2011

Acceleration and Stabilization Techniques for Column Generation Applied to Capacitated Resource Management Problems

Alper Uygur
Lehigh University

ACCELERATION AND STABILIZATION TECHNIQUES FOR COLUMN GENERATION APPLIED TO CAPACITATED RESOURCE MANAGEMENT PROBLEMS

by

Alper Uygur

Presented to the Graduate and Research Committee
of Lehigh University
in Candidacy for the Degree of
Doctor of Philosophy

in
Industrial Engineering

Lehigh University

September 2011

Approved and recommended for acceptance as a dissertation in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

_____
Date

_____
Dr. George R. Wilson, Dissertation Advisor

_____
Accepted Date

Committee:

_____
Dr. George R. Wilson, Chairman

_____
Dr. Tamas Terlaky

_____
Dr. Robert Storer

_____
Dr. Oktay Gunluk

_____
Dr. Michael Magent

Acknowledgments

I would like to first express my sincerest gratitute to my dear Ph.D. Advisor, Dr. George Wilson, primarily for introducing me to this research subject, for his guidance, and patience. Without his support, collaboration, and brilliant ideas from his vast knowledge, none of this could have materialized. I am also forever indebted to Dr. Tamas Terlaky, Dr. Robert Storer, Dr. Oktay Gunluk and Dr. Michael Magent for being on my committee, for their understanding, and suggestions for research directions. I truly learned a great deal for how to be more professional, careful, and more organized, thanks to them.

I am also grateful that IBM allowed me to use their business problem as the application area for my Ph.D. thesis, with some modelling changes and of course with randomized data. I would like to thank my manager Terry Hammaker, my supervisor Mark Booth, and my colleagues Eugene Kelton, Derek Jones, Gardner Pomper, and Christina Ma for letting me learn a lot during my time at IBM. From the BNSF side, my manager Pooja Dewan showed gracious consideration during the final stages of the thesis; I cannot thank her enough.

Regarding technical and coding related assistance in DIP, Matthew Galati's help was invaluable. For introducing me to Column Generation, Pietro Belotti's efforts will never be forgotten. For everything they taught me during my Ph.D. education at Lehigh, I appreciate the wisdom of Dr. Aurelie Thiele, Dr. Ted Ralphs, Dr. Larry Snyder, Dr. Lou Plebani, and Dr. Jeff Linderoth.

In the I&SE department, I owe so much to Rita Frey, Kathy Rambo and Amanda Fabrizio for being so helpful when needed, at all times. From the College of Engineering, Brie Lisk, I thank you very much for your deadline reminders, and never showing exhaustion while replying to my endless emails. I also owe Dr. David Wu and Dr. Joseph Hartman, for granting me several years of I&SE department funding.

Last but not the least, I owe a debt of gratitude to my loving family, for helping me persevere in this long journey, which could have become awfully difficult without the support of dear friends, as well. Having read everything above, I realize I owe so much to so many people. Thank you all. I promise I will pay.

Contents

List of Tables

List of Figures

Abstract

This research presents a very efficient method of solving a broad class of large-scale capacitated resource management problems by introducing a new formulation and decomposition. A heuristic called Likelihood of Assignment is utilized not only to find high quality initial integer feasible solutions, but also to guide the Branch-and-Price (B&P) Algorithm towards stabilization. Although Column Generation (ColGen) is thought to be the ideal approach to attack large-scale linear problems, it has been found that the textbook variety of ColGen algorithms are rather problematic during the Heading-in Phase. To alleviate the Heading-In Effect, and to start with good and valid lower-bounds, strategies for constructing the initial dual vector are provided, which is guaranteed to be feasible for the first Restricted Master Problem (RMP). Dual vectors obtained from RMP's at any iteration are further manipulated in order to prevent parallel and astray columns, which cause degeneracy and stalling. A subproblem selection scheme is also proposed to accelerate the convergence. The proposed techniques are thoroughly covered in the first two chapters of this thesis for the Workforce Planning with Cross-Training Problem and they provide superior results to CPLEX by being 10 times faster on average and having 20% better solution gap quality in the same amount of time.

In the third chapter of the dissertation, an Agent-Based Tabu Algorithm for the Workforce Scheduling Problem with Stochastic Demand is introduced to yield quality starting solutions. These starting solutions then feed a stabilized Nested ColGen Algorithm. The Nested ColGen algorithm not only provides a valid lower bound, but also, improve the integer feasible solution towards optimality with high quality incoming columns. Likelihood of Assignment's effectiveness in this particular problem structure for accelerating and stabilizing the ColGen Algorithm is also showcased.

In the final chapter, we show that, the applicability of Likelihood of Assignment and the aforementioned acceleration and stabilization schemes are indeed viable for very well-known capacitated resource management problems such as Train (Vehicle) Routing, Capacitated Facility Location, and Airline Crew Pairing and Rostering Problems. Therefore, the approach presented in this research is not designed solely for Workforce Planning and Scheduling Problems, but it can be slightly modified to solve any type of Capacitated Resource Management Problem with block angular structure. Recommendations for parameter setup and performance tuning are also given for different problem characteristics and data input.

0.1 Introduction and Motivation

The main purpose of this research is to solve the various stages of Workforce Planning Problem. The stages can be defined as: The determination of the number of employees, their appropriate cross-skill training and their customer assignments for a given service territory having a distinct demand structure, and scheduling of these employees to weekly shifts constrained by work rules, back-up strategies, and contract types.

The Maintenance and On-Site Technical Support Business for electronic systems is chosen as the application area of this research. The reason is that, service support is an ever-expanding business in the United States (currently over $10 billion in revenue per year) and in the world. For a healthy business, good maintenance and upkeep of computers, servers, and printers are necessary for any company benefiting from such equipment. Therefore, companies like IBM, HP, Geek Squad, Sun Systems, Computer Patrol, and many more are competing to seize the greatest share of this market. Since the repair durations of broken electronics do not vary from company to company, the only way these firms can attract customers and provide an edge is if they offer quicker response at the facility where some type of a computer system is broken. This prompts the idea of selling time-based service warranties; i.e., 4-hour, 8-hour or same day contracts to customers. Of course the 4-hour contract would be the most expensive, because if the service company cannot fulfill their obligation of fixing that machine in 4-hours, there is a corresponding penalty. Although, these service companies are trying to sell as many high-revenue generating 4-hour contracts as possible, not meeting the demand on time also ruins the reliability of service and endangers future contract renewals.

For 4-hour contracts to be successfully fulfilled, the service company must have enough service technicians with the correct skills to fix the machines, and located sufficiently close to the customer so that the travelling time is within the contract time window. In spite of the fact that 4-hour contracts are lucrative, they either entail more employees being closer to the demand points, or, if management chooses to go with fewer employees, the travelling times become a serious issue along

with extra machine training costs and skill upgrade costs for a more diversely skilled labor force.

Considering all these factors, one must carefully choose the employee group size, locate them carefully and strategically on the service region, prescribe appropriate training based on demand type requirements, and schedule them to shifts, taking into account the contract types. This problem as a whole, however, would be extremely hard to solve all at once, owing to the number of possible workshifts, skills, training requirements, start of the day launching points and assignment combinations. Therefore, we propose to solve the problem by a divide and conquer strategy. A strategic/tactical level deterministic planning problem is solved first, and the solution of this first stage problem provides input to the shift scheduling problem with stochastic demand.

By solving the first stage problem, management would have a concrete idea of the total headcount they need, where these technicians are to be located at the beginning of every work day, their skill combinations, and their potential customer assignments and workload at every demand location. The demand data is given in terms of monthly service call rates; therefore, one of the primary assumptions of this first stage is that the work assigned to the technicians are to be completed within the 125 hours of their monthly available capacity. There are no queueing constraints in this model, as long as the demand is fulfilled in that particular month by a capable SSR, the demand is considered to be satisfied.

The second stage problem is more on the operational side of decision making. It has more granular data and details; for instance, the demand data is given with an additional index of what time of the day that the service call is expected. This finer granularity actually requires us to use and fix the headcount and skill training combinations found in the planning problem, so that the stochastic scheduling model stays tractable. The stochastic demand data entails an approximate recourse action which is assigning overtime shifts to the SSRs. In this research, the random stochastic demand is generated in such a way that every scenario is guaranteed to be completely covered by the set of SSRs that is found in the first stage.

0.2 Literature Survey

An aim of this research is to solve a national workforce determination problem for a large corporation, starting with the headcount decision and incorporating technician coverage territories, training and finally scheduling. There have been several attempts using various techniques to solve similar problems in the literature. Almost all of them assume the solution of a particular stage of the problem is already known, and inputted as a given for the subsequent stage.

A good survey for staff scheduling and rostering is found in Ernst et al. [23]. Among the solution approaches, the following are most commongly taken: Fuzzy set theories to aid greedy heuristics [44], neighborhood search to improve existing solutions [45], Constraint Programming for highly constrained problems and when any feasible solution would suffice even if nonoptimal [55]. Constraint Programming is also used as a preprocessing stage to reduce the size of the original feasible region and then followed with traditional OR techniques in [30] and [10]. Metaheuristics like Simulated Annealing [8], Genetic Algorithms [9], Tabu Search, Ant Colony Optimization, etc., have also been tried. Although they proved to be robust, optimality is rarely obtained and almost never guaranteed, (refer to [49] and [51]).

A few researchers have attempted to model the staff planning and the scheduling problems together [1], by introducing hierarchial decomposition. Some have developed their own algorithm in the implementation of their approach [2]; however, those proposed approaches cannot be applied to the size of problems we are dealing with.

Although Volume Algorithm [4], Lagrangean Decomposition with Variable Splitting [28], Benders' Decomposition [7], and Cross Decomposition [60] are representative of a limited number of other proven techniques for attacking large-scale problems, our experience with them has been disappointing. Their main shortcoming is the lack of the capability to provide a MIP upperbound; and requiring the need to find our own external heuristic for feasible solutions. While Lagrangean Decomposition and Cross Decomposition suffered from non-converging dual multipliers, Volume

Algorithm did not yield quality feasible solutions, mostly because we did not have a heuristic well-tailored to use the approximate solutions that it yields. Unfortunately, the objective function of the problem behaves very poor when Rounding Heuristics are used. Benders' Decomposition, was tested; but, our problem did not have the proper structure to exploit benefits from it. The so-called first stage variables became the continuous assignment variables in order for the subproblem decomposition to be formed for every technician; and the second stage variables turned out to be the binary variables. Benders decomposition benefits when the first stage variables are integer/binary and fixed, and the second stage subproblems are simple linear problems. All this has inspired the development of a stabilized column generation routine.

For the planning phase of our problem, mathematical programming approaches mostly use the Dantzig set covering or set partitioning formulations [15] with possible variations. Due to the formidable size of the problems being considered, two different methods have been developed in order to handle the exponential explosion in number of variables: generating a limited number of columns and forming a reasonably sized formulation ([24], [29]), and partially generating all possible columns using column generation approaches ([21], [37]). Since the problem at hand is mixed-integer linear, Branch-and-Cut [32] or Branch-and-Price ([5], [27]) is followed until either optimal or a high-quality solution is found.

In the planning stage of the problem, the objective is to find the least costly combination of number of employees needed, the machine trainings they are required to attain with associated skill upgrade salary costs incurred, assignment of these employees to customer locations, and allocating the jobs among the properly skilled SSRs considering travel costs. These problem attributes clearly distinguish the Workforce Planning Problem from Aircrew Pairing and Rostering Problems ([26], [37], [32], [27], [3]), Vehicle Routing Problems ([21], [17], [34], [20], [46], [19]) and Multi-Commodity Flow determinations within Capacitated Facility Location Problems (CFLP) ([47], [39], [56], [35]) as described in the following.

Most Aircrew Pairing and Rostering Problems do not have training possibilities for the pilots and the crew; they are assumed to be capable of flying any type of aircraft, and the crew size is

already known and fixed. Their subproblems are constrained shortest path problems where very efficient dynamic programming algorithms are readily available. In spite of the fact that the number of nodes and arcs may seem to make the subproblems intractable; flight pairing rules, work, rest and scheduling rules, and pre-assignments, which are dictated by the airline, simplify the subproblems. Flow conservation constraints for each node allow the elimination of redundant and dominated arcs.

Vehicle Routing Problems are also formulated using a set partitioning scheme, like Aircrew Pairing and Rostering Problems, where a route is proposed for each vehicle until all demands are met. They benefit from the network structure of the subproblems, for which efficient algorithms exist. By contrast, Workforce Planning Problem's subproblems are knapsack problems with side constraints, and do not possess a network structure.

The formulation for the Workforce Planning Problem closely resembles the Multi-Commodity Capacitated Facility Location Problem (CFLP) with side constraints (those regarding machine training and skill upgrades). The Capacitated Facility Location and the p-median problems have been attacked with Column Generation very rarely in the literature, and even problem sizes of 250 customers to 100 facilities are being reported as very difficult to converge with their standard approaches ([47], [39], [56], [35]).

Several important distinctions have to be made, though, that in our problem definition, a customer can be given service by several employees based in different zipcodes throughout the course of a month; therefore, the demand points do not have to be assigned to a 'Single Source'. This condition adds difficulty to the problem in the sense that the LP relaxation of the strong formulation does not necessarily yield integer solutions anymore. Moreover, the LP relaxation's objective function is no longer close to the optimal MIP solution. Another reason for the difficulty is that the capacity of the employees is depleted while travelling to another customer location, which is not the case in CFLP.

Despite the fact that Dantzig-Wolfe Decomposition [14] is actually straightforward for our problem (since it is block angular by Service Support Representatives (SSR)), the textbook version column generation performed very badly in our case. It is clearly observed that the performance of the

algorithm is extremely dependent on the quality and variety of the initial feasible solutions, and also very much reliant on the incoming columns at every iteration, as stated in [40]. The default D-W decomposition demonstrated all the bad traits given in the textbooks [18]:

- Heading-in issues: No valid (useful) lowerbounds at the beginning.
- Yo-Yo effect: No monotonic lowerbound improvement.
- Plateau effect and Tailing-off: Stalling in the RMP upperbound and no significant upperbound improvements towards the end.

Although there have been stabilization methods using boxing techniques ([42], [22], [50]), by taking convex combination of duals ([62], [31]), using interior point methods [52] and introducing dual-optimal inequalities ([16], [6]) proposed to overcome issues with the convergence of column generation, they are shown to be not trivial, problem specific, and entailing too much effort for parameter tuning. In this research, we are proposing a generic stabilization and acceleration routine using the Likelihood of Assignment Heuristic as the first step for initializing the RMP, estimating the initial dual vector, choosing subsets of suproblems to solve and manipulating the intermediary dual vectors. Substantial performance enhancements in terms of lowerbound quality, global upperbound improvements, and convergence rates result from methods developed in this research.

The second stage of the problem is called Workforce Scheduling with Stochastic Demand. For this stage, it is assumed that the correct headcount, along with SSR ids with proper skills and their start of the day launch locations are known. The objective of the problem is, then, to minimize the total overtime hours assigned and to find the best weekly shift assignments while providing coverage to a sample set of equally likely demand scenarios.

There are a number of well-known solution methods for Stochastic Integer Problems: Integer L-Shaped Method [36] is similar to the Benders' Decomposition. Dual Decomposition [11] is a Lagrangean-based decomposition where each subproblem is composed of individual scenarios; nonetheless, it does not guarantee optimality. Structured Enumeration and Stochastic Branch-and-Bound [48] is an algorithm where the performance is strictly dependent on the initial partitioning of

the feasible set and the estimates on the upper and lower bounds. Progressive Hedging Algorithm [38] is a Tabu Search-based heuristic where integer feasible solutions are obtained with variable fixing.

The reason we are attacking this problem with column generation is that there are very few publications using ColGen in Stochastic Integer Programs (SIP) ([59], [58], [13], [57], [41]) and there exists none for Stochastic Workshift Scheduling. The deterministic equivalent of the Stochastic Shift Scheduling Problem is appropriately decomposed into Master and Subproblems for us to be able to deploy the stabilization and acceleration techniques proposed in this thesis.

The thesis is organized in the following manner: The first chapter describes the planning problem and the necessity for a heuristic solution. Details of how and why the Likelihood of Assignment Heuristic works are provided. The second chapter gives details about the reformulation of the problem, and provides the exact B&P algorithm which is stabilized and accelerated using the findings from Chapter 1. Chapter 3 is where the Stochastic Mixed Integer Model for the Scheduling Problem is introduced. An Agent-based Tabu search algorithm which uses Likelihood of Assignment's findings for taboo lists, and a stabilized ColGen algorithm are developed. Finally in Chapter 4, we show that our algorithm could be applicable to Train Routing, CFLP, and Aircrew Pairing and Rostering Problems to demonstrate its flexibility and robustness. The second half of this chapter is devoted to give parametric setup recommendations and their impact on solution times.

For all computational experiments, CPLEX 12.2 is used as the LP and MIP solver throughout the thesis. The default solver settings can be found in the User's Manual [33]. Although CPLEX selects the Dual Simplex Algorithm by default for solving LPs, Barrier Algorithm was invoked when the largest test cases were being considered. This is because the Barrier Algorithm has superior performance to the Primal or Dual Simplex Algorithms as the problem sizes increase.

The contributions achieved by this research are important: A generic Restricted Feasible Domain Heuristic is developed, which allows finding high quality feasible solutions in a very short amount of time, and also stabilizes and accelerates the B&P algorithm for most of the prominent Capacitated Resource Management Problems. The heuristic is easy to set-up and the stabilization

techniques are already embedded in DIP (Decomposition for Integer Programming framework, an open source COIN-OR initiative) for other researchers who may want to benefit from the approach. Once the heuristic finds good quality initial solutions, the stabilization and acceleration routines steadily approach optimality.

Chapter 1

An Efficient Heuristic for Workforce Planning Problems with Cross-Training

1.1 Necessity for a Heuristic

Experience with various sizes of the Workforce Planning and Scheduling problem showed that even after the problem is broken into 2 stages, neither stage provides a quality integer solution in a reasonable amount of time in their monolithic forms. Consideration was given to ”Restricted Feasible Domain” type heuristics, where the original model’s feasible domain is reduced to a size which can be solved faster and more efficiently. As will be seen later in this chapter, considering and/or creating all variables explicitly is not necessary in order to discover a high quality integer solution in a short amount of time. Moreover, a logic will be devised to eliminate the need to keep the complicating binary variables in the mathematical model representation, so that a solution from a LP formulation will yield us an integer solution after some post-processing.

In this heuristic, ”Likelihood of Assignment” (LoA), it is decided which variables may actually be included in the core solution, and which variables are to be flagged as least likely to be in the optimal solution. Later on in Chapter 2, column generation will help us detect the variables which will surely improve the current solution at hand and bring back those variables which might have

been eliminated by LoA.

1.2 Mathematical Model for the Planning Problem

Before the mathematical model is introduced, the notion of "launch zipcode" has to be defined. Based on our assumption of problem and solution construction, the launch zipcode will dictate the SSR to start the workday at this location and he will always return to this place after completing each repair task at the demand zipcode. That means, in our model, a SSR never travels directly from one customer site to another, instead, always returns to the launch zipcode before taking on the next task. Then, the first stage of the problem is the assignment of employees to their launch and work zipcodes considering demand rates, SSR-machine training and skill upgrade costs, and distances between SSR launch zipcodes and customer sites. The location index of the input demand is provided at the zipcode level, because a finer granularity makes the problem intractable. The objective is to find the least cost solution which has the best balance of salary, training, and travelling costs while satisfying all demand. The model is further explained through the following definitions:

Decision Variables:

- mu_i : binary variable if SSR_i is activated or not, where $i \in S$, S is the set of all potential SSRs
- $ys_{i,k}$: binary variable if SSR_i has a job assignment in skill group k or not, where $k \in Sg$, Sg is the set of all skill groups
- $yk_{i,m}$: binary variable if SSR_i requires machine training for machine m or not, where $m \in M$, M is the set of all machines with positive demand
- $x_{m,j,i}$: continuous variable which denotes the number of calls for service for machine m at zipcode j allocated to SSR_i in a month, where $j \in Z$, Z is the set of all zipcodes with positive demand

Parameters:

- ch_i : hourly salary of SSR_i
- cs_k : monthly extra pay for working on skill group k
- ct_m : training cost for machine m, scaled for 1 month
- cd : various expenses of operating a service vehicle for 1 hour (gas, tolls, depreciation etc.)
- $t_{j,i}$: total time it takes to travel from SSR_i's base zipcode to demand zipcode j (round trip)
- $d_{m,j}$: average number of calls expected associated with machine type m at zipcode j in a month
- $rt_{m,j}$: average repair time for a machine of type m at zipcode j

Then the mathematical model is the following:

Problem **Monolithic MIP:**

minimize total_cost :

$$\sum_{i \in S}(125\, ch_i\, mu_i)+$$

$$\sum_{i \in S}\sum_{k \in Sg}(cs_k\, ys_{i,k})+$$

$$\sum_{i \in S}\sum_{m \in M}(ct_m\, yk_{i,m})+$$

$$\sum_{m \in M}\sum_{j \in Z}\sum_{i \in S}(cd\, t_{j,i}\, x_{m,j,i}) \tag{1.1}$$

subject to:

$$\sum_{i \in S} x_{m,j,i} = d_{m,j} \qquad \forall m \in M, j \in Z \tag{1.2}$$

$$\sum_{m \in M}\sum_{j \in Z} x_{m,j,i}(rt_{m,j} + t_{j,i}) \leq 125 mu_i \qquad \forall i \in S \tag{1.3}$$

$$x_{m,j,i} \leq d_{m,j} mu_i \qquad \forall m, j, i \tag{1.4}$$

$$x_{m,j,i} \leq d_{m,j} yk_{i,m} \qquad \forall m, j, i \tag{1.5}$$

$$yk_{i,m} \leq ys_{i,k} \qquad \forall i,\ m,\ k\ \text{—}\ m \text{ belongs to skill group } k \tag{1.6}$$

$$x_{m,j,i} >= 0 \qquad \forall m, j, i \tag{1.7}$$

$$mu_i,\ yk_{i,m},\ ys_{i,k} \quad binary \qquad \forall i, (i,m), (i,k) \tag{1.8}$$

Objective function (1.1) says that if a SSR is activated, there will be a fixed cost of 125 hours times the hourly wage, skill upgrade payments for each skill group he works in, machine training costs for each machine he needs training on, and transportation costs will be incurred for each time he provides service to a customer based on the distance between the launch point and the demand point. Constraint set (1.2) requires that all demands have to be satisfied, (1.3) prescribe 125 hour capacity per SSR in a month, (1.4) is a strengthening constraint set for SSR activation, (1.5) is for machine training activation, (1.6) is to impose machine training on skill upgrades, and

(1.7) and (1.8) are nonnegativity and binary declarations. When the strong constraints are used instead of their weak counterparts, the number of variables in the problem turns out to be less than that of the constraints. There are 2 reasons that we decided to use the strong constraints. First of all, the LP relaxation of this formulation is very weak when strong constraints are abandoned, as expected. Secondly, when the problem is decomposed, the strong constraints will be placed inside the subproblems, and the subproblem solution speed turns out to be even better.

1.3 Likelihood of Assignment Heuristic

A typical problem would have 150 zipcodes with positive demand for a variety of machines (m/c). A feasible solution can locate more than 1 SSR at particular zipcode where demand is very dense, which means there can be a multiple of 150 potential SSRs in the problem. The maximum number of potential SSRs to locate at a zipcode can be roughly calculated by adding up the total demand in a 4 hour radius, and dividing by 62.5 (where 62.5 hours is half of 125, the total available time per month per SSR, owing to the fact that SSRs never tend to use more than 50% of their available capacities for driving in the optimal solutions). In actual practice, the field managers of these service companies add extra SSRs to those zipcodes where they detect total travelling hours to be more than 50% of available time in a given month. The 4-hour radius is an important limit, which permits the results of this planning model to be usable by the second stage scheduling model. The second stage model assumes the work shifts are 8 hours, therefore a round-trip should never exceed 8 hours in total. There are approximately 1000 distinct machines which require special training, and each of those 1000 machines belong to 1 of 30 skill groups. SSRs are paid extra, based on the skill group they are assigned to, and they can belong to more than one.

Thus, the expected number of variables can look like the following, as motivated by the industrial (urban) data sets used:

- 150 zipcodes with positive demand points translate to 300 SSRs to be potentially located on the map, on the average 2 for each zipcode

1.3. LIKELIHOOD OF ASSIGNMENT HEURISTIC

- mu_{SSR}, binary

- 300 SSRs * 1000 machines = 300K possible m/c training variables

  - $yk_{SSR,m/c}$, binary

- 300 SSRs * 30 skill groups = 9K possible skill upgrade variables

  - $ys_{SSR,skill}$, binary

- Average of 20 demand points at every urban zipcode * 150 zipcodes * 300 potential SSRs = 900K variables

  - $x_{m/c,zip,SSR}$, continuous

- TOTAL = **1,209.3K decision variables**

The number of constraints:

- 20 machines * 150 zipcodes

  - 3,000 for demand fulfillment: const. (1.2)

- 300 potential SSRs with limited available hours (125 hours/month)

  - 300 for SSR capacity: const. (1.3)

- SSR activation constraints (as many as x variables)

  - 900K SSR activation and strengthening: const. (1.4)

- Training constraints, (as many as x variables)

  - 900K m/c training activation: const. (1.5)

- 300 potential SSRs * 1000 machines

  - 300K machine training and skill upgrade associations: const. (1.6)

- TOTAL = **2,103.3K constraints**

1.3. LIKELIHOOD OF ASSIGNMENT HEURISTIC

This problem is too big to solve with an acceptable mipgap ($\leq 5\%$) in a reasonable amount of time by CPLEX 12.0, on a computer with 4 GB RAM and Pentium Xeon 3.0 GHz(x2) CPU. After 12 hours, the mipgap is still at 45%. Therefore, to understand how the optimal solutions behave, we have investigated results on smaller instances e.g. with 20 zipcodes, 40 SSRs, 30 machines, 30 skill groups, 10 demand points per zipcode, hence 8,000 assignment variables on the average. For similar urban scenario inputs of the same size data instances, the striking observation is that out of 2,400 ys and yk variables, approximately 115 of them appear in the optimal solution, which is less than 5%. A similar arguement is valid for the x variables, approximately 575 out of 8,000 are active in the solution on the average, when 20 small instances are solved.

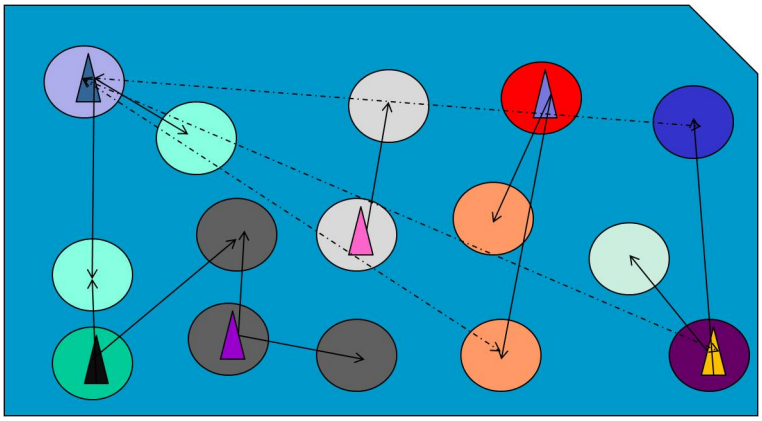


Figure 1.1: An Example of Non-Optimal Assignments

"The Likelihood of Assignment" idea emerged from the fact that we need to start attacking the problem, knowing that only a small portion of the variables are actually going to be candidates for the optimal solution. If one takes a look at the Figure 1.1 above, the circles with different color tones denote the demand points for distinct machines, and different colored triangles represent the SSRs that are launched at those zipcodes with particular skills. While the solid lines between the triangles and the circles are potentially good work assignments, the dashed lines would seem to be the assignments that will possibly not show up in the optimal solution. Since the x variables are the actual triggering ones for ys and yk, if we can figure out which x's are potentially good and which are potentially useless, we can reduce the size and the complexity of the problem substantially. To

accomplish this, a penalty function will be developed to be used for every assignment variable x.

Assessing the value of Likelihood of Assignment for a SSR and a m/c at a zipcode, depends on the following factors, of which the ones with (+) sign denote the factors that adversely affect (hence increase the penalty for assignment), (-) sign denotes the beneficial factors (hence decrease the penalty):

i. Salary of SSR (+) : if salaries differ based on zipcodes: $factor1$

ii. Skill upgrade cost required (+): $factor2$

iii. Machine training cost required (+): $factor3$

iv. Distance * number of demand calls (+): $factor4$

v. Sum of same m/c demand in a 1 and a 2-hour radius (-): $factor5$ and $factor6$

vi. Sum of same skill demand in a 1 and a 2-hour radius (-): $factor7$ and $factor8$

vii. Sum of all closer demands with cheaper m/c training and cheaper skill upgrade (+): $factor9$

The items with the plus sign denote that they have an adverse effect on the Likelihood processes. Thus, the Likelihood penalty will be higher, which means an unlikely assignment. For instance, a SSR would not want to associate himself with a machine that has high skill upgrade and high machine training cost at a remote zipcode. He would instead want to be assigned to a m/c he is already trained on. Therefore, if there are more jobs in the nearby area (in 1-hour radius more preferably, and within at most 2-hour radius) requiring the same training and skills, he would resist new m/c training and skill upgrade costs. Costs are incurred from the perspective of the system, not from the SSR's point of view. The likelihood multipliers are found using the following formula:

$$wR_{m,j,i} = \frac{\sum_{f} \left(100\, w_f \frac{f\_value_{f,m,j,i}}{avg\_f\_value_f} \right)}{|\sum_{f} w_f|} \tag{1.9}$$

(1.9) basically asserts that the assignments are penalized for the SSR-zipcode-m/c associations which demonstrate worse deviation (in terms of $f\_value_{f,m,j,i}$) from their respective average_factor (avg_f_value) for the (+) factors. Moreover, the factors with the minus sign are subject to a decrease in penalty based on their divergence from their averages; hence, making this assignment more likely. Factor weights (w_f) are found by analyzing the importance of each factor in smaller scale problems' optimal solutions. When a factor does not show a significant divergence compared to its average in a favorable or optimal solution, then it has an inconclusive effect on the outcome, a unit weight (equals to 1) is assigned to it. For all other weights for the factors, first a base value is determined (for instance, it is 4 in the example below), and then, each factor weight is found by multiplying the base factor by a multiplier. The multiplier equals the divergence percentage of the factor divided by the base divergence percentage.

The active x variables in various smaller sized problems' optimal solutions demonstrated the following properties, when solved for 20 randomly generated instances with 150 demand points and 30 SSRs:

i. 90% had smaller $factor9$ than the average value of factor9 for all possible combinations.
   $weight_{f9} = 6$

ii. 76% had larger $factor8$ than their corresponding average.
   $weight_{f8} = -5.1$

iii. 74% had larger $factor7$ than their corresponding average.
   $weight_{f7} = -4.9$

iv. 73% had larger $factor6$ than their corresponding average.
   $weight_{f6} = -4.8$

v. 70% had larger $factor5$ than their corresponding average.
   $weight_{f5} = -4.5$

vi. 60% had smaller $factor4$ than their corresponding average.

$weight_{f4} = 4$, the base factor weight.

vii. Factors 1,2 and 3 had inconclusive effects regarding their expected influences; therefore, they will be kept in the penalty function, with a unit weight of 1.

$weight_{f3} = weight_{f2} = weight_{f1} = 1$

Based on the analysis above, the most important factor turns out to be "Sum of all closer demands with cheaper m/c training and cheaper skill upgrade". Sums of same m/c and skill demands in 1 and 2-hour radius are the next significant weight factors for calculating the LoA penalties. Nevertheless, the combination of these factors dictate the Likelihood of Assignment, as laid out in the following simple numerical example:

Suppose the sum of all closer demands with cheaper m/c training and cheaper skill upgrade, $factor9$ for SSR_1 calculated for machine $m1$ at zipcode $j1$, is 300 hours in a month; i.e., $f\_value_{9,m1,j1,SSR_1} = 300$. Suppose, also, that the average for this factor among all SSRs and all machines at all zipcodes is $avg\_f\_value_9 = 200$. That means this demand point $(m1, j1)$ "seems" bad for this SSR_1 compared to the average. On the other hand, let the sum of same m/c demand in 2-hour radius, $factor6$, is 100 hours in a month; i.e., $f\_value_{6,m1,j1,SSR_1} = 100$, and the average for this statistic is $avg\_f\_value_6 = 50$ hours for all SSRs, considering every machine at every zipcode separately. This $factor6$ evaluation, pulls the work assignment towards the favorable side when calculated using the formula in (1.9):

$$wR_{m1,j1,SSR_1} = \frac{(6 * 100 * \frac{300}{200}) + (-4.8 * 100 * \frac{100}{50})}{|6 + (-4.8)|} = \frac{900 - 960}{1.2} = -50.$$

A negative wR actually means a reward for such an assignment in our problem setting. The numerator of the formula decides if the assignment is favorable or not, and the denominator only scales the magnitute; therefore, there is an absolute value sign in the denominator.

Once all wR parameters are calculated, 2 types of lists are creates and stored:

1.3. LIKELIHOOD OF ASSIGNMENT HEURISTIC

i. Best potential SSRs for each zipcode-m/c (sorted list of SSRs for each zipcode-m/c, the SSR with the smallest $wR_{m,j,i}$ is at the top): $type_{1,m,j}$ list

ii. Best potential zipcode-m/c's for each SSR_i (sorted list of zipcode-m/c's for each SSR, the zipcode-m/c with the smallest $wR_{m,j,i}$ is at the top): $type_{2,i}$ list

Our target is to find a quality feasible solution which will also be used as a starting solution for the monolithic problem; therefore, if variables that would unlikely be in the optimal solution are identified and eliminated, and the ones that are potential candidates to appear in the optimal solution are kept, the problem can become simpler to solve. In order to achieve this, take a look at the $type_{1,m,j}$ list introduced above. The first element is the smallest $wR_{m,j,i}$ which corresponds to the SSR_i who will "most likely" provide service to this machine m at this demand zipcode j. Towards bottom of the list, comes the worst assignment possibilities, which are to be eliminated. However, the elimination logic should keep as many x variables as possible in the system so that feasibility is guaranteed. The formal algorithm is given to decide which demand point (m, j) is to be possibly serviced by which SSR_i, and which associations to be killed at this stage:

- **Step 0:** Check if there are any unassigned (m, j) demand points. If none exists, algorithm terminates. Otherwise, continue with Step 1.
- **Step 1:** Find the $type_{1,m,j}$ list with the smallest $wR_{m,j,i}$ as its first element.
- **Step 2:** Assign as much of the demand ($d_{m,j}$) to the SSR_i who is at the top of the $type_{1,m,j}$ list for this demand point.
- **Step 3:** Update $d_{m,j}$ for the remaining amount. Remove the repair hours and travelling times from SSR_i's capacity. Remove SSR_i from the top of $type_{1,m,j}$ list.
- **Step 4:** If $d_{m,j}$ is completely fulfilled, and if the last chosen SSR_i for this (m, j) was at the top 10% of updated $type_{1,m,j}$, add the remaining SSR_i's who are also at the top 10% of the updated $type_{1,m,j}$. These additions secure the maintainance of feasibility and let the Problem

Integer_Yielding_LP explore more options. All other assignment variables associated with this demand point are then fixed to 0.

($x_{m,j,ii} = 0, \quad ii \notin$ chosen set of SSR_i's).

Go to Step 4b.

- **Step 4b:** If SSR_i's capacity is fully depleted, remove SSR_i from all $type_{1,m,j}$ lists, i.e. his assignment variables regarding the remaining demand points are set to 0.

  ($x_{mm,jj,i} = 0, \quad (mm, jj) \in$ unassigned demand points set).

  Go to Step 0. If not fully depleted, go to Step 5.

- **Step 5:** If $demand_{m,j} > 0$, go to Step 2.

In spite of the fact that assignments are greedily made to SSRs at **Step 3**, the primary purpose was only to make sure the model has enough SSRs and assignment variables in the Restricted Feasible Domain, which is the whittled down version of the original feasible region. Actual assignments will be made in the next section, where a LP will be used in order to determine the actual rationing of the jobs among the SSRs. This LP will not be using the real costs in the original model.

By using the heuristic above, the problem reduces to approximately 5-10% of its original size depending on the input data, which is a little more than a good quality solution's typical number of active variables.

1.3.1 Finding an Initial Feasible Solution

Now, assume we were able to represent all the distinct costs of salaries, m/c training, skill upgrades, and travelling for a particular SSR to a m/c at a demand zipcode in one expression, $wR_{m,j,i}$, the Likelihood of Assignment parameter previously discussed. Remember that as the value of wR increases, the likelihood decreases, just as in a minimization problem where if a cost associated with a variable gets higher, the possibility of that variable being in the optimal solution becomes lower.

Therefore, if the salary, machine training, and skill upgrade costs are approximately represented in the wR parameter, then the binary variables corresponding to these costs are not needed explicitly in the model to find a feasible assignment. Furthermore, post-processing the work assignment variables x will result in which mu, ys and yk binary variables actually get to be activated in the original model, using constraints (1.4), (1.5) and (1.6).

The resulting model would be the following:

Problem **Integer_Yielding_LP**:

minimize LoA penalties:

$$\sum_{m\in M}\sum_{j\in Z}\sum_{i\in S} x_{m,j,i} wR_{m,j,i} \tag{1.10}$$

subject to :

$$\sum_{i\in S} x_{m,j,i} = d_{m,j} \qquad \forall m,j \tag{1.11}$$

$$\sum_{m\in M}\sum_{j\in Z} x_{m,j,i}(rt_{m,j} + t_{j,i}) \leq 125 \qquad \forall i \tag{1.12}$$

$$x_{m,j,i} >= 0 \qquad \forall m,j,i \tag{1.13}$$

The optimal objective function value of this problem is neither an upper nor a lower bound for the original problem. After post-processing the flow variables to retrieve the value of the binary variables, the upperbound will be calculated from a core solution which will be improved upon in Chapter 2.

1.4 Computational Results

Computational results for problem sizes of 200x20, 300x30, 400x40, 600x60, 1000x100, 1500x300 (demand points x SSRs) in 3 types of demand scenario settings are presented:

i. Assume uniform demand over all zipcodes for all machines (extreme scenario - very unlikely

to happen in real life)

ii. Assume high volume demand for certain machines at certain zipcodes, very low or no demand in others (rural scenario)

iii. Assume high volume demand for certain machines at certain zipcodes, normal level demands for others (urban scenario)

LoA performs quite well with the urban and rural scenario settings. The urban scenario has higher volume demand located at more zipcodes, which corresponds to more variables and hence more possibilities to choose from; therefore, both CPLEX and LoA solve the urban scenario more slowly than the rural scenario problems. LoA's solutions for urban scenarios are slightly worse than the rural setting, again due to the size of the problem getting larger.

However, for the extreme scenario, where all demands for all machines are uniformly distributed throughout the territory, LoA still solves rapidly compared to CPLEX, though, the solution quality is worse than for the other cases. That is a drawback of our heuristic; which is observed when the weight factors defined at the beginning of the chapter are not discernably different among competing assignments. The assignments will be made mostly based on distances; the skill upgrades and machine trainings will have very insignificant effect on the solution.

The following tables are constructed by solving 20 randomly generated instances for each problem size for each of these 3 scenarios on a computer with 4 GB RAM and Pentium Xeon 3.0 GHz(x2) CPU. The solution times and best mipgaps are averaged for each problem size. The formula for $mipgap$ is given in the CPLEX User's Manual [33] as:

$$mipgap = \frac{|UB_{best} - LB_{best}|}{|UB_{best}| + 1e^{-10}}$$

#vars denotes the number of variables in the original problem. Likewise, #vars_LoA denotes the number of variables in the Problem Integer_Yielding_LP, which is approximately 10% of the original size, regardless of scenario type. Cplex_T is the CPLEX solution time obtained at 1% mipgap. Best

mipgaps obtained for LoA (LoA_Gap) are calculated from the best lowerbound CPLEX reports. LoA_T is the total time it takes to set up the heuristic (reading data, sorting the lists, file outputs for fixings and eliminations), solve it to optimality and post-process to attain original variable values. Cplex_T_LoA is the time it takes CPLEX to discover a feasible solution with the same mipgap LoA finds. All solution times are reported in seconds.

A careful inspection of the tables below would help the reader to realize the significance of the need for a heuristic to solve this problem. CPLEX performance is not at acceptable levels starting from the (400x40) test case. Our LoA heuristic, on the other hand, solves all test cases in less than 100 seconds, with an average mipgap of 7.5% for rural and urban scenarios. The same quality solutions are obtained by CPLEX after more than half an hour for (400x40), after 2 hours for (600x60), after 5 hours for (1000x100). More than 2 days of run time is required to match the LoA solution by CPLEX, for (1500x300) size problems.

Table 1.1: LoA vs CPLEX for Urban Demand Scenarios

Size	#vars	Cplex_T	#vars_LoA	LoA_Gap	LoA_T	Cplex_T_LoA
200x20	8,620	65	749	5.4%	12	14
300x30	18,930	369	1,498	6.1%	13	275
400x40	33,240	2,711	2,568	7.5%	30	2,114
600x60	73,860	8,753	5,671	9.9%	46	7,499
1000x100	203,100	24,543	15,408	10.8%	66	19,345
1500x300	1,209,300	224,884	91,699	11.8%	85	187,234

Table 1.2: LoA vs CPLEX for Rural Demand Scenarios

Size	#vars	Cplex_T	#vars_LoA	LoA_Gap	LoA_T	Cplex_T_LoA
200x20	8,330	61	1,070	4.9%	11	13
300x30	17,630	346	2,247	5.5%	12	223
400x40	32,640	2,551	3,531	6.8%	28	1,904
600x60	71,960	8,235	6,313	8.5%	42	7,129
1000x100	201,200	22845	15,408	9.1%	60	17,825
1500x300	1,202,150	219,065	76,505	9.5%	80	168,766

Table 1.3: LoA vs CPLEX for Uniform Demand Scenarios

Size	#vars	Cplex_T	#vars_LoA	LoA_Gap	LoA_T	Cplex_T_LoA
200x20	9,520	72	762	13.2%	13	13
300x30	20,130	410	1,651	14.2%	14	320
400x40	35,040	3,013	2,667	16.4%	31	2,544
600x60	74,820	8,502	5,715	19.1%	46	7,677
1000x100	205,760	23,903	15,621	24.2%	69	20,320
1500x300	1,232,800	232,244	93,472	33.2%	91	77,890

1.5 Discussion

Likelihood of Assignment is a greedy algorithm; but, its foundation is very flexible, which allows for user-specific, problem-specific and data-specific tunings. In Chapter 4, guidance for how to increase the quality of the solutions will be provided.

As can be seen from the computational results, the heuristic being proposed is finding high quality solutions very fast for the rural and urban scenarios, for the purpose of initializing the B&P algorithm to be introduced in the next chapter. The $wR_{m,j,i}$ parameters are randomized with a

multiplier from $Uniform \sim [0.90, 1.10]$ to obtain diverse (and sometimes even better) columns. These columns are used in the initial basis of the first Restricted Master Problem of the ColGen procedure. The byproducts of LoA, for instance the $type_{1,m,j}$ lists, will also be used to accelerate and stabilize the ColGen Algorithm.

To conclude, most real life problems do not possess the extreme scenario demand structure; therefore, we are confident that the LoA heuristic performs quite well under the predominant urban and rural scenario cases.

Chapter 2

An Exact and Stabilized Branch-and-Price Algorithm for Workforce Planning Problems with Cross-Training

2.1 Introduction

As shown in the previous chapter, although a reasonably good quality solution was obtained very quickly using the Likelihood of Assignment (LoA) heuristic, the optimal solution was not found in any of the instances. In this chapter, an exact Branch-and-Price (BP) algorithm will be developed in order to achieve that goal. Despite the fact that the direct application of the Column Generation (ColGen) procedure, as defined in the textbooks, failed in terms of convergence speed and lower bound quality, 3 techniques to make ColGen effective for the Workforce Planning Problems with Cross-Training will be proposed. The LoA heuristic will play a critical role, not only in initializing the core solution to the Restricted Master Problem (RMP) of the ColGen, but also, the findings of

LoA will be used in the stabilization and acceleration methods, that will be explained later in this chapter.

The outline of this chapter follows: Firstly, a reformulation of the Workforce Planning Problem with Cross-Training will be given. This is going to allow the reader to capture the structure of the dual perspective of the problem and enable an intuitive understanding of the methods proposed. Secondly, the RMP and the pricing problems (subproblems) will be introduced. Next, the acceleration and the stabilization methods will be shown with their effects at the root node of the ColGen. Finally, the complete BP algorithm will be displayed for obtaining optimality.

2.2 Reformulation for Column Generation

Recall the original formulation of the Workforce Planning Problem:

minimize total_cost:

$$\sum_{i \in S}(125\, mu_i\, ch_i)+$$

$$\sum_{i \in S}\sum_{k \in Sg}(cs_k\, ys_{i,k})+$$

$$\sum_{i \in S}\sum_{m \in M}(ct_m\, yk_{i,m})+$$

$$\sum_{m \in M}\sum_{j \in Z}\sum_{i \in S}(cd\, t_{j,i}\, x_{m,j,i}) \tag{2.1}$$

subject to:

$$\sum_{i \in S} x_{m,j,i} = d_{m,j} \qquad \forall m \in M, j \in Z \tag{2.2}$$

$$\sum_{m \in M}\sum_{j \in Z} x_{m,j,i}\,(rt_{m,j} + t_{j,i}) \leq 125\, mu_i \qquad \forall i \in S \tag{2.3}$$

$$x_{m,j,i} \leq d_{m,j}\, mu_i \qquad \forall m, j, i \tag{2.4}$$

$$x_{m,j,i} \leq d_{m,j}\, yk_{i,m} \qquad \forall m, j, i \tag{2.5}$$

$$yk_{i,m} \leq ys_{i,k} \qquad \forall i,\ m,\ k, \text{ and } m \text{ belongs to skill group } k \tag{2.6}$$

$$x_{m,j,i} >= 0 \qquad \forall m, j, i \tag{2.7}$$

$$mu_i,\ yk_{i,m},\ ys_{i,k} \quad binary \qquad \forall i, (i,m), (i,k) \tag{2.8}$$

When the constraint sets are carefully inspected, it can be seen that the demand constraints (2.2) are the only ones that bind all SSRs. This block angular property allows the problem to be decomposed by SSR, as it is done in Dantzig-Wolfe Decomposition [14] where the demand constraints (2.2) will be handled inside the Restricted Master Problem, and the remaining constraints (2.3-2.8) will be put into SSR subproblems. Although we could continue with a direct application of D-W decomposition scheme, a new formulation will be introduced in order to facilitate the understanding of what is to happen in the algorithm to be proposed.

The reformulation is based on this new binary variable $Yt_{i,q}$, representing a SSR_i with all the necessary attributes attached to him found during a ColGen iteration q (or during the initialization phase by LoA). This method is widely used in the Aircrew Planning [26] and Vehicle Routing Problems [21] where this variable denotes a pilot's schedule of flights, and a vehicle's route, respectively. In our context, Yt will be representing a SSR's set of machine trainings, skill upgrades, and the rate of service he provides at the customer locations. Thus, every solution to the SSR subproblems will be bundled as a Yt variable and the RMP will decide if the SSR corresponding to this Yt variable is beneficial enough to be activated or not.

Each $Yt_{i,q}$ has its own cost parameter $Ct_{i,q}$, which is computed in the following formula after finding which machines and skills and zipcodes SSR_i is associated with during iteration q:

$$\begin{aligned} Ct_{i,q} =& 125\, ch_i\, mu_{i,q} + \\ & \sum_{k \in Sg} (cs_k\, ys_{i,k,q}) + \\ & \sum_{m \in M} (ct_m\, yk_{i,m,q}) + \\ & \sum_{m \in M} \sum_{j \in Z} (cd\, t_{j,i}\, x_{m,j,i,q}) \end{aligned} \tag{2.9}$$

Having defined $Ct_{i,q}$, the objective function transformation is now complete. Since the constraints (2.3) through (2.8) are to be moved into the subproblems, only the demand constraints (2.2) need to be transformed:

$$\sum_{i \in S} \sum_{q \in itrs} x_{m,j,i,q}\, Yt_{i,q} >= d_{m,j} \qquad (\pi_{m,j,q}) \qquad \forall m, j \tag{2.10}$$

The summation in (2.10) ensures that the demand for every (m, j) is satisfied by either the columns obtained from LoA initialization or the ones found during column generation iterations. The set $itrs$ will be used to denote the set of columns that currently exist in the RMP regardless of where they are found. The $\geq$ sign is used in (2.10) to obtain more stabilized duals ($\pi_{m,j,q}$) for these

constraints, as opposed to these constraints originally being the equality type. This conversion, however, imposes no change on the optimal solution of the problem due to the fact that it is a minimization problem after all.

The last set of constraints to be put inside the reformulation are the so-called convexity constraints:

$$\sum_{q \in itrs} Yt_{i,q} \leq 1 \qquad \theta_{i,q} \qquad \forall i \in S \tag{2.11}$$

(2.11) ensures that no more than 1 bundle of decisions for each SSR is to be chosen when the RMP is discretized and solved for an integer solution. $\theta_{i,q}$ are the corresponding dual variables. Convexity constraints regarding the extreme rays will not be included in the reformulation, because it will be proved in the next section that the subproblem feasible regions are all bounded. Thus, the Problem Reformulation of the original monolithic problem is finalized:

Problem **RMP_integer**:

minimize total_cost:

$$\sum_{i \in S} \sum_{q \in itrs} Ct_{i,q}\, Yt_{i,q} \tag{2.12}$$

subject to:

$$\sum_{i \in S} \sum_{q \in itrs} x_{m,j,i,q}\, Yt_{i,q} \geq d_{m,j} \qquad (\pi_{m,j,q}) \qquad \forall m, j \tag{2.13}$$

$$\sum_{q \in itrs} Yt_{i,q} \leq 1 \qquad (\theta_{i,q}) \qquad \forall i \in S \tag{2.14}$$

$$Yt_{i,q} \quad binary \qquad \forall i \in S,\ q \in itrs \tag{2.15}$$

When only a subset of all possible q's are considered, this condensed and small version (in terms of number of variables) of the problem is tractable and simple to solve, because it is going to be the Restricted Master Problem that will be solved every time new improving columns are brought in. If

the objective function and the constraints are expanded to cover all possible q's, then, the restricted problem becomes the extensive formulation. It is equivalent to the original monolithic formulation, for which the formal proof will be given after the subproblems are presented, using the well-known Resolution Theorem [54] and Dantzig-Wolfe Decomposition [14] rules. Within that context, each $Yt_{i,q}$ then denotes the weight of the solution (extreme point) found in the corresponding subproblem of SSR_i.

There is a slight issue concerning the initialization of the RMP. In order to claim that any feasible integer solution to Problem RMP_integer is also feasible for the original problem, the RMP has to be initialized with a set of $Yt_{i,q}$ that conforms to all constraints (2.3-2.8) in the subproblems along with the demand constraints (2.10) and the convexity constraints (2.11). Consequently, a ColGen user, most of the time, tries to initialize the RMP with the integer solution in mind.

On the other hand, ColGen at each node of the Branch-and-Price tree actually works toward optimizing the LP relaxation of the RMP at that iteration. Therefore, even an excellent integer solution as the initial basis is detrimental to solving a linear program by Colgen [61]. Poorly chosen initial columns lead the algorithm astray when they do not resemble the structure of the possible optimal solution of the RMP. That is why after the best heuristically found integer solution from our Likelihood of Assignment is obtained, to enrich the solution pool, more instances of the Integer_Yielding_LP are solved with $wR[m, j, l]$ multipliers which are randomized by $Uniform \sim [0.90, 1.10]$ distribution. After the randomization, there are instances where new solutions are even better in terms of the original objective function value. The largest test instance has on the average of 50 active SSRs in the optimal solution, which is the approximate headcount obtained from the Problem Integer_Yielding_LP as well. In Chapter 4, it is shown that solving Problem Integer_Yielding_LP 20 times (i.e. initializing the RMP with approximately 1,000 columns) outweighs the solving fewer times, because too few columns produce non-useful duals (π's and θ's) owing to the fact that the dual feasible region is in that case very relaxed and not tight. On the contrary, too many columns make it harder and longer to solve the LP of the RMP and RMP_integer. Further guidelines on randomization and number of initial columns are given in Chapter 4 for distinct cases

and data inputs.

For a better comprehension of how the dual vectors obtained from the RMP affect the performance of the ColGen procedure, the next section explains the pertinence in primal versus dual domain relationships.

2.2.1 Dual of RMP

The dual of the RMP is the following model:

Problem **RMP_dual:**

maximize:

$$\sum_{m \in M} \sum_{j \in Z} d_{m,j}\, \pi_{m,j,q} + \sum_{i \in S} \theta_{i,q} \tag{2.16}$$

subject to:

$$\sum_{m \in M} \sum_{j \in Z} x_{m,j,i,q}\, \pi_{m,j,q} + \theta_{i,q} \leq Ct_{i,q} \qquad (Yt_{i,q}) \qquad \forall i \in S,\, q \in itrs \tag{2.17}$$

$$\pi_{m,j,q} \geq 0 \qquad \forall m, j \tag{2.18}$$

$$\theta_{i,q} \leq 0 \qquad \forall m, j \tag{2.19}$$

The dual variable $\theta_{i,q}$ corresponds to the convexity constraints (2.14) for every SSR_i and $\pi_{m,j,q}$ is associated with constraints (2.13) for every demand point (m, j). Conversely, $Yt_{i,q}$ variables correspond to the dual feasibility constraints (2.17), above.

A careful reader would notice that when the dual feasibility constraints (2.17) are rearranged,

the reduced cost function for every $Yt_{i,q}$ is obtained and will be referred to as $\bar{Yt}_{i,q}$:

$$\sum_{m\in M}\sum_{j\in Z} x_{m,j,i,q}\,\pi_{m,j_q} + \theta_{i,q} \leq Ct_{i,q} \qquad \forall i \in S,\, q \in itrs \tag{2.20}$$

is equivalent to:

$$Ct_{i,q} - (\sum_{m\in M}\sum_{j\in Z} x_{m,j,i,q}\,\pi_{m,j,q} + \theta_{i,q}) \geq 0 \qquad \forall i \in S,\, q \in itrs \tag{2.21}$$

then the reduced cost of $Yt_{i,q}$ is equal to:

$$\bar{Yt}_{i,q} = Ct_{i,q} - (\sum_{m\in M}\sum_{j\in Z} x_{m,j,i,q}\,\pi_{m,j,q} + \theta_{i,q}) \qquad \forall i \in S,\, q \in itrs \tag{2.22}$$

In a minimization problem, a solution is optimal when all variables have positive reduced costs, that means expression (2.21) has to hold for every $Yt_{i,q}$, namely $\bar{Yt}_{i,q} \geq 0$. This is also equivalent to saying that the dual of the original problem is also feasible (and hence optimal), since all $\bar{Yt}_{i,q}$'s being greater than 0 means all constraints of type (2.17) hold.

In order to check if a variable has a negative or positive reduced cost for a given RMP at some iteration q, the dual variables in the evaluation of the expression (2.22) must be feasible to the Problem RMP_dual at that ColGen iteration. The importance of this check will be more apparent when the initial dual vector is estimated in Section **2.3.1**. Briefly, an infeasible dual vector can cause the subproblems either to bring already existing columns (claiming they have negative reduced cost) or produce no incoming columns which means a false optimality is reached. Moreover, a lower bound calculated using an infeasible dual vector would not be valid.

2.2.2 Pricing Subproblems

At any given iteration of ColGen, a solution of the LP of the RMP provides the best combination of Yt variables and dual variables π and θ for their corresponding constraints. The next check that needs to be done is if this solution is optimal or not. In a default ColGen algorithm, these dual variables are used in the pricing subproblems to find whether there are any variables that have not

been considered yet and if they have a negative reduced cost or not. Thus, the pricing subproblems are utilized either to do this optimality check when all available subproblems are used, or to find the best incoming column from every subproblem chosen to be solved.

Dantzig-Wolfe [14] recommends to bring the column with the most negative reduced cost at every iteration. That is equivalent to adding a constraint (2.17) which is most violated in the dual domain. This column also needs to satisfy all the constraints that are not covered in the RMP. Recall that the original problem can be decomposed by SSR_i owing to the nice block angular structure. Therefore, when the pricing subproblems yield columns with negative reduced cost, the practical meaning of this is SSRs with the best combination of machine training, skill upgrade, and work assignments are found, while complying with their capacity constraints. This entails that the objective function of the pricing subproblems should be the reduced cost expression introduced in (2.22). The constraints of the pricing subproblems then become (2.3-2.8).

Pricing Subproblem for SSR_i at iteration q:

minimize reduced cost:

$$\bar{Yt}_{i,q} = Ct_{i,q} - \Big(\sum_{m\in M}\sum_{j\in Z} x_{m,j,i,q}\,\pi_{m,j,q} + \theta_{i,q}\Big) \tag{2.23}$$

subject to:

$$\sum_{m\in M}\sum_{j\in Z} x_{m,j,i,q}\,(rt_{m,j} + t_{j,i}) \le 125\,mu_{i,q} \tag{2.24}$$

$$x_{m,j,i,q} \le d_{m,j}\,mu_{i,q} \qquad \forall m, j \tag{2.25}$$

$$x_{m,j,i,q} \le d_{m,j}\,yk_{i,m,q} \qquad \forall m, j \tag{2.26}$$

$$yk_{i,m,q} \le ys_{i,k,q} \qquad \forall\ m,\ k, \text{ and } m \text{ belongs to skill group } k \tag{2.27}$$

$$x_{m,j,i,q} >= 0 \qquad \forall m, j \tag{2.28}$$

$$mu_i, q,\ yk_{i,m,q},\ ys_{i,k,q} \quad binary \qquad \forall (i,m), (i,k) \tag{2.29}$$

where $Ct_{i,q}$ is defined in (2.9).

A quick inspection of the pricing subproblem for SSR_i reveals that it is almost identical to the original monolithic MIP model without the demand constraints, and written only for 1 SSR. The demand constraints are not inside SSR subproblems because they are already taken care of inside the RMP. The practical explanation of the objective function is that the dual variables of type π from the RMP will behave as the SSR's reward to fulfill a job, and $Ct_{i,q}$ is composed of all the pertinent costs for providing these services in (2.9). On the other hand, the dual variable $\theta_{i,q}$ from the convexity constraint (2.14) impose a barrier for this SSR's activation. The 125 hour availability is the capacity constraint. Machine training and skill upgrade constraints are maintained in order to keep track of necessary binary associations.

The pricing subproblem either concludes with a negative reduced cost yielding column, or a 0 value for the objective function. The latter means, this subproblem SSR_i cannot find the dual constraint which violates dual feasibility. An industrial explanation is that the rewards provided to SSR_i are not large enough considering the π vector; therefore, SSR_i chooses not to work in this iteration.

This subproblem, therefore, can be classified as a knapsack type problem, where a SSR_i is expected to fill his 125 hour capacity with the most rewarding and least costly jobs. Even for the largest test case, this subproblem has at most 2,500 variables and 4,000 constraints, which can easily be solved to optimality in less than 0.01 seconds on a computer with 4 GB RAM and Pentium Xeon 3.0 GHz(x2) CPU.

Before the formal steps of a generic ColGen algorithm are provided, a brief description of such an algorithm is provided: At every iteration of a ColGen algorithm, a RMP is solved in order to obtain the best values of the primal variables. To check if this is the optimal solution, pricing problems are solved to detect if there are any other primal columns with negative reduced cost. This has the same meaning as to find if there are any constraints that are violated in the dual domain of the original problem. Once such primal variables are found, they are added to the primal RMP (therefore, the corresponding constraints are automatically added to the Problem RMP_dual). This procedure guarantees that a new dual variable is obtained at every new RMP, and the dual vector

causing the dual infeasibility is removed. However, the objective function is not guaranteed to improve on the primal side, because the problem may be highly degenerate. The algorithm terminates when no implicit column has a negative reduced cost. Nevertheless, it is a very common experience among ColGen users that the issue known as *"tailing-off"* is encountered as the iterations progress. Tailing-off is detected when the incoming columns either do not improve the upperbound for the RMP for over a certain number of iterations, or the improvements become numerically insignificant.

2.2.3 A Generic Colgen Algorithm

Certain statistics need to be defined in order to construct a ColGen Algorithm:

- $Yt^*_{i,qq}$ is the primal solution to the RMP, $\forall(i, qq)$.
- $\hat{Yt_{i,qq}}$ is the integer feasible solution to the RMP_integer.
- UB^q_n is the upper bound for the RMP at iteration q, i.e. the objective function value (2.12) of the optimal solution to the RMP at iteration q provides this number. This number is expected to monotonically decrease as the ColGen iterations progress.
- LB^q_n is the lower bound for the RMP at iteration q, and monotonic increase is not guaranteed. It is calculated with the following expression:

$$\begin{aligned} LB^q_n =& UB^q_n + \sum_{i \in S} \bar{Yt}_{i,q} \\ =& \sum_{i \in S} \sum_{qq \in itrs} Ct_{i,qq}\, Yt^*_{i,qq} + \sum_{i \in S} \bar{Yt}_{i,q} \end{aligned} \tag{2.30}$$

- UB_n is the best upper bound found; i.e., $UB_n = min\{UB^q_n | q \in itrs\}$, and guaranteed to monotonically decrease.
- LB_n is the best lower bound found; i.e., $LB_n = max\{LB^q_n | q \in itrs\}$, and guaranteed to monotonically increase.

- UB_{mip} is the objective function value corresponding to the best integer feasible solution found by solving Problem RMP_integer.

Then, the steps of the algorithm are the following:

- **Step 0:** Initialize RMP, with a heuristic solution or any feasible solution out of phase1 of the Simplex Algorithm. Initial columns form the set:
  $\{Yt_{i,q} \mid \text{all } (i,q) \text{ couples are enumerated from } q = 0 \text{ and distinct}\}$. Set node $n = 0, UB_n = \infty, LB_n = -\infty$.

- **Step 1:** Solve RMP. Obtain UB_n^q, $Yt_{i,qq}^*$, $\pi_{m,j,qq}^*$ and $\theta_{i,qq}^*$.

- **Step 2:** Update $UB_n = min\{UB_n^q | q \in itrs\}$.

- **Step 3:** Solve Pricing Subproblem SSR_i, $\forall i \in S$. Obtain $\bar{Yt}_{i,q}$, and $x_{m,j,i,q}$, $mu_{i,q}$, $yk_{i,m,q}$ and $ys_{i,k,q}$, which are then used to calculate $Ct_{i,q}$.

- **Step 4:** Calculate LB_n^q.

- **Step 5:** Update $LB_n = max\{LB_n^q | q \in itrs\}$.

- **Step 6:** Optimality check:
  if $UB_n = LB_n + \epsilon$, where $\epsilon = 0.001$ terminate. Go to Step 7.
  Else update set $\{Yt_{i,q}\}$ with (i,q) having $\bar{Yt}_{i,q} < 0$. Update $q = q + 1$. Go to Step 1.

- **Step 7:** Solve RMP_integer. Obtain UB_{mip} and $\hat{Yt}_{i,qq}$. Extract the values for $\hat{x_{m,j,i,q}}$, $\hat{mu}_{i,qq}$, $\hat{yk_{i,m,qq}}$ and $\hat{ys_{i,k,qq}}$ from the construction of $\hat{Yt}_{i,qq}$.

Our experience shows very problematic run times even for small sized problems using this generic setting. The numerical results are provided for convenience in **Section 2.4**. These unacceptable run times are the primary reason why the following Acceleration and Stabilization Techniques are proposed in the next section. The reasons for the bad convergence will also be explained.

2.3 Acceleration and Stabilization Proposals for Column Generation

2.3.1 Estimating the Initial Dual Vector

One of the most common critisms that ColGen receives is that the generic algorithm fails to provide a valid (non-negative) lower bound at the beginning of the process. This workforce planning model's objective function must have a positive value as long as all the demand constraints are met. That implies even the best solution will have a positive cost with loose lower bound of 0. A generic ColGen algorithm, unfortunately, starts with a negative lower bound, and then takes 50-100 iterations to bring the lower bound into the positive territory. We claim that if we can estimate a "balanced" initial feasible dual vector, this so-called heading-in issue could be avoided. Most of the entries in a balanced dual vector are in the same order of magnitude, and very few of them are 0.

A lower bound can be calculated at any iteration q of ColGen, if and only if, all pricing subproblems for $i \in S$ are solved after the RMP. An upper bound can be calculated from both the primal objective function and dual objective function (by strong duality):

$$\begin{aligned} UB_n^q &= \sum_{i \in S} \sum_{qq \in itrs} Ct_{i,qq}\, Yt^*_{i,qq} \\ &= \sum_{m \in M} \sum_{j \in Z} d_{m,j}\, \pi^*_{m,j,q} + \sum_{i \in S} \theta^*_{i,q} \end{aligned} \tag{2.31}$$

$$
\begin{aligned}
LB_n^q =& UB_n^q + \sum_{i \in S} Y\bar{t}_{i,q} \\
=& \sum_{i \in S} \sum_{qq \in itrs} Ct_{i,qq}\, Yt^*_{i,qq} + \sum_{i \in S} Y\bar{t}_{i,q} && (2.32) \\
=& \sum_{m \in M} \sum_{j \in Z} d_{m,j}\, \pi^*_{m,j,q} + \sum_{i \in S} \theta^*_{i,q} + \sum_{i \in S} Y\bar{t}_{i,q} && (2.33) \\
=& \sum_{m \in M} \sum_{j \in Z} d_{m,j}\, \pi^*_{m,j,q} + \sum_{i \in S} \theta^*_{i,q} + \sum_{i \in S} \left(Ct_{i,q} - (\sum_{m \in M} \sum_{j \in Z} x_{m,j,i,q}\, \pi^*_{m,j,q} + \theta^*_{i,q}) \right) \\
=& \sum_{m \in M} \sum_{j \in Z} d_{m,j}\, \pi^*_{m,j,q} + \sum_{i \in S} \left(Ct_{i,q} - (\sum_{m \in M} \sum_{j \in Z} x_{m,j,i,q}\, \pi^*_{m,j,q}) \right) && (2.34)
\end{aligned}
$$

Equation (2.32) shows the formula from the primal side, equation (2.33) from the dual perspective, and equation (2.34) is the expanded dual view. A careful examination of the expanded dual view gives hints about why the first solution to the dual vector from the first RMP might cause negative lowerbounds.

When some of the $\pi^*_{m,j,q}$, which are obtained by the solution of the RMP have very large values (rewards) in terms of absolute magnitude, these cause the pricing subproblems to propose more than the necessary number of SSRs working on the same (m, j) demands with $x_{m,j,i,q}$ variables over and over again. Whereas, in practice and in optimal solutions, only those SSRs, who have the best combination of skills and distance relationships should be assigned to these (m, j) demand points. When $\pi^*_{m,j,q}$ have extremely large values for reward, the expression with the minus sign in front in the equation (2.34) grows too fast. Since the amount of supplied service exceeds the demand too much, the first part of the equation $(\sum_{m \in M} \sum_{j \in Z} d_{m,j}\, \pi^*_{m,j,q})$ cannot balance that negative sum.

It is also known that $Ct_{i,q}$ grows with the increasing values of $x_{m,j,i,q}$ (see the formula (2.9) for reference). However, the rewards already have to be larger than the cost factors inside $Ct_{i,q}$ to propose activated SSRs. That means, since the reason for negative lower bounds is known, a remedy

can be proposed.

The remedy is simply estimating a balanced dual vector; balanced in the sense that the occurrence in which some of the $\pi^*_{m,j,q}$ being extremely large, and most of them being 0 should be avoided. This estimation has to comply with Problem RMP_dual's constraints:

$$\sum_{m\in M}\sum_{j\in Z} x_{m,j,i,q}\,\pi_{m,j,q} + \theta_{i,q} \le Ct_{i,q} \qquad (Yt_{i,q}) \qquad \forall i \in S,\, q \in itrs \tag{2.35}$$

$$\pi_{m,j,q} \ge 0 \qquad \forall m, j \tag{2.36}$$

$$\theta_{i,q} \le 0 \qquad \forall m, j \tag{2.37}$$

Knowing $x_{m,j,i,q}$ and $Ct_{i,q}$ values from the pricing subproblems and rearranging (2.35), the feasibility condition is obtained for each estimate in the dual vector :

$$0 \le Ct_{i,q} - \sum_{m\in M}\sum_{j\in Z} x_{m,j,i,q}\,\pi_{m,j,q} - \theta_{i,q} \qquad \forall i \in S \tag{2.38}$$

$$x_{m,j,i,q}\,\pi^*_{m,j,q} \le Ct_{i,q} - \sum_{mm\in M}\sum_{jj\in Z} x_{mm,jj,i,qq}\,\pi_{mm,jj,qq} - \theta^*_{i,q} \qquad \forall i \in S \tag{2.39}$$

$$\pi^*_{m,j,q} \le \left(Ct_{i,q} - \sum_{mm\in M}\sum_{jj\in Z} x_{mm,jj,i,qq}\,\pi_{mm,jj,qq} - \theta^*_{i,q} \right) / x_{m,j,i,q} \qquad \forall i \in S \tag{2.40}$$

Equation (2.40) asserts that the value for the new estimate for $\pi^*_{m,j,q}$ depends on the other $\pi_{mm,jj,q}$ values that are either previously set or estimated. Those $\pi^*_{m,j,q}$ which are found to be extremely large from the solution of the RMP will be reduced to a smaller value, and will become the $\pi_{mm,jj,q}$ in the above equation. Then, we will only try to estimate positive values for the duals which are found to be 0 originally. In order to accomplish that the following algorithm is proposed:

- **Step 1:** Obtain $\pi^*_{m,j,q}$ and $\theta^*_{i,q}$ from the first RMP solution. All constraints initially to be marked as in set SNP, those with set of duals not positive.

- **Step 2:** Calculate the average dual entry value: $\bar{\pi_{m,j,q}} = \dfrac{\sum_{m\in M}\sum_{j\in Z}\pi_{m,j,q}d_{m,j}}{\sum_{m\in M}\sum_{j\in Z}d_{m,j}}$. If $\pi^*_{m,j,q} > \bar{\pi_{m,j,q}}$, set $\pi^*_{m,j,q} = \bar{\pi_{m,j,q}}$.

- **Step 3:** Mark the constraints whose dual variables are set to a positive value and remove from set SNP. If set $SNP = \emptyset$, Stop.

- **Step 4:** Sort the demand $d_{m,j}$ values for all (m, j) in descending order in set SNP.

- **Step 5:** For (m, j) at the top of the list, use the following formula to obtain the estimate for $\pi_{m,j,q}$:

$$\pi_{m,j,q} = \left(min \left\{ \frac{Ct_{i,q} - \sum_{mm\in M}\sum_{jj\in Z} x_{mm,jj,i,qq}\,\pi_{mm,jj,qq} - \theta^*_{i,q}}{x_{m,j,i,q}} | \forall i, q; x_{m,j,i,q} > 0 \right\} \right) \frac{1}{|SNP|} \tag{2.41}$$

- **Step 6:** Go to Step 3.

Step 5 simply makes sure that, knowing that the same $\pi_{m,j,q}$ is to be used in the calculation of dual feasibility constraints (2.35) for all SSRs, the estimate for $\pi_{m,j,q}$ must satisfy the most restrictive one. Moreover, it is scaled by the number of unassigned dual entries $|SNP|$ in the vector, so that a large estimate does not limit the value of other upcoming dual estimates, owing to the fact that this is an iterative scheme, following the sorted order for the constraints.

This algorithm ensures that there are no "0" entries in the initial dual vector π and the values are prevented from being too distant from each other. Thus, the negative lower bounds are avoided while $\theta_{i,q}$ is kept the same. There can be many other versions of this algorithm, some of which will be explained in Chapter 4.

2.3.2 Manipulating the Dual Vector

Although column generation provides a monotonic improvement in the upper bound of the RMP, the lower bound progress is, unfortunately, non-monotonic. That is largely due to the fact that duals obtained from RMP in the early iterations of the procedure do not resemble the optimal dual structure, and also, they tend to jump from one extreme solution of the dual domain to another distant one. To stabilize the dual movement, Wentges [62] suggested that a convex combination of the most recent dual vector and the one which provided the best lower bound should be used instead. Then, the dual vector to be used in the pricing subproblems at iteration q is:

$$\pi^{new}_{m,j,q} = \alpha\, \pi_{m,j,q_{best}} + (1-\alpha)\, \pi_{m,j,q} \qquad \forall m, j \qquad (2.42)$$

$$\theta^{new}_{i,q} = \alpha\, \theta_{i,q_{best}} + (1-\alpha)\, \theta_{i,q} \qquad \forall i \qquad (2.43)$$

α is chosen to be equal to 0.5, which appeared to behave very well in our experiments compared to others. Chapter 4 elaborates on selecting the value of α. This smoothing will basically let us explore new dual vectors, but not drift too far away from the last good dual vector.

By applying Wentges' smoothing, the yo-yo appearance of the lower bound obtained at every iteration of ColGen is alleviated and stabilized. However, during the very first iterations, the RMP is rather short-sighted and the dual vectors it proposes are not balanced; i.e., most of the duals corresponding to the demand constraints are 0 and the nonzero ones are extremely large. When such dual vectors are used in the pricing problems, they would unfortunately cause astray columns to be proposed. Astray columns have no beneficial effects in either finding better feasible integer solutions nor or optimizing the RMP in the long run. A simple example is given:

Suppose at iteration q, for machine $m1$ at zipcode $j1$, the dual value is found to be $\pi_{m1,j1,q}$ = 20,000. Also suppose, the first entry in $type_{1,m1,j1}$ list is SSR_1 and SSR_2 is the 50^{th} best choice to assign this demand to. Assume there is also a machine $m2$ at zipcode $j2$ with dual value $\pi_{m2,j2,q}$ = 16,000 and $type_{1,m2,j2}$ list shows the SSR_2 to be at the top. When the pricing subproblem for SSR_1 is solved, all assignments look fine; however, the pricing subproblem SSR_2 produces

unwanted results. Since the same dual vector is loaded as a whole in each subproblem, rather than assigning SSR_2 to $(m2, j2)$, the subproblem assigns him to $(m1, j1)$. This proposal for SSR_2 is what is called an astray column (because he is the 50^{th} best choice for this $(m2, j2)$ demand point), and has a very low possibility of being useful. To avoid this occurrence from happening, we propose the following modification in the dual vector:

For each subproblem SSR_i:

$$\pi_{m,j,q_i} = \pi_{m,j,q} \frac{\text{number of SSRs} - \text{ranking in the } type_{1,m,j}list}{\text{number of SSRs}} \tag{2.44}$$

Simply put, this manipulation would be reducing the magnitude of the reward for unwanted associations. This in turn results in fewer unnecessary columns to be brought into the RMP, cuts down the solution time, and allows the Problem RMP_Integer to discover high quality integer solutions right from the beginning of the algorithm.

Figures 2.1 and 2.2 show the SSR coverage areas and customer assignments obtained from a particular set of subproblems. Figure 2.1 depicts the outcome when no dual manipulation is in place. Figure 2.2, on the other hand, shows what happens after the dual manipulation, described above, is applied. The undesirable assignments are avoided as it can be clearly seen in the map for SSR 14, 12 and 9.

Wentges' smoothing is applied after a separate dual vector (π_{m,j,q_i}) is created for every subproblem. As stated in Wentges' paper [62], when the pricing subproblems do not yield any columns with negative reduced cost although $UB_n \neq LB_n$, there are 2 options while designing the algorithm at hand: We can either choose to smooth the π_{m,j,q_i} once again with less weight on the best lower bound yielding dual:

$$\pi^{new}_{m,j,q_i} = \beta\, \pi_{m,j,q_{best}} + (1-\beta)\, \pi_{m,j,q_i} \qquad \forall m, j \text{where } \beta < \alpha$$

and this is proven in his paper to break the cycle. Otherwise, the dual vector $\pi_{m,j}$ found from solving

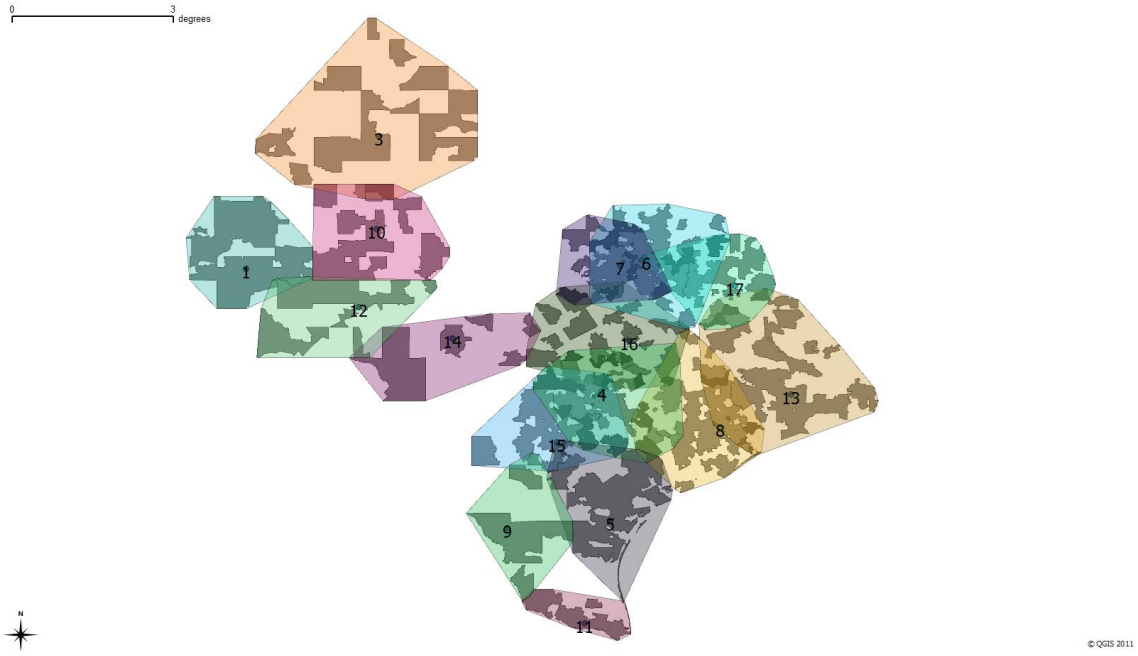


Figure 2.1: SSR Coverages before Dual Manipulation

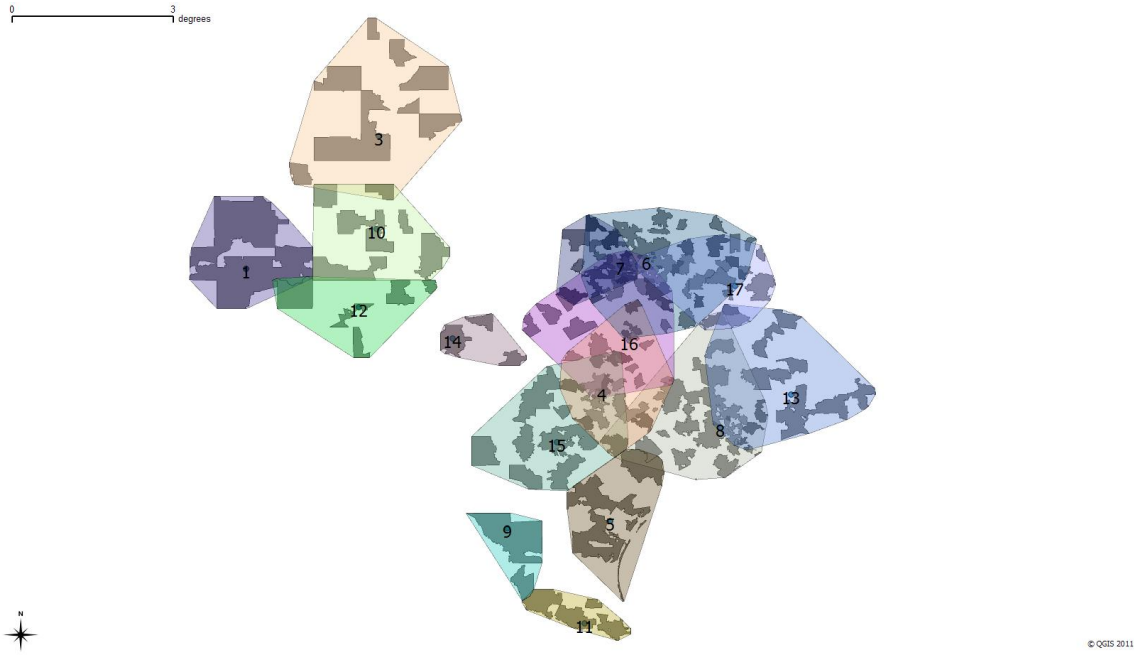


Figure 2.2: SSR Coverages after Dual Manipulation

the RMP is used without any smoothing or manipulation. In our algorithm design, we chose to use the default $\pi_{m,j}$ (from RMP) in the very rare occasions that no columns with negative reduced cost were found.

Wentges' smoothing and Dual Manipulation are not utilized when all subproblems are solved to calculate LB_n^q, where the original dual vector needs to be used.

2.3.3 Subproblem Selection

In theory, for ColGen to work, only 1 incoming column with a negative reduced cost is sufficient. However, 1 column at a time does not justify solving the RMP which is an ever-growing LP. Although solving 1 pricing subproblem takes around 0.01 seconds, for the largest test case where there are 300 SSRs, it is unnecessary to bring that many columns into the RMP. It hampers the solve time of the RMP, significantly, if all 300 columns are brought in every iteration. Therefore, a subproblem selection logic at any iteration q is proposed to mitigate this event. It is proposed to solve at most 10% of all subproblems; but, performance measures are also provided when other percentages are used, in Chapter 4.

The primary objective of this section is to try to choose the subproblems that will most likely yield the most negative reduced cost. Recall the expression for the reduced cost:

$$\bar{Yt}_{i,q} = Ct_{i,q} - (\sum_{m \in M} \sum_{j \in Z} x_{m,j,i,q}\, \pi_{m,j,q} + \theta_{i,q}) \tag{2.45}$$

Equation (2.45) asserts that large values of $\pi_{m,j,q}$ and $x_{m,j,i,q}$ are necessary to attain negative reduced costs. Therefore, using $type_{1,m,j}$ lists, the selection logic will try to pick the best SSR subproblem to solve, based on the dual value ($\pi_{m,j,q}$) and demand ($d_{m,j}$) of some (m, j):

i. Obtain $\pi_{m,j,q}\ \forall (m, j)$.

ii. Calculate $\pi_{m,j,q} * d_{m,j}$ and sort in descending order.

iii. Starting from the top of the list, pick the first and 75^{th} percentile SSR in the $type_{1,m,j}$ list

until the number of subproblems chosen reach 10% of the total number of subproblems. Do not repeat SSRs for distinct (m, j); rather, continue with rank 2, 3 etc. until an uncalled SSR is found.

The reason for calling the 75^{th} percentile SSR along with the best SSR for that particular demand (m, j) is to bring diverse set of columns back into the RMP, and avoid all columns being biased by LoA findings. Many combinations have been tested to detect the best percentile for the second SSR subproblem. The subproblems for SSRs below the 66^{th} percentile provided negative reduced costs with very low absolute values. On the contrary, the subproblems corresponding to the SSRs above the 90^{th} percentile caused incoming columns to be very similar due to being too heavily biased by LoA manipulations. Similar (parallel) incoming columns cause degeneracy in the dual domain. The subproblems for the 75^{th} percentile SSR in the $type_{1,m,j}$ list seemed to balance out the impact from both sides.

The Subproblem Selection (SS) routine cannot be used when the algorithm needs to compute LB_n^q. Therefore, once in a while (for instance every 50 iterations) ALL subproblems are solved to calculate the lower bound. Moreover, at an iteration where the algorithm needs to check for optimality, SS cannot be used either, because all subproblems need to be checked for any potentially incoming columns. Towards the end of ColGen algorithms, the number of subproblems that yield a negative reduced cost lessens significantly. When subproblems chosen by SS do not provide incoming columns despite the RMP status being non-optimal, the iteration is re-invoked with all subproblems.

2.3.4 Steps for $Colgen_{LoA}$

Now that all enhancements are introduced, the following are the steps for implementing $Colgen_{LoA}$ to solve the root node:

- **Step 1:** Initialize the RMP by solving Problem **Integer_Yielding_LP** as 20 times (with randomized wR penalty vector) to start with at least 1,000 columns. Form the set $\{Yt_{i,q}\}$ and

record all $x_{m,j,i,q}$ associated with the corresponding $Yt_{i,q}$.

- **Step 2:** Set node $n = 0$, iteration counter $q = 1$. Set $UB_n = \infty, LB_n = -\infty$.
- **Step 3:** if $q \neq 1$: Deploy reduced cost fixing: [12]. Remove $Yt_{i,q}$ for $\bar{Yt}_{i,q} > UB_{mip} - LB_n$.
- **Step 4:** Solve RMP. Obtain UB_n^q, $Yt_{i,qq}^*$, $\pi_{m,j,q}^*$ and $\theta_{i,q}^*$. Update $UB_n = min\{UB_n^q | q \in itrs\}$
- **Step 5:** if $q = 1$, Call initDualVector() function to initialize the dual vector.
- **Step 6:** if $q \neq 1$ or q mod $50 \neq 0$: Call dualManip() function to manipulate the dual vector.
- **Step 7:** Use Wentges' smoothing as a final touch on the duals.
- **Step 8:** if $q = 1$ or q mod $50 = 0$:

  Solve all pricing subproblems for SSR_i, $\forall i \in S$. Obtain $\bar{Yt}_{i,q}$ and $x_{m,j,i,q}$.

  Else:

  Call solveRelaxedWhich() function to select a subset of subproblems to solve. The size of the subset not to exceed 10% of all subproblems.
- **Step 9:** Postprocess to obtain $mu_{i,q}$, $yk_{i,m,q}$ and $ys_{i,k,q}$. Using those, calculate $Ct_{i,q}$.
- **Step 10:** if $q = 1$ or q mod $50 = 0$: Calculate LB_n^q using (2.34).

  Update $LB_n = max\{LB_n^q \,|\, q \in itrs\}$. If LB_n is updated, set $\pi_{m,j,q_{best}} = \pi_{m,j,q}^*$ and $\theta_{i,q_{best}} = \theta_{i,q}^*$.
- **Step 11:** Termination criteria check: if $(|UB_n - LB_n|)/(|LB_n| + 1) < 1.0\%$, Go to Step 13. Else update set $\{Yt_{i,q}\}$ with (i, q) having $\bar{Yt}_{i,q} < 0$. Update $q = q + 1$.
- **Step 12:** if $q = 1$ or q mod $50 = 0$: Go to Step 13.

  Else: Go to Step 3.
- **Step 13:** Solve RMP_integer. Obtain UB_{mip} and $\hat{Yt}_{i,qq}$.

- **Step 14:** Postprocess to obtain $m\hat{u}_{i,qq}$, $\hat{yk_{i,m,qq}}$ and $\hat{ys_{i,k,qq}}$. If Termination criteria is fulfilled, Stop.

  Else: Go to Step 3.

2.3.5 Certificate of Optimality for the $Colgen_{LoA}$ Procedure

Proposition 1: The Column Generation procedure using LoA enhancements finds a stronger lower bound than the LP of the original monolithic problem. It also solves RMP to optimality (when not stopped at 1% duality gap) in a finite number of steps.

Proof: The convergence of a generic ColGen algorithm at some node n is proven in [14]. Then, it is sufficient to show that none of the enhancements proposed in this thesis is breaking the underlying logic of ColGen. In essence, ColGen converges to an optimal solution for a RMP, provided that the columns are brought back from a pool of finite size and the pricing subproblems continuously recommend new incoming variables. Since only the objective function is changing from iteration to iteration due to new dual vectors every iteration, those subproblem polyhedrons stay unchanged throughout the ColGen iterations. It is also known that those polyhedrons are composed of finitely many linear inequalities and all variables are bounded from above; hence, they have finite number of extreme points.

LoA basis initialization and estimation of the initial dual vector have no adverse effect in the procedure; on the other hand, dual manipulation and subproblem selection might end up in iterations with no columns with negative reduced cost found. However, since our reaction to such an event is a direct reinvoking of the full subproblem set to be solved, the algorithm surely terminates in finite number of steps. Even in the worst case where the reinvoking of all subproblems recur at all iterations in which DM and SS are both used, then it becomes a generic ColGen algorithm that terminates when no new columns are proposed from the finite size column pool. Thus, cycling can never happen if a feasible dual vector to the last RMP is used.

Lubbecke [40] shows that for a generic Colgen algorithm, the optimal objective function of the RMP will be at least as large as the LP relaxation of the original MIP. Equality only happens, if this

problem had the ”Integrality property”, which it does not.

2.4 Computational Results for the Root Node

In this section computational results for problem sizes of 200x20, 300x30, 400x40, 600x60, 1000x100, 1500x300 (demand points x SSRs) will be presented for the urban scenario setting. All the entries in the following tables are averaged for 20 randomly generated instances for each problem size.

This section is organized in the following way: First, results from CPLEX vs Default ColGen with no LoA enhancements are compared. Table 2.1 provides a convincing numerical analysis that neither CPLEX nor default ColGen algorithms are doing a good job while trying to solve the root node of this problem. Then, one by one, the effects of the enhancements are tested in the order of ”estimation of the initial dual vector (ID)”, ”intermediate dual vector manipulation (DM)” and ”Subproblem Selection (SS)”. Next, paired combinations of those constructs are tested separately (ID+DM, ID+SS, DM+SS). Finally, the results for ColGen with all LoA enhancements (ID+DM+SS) are presented.

Table 2.1: CPLEX vs Default ColGen

	CPLEX				Default ColGen		
Size	#vars	Time_1%	Time_rt	Gap	Gap	Time	its
200x20	8,620	65	21	18.4%	35.4%	120	350
300x30	18,930	369	95	23.2%	38.3%	301	1,113
400x40	33,240	2,711	211	27.6%	41.1%	554	1,989
600x60	73,860	8,753	363	34.2%	46.7%	1,409	3,602
1000x100	203,100	24,543	890	43.5%	59.3%	5,502	16,250
1500x300	1,209,300	224,884	2,777	n/a	64.2%	44,251	49,313

In Table 2.1, #vars denotes the number of variables in the monolithic problem and Time_1% is the time it takes CPLEX to discover a solution within 1% of the best lower bound found. Time_rt, on the other hand, is the statistic that our algorithm is racing against, which is the time it takes

CPLEX to solve the root node of the monolithic problem while using Branch-and-Bound with default CPLEX settings. Cplex Gap is the quality of the integer feasible solution obtained by CPLEX when the root node is solved.

To obtain performance measurements for a Default ColGen algorithm which has no special initial feasible solution finding scheme (e.g LoA), and has no acceleration and stabilization schemes (ID/DM/SS), we turned off all enhancements that are proposed in this thesis in DIP. DIP is the Decomposition for Integer Programming framework designed for Column Generation [25].

Starting from size 400x40, both CPLEX and Default ColGen have a hard time finding good quality, integer, feasible solutions in a reasonable amount of time. Default Colgen Gap shows the mipgap for the best feasible solution found with respect to the best lower bound found at the end of the root node solve. There are 2 possible explanations for those rather bad numbers: Explanation 1 is trivial in that Default ColGen uses phase1 simplex algorithm to initialize the first RMP, not a special heuristic. Explanation 2 is that every iteration of the Default ColGen is aimed towards optimization of the RMP (which is a LP), and not aimed at discovering new integer feasible solutions on the side. Even though Default ColGen can solve the largest test cases for (1500x300), the mipgap obtained is almost useless. The solution times reported for Default Colgen are a lot worse than CPLEX Time_rt. The quantity, *its*, is the number of ColGen iterations to solve the root node, and it will be used as a benchmark for comparison against the proposed enhancements.

Table 2.2: initDualV vs DualManip vs SubSel for the Root Node

	ID			DM			SS		
Size	Gap	Time	its	Gap	Time	its	Gap	Time	its
200x20	4.9%	91	305	2.2%	74	191	5.2	65	420
300x30	5.5%	211	1,021	3.6%	183	678	5.8	160	1,333
400x40	6.8%	376	1,703	4.8%	331	1,302	7.2	276	2,593
600x60	9.1%	1,042	2,765	5.1%	856	2,113	9.5	520	6,078
1000x100	9.9%	4,078	12,078	6.2%	3,204	9,885	10.4	1,819	19,222
1500x300	10.8%	32,015	37,554	10.3%	30,430	36,884	11.6	18,879	61,883

With a little help from the enhancements, Colgen begins to become favorable, but the results are not good enough to declare victory when they are deployed one by one. Each of the 3 enhancements is applied on top of LoA initialization. Inspecting Table 2.2, estimating the initial dual vector (ID) reduces the solution time by approximately 25%, intermediate dual manipulations (DM) causes 40% reduction on solution time and almost 50% better mipgaps except in the largest test cases. Subproblem selection, on the other hand, cuts down the solution time by around 33%, but, increases the number of iterations.

Table 2.3: (initDualV + DualManip) vs (initDualV + SubSel) vs (DualManip+ SubSel) for the Root Node

	ID+DM			ID+SS			DM+SS		
Size	Gap	Time	its	Gap	Time	its	Gap	Time	its
200x20	2.1%	55	156	5.1%	45	320	2.4%	61	221
300x30	3.5%	120	366	5.6%	120	1,030	3.9%	162	774
400x40	4.7%	247	1,228	7.0%	191	1,776	5.2%	290	1,633
600x60	5.0%	607	2,011	9.2%	400	4,213	5.4%	724	2,588
1000x100	6.0%	2,509	8,886	10.1%	1,473	14,041	6.6%	2,843	10,113
1500x300	10.1%	23,443	27,432	11.2%	16,559	44,365	10.7%	22,112	40,984

When the enhancements are tested in pairs, the improvements are even better. ID+DM solution times are around 50% smaller compared to Default Colgen, ID+SS is almost 66% faster. Dm+SS is both 50% faster and mipgap is reduced by 50%. Each pair providing better solution times has encouraged us to test them all at once. The results are shown in the next table.

Table 2.4 shows that, when ColGen is accompanied with LoA+ID+DM+SS, both the solution times and solution qualities are far superior than Default Colgen and CPLEX. The Default ColGen is accelerated 8 times, and the root node is solved faster than CPLEX (except for the smallest test cases and the largest test cases), with excellent quality integer feasible solutions. The dramatic jump in the solution time from problem 1000x100 to 1500x300 is explained by the exponential increase in the number of variables: in the former problem, there are approximately 200K variables, whereas

the latter has 1.2M variables on the average.

Table 2.4: CPLEX vs Default ColGen vs Colgen with (LoA+ID+DM+SS)

	CPLEX		Default ColGen			LoA+ID+DM+SS		
Size	Time_rt	Gap	Gap	Time	its	Gap	Time	its
200x20	21	18.4%	35.4%	120	350	1.9%	35	310
300x30	95	23.2%	38.3%	301	1,113	3.4%	85	790
400x40	211	27.6%	41.1%	554	1,989	4.5%	95	1,123
600x60	363	34.2%	46.7%	4,213	3,602	4.9%	270	2,611
1000x100	890	43.5%	59.3%	14,041	16,250	5.6%	775	9,233
1500x300	2,777	n/a	64.2%	44,365	49,313	9.7%	8,994	31,313

In the next section, after the specialized B&P algorithm is introduced, computational results will be provided for the full B&P tree. Having shown that our enhancements are valid and numerically sound, Colgen with LoA+ID+DM+SS will be used at each node of the B&P tree for the exact solution of the monolithic problem.

2.5 Branch-and-Price Algorithm

When ColGen terminates at the root node, there is no guarantee that $Yt^*_{i,q}$'s are integer. Actually, for this problem, it never turned out to be the case that the optimal root node solution was integer feasible. Since the original problem is of type MIP which requires integer solutions for the binary variables, a problem specific Branch-and-Price algorithm is proposed.

2.5.1 Node Generation

The solution of RMP is, unfortunately, non-integer. Therefore, similar to solving MIP problems using Branch-and-Bound, valid cuts that avoid the current non-integer solution are needed. As laid out in Barnhart, et. al. [5], these cuts should divide the feasible region in a balanced manner for the efficiency of node generation. Depth-first strategy will be adopted in our B&P implementation,

which raises the importance of cut selection and the order of addition in the RMP even more. This is due to the fact that futile effort might be spent if the initial cuts are not effective, for instance, selecting the most fractional $Yt_{i,q}$ to branch on (closest to 0.5), which is the case for most of the B&P frameworks, instead of choosing a critical Yt that might have cascading effects in the model. The following are the cuts to be generated along the way for the Branch-and-Price tree in our design:

$$\sum_{i \in S} \sum_{q \in itrs} Yt_{i,q} \leq \lfloor \alpha \rfloor \tag{2.46}$$

$$\sum_{i \in S} \sum_{q \in itrs} Yt_{i,q} \geq \lceil \alpha \rceil \tag{2.47}$$

$$\sum_{i \in S} \sum_{q \in itrs} (Yt_{i,q} \sum_{k \in Sg} ys_{i,k,q}) \leq \lfloor \beta \rfloor \tag{2.48}$$

$$\sum_{i \in S} \sum_{q \in itrs} (Yt_{i,q} \sum_{k \in Sg} ys_{i,k,q}) \geq \lceil \beta \rceil \tag{2.49}$$

$$\sum_{i \in S} \sum_{q \in itrs} (Yt_{i,q} \sum_{m \in M} yk_{i,m,q}) \leq \lfloor \gamma \rfloor \tag{2.50}$$

$$\sum_{i \in S} \sum_{q \in itrs} (Yt_{i,q} \sum_{m \in M} yk_{i,m,q}) \geq \lceil \gamma \rceil \tag{2.51}$$

- α in (2.46) and (2.47) is the sum of activated SSRs found in the root node of the RMP. Say α turned out to be 200.5, the optimal MIP solution will either have 'less than or equal to' 200 or 'more than or equal to' 201 ssrs. Inequality (2.47) will be the first cut to be introduced to the RMP, because it is more likely that more SSRs will be needed than the fractional sum α. Inequalities of type (2.46) will be used when backtracking from the node with inequality (2.47).

- β in (2.48) and (2.49) is the sum of all skill upgrades attained by all SSRs in the root node of the RMP. $ys_{i,k,q}$ is the skill upgrades found in the pricing subproblem for SSR_i; therefore, it is used as a parameter in the RMP. Since it is not a variable, the linearity of the RMP is still preserved. If using (2.46) and (2.47) does not provide an integer feasible solution, (2.48) and

(2.49) will be introduced to the RMP, the greater than or equal to constraint being first.

- γ in (2.50) and (2.51) is the sum of machine trainings by all SSRs in the root node of the RMP. $yk_{i,m,q}$ is the machine trainings found in the pricing subproblem for the corresponding SSR_i . If (2.48) and (2.49) do not enforce in integer feasible solution, (2.50) and (2.51) will be introduced, the greater than or equal to constraint being first.

Experience has shown that high quality integer feasible solutions with mipgap $\leq 1\%$ are quickly achieved, after adding (2.46), (2.47), (2.48) , (2.49), (2.50) and (2.51); however, for the case of completeness, the following cuts have to be incorporated if MIP optimality or 1% mipgap have not been achieved in a reasonable amount of time:

$$\sum_{i \in S} \sum_{q \in itrs} Yt_{i,q} ys_{i,k,q} \leq \lfloor \beta_k \rfloor \qquad \text{for a specific skill } k \qquad (2.52)$$

$$\sum_{i \in S} \sum_{q \in itrs} Yt_{i,q} ys_{i,k,q} \geq \lceil \beta_k \rceil \qquad \text{for a specific skill } k \qquad (2.53)$$

$$\sum_{i \in S} \sum_{q \in itrs} Yt_{i,q} yk_{i,m,q} \leq \lfloor \gamma_m \rfloor \qquad \text{for a specific machine } m \qquad (2.54)$$

$$\sum_{i \in S} \sum_{q \in itrs} Yt_{i,q} yk_{i,m,q} \geq \lceil \gamma_m \rceil \qquad \text{for a specific machine } m \qquad (2.55)$$

For (2.52) and (2.53), β_k denotes the sum of upgrades by all SSRs for a particular skill k, which is a decimal; whereas, for (2.54) and (2.55), γ_m is the sum of trainings by all SSRs for a particular machine m, again, γ_m is a decimal. There is a decision that needs to be made about which particular skill or machine to branch on. One simple way is to start from the skill group which has the largest total demand value. Once there are no more (2.52)and (2.53) inequalities to deploy (i.e. all β_k's are integer), the algorithm continues with (2.54) and (2.55), where the first cut starts with the machine with the largest total demand value.

The cuts above were sufficient in the data sets to examine and allow stopping iterations either because MIP optimality was proven (small problems) or discovered an integer feasible solution with a high-quality mipgap ($\leq 1\%$). To claim we have an exact Branch-and-Price algorithm, the default

inequalities are necessary:

$$Yt_{i,q} \leq 0 \qquad \text{for a particular } (i,q) \tag{2.56}$$

$$Yt_{i,q} \geq 1 \qquad \text{for a particular } (i,q) \tag{2.57}$$

These 2 cuts above ensