall i \in I, \forall j \in J(i) : x_{ij} \text{ that are fixed to one in the path of the current node}\} \quad (6.11b)$$

$$\hat{W}^0 = \{w_i, \forall i \in I : w_i \text{ that are fixed to zero in the path of the current node}\} \quad (6.11c)$$

$$\hat{W}^1 = \{w_i, \forall i \in I : w_i \text{ that are fixed to one in the path of the current node}\} \quad (6.11d)$$

We can implement the node selection strategy for the **SSTPP** as follows.

Step 1. At each remaining node, count the number of elements in $\hat{W}^0$. Make a list of the nodes with the largest $\left|\hat{W}^0\right|$, say, $L(W^0)$. If $L(W^0)$ has just one node in the list, choose the node as the next node to investigate, and finish the node selection process. Otherwise, if $L(W^0)$ contains more than one node in the list, pass only the nodes in $L(W^0)$ to step 2, or if $L(W^0)$ contains no node, then let $L(W^0)$ contain all remaining nodes, and pass it to step 2.

Step 2. Among the nodes in $L(W^0)$, make a list of the nodes with the largest $\left|\hat{X}^1\right|$, say, $L(X^1)$. If $L(X^1)$ includes just one node in the list, choose the node as the next node to investigate, and finish the node selection process. Otherwise, if $L(X^1)$ contains more than one node in the list, pass only the nodes in $L(X^1)$ to step 3, or if $L(X^1)$ contains no node, let $L(X^1)$ equal to $L(W^0)$, and pass it to step 3.

Step 3. Among the nodes in $L(X^1)$, make a list of the nodes with the largest $\left|\hat{W}^1\right|$, say, $L(W^1)$. If $L(W^1)$ includes just one node in the list, choose the node as the next node to investigate, and finish the node selection process. Otherwise, if $L(W^1)$ contains more than one node in the list, pass only the nodes in $L(W^1)$ to step 4, or if $L(W^1)$ contains no node, let $L(W^1)$ equal to $L(X^1)$, and pass it to step 4.

Step 4. Among the nodes in $L(W^1)$, make the list for the nodes with the largest $\left|\hat{X}^0\right|$, say, $L(X^0)$. If $L(X^0)$ includes just one node in the list, choose the node as the next node to investigate, and finish the node selection process. Otherwise, if $L(X^0)$ contains more than a node in the list, break ties arbitrarily.

In step 4, $L(X^0)$ cannot be empty because all children nodes should have at least one fixed variable. Therefore, with this node selection strategy, we always have exactly one node for the next iteration as long as the remaining nodes exist.

6.2.4. A Greedy Heuristic for the SSTPP

We can use a greedy heuristic for the formulation **SSTPP2** to obtain an initial feasible solution at each node, like for the formulation SGAP2 in Section 3.4.2. it is as follows:

Step 1. Set $\hat{I} = \{i, \forall i \in I\}$, $I \otimes J = \{(i, j), \forall i \in I, \forall j \in J(i)\}$, and $\overline{T}_{is} = T$, $\overline{TP}_{is} = TP$, $\forall i \in I$, $\forall s \in S$. Go to step 2.

Step 2. Set w_i and x_{ij} to zero or one based on the node information. Adjust $I \otimes J$, $\hat{I}$, and $\overline{T}_{is}$ $\forall i \in I$, $\forall s \in S$ as follows: If x_{ij} is set to one, remove i from $\hat{I}$ and (i', j) $\forall i' \in I(j)$ from all $I \otimes J$. Let $\xi_{ij} = \{(i, j) \mid x_{ij} = 1\}$. Then, let $\overline{T}_{is} = \text{Max}(0, T - \sum_{(i,j') \in \xi_{ij}} d_{ij's})$, and $\overline{TP}_{is} = Max\{0, \sum_{(i,j') \in \xi_{ij}} d_{ij's} - T\}$, $\forall i \in I$, $\forall s \in S$. Go to step 3.

Step 3. For all $i \in I$, if $i \in \hat{I}$, set $\bar{c}_{ij} = h_i + c_{ij}$; otherwise, set $\bar{c}_{ij} = c_{ij}$. Go to step 4.

Step 4. Calculate

$$R_{ij} = \left\{ \frac{\bar{c}_{ij}}{\sum_{s \in S} p_s d_{ijs}} + \frac{\sum_{s \in S} p_s e_i \text{Max}(0, \overline{TP}_{is} + d_{ijs} - \overline{T}_{is})}{\sum_{s \in S} p_s d_{ijs}} \right\}. \tag{6.12a}$$

Find $(i, j) \in I \otimes J$ which has the least value of R_{ij}. Go to step 5.

Step 5. Break ties in the following order of preference:

i. increasing order of $\dfrac{\sum_{s \in S} p_s e_i \text{Max}(0, \overline{TP}_{is} + d_{ijs} - \overline{T}_{is})}{\sum_{s \in S} p_s d_{ijs}}$, (6.12b)

ii. increasing order of $\sum_{s \in S} p_s e_i \text{Max}(0, \overline{TP}_{is} + d_{ijs} - \overline{T}_{is})$, (6.12c)

iii. increasing order of e_i,

iv. increasing order of $\text{Max}(0, \overline{TP}_{is} + d_{ijs} - \overline{T}_{is})$, (6.12d)

v. increasing order of $\dfrac{\bar{c}_{ij}}{\sum_{s\in S} p_s d_{ijs}}$, (6.12e)

vi. increasing order of $\bar{c}_{ij}$, and

vii. randomly.

Go to step 6.

Step 6. Set $x_{ij}=1$, and $x_{i'j}=0 \quad \forall i'\in I\setminus i$. Set $\overline{TP}_{is}=Max\{0,\overline{TP}_{is}+d_{ijs}-\bar{T}_{is}\}$ and $\bar{T}_{is}=\mathrm{Max}(0,\bar{T}_{is}-d_{ijs})$. and remove all the elements related to job j from the set $I\otimes J$. If $i\in\hat{I}$, remove i from $\hat{I}$, and set $w_i=1$. Go to step 7.

Step 7. If $I\otimes J=\phi$, go to step 8; otherwise, go to step 3.

Step 8. Find the values of y_{is}, $\forall i\in I$, $\forall s\in S$ based on the values of x_{ij}, $\forall i\in I$, $\forall j\in J$ that are fixed in step 1 through step 6.

Note that in the above heuristic, we have used the following notation.

$\hat{I}$: Set of contractors unassigned to any job, i.e., contractors who are not hired yet. Therefore, if $x_{ij}=1$ for any j, $j\in J(i)$ and $i\in\hat{I}$, i should be removed from $\hat{I}$. Note that in this case, the cost for hiring contractor i, h_i, is incurred, but if $i\notin\hat{I}$, i.e., if i must have been already removed from $\hat{I}$, h_i is not incurred again since contractor i is already hired. Step 3 implies this with $\bar{c}_{ij}$.

$\bar{T}_{is}$: Available work hours for contractor i during regular duration under scenario s

$\overline{TP}_{is}$: Available work hours from contractor i during overtime under scenario s. Note that it should always be nonnegative so as to keep the problem feasible.

This greedy heuristic is associated with the column generation method in the BNP method for **SSTPP**, and we expect it to help reduce the computational effort to find the solution at each

node. In the meantime, this greedy heuristic is guaranteed to find an initial feasible solution at each node in the branch-and-bound tree as we show next.

Proposition 6.3. If the assignments at the preceding nodes of a node provide a feasible solution so far, then the above greedy heuristic guarantees a feasible solution to the formulation **SSTPP2** at that node, and hence, its objective function value provides an upper bound for **SSTPP2**.

Proof. Assume that the heuristic ends up with an infeasible solution. It implies that the solution violates at least one of the constraints in **SSTPP2**.

Note that steps 2 and 6 assert that no job is assigned to more than one contractor, and step 7 guarantees that no job is left unassigned. Therefore, job assignment constraint set (6.3b) is satisfied. Moreover, the contractor configuration constraint set (6.3d) is satisfied at step 2 and step 6, which makes sure that $w_i = 1$ if $x_{ij} = 1$ for any j, $j \in J(i)$.

For a given solution $\bar{x}_{ij}$, $\forall i \in I$, $\forall j \in J(i)$, the corresponding values of y_{is}, denoted by $\bar{y}_{is}$, $\forall i \in I$, $\forall s \in S$, can be obtained while satisfying overtime allocation constraint set (6.3c) by

$$\bar{y}_{is} = \text{Max}(0, \sum_{j \in J} d_{ijs} \bar{x}_{ij} - T) \qquad \forall i \in I,\ \forall s \in S. \tag{6.12f}$$

Moreover, as long as $\sum_{j \in J} d_{ijs} \bar{x}_{ij} - T \le TP$, then (6.3c) will be satisfied. In step 4, we only allow a pair of contractor i and job j for which $\overline{TP}_{is} - d_{ijs} > 0$, $\forall s \in S$, which implies that the pair will not be allowed if it violates the maximum overtime constraint. Therefore, (6.3c) is not violated by the solution.

The fact that all three constraint sets are satisfied contradicts the assumption. Hence, a solution obtained by the greedy heuristic is always feasible to **SSTPP2**, and, therefore, its objective function value provides an upper bound on the optimal value of **SSTPP2**. □

6.3. Branch-and-Price Method for the SSTPP

Figure 6.1 shows the BNP method for the formulation **SSTPP2**. It consists of the branch-and-bound method, the column generation method (Section 6.2.1), the dual stabilization technique (Section 6.2.2), the branching variable selection strategy (Section 6.3.1.2), the node selection strategy (Section 6.3.1.3), and the greedy heuristic (Section 6.3.2). The steps belonging to each of these parts of the algorithm are shown in yellow, blue, brown, red, green, and purple colors.

First, we solve the relaxed **SSTPP2** at each node (step 1). Second, we find an initial feasible solution using the greedy heuristic (step 3), instead of the Phase II method. With a feasible solution, we build the first master problem for which the DST is also implemented (step 3 and step 4). The dual prices obtained from the master problem are transferred to the pricing problems (step 4). The DST parameters are updated (step 5). We solve the pricing problems (step 6). We repeat this process until there is no valid column available from pricing problems (step 7). If the solution obtained from the column generation process is feasible to the relaxed **SSTPP2**, i.e., all the $z_j^{(-)}$ and $z_j^{(+)}$ variables for the DST become zero (step 8), then the branch-and-bound process is initiated. We evaluate the integrality of the solution (step 11), the value of the objective value for the lower bound (step 10) and the upper bound (step 13), and the remaining nodes (step 17). If there is still at least one node left to be searched, we select a new branching variable and the next node to search by our branching variable (step 18) and node selection (step 19) strategies, respectively. We iterate the entire process until all the nodes are pruned.

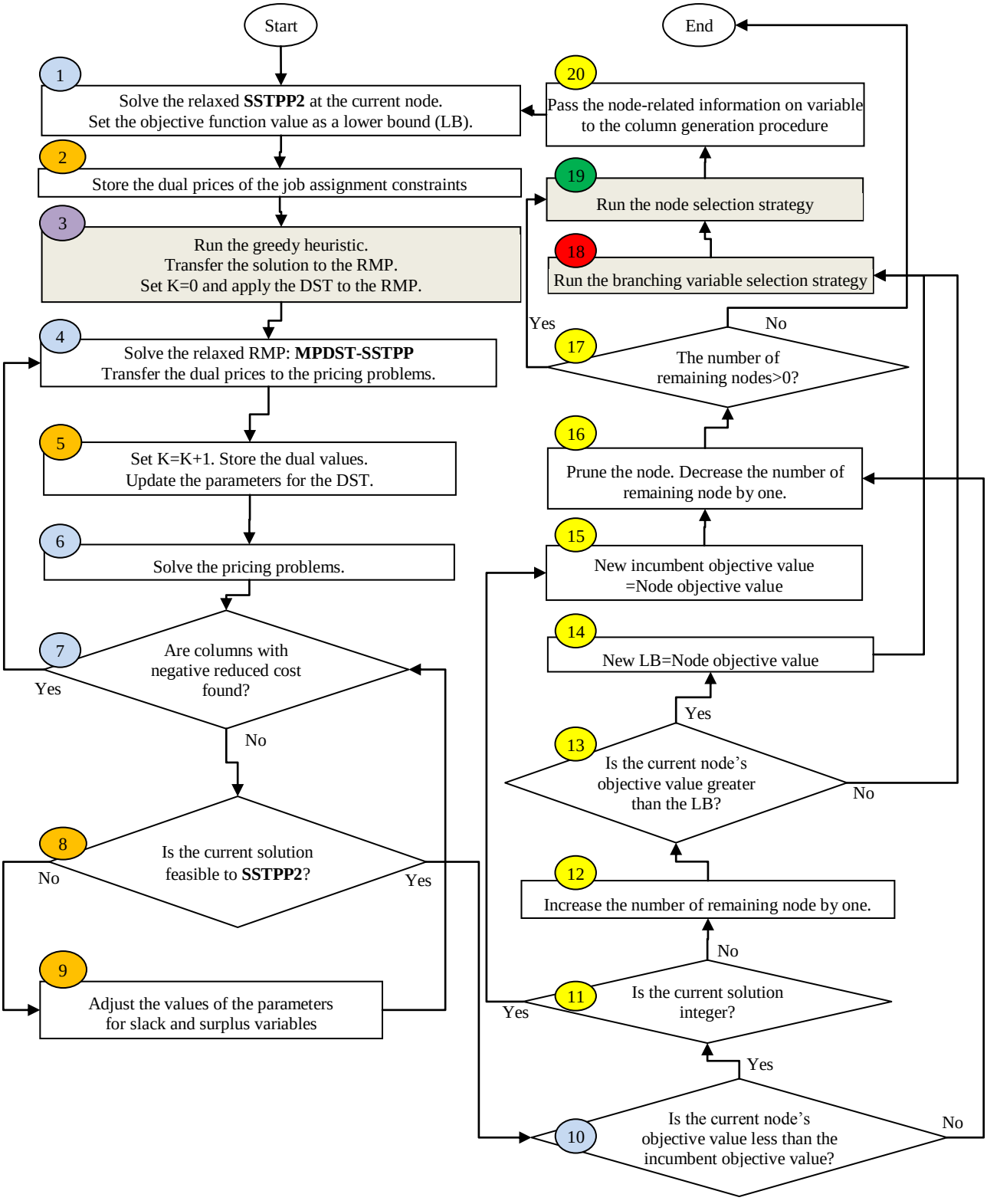

Figure 6.1. BNP Method for the SSTPP with Dual Stabilization Technique, Greedy Heuristic, and the Branching Variable and Node Selection Strategies

6.4. Experimental Data

The complexity of the SSTPP is associated with the number of a candidate pool of contractors ($|I|$), the number of jobs ($|J|$) being planned to assign to future contractors, and the number of jobs that can be performed by a contractor as they determine the number of scenarios. In our experimentation, we vary values of these three factors.

We also vary values of the parameters presented in the formulation **SSTPP2**. These include overtime cost that a company has to pay to a contractor, the set of jobs that each contractor is able to perform, and the stochastic processing time, while for every possible contractor we set the length of regular work duration (T) to 8 and that of maximum overtime (TP) to 4.

We conduct experimentation by using the branch-and-cut (BNC) method and the branch-and-price (BNP) method described in Section 6.3 and denote as the BNC-SSTPP and BNP-SSTPP methods, respectively. We compare performances of these methods based on the cpu time required as well as the ability of each method to solve the problem in view of memory requirements. We vary and observe the following:

- Number of contractors
- Number of jobs
- Overtime cost for each contractor
- Problems: Number of variables and Number of constraints
- Comparative performances of the BNC and the BNP methods
- Efficiency of each method

Table 6.2 depicts the data that are used for our experimentation. The number of contractors ($|I|$) and the number of jobs ($|J|$) are changed from 7 to 11 and from 7 to 17, respectively. Consequently, the number of scenarios ($|I|$) varies from 128 to 2048, while the number of variables and the number of constraints vary from 952 to 22,726 and from 910 to 22,556, respectively.

Furthermore, the coefficients in the formulation **SSTPP2** are determined as follows:

Table 6.2. Experimental Data for the SSTPP

Experiment #	$\|I\|$	$\|J\|$	$\|S\|$	Number of Variables	Number of Constraints
1	7	7	128	952	910
2	7	8	128	959	911
3	7	9	128	966	912
4	7	10	128	973	913
5	7	11	128	980	914
6	7	12	128	987	915
7	7	13	128	994	916
8	8	8	256	2,120	2,064
9	8	9	256	2,128	2,065
10	8	10	256	2,136	2,066
11	8	11	256	2,144	2,067
12	8	12	256	2,152	2,068
13	8	13	256	2,160	2,069
14	8	14	256	2,168	2,070
15	9	9	512	4,698	4,626
16	9	10	512	4,707	4,627
17	9	11	512	4,716	4,628
18	9	12	512	4,725	4,629
19	9	13	512	4,734	4,630
20	9	14	512	4,743	4,631
21	9	15	512	4,752	4,632
22	10	10	1024	10,350	10,260
23	10	11	1024	10,360	10,261
24	10	12	1024	10,370	10,262
25	10	13	1024	10,380	10,263
26	10	14	1024	10,390	10,264
27	10	15	1024	10,400	10,265
28	10	16	1024	10,410	10,266
29	11	11	2048	22,660	22,550
30	11	12	2048	22,671	22,551
31	11	13	2048	22,682	22,552
32	11	14	2048	22,693	22,553
33	11	15	2048	22,704	22,554
34	11	16	2048	22,715	22,555
35	11	17	2048	22,726	22,556

- Hiring cost (h_i) and overtime cost (e_i) for contractor i

  $h_i = 200\ (i + 1)$ $\forall i \in I$, and

  $e_i = 100\ (|I| - i)$ $\forall i \in I$.

- Processing cost incurred by contractor i for job j

  $c_{ij} = 10\ (|I| + |J| - i - j)$ $\forall i \in I$.

- Processing time of the resource required by contractor i to complete job j

  $d_{ij} = 0.5\ (i + j + 2)$ $\forall i \in I, \forall j \in J$.

6.5. Results and Analysis

As we have done for the SGAP and the HSSP, we compared the performances of the BNC-based and BNP-based methods for the SSTPP, which are designated as **BNC-SSTPP** and **BNP-SSTPP**, respectively. Table 6.3 depicts the results for both of these methods. The 4th column contains the optimal objective function value, the 5th and the 6th columns present the cpu times for **BNC-SSTPP** and **BNP-SSTPP**, respectively.

As shown in the table, the BNP method performs betters than the BNC method in 26 out of 35 experiments (a method is highlighted when its cpu time is smaller). In addition, the range of the cpu times for the BNC method is much larger than that for the BNP method. Therefore, **BNP-SSTPP** is more stable than **BNC-SSTPP**.

Figure 6.2 displays a plot of the ratio of the cpu time of **BNC-SSTPP** to that of **BNP-SSTPP**, presented in the 5th and 6th columns, respectively. The number on each dot in the graph represents this ratio. On average, **BNP-SSTPP** results in over four-fold savings in cpu time over **BNC-SSTPP**. Another observation that is worth noting is that for harder problems, the difference in performance between the two methods gets bigger. When $|I|$=7, 8, and 9, **BNP-SSTPP** performs better than **BNC-SSTPP** by 40% on average. However, **BNP-SSTPP** performs better than **BNC-SSTPP** by almost 10 folds on average when $|I|$=10 and 11.

Table 6.3. Results for **BNC-SSTPP** and **BNP-SSTPP**

Experiment #	$\|I\|$	$\|J\|$	$\|S\|$	Optimal Objective Function Value	CPU Time (Seconds)	
					BNC-SSTPP	**BNP-SSTPP**
1	7	7	128	2862.5	2.5811	1.9690
2	7	8	128	4122.5	1.2405	1.9813
3	7	9	128	5757.5	2.0561	2.7052
4	7	10	128	7530	2.7361	1.8243
5	7	11	128	9872.5	2.0041	2.3879
6	7	12	128	12942.5	4.0937	3.1722
7	7	13	128	17187.5	1.2025	3.5744
8	8	8	256	4265	6.2796	4.9787
9	8	9	256	6072.5	9.7256	5.8760
10	8	10	256	8105	5.7588	7.4397
11	8	11	256	10212.5	5.9219	8.1594
12	8	12	256	13125	4.9850	2.2787
13	8	13	256	16915	9.7795	8.5927
14	8	14	256	22292.5	3.5293	1.9395
15	9	9	512	6387.5	16.0024	17.3358
16	9	10	512	8667.5	30.8635	21.1302
17	9	11	512	11037.5	30.1091	25.3661
18	9	12	512	13615	40.3567	24.9230
19	9	13	512	17235	17.2265	10.1000
20	9	14	512	21995	55.1152	26.0897
21	9	15	512	28772.5	21.1440	6.0177
22	10	10	1024	9230	1768.2975	98.4007
23	10	11	1024	11862.5	1709.1536	117.0240
24	10	12	1024	14827.5	842.7867	98.0997
25	10	13	1024	18125	238.4778	81.8800
26	10	14	1024	22765	69.9499	70.7926
27	10	15	1024	28895	451.9913	98.6401
28	10	16	1024	37205	85.6693	24.8822
29	11	11	2048	12947.5	11984.9307	822.9282
30	11	12	2048	16435	22664.9475	708.1734
31	11	13	2048	20255	13989.6994	661.0506
32	11	14	2048	24455	2248.3723	602.4312
33	11	15	2048	30340	489.0593	562.3288
34	11	16	2048	37915	1782.0897	377.9956

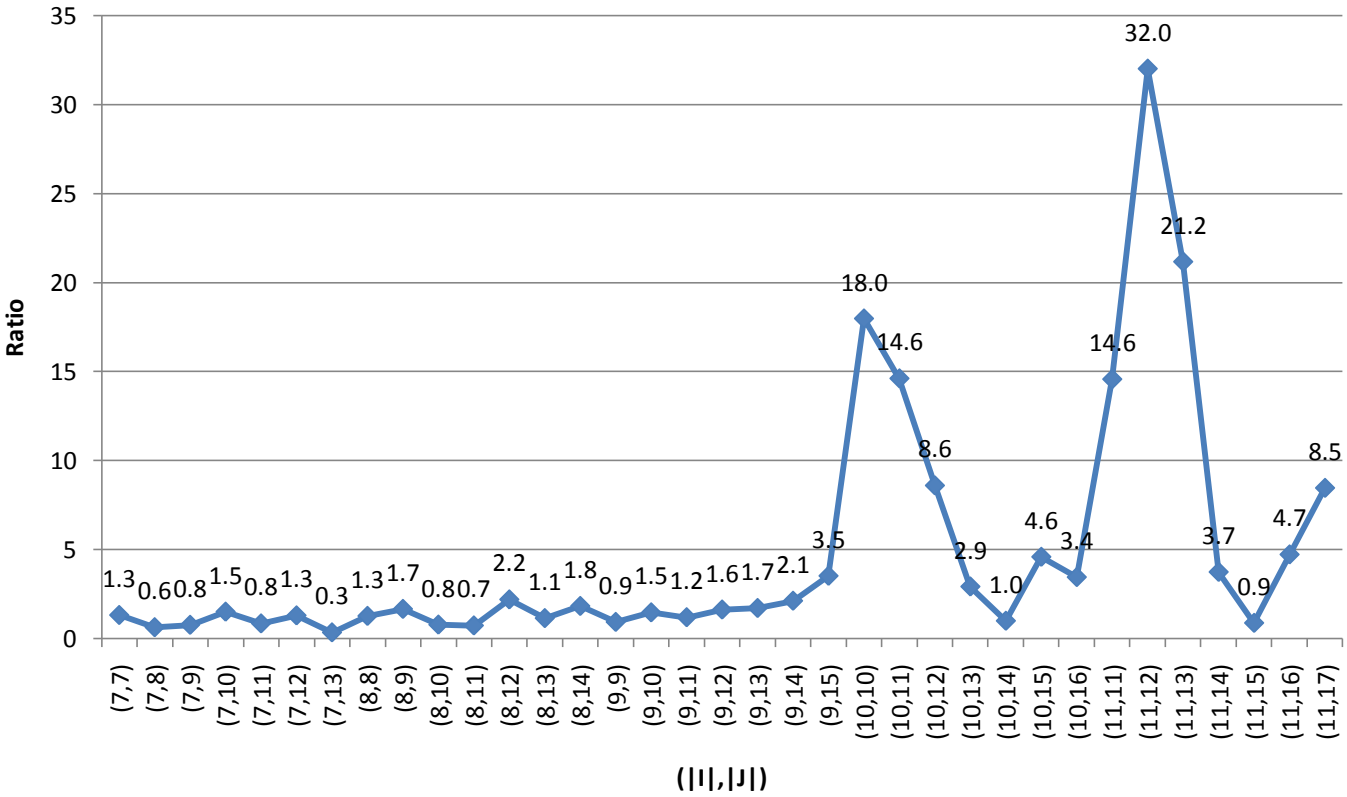

Figure 6.2. Ratios of CPU Times for **BNC-SSTPP** and **BNP-SSTPP**

6.6. Concluding Remarks

In this chapter, we have addressed the stochastic short-term personnel planning problem (SSTPP). We have introduced and developed the BNC- and BNP-based methods for the SSTPP, designated as **BNC-SSTPP** and **BNP-SSTPP**, respectively. We have applied the dual stabilization technique, the branching variable and node selection strategies, and the greedy heuristic to **BNP-SSTPP**. We have compared the performances of these two methods by the cpu times required. **BNP-SSTPP** performs better than **BNC-SSTPP** in almost 75% of the experiments, and **BNP-SSTPP** is much more stable because of lower variability in cpu times than **BNC-SSTPP**. On average, it is able to solve harder problems in much smaller cpu times than that required by **BNC-SSTPP**. Therefore, we can say that the BNP method is a better approach for **SSTPP2** than the BNC method.

Chapter VII. Concluding Remarks and Future Research

In this dissertation, we have investigated the use of the branch-and-price (BNP) method for the solution of the stochastic generalized assignment problem (SGAP), the hospital staff scheduling problem (HSSP), the stochastic HSSP (SHSSP), and the stochastic short-term personnel planning problem (SSTPP). We have developed the BNP method as an exact method for the solution of these problems. For each stochastic problem, we have introduced a recourse model, which we have proven to be a complete recourse model and a relatively complete recourse model for the SGAP and the SHSSP, respectively, and for the SSTPP, it is neither of them. We have developed the dual stabilization technique (DST) and greedy heuristic (GH) for the SGAP and the SSTPP to speed up their solution by the BNP method. We have developed the MCM for the SGAP and the SHSSP to reduce the number of scenarios used for their solutions, and hence, enhance efficiency of the solution method. We have also developed effective branching variable selection and node selection strategies for the SGAP and the SSTPP. Our computational experimentation has revealed the superiority of the BNP method over the BNC method for each of these problems. Also, we have shown that both the DST and the GH make a positive impact on the solution of the SGAP.

The first problem that we address is the SGAP for which the processing time of a job on a machine is assumed to be stochastic. Even though the generalized assignment problem (GAP) has been solved using the BNP method, yet no study has been reported in the literature on the use of the BNP method for the solution of the SGAP. Our work has been motivated by the desire to fill this gap.

We begin by showing that it is better to solve the SGAP as a stochastic program in contrast to solving it by using the expected values of the times required to process the jobs on the machines. Then, we show that the stochastic model of the SGAP is a complete recourse model – a useful property which permits the first stage decisions to produce feasible solutions for the recourse problems. We develop three BNP-based methods for the solution of the SGAP. The first of these is **BNP-SGAP**, which is a combination of branch-and-bound and column generation methods. The pricing problem of **BNP-SGAP** is separable with regard to each machine, and it is

a multiple-constraint knapsack problem. The second method is **BNP-SGAP** implemented in concert with the dual stabilization technique (DST), and it is designated as **BNPDST-SGAP**. We have introduced a new DST by modifying the Boxstep method of Pigatti et al. [76]. We have shown that our method performs better than the method of Pigatti et al. [76] resulting in over two-fold savings in cpu times on average. The third method that we develop for the solution of the SGAP is **BNPDST-SGAP** implemented with an advanced start to obtain an initial feasible solution. We use a greedy heuristic to obtain this solution, and this heuristic is a modification of a similar method used for the knapsack problem. It relies on the information available at a node of the underlying branch-and-bound tree. We have shown that this procedure obtains an initial feasible solution, if it exists at that node. We designate this method as **BNPDSTKP-SGAP**. We have also developed a BNC method to solve the SGAP using CPLEX 9.0. We have compared the performances of the BNP and BNC methods on various problem instances obtained by varying the number of machines, the ratio of the number of machines to the number of jobs, the machine capacity, and the penalty cost per unit of extra resource required at each machine. Our results show that all BNP-based methods perform better than the BNC method, with the best performance obtained for **BNPDSTKP-SGAP**.

An issue with the use of the scenario-based methods that we have employed for the solution of the SGAP is that the number of scenarios generally grows exponentially in problem parameters, which gives rise to a large-size problem. To overcome the complexity caused by the presence of a large number of scenarios for the solution of the SGAP, we introduce the use of the Monte Carlo method (MCM) within the BNP scheme. We designate this method as **BNPDSTKP-SGAP with MCM**. It affords the use of a small subset of scenarios at a time to estimate the "true" optimal objective function value. Replications of the subsets of scenarios are carried out until the objective function value satisfies a stopping criterion. We have established theoretical results for the use of the MCM. These pertain to determining unbiased estimates of: (i) lower and upper bounds of the "true" optimal objective function value, (ii) the "true" optimal solution, and (iii) the optimality gap. We have also provided the $100(1-\alpha)$ confidence interval on the optimality gap. Our experimental investigation has shown the efficacy of using this method. It obtains almost optimal solutions, with the objective function value lying within 5% of the "true" optimal objective function value, while giving almost ten-fold savings in cpu time. Our experimentation has also revealed that an increment in the number of scenarios in each

replication makes a greater impact on the quality of the solution obtained than an increment in the number of replications. We have also observed the impact of a change in the variance of a processing time distribution on cpu time. As expected, the optimal objective function value increases with increment in processing time variability. Also, by comparing the results with the expected value solution, it is observed that the greater the variability in the data, the better it is to use the stochastic program.

The second problem that we study is the hospital staff scheduling problem. We address the following three versions of this problem: HSSP (General): Implementation of schedule incorporating the four principal elements, namely, surgeons, operations, operating rooms, and operation times; HSSP (Priority): Inclusion of priority for some surgeons over the other surgeons regarding the use of the facility in HSSP (General); HSSP (Pre-arranged): Implementation of a completely pre-fixed schedule for some surgeons. The consideration of priority among the surgeons mimics the reality. Our BNP method for the solution of these problems is similar to that for the SGAP except for the following: (i) a feasible solution at a node is obtained with no additional assignment, i.e., it consists of the assignments made in the preceding nodes of that node in the branch-and-bound tree; (ii) the columns with positive reduced cost are candidates for augmentation in the CGM; and (iii) a new branching variable selection strategy is introduced, which selects a fractional variable as a branching variable by fixing a value of which we enforce the largest number of variables to either 0 or 1. The priority problem is separable in surgeons.

The results of our experimentation have shown the efficacy of using the BNP-based method for the solution of each HSSP as it takes advantage of the inherent structure of each of these problems. We have also compared their performances with that of the BNC method developed using CPLEX. For the formulations **HSSP (General)**, **HSSP (Priority)**, and **HSSP (Pre-arranged)**, the BNP method gives better results for 22 out of 30, 29 out of 34, and 20 out 32 experiments over the BNC method, respectively. Furthermore, while the BNC method fails to obtain an optimal solution for 15 experiments, the BNP method obtains optimal solutions for all 96 experiments conducted. Thus, the BNP method consistently outperforms the BNC method for all of these problems.

The third problem that we have investigated in this study is the stochastic version of the HSSP, designated as the Stochastic HSSP (SHSSP), in which the operation times are assumed to be stochastic. We have introduced a formulation for this formulation, designated as **SHSSP2**

(General), which allows for overlapping of schedules for surgeons and operating rooms, and also, allows for an assignment of a surgeon to perform an operation that takes less than a pre-arranged operation time, but all incurring appropriate penalty costs. A comparison of the solution of **SHSSP2 (General)** and its value with those obtained by using expected values (the corresponding problem is designated as **Expected-SHSSP2 (General)**) reveals that **Expected-SHSSP2 (General)** may end up with inferior and infeasible schedules. We show that the recourse model for **SHSSP2 (General)** is a relatively complete recourse model. Consequently, we use the Monte Carlo method (MCM) to reduce the complexity of solving **SHSSP2 (General)** by considering fewer scenarios. We employ the branch-and-cut (BNC) method in concert with the MCM for solving **SHSSP2 (General)**. The solution obtained is evaluated using tolerance ratio, closeness to optimality, length of confidence interval, and cpu time. The MCM substantially reduces computational effort while producing almost optimal solutions and small confidence intervals.

We have also considered a special case of **SHSSP2 (General)**, which considers no overlapping schedules for surgeons and operating rooms and assigns exactly the same operation time for each assignment under each scenario, and designate it as **SHSSP2 (Special)**. With this, we consider another formulation that relies on the longest operation time among all scenarios for each assignment of a surgeon to an operation in order to avoid scheduling conflicts, and we designate this problem as **SHSSP (Longest)**. We show **SHSSP (Longest)** to be equivalent to deterministic HSSP, designated as **HSSP (Equivalent)**, and we further prove it to be equivalent to **SHSSP (General)** in terms of the optimal objective function value and the optimal assignments of operations to surgeons. The schedule produced by **HSSP (Equivalent)** does not allow any overlap among the operations performed in an operating room. That is, a new operation cannot be performed if a previous operation scheduled in that room takes longer than expected. However, the schedule generated by **HSSP (Equivalent)** may turn out to be a conservative one, and may end up with voids due to unused resources in case an operation in an operating room is completed earlier than the longest time allowed. Nevertheless, the schedule is still a feasible one. In such a case, the schedule can be left-shifted, if possible, because the scenarios are now revealed. Moreover, such voids could be used to perform other procedures (e.g., emergency operations) that have not been considered within the scope of the SHSSP

addressed here. Besides, such a schedule can provide useful guidelines to plan for resources ahead of time.

The fourth problem that we have addressed in this dissertation is the stochastic short-term personnel planning problem, designated as Stochastic STPP (SSTPP). This problem arises due to the need for finding appropriate temporary contractors (workers) to perform requisite jobs. We incorporate uncertainty in processing time or amount of resource required by a contractor to perform a job. Contrary to the SGAP, the recourse model for this problem is not a relatively complete recourse model. As a result, we cannot employ a MCM method for the solution of this problem as it may give rise to an infeasible solution. The BNP method for the SSTPP employs the DST and the advanced start procedure developed for the SGAP, and due to extra constraints and presence of binary decision variables, we use the branching variable selection strategy developed for the HSSP models. Because of the distinctive properties of the SSTPP, we have introduced a new node selection strategy. We have compared the performances of the BNC-based and BNP-based methods in cpu time. The BNP method outperforms the BNC method in 75% of the experiments conducted, and the BNP method shows a strong stability with much smaller variance in cpu times than those for the BNC method. It affords solution of difficult problems within smaller cpu times than those required for the BNC method.

In this dissertation, we have demonstrated the effectiveness of using both the branch-and-price and Monte Carlo methods to solve some combinatorial optimization problems. These can further be exploited to solve other problems in scheduling, production planning, and set covering.

Appendix

A. Tables of Experimental Designs for BNC-SGAP, BNP-SGAP, BNPDST-SGAP, and BNPDSTKP-SGAP

Tables A.1, A.2, A.3, A.4, and A.5 illustrate the examples of the experimental designs. Note that the processing time of a job on a machine and the cost for each assignment does not change for all the experiments. In other words, d_{ij} and c_{ij} are the same in all experiments. For example, d_{ij} is fixed to 10, and c_{ij} is fixed to 10 in every experiment.

Table A.1 describes the example of the experimental designs where the ratio of the number of jobs to the number of machines ($|J|/|I|$) is equal to 1, the penalty per extra hour at machine 1 (e_1) in the fourth column is 10, the capacity of machine 1 (b_1) in the fifth column increases by 3 starting from 3. We increase the number of machines by one until we have the same objective values. Tables A.2 and A.3 are different from Table A.1 in e_1.

Table A.2 is used to compare with Tables A.4 and A.5, which are different in the ratio of the number of jobs to the number of machines. In Tables A.2, A.4, and A.5, the ratio is equal to 1, 1.5, and 2, respectively. Note that in Table A.4 we round up the value of the number of jobs after we multiply the number of machines by 1.5. For example, when $|I|=3$, $|J|=5$ in Table A.4.

Table A.1. Experimental Designs for SGAP with $|J|/|I|$=1.0 and e_1=10 and Changes in $|I|$,$|J|$, and b_1

#	$\|I\|$	$\|J\|$	$\|S\|$	e_1	b_1	Optimal Objective Value	CPU Time of Algorithms (Seconds)			
							BNC-SGAP	BNP-SGAP	BNPDST-SGAP	BNPDSTKP-SGAP
1	7	7	128	10	3					
2					6					
3					9					
4					12					
5	8	8	256	10	3					
…					…					
9					15					
10	9	9	512	10	3					
…					…					
14					15					
15	10	10	1024	10	3					
…					…					
20					18					
21	11	11	2048	10	3					
…					…					
26					18					

Table A.2. Experimental Designs for SGAP with $|J|/|I|$=1.0 and e_1=100 and Changes in $|I|$,$|J|$, and b_1

#	$\|I\|$	$\|J\|$	$\|S\|$	e_1	b_1	Optimal Objective Value	CPU Time of Algorithms (Seconds)			
							BNC-SGAP	BNP-SGAP	BNPDST-SGAP	BNPDSTKP-SGAP
1	7	7	128	100	3					
2					6					
3					9					
4					12					
5	8	8	256	100	3					
…					…					
9					15					
10	9	9	512	100	3					
…					…					
14					15					
15	10	10	1024	100	3					
…					…					
20					18					
21	11	11	2048	100	3					
…					…					
26					18					

Table A.3. Experimental Designs for SGAP with $|J|/|I|$=1.0 and e_1=1,000 and Changes in $|I|$,$|J|$, and b_1

#	$\|I\|$	$\|J\|$	$\|S\|$	e_1	b_1	Optimal Objective Value	CPU Time of Algorithms (Seconds)			
							BNC-SGAP	BNP-SGAP	BNPDST-SGAP	BNPDSTKP-SGAP
1	7	7	128	1000	3					
2					6					
3					9					
4					12					
5	8	8	256	1000	3					
…					…					
9					15					
10	9	9	512	1000	3					
…					…					
14					15					
15	10	10	1024	1000	3					
…					…					
20					18					
21	11	11	2048	1000	3					
…					…					
26					18					

Table A.4. Experimental Designs for SGAP with $|J|/|I|=1.5$ and $e_1=100$ and Changes in $|I|$,$|J|$, and b_1

#	$\|I\|$	$\|J\|$	$\|S\|$	e_1	b_1	Optimal Objective Value	CPU Time of Algorithms (Seconds)			
							BNC-SGAP	BNP-SGAP	BNPDST-SGAP	BNPDSTKP-SGAP
1	7	11	128	100	3					
2					6					
…					…					
6					18					
7	8	12	256	100	3					
…					…					
13					21					
14	9	14	512	100	3					
…					…					
21					24					
22	10	15	1024	100	3					
…					…					
29					24					
30	11	17	2048	100	3					
…					…					
38					27					

Table A.5. Experimental Designs for SGAP with $|J|/|I|=2.0$ and $e_1=100$ and Changes in $|I|$,$|J|$, and b_1

#	$\|I\|$	$\|J\|$	$\|S\|$	e_1	b_1	Optimal Objective Value	CPU Time of Algorithms (Seconds)			
							BNC-SGAP	BNP-SGAP	BNPDST-SGAP	BNPDSTKP-SGAP
1	7	14	128	100	3					
2					6					
…					…					
9					27					
10	8	16	256	100	3					
…					…					
18					27					
19	9	18	512	100	3					
…					…					
27					27					
28	10	20	1024	100	3					
…					…					
36					27					
37	11	22	2048	100	3					
…					…					
45					27					

B. Tables of Experimental Designs for BNPDSTKP-SGAP with MCM

Tables B.1, B.2, B.3, B.4, and B.5 represent the experimental designs for the **BPDSTKP-SGAP with MCM** method, described in the section 3.5. Each of them counters Tables A.1, A.2, A.3, A.4, and A.5, respectively, where each has the same conditions with its counter table in terms of the number of machines (|*I*|), the number of jobs (|*J*|), the ratio of |*J*| to |*I*|, the penalty for overtime, and machine capacity.

The first columns represent experiment number that is given by its counter table. The second, third, and fourth columns represent the number of machines, the number of jobs, and the number of scenarios. The eighth and ninth columns contain various initial and final values of N, $\tilde{N}$, and M, respectively.

The columns for "Optimal Objective Value", "Best Objective Value", "Average Objective Value", and "Tolerance Ratio" represent h, $h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)$, $\bar{h}_N$, and $\frac{h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)}{\bar{h}_N}-1$, respectively. The column for "Near-Optimality", representing $\frac{h_{\tilde{N}}\left(\left(\hat{X}_N\right)^*\right)}{h}$, provides the near-optimality of the best solution, $\left(\hat{X}_N\right)^*$, and the higher the value of this ratio is, the better the quality of the solution is.

Table B.1. Experimental Designs for **BNPDSTKP-SGAP with MCM** Considering ε=0.20 for the Experiments in Table A.1

#	$\|I\|$	$\|J\|$	$\|S\|$	e_1	b_1	Optimal Objective Value	Initial $(N,\tilde{N},M)$	Final $(N,\tilde{N},M)$	Best Objective Value	Average Objective Value	Tolerance ratio	Near-Optimality	CPU Time of BPDSTKP-SGAP with MCM (Seconds)
15	10	10	1024	10	3		(128,256,2)						
…					…								
20					18								
21	11	11	2048	10	3		(256,512,2)						
…					…								
26					18								

Table B.2. Experimental Designs for **BNPDSTKP-SGAP with MCM** Considering ε=0.20 for the Experiments in Table A.2

#	$\|I\|$	$\|J\|$	$\|S\|$	e_1	b_1	Optimal Objective Value	Initial $(N,\tilde{N},M)$	Final $(N,\tilde{N},M)$	Best Objective Value	Average Objective Value	Tolerance ratio	Near-Optimality	CPU Time of BPDSTKP-SGAP with MCM (Seconds)
15	10	10	1024	100	3		(128,256,2)						
…					…								
20					18								
21	11	11	2048	100	3		(256,512,2)						
…					…								
26					18								

Table B.3. Experimental Designs for **BNPDSTKP-SGAP with MCM** Considering ε=0.20 for the Experiments in Table A.3

#	$\|I\|$	$\|J\|$	$\|S\|$	e_1	b_1	Optimal Objective Value	Initial $(N,\tilde{N},M)$	Final $(N,\tilde{N},M)$	Best Objective Value	Average Objective Value	Tolerance ratio	Near-Optimality	CPU Time of BPDSTKP-SGAP with MCM (Seconds)
15	10	10	1024	1000	3		(128,256,2)						
…					…								
20					18								
21	11	11	2048	1000	3		(256,512,2)						
…					…								
26					18								

Table B.4. Experimental Designs for **BNPDSTKP-SGAP with MCM** Considering ε=0.20 for the Experiments in Table A.4

#	$\|I\|$	$\|J\|$	$\|S\|$	e_1	b_1	Optimal Objective Value	Initial $(N,\tilde{N},M)$	Final $(N,\tilde{N},M)$	Best Objective Value	Average Objective Value	Tolerance ratio	Near-Optimality	CPU Time of BPDSTKP-SGAP with MCM (Seconds)
22	10	15	1024	100	3		(128,256,2)						
…					…								
29					24								
30	11	17	2048	100	3		(256,512,2)						
…					…								
38					27								

Table B.5. Experimental Designs for **BNPDSTKP-SGAP with MCM** Considering ε=0.20 for the Experiments in Table A.5

#	$\|I\|$	$\|J\|$	$\|S\|$	e_1	b_1	Optimal Objective Value	Initial $(N,\tilde{N},M)$	Final $(N,\tilde{N},M)$	Best Objective Value	Average Objective Value	Tolerance ratio	Near-Optimality	CPU Time of BPDSTKP-SGAP with MCM (Seconds)
28	10	20	1024	100	3		(128,256,2)						
…					…								
36					27								
37	11	22	2048	100	3		(256,512,2)						
…					…								
45					27								

C. Tables of Experimental Designs for HSSP Models

Table C.1. Experimental Design for Comparison between **BNC-HSSP (General)** and **BNP-HSSP (General)**

Experiment #	$\|M\|$	$\|I\|$	$\|J\|$
1	2	2	2
2	2	2	4
3	2	2	6
⋮	⋮	⋱	⋱
10	2	8	8
11	3	3	3
12	3	3	6
13	3	3	9
⋮	⋮	⋱	⋱
20	3	12	12
21	4	4	4
22	4	4	8
23	4	4	12
⋮	⋮	⋱	⋱
30	4	16	16

Table C.2. Experimental Design for Comparison between **BNC-HSSP (Priority)** and **BNP-HSSP (Priority)**

Experiment #	$\|M\|$	$\|I_1\|$	$\|I_2\|$	$\|J_1\|$	$\|J_2\|$
1	2	1	2	2	4
2	2	1	2	2	6
3	2	1	2	2	8
4	2	1	2	2	10
5	2	1	4	2	6
6	2	1	4	2	8
7	2	1	6	2	8
8	2	1	6	2	10
9	2	1	8	2	10
10	3	1	3	2	6
⋮	⋮	⋱	⋱	⋱	⋱
18	3	1	12	2	15
19	4	2	4	4	8
⋮	⋮	⋱	⋱	⋱	⋱
27	4	2	16	4	20
28	5	2	5	4	10
⋮	⋮	⋱	⋱	⋱	⋱
34	5	2	15	4	20

Table C.3. Experimental Designs for Comparison between

BNC-HSSP (Pre-arranged) and **BNP-HSSP (Pre-arranged)**

Experiment #	$\|M\|$	$\|I_1\|$	$\|I_2\|$	$\|J_1\|$	$\|J_2\|$
1	3	1	1	2	2
2	3	1	1	2	4
3	3	1	1	2	6
4	3	1	1	2	8
5	3	1	2	2	2
6	3	1	2	2	4
7	3	1	2	2	6
8	3	1	2	2	8
9	3	1	3	2	2
10	3	1	3	2	4
11	3	1	3	2	6
12	3	1	3	2	8
13	3	1	4	2	2
14	3	1	4	2	4
15	3	1	4	2	6
16	3	1	4	2	8
17	4	2	2	4	4
18	4	2	2	4	8
⋮	⋮	⋮	⋮	⋮	⋮
32	4	2	8	4	16

References

[1] M. Albareda-Sambola and E. Fernandez, "The Stochastic Generalized Assignment Problem Bernoulli Demands," *TOP*, vol. 8(2), pp. 165-190, 2000.

[2] M. Albareda-Sambola, V. D. Vlerk, H. Maarten, and E. Fernandez, "Exact solutions to a class of stochastic generalized assignment problems," *European Journal of Operational Research*, vol. 173(2), pp. 465-487, 2002.

[3] A. Ali, J. Kennington, and T. Liang, "Assignment with en route training of Navy personnel," *Naval Resource Logistics*, vol. 40, pp. 581-592, 1993.

[4] H. B. Amor, J. Desrosiers, and J. M. V. Carvalho, "Dual-Optimal Inequalities for Stabilized Column Generation," *Operations Research*, vol. 54(3), pp. 454-463, 2006.

[5] D. Avis and K. Fukuda, "A Pivoting Algorithm for Convex Hulls and Vertex Enumeration of Arrangements and Polyhedra," *Discrete and Computational Geometry*, vol. 8, pp. 295-313, 1992.

[6] M. N. Azaiez and S. S. Al Sharif, "A 0-1 goal programming model for nurse scheduling," Computers and Operations Research, vol. 32, pp. 491-507, 2005.

[7] J. F. Bard and H. W. Purnomo, "Preference scheduling for nurses using column generation," *European Journal of Operational Research*, vol. 164, pp. 510–534, 2005.

[8] S. Barkin, "Technical change and manpower planning; coordination at enterprise level: a series of national case studies," *Organization for Economic Co-operation and Development*, 1967.

[9] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance, "Branch-and-price: Column generation for solving huge integer programs," *Operations Research*, vol. 46(3), pp. 316-329, 1998.

[10] J. Beliën and E. Demeulemeester, "Scheduling trainees at a hospital department using a branch-and-price approach," *European journal of operational research*, vol. 175 (1), pp. 258-278, 2006.

[11] J. Beliën, "Exact and heuristic methodologies for scheduling in hospitals: problems, formulations and algorithms," *4OR: A Quarterly Journal of Operations Research*, vol. 5(2), pp. 157-160, 2007.

[12] J. Beliën and E. Demeulemeester, "Building cyclic master surgery schedules with leveled resulting bed occupancy," *European Journal of Operational Research*, vol. 176(2), pp. 1185-1204, 2007.

[13] B. A. Berg, "Markov Chain Monte Carlo Simulations and Their Statistical Analysis with Web-Based Fortran Code," *World Scientific*, 2004.

[14] I. Berrada, J. Ferlan, and P. Michelon, "A multi-objective approach to nurse scheduling with both hard and soft constraints," *Socio-Economic Planning Science*, vol. 30, pp. 183-193, 1996.

[15] J. T. Blake and J. Donald, "Mount Sinai hospital uses integer programming to allocate operating room time," *Interface*, vol. 32, pp. 63-73, 2002.

[16] T. Blanco and R. Hillery, "A sea story: Implementing the Navy's personnel assignment system," *Operations Research*, vol. 42(5), pp. 814-822, 1994.

[17] I. Blöchliger, "Modeling staff scheduling problems. A tutorial," *European Journal of Operational Research*, vol. 158, pp. 533-542, 2004.

[18] F. Bosi and M. Milano, "Enhancing constraint logic programming branch and bound techniques for scheduling problems," *Software Practice and Experience*, vol. 31, pp. 17–42, 2001.

[19] E. K. Burke, P. D. Causmaecker, S. Petrovic, and G. V. Berghe, "A multi-Criteria metaheuristic approach to nurse rostering," *Proceeding of the 2002 Congress on Evolutionary Computation (CEC2002)*, pp. 1197-1202, 2002.

[20] E. K. Burke, P. D. Causmaecker, G. V. Berghe, and H. V. Landeghem, "The State of Art of Nurse Rostering," *Journal of Scheduling*, vol. 7(6), pp. 441-449, 2004.

[21] P. Cappanera and G. Gallo, "A multi-commodity flow approach to the crew rostering problem," Dip. di Informatica, Univ. di Pisa, Technical Report TR-01-08, 2001.

[22] A. Chabrier, "Vehicle Routing Problem with elementary shortest path based column generation," *Computers & Operations Research*, vol. 33(10), pp. 2972-2990, 2006.

[23] A. Charnes, W.W. Cooper, and R.J. Niehaus, "Mathematical Models for Manpower and Personnel Planning," *Texas University Austin Center for Cybernetic Studies*, 1971.

[24] B. Cheang, H. Li, A. Lim, and B. Rodrigues, "Nurse rostering problems—A bibliographic survey," *European Journal of Operational Research*, vol. 151, pp. 447–460, 2003.

[25] J. G. Chen and T. W. Yeung, "Hybrid expert-system approach to nurse scheduling," *Computers in Nursing*, vol. 11(4), pp. 183-190, 1993.

[26] H. Christos and K. S. Papadimitriou, "Approximation Algorithms," in *Combinatorial Optimization: Algorithms and Complexity*, Courier Dover Publications, pp. 419-430, 1998.

[27] R. Cohen, L. Katzir, and D. Raz, "An efficient approximation for the Generalized Assignment Problem," *Information Processing Letters*, vol. 100(4), pp. 162-166, 2006.

[28] E. Danna and C. L. Pape, "Branch-and-price heuristics: A case study on the vehicle routing problem with time windows," in *Column Generation*, J. D. G. Desaulniers, M. M. Solomon, Eds. Berlin: Springer, pp. 99-129, 2005.

[29] Z. Degraeve and R. Jans, "A New Dantzig-Wolfe Reformulation and Branch-and-Price Algorithm for the Capacitated Lot Sizing Problem with Set Up Times," *Erasmus Research Institute of Management (ERIM)*, RSM Erasmus University, 2005.

[30] F. Dexter, A., Macario, and R. D. Traub, "Which algorithm for scheduling add-on elective cases maximizes operating room utilization?" *Anesthesiology*, vol. 91, pp. 1491-1500, 1999.

[31] F. Dexter, R. D. Traub, and P. Lebowitz, "Scheduling a delay between different surgeons' cases in the same operating room on the same day using upper prediction bounds for case durations," *Anesthesia and Analgesia*, vol. 92, pp. 943-946, 2001.

[32] F. Dexter and R. D. Traub, "How to schedule elective surgical cases into specific operating rooms to maximize the efficiency of use of operating room time," *Anesthesia and Analgesia*, vol. 94, pp. 943-946, 2002.

[33] W. R. Dill, D. P. Gaver, and W. L. Weber, "Models and Modelling for Manpower Planning," *Management Science*, vol. 13(4), pp. 142-167, 1966.

[34] H. A. Eiselt and V. Marianov, "Employee positioning and workload allocation," *Engineering Optimization*, vol. 40(11), pp. 1051-1066, 2008.

[35] A. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier, "Staff scheduling and rostering: A review of applications, methods and models," *European Journal of Operational Research*, vol. 153, pp. 3-27, 2004.

[36] G. S. Fishman, *Monte Carlo: Concepts, Algorithms, and Applications*, Springer Verlag, New York, 1995.

[37] L. Fleischer, M. Goemans, V. S. Mirrokni, and M. Sviridenko, "Tight Approximation Algorithms for Maximum General Assignment Problems," *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pp. 611-620, 2006.

[38] A. M. Geoffrion, "Lagrangian relaxation and its uses in integer programming," Mathematical Programming Study, vol. 2 82-114, 1974

[39] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright, "Constrained nonlinear programming," in *Optimization*, G. L. Nemhauser, A.H.G.R. Kan, M. J. Todd, Eds. New York: Elsevier North-Holland, Inc., pp. 171-210, 1989.

[40] A. Guinet and S. Chaabane, "Operating theatre planning," *International Journal of Production Economics*, vol. 85, pp. 69-81, 2003.

[41] F. Hanssmann and S. W. Hess, "A linear programming approach to production and employment scheduling," *Management Technology*, vol. 1(1), pp. 46-51, 1960.

[42] A. Holder, "Navy Personnel Planning and the Optimal Partition," *Operations Research*, vol. 53(1), pp. 77-89, 2005.

[43] C. C. Holt, F. Modigliani, J. F. Muth, and H. A. Simon, *Planning production, inventories and work force*, Prentice-Hall, Englewood Cliffs, New Jersey, 1960.

[44] H. Hoyle and R. J. Stubbs, "Management Stocktaking: An Approach to Manpower Planning in Banking," *Long Range Planning*, vol. 2, pp. 18–23, 1969.

[45] A. Jaszkiewicz, "A metaheuristic approach to multiple objective nurse scheduling," *Foundations of Computing and Decision Science*, vol. 22(3), pp. 169-184, 1997.

[46] B. Jaumard, F. Semet, and T. Vovor, "A generalized linear programming model for nurse scheduling," *European Journal of Operational Research*, vol. 107, pp. 1–18, 1998.

[47] A. Jebali, A. B. H. Alouane, P. Ladet, "Operating rooms scheduling," *International Journal of Production Economics*, vol. 99, pp. 52-62, 2006.

[48] P. Jirutitijaroen and C. Singh, "Reliability constrained multi-area adequacy planning using stochastic programming with sample-average approximations," *IEEE Transactions on Power Sytems*, vol. 23(2), pp. 504-603, 2008.

[49] B. Kallehauge, J. Larsen, O. B. G. Madsen, and M. M. Solomon, "Vehicle Routing with Time Windows," in *Column Genration*, J. D. G. Desaulniers and M. M. Solomon, Eds. Kluwer Academic Publishers, pp. 31-98, 2004.

[50] B. Kallehauge, J. Larsen, and O. B. G. Madsen, "Lagrangean Duality Applied to the Vehicle Routing with Time Windows," *Computers and Operations Research*, vol. 33(5), pp. 1464-1487, 2006.

[51] A. Kleywegt and A. Shapiro, "The sample average approximation method for stochastic discrete optimization," *SIAM Journal on Optimization*, vol. 12(2), 2000.

[52] K. Kogan, E. Khmelnitsky and T. Ibaraki, "Dynamic Generalized Assignment Problems with Stochastic Demands and Multiple Agent--Task Relationships," *Journal of Global Optimization*, vol. 31(1), pp. 17-43, 2005.

[53] M. Lamiri, X. Xie, A. Dolgui, and F. Grimaud, "A stochastic model for operating room planning with elective and emergency demand for surgery," *European Journal of Operations Research*, vol. 185(3), pp. 1026-1037, 2008.

[54] S. D. Lapierre, C. Batson, C. and S. McCaskey, "Improving on time performance in health care organizations: A case study," *Health Care Management Science*, vol. 2, pp. 27-34, 1999.

[55] S. Lee, C. Parakornkule, M. M. Domach, and I. E. Grossmann, "Recursive MILP model for finding all the alternate optima in LP models for metabolic networks," *Computers and Chemical Engineering*, vol. 24, pp. 711-716, 2000.

[56] T. Liang and B. Buclatin, "Improving the utilization of training resources through optimal personnel assignment in the U.S. Navy," *European Journal of Operations Research*, vol. 33, pp. 183-190, 1988.

[57] T. Liang and T. Thompson, "A large-scale personnel assignment model for the Navy," *Decision Science*, vol. 18(2), pp. 235-250, 1987.

[58] E. Litvak and M. C. Long, "Cost and quality under managed care: Irreconcilable differences," *The American journal of managed care*, vol. 6(3), pp. 305–312, 2000.

[59] M. E. Luebbecke and J. Desrosiers, "Selected Topics in Column Generation," *Operations Research*, vol. 53(6), pp. 1007-1040, 2005.

[60] W. K. Mak, D. P. Morton, and R. K. Wood, "Monte Carlo bounding techniques for determining solution quality in stochastic programs," *Operations Research Letters*, vol. 24(1), pp. 47-56, 1999.

[61] A. J. Mason and M. C. Smith, "A nested column generator for solving rostering problems with integer programming: Techniques and applications," *International Conference on Optimization*, pp. 827–834, 1998.

[62] E. Marcon, S. Kharraja, and G. Simonnet, "The operating theatre planning by the follow-up of the risk of no realization," *International Journal of Production Economics*, vol. 85, pp. 83-90, 2003.

[63] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*, John Wiley & Sons, 1990.

[64] R. E. Marsten, W. W. Hogan, and J. W. Blankenship, "The Boxstep Method for Large-Scale Optimization," *Operations Research*, vol. 23(3), pp. 389-405, 1975.

[65] A. Martel and A. Al-Nuaimi, "Tactical Manpower Planning via Programming Under Uncertainty," *Operational Research Quarterly*, vol. 24(4), pp. 571-585, 1973.

[66] T. H. Matheiss and D. S. Rubin, "A Survey and Comparison of Methods for Finding All Vertices of Convex Polyhedral Sets," *Mathematics of Operations Research*, vol. 5(2), pp. 167-185, 1980.

[67] A. Mehrotra, K. E. Murphy, and M. A. Trick, "Optimal shift scheduling: A branch-and-price approach," *Naval Research Logistics*, vol. 47, pp. 185–200, 2000.

[68] O. D. Merle, D. Villeneuve, and J. Desrosiers, "Stabilized column generation," Discrete Math, vol. 194, pp. 229-237, 1999.

[69] N. Metropolis and S. Ulam, "The Monte Carlo Method," *Journal of the American Statistical Association*, vol. 44(247), pp. 335–341, 1949.

[70] H. Mine, M. Fukushima, K. Ishikawa, and I. Sawa, "An algorithm for the assignment problem with stochastic side constraints: Memoirs of the Faculty of Engineering," Kyoto University, 1983.

[71] O. Oguz, "Generalized Column Generation for Linear Programming," *Management Science*, vol. 48(3), pp. 444-452, 2002.

[72] D. V. Olivier du Merle, J. Desrosiers, and P. Hansen, "Stabilized column generation," *Discrete Mathematics*, vol. 194(1-3), pp. 229-237, 1999.

[73] I. Ozakarahan, "Flexible nurse scheduling support systems," *Computer Methods and Programs in Biomedicine*, vol. 30, pp. 145-153, 1989.

[74] I. Ozakarahan, "Allocation of surgical procedure to operating rooms," *Journal of Medical Systems*, vol. 19(4), pp. 333-352, 1995.

[75] I. Ozakarahan, "Allocation of surgeries to operating rooms using goal programming," *Journal of Medical Systems*, vol. 24(6), pp. 339-378, 2000.

[76] A. Pigatti, M. P. D. Aragao, and E. Uchoa, "Stabilized Branch-and-cut-and-price for the Generalized Assignment Problem," *Annals of GRACO*, vol. 19, pp. 389-395, 2004.

[77] D. Pisinger, "Algorithms for Knapsack Problems," University of Copenhagen, *E-print available at* http://www.diku.dk/publikationer/tekniske.rapporter/1995/95-1.ps.gz, 1995.

[78] D. Pisinger, "Linear Time Algorithms for Knapsack Problems with Bounded Weights," *Journal of Algorithms*, vol. 33(1), pp. 1-14, 1999.

[79] C. P. Robert and G. Casella, *Monte Carlo Statistical Methods*, New York: Springer-Verlag, 2004.

[80] R. Y. Rubinstein and D. P. Kroese, *Simulation and the Monte Carlo Method*, New York: John Wiley & Sons, 2007.

[81] P. Santibanez, M. Begen, and D. Atkins, "Managing surgical waitlists for a British Columbia health authority," Centre for Operations Excellence, Sauder School of Business, University of British Columbia, Canada, Research Report, 2005.

[82] M. Savelsbergh, "A Branch-and-Price Algorithm for the Generalized Assignment Problem," *Operations Research*, vol. 45(6), pp. 831-841, 1997.

[83] S. Sen, J. L. Higle, "An introductory tutorial on stochastic linear programming," Interfaces, Vol. 29 (2), pp. 33-61, 1999.

[84] A. Shapiro, "Monte Carlo simulation approach to stochastic programming," *Proceedings of the 2001 Winter Simulation Conference*, pp. 428-431, 2001.

[85] A. Shapiro, "Monte Carlo sampling approach to stochastic programming," *ESAIM: Proceedings*, vol. 13, pp. 65-73, 2003.

[86] J. Singhal and K. Singhal, "Holt, Modigliani, Muth, and Simon's work and its role in the renaissance and evolution of operations management," *Journal of Operational Management*, vol. 25(2), pp. 300-309, 2007.

[87] V. H. Tran, G. Reinelt, and H. G. Bock, "Boxstep methods for crew pairing problems," *Optimization and Engineering*, vol. 7(1), pp. 33-46, 2006.

[88] V. M. Trivedi and D.M. Warner, "A branch and bound algorithm for optimum allocation of float nurses," *Management Science*, vol. 22(9), pp. 972-981, 1976.

[89] F. Vanderbeck, "On Dantzig-Wolfe Decomposition in Integer Programming and ways to Perform Branching in a Branch-and-Price Algorithm," *Operations Research*, vol. 48(1), pp. 111-128, 2000.

[90] F. Vanderbeck and M. W. P. Savelsbergh, "A generic view of Dantzig-Wolfe decomposition in mixed integer programming," *Operations Research Letters*, vol. 34(3), pp. 296-306, 2006.

[91] F. Vanderbeck and L. A. Wolsey, "An exact algorithm for IP column generation," *Operations Research Letters*, vol. 19(4), pp. 151-159, 1996.

[92] V. V. Vazirani, "A Pseudo-polynomial time algorithm for knapsack," in *Approximation Algorithms*, Springer, pp. 68-73, 2001.

[93] R. Venkataraman and M. J. Brusco, "An integrated analysis of nurse staffing and scheduling policies," *Omega*, vol. 24, pp. 57-71, 1996.

[94] B. Verweij, S. Ahmed, A. J. Kleywegt, G. Nemhauser, and A. Shapiro, "The sample average approximation method applied to stochastic routing problems: A computational study," *Computational Optimization and Application*, vol. 24, pp. 289-333, 2003.

[95] E. N. Weiss, "Models for determining estimated start times and case orderings in hospital operating rooms," *IIE Transactions*, vol. 22, pp. 143-150, 1990.

[96] A. Westerlund, G. L. Maud, L. Torbjörn, "A stabilized column generation scheme for the traveling salesman subtour problem," *Discrete Applied Mathematics*, vol. 154(15), pp. 2212-2238, 2006.

[97] W. E. Wilhelm, "A Technical Review of Column Generation in Integer Programming," *Optimization and Engineering*, vol. 2(2), pp. 159-200, 2001.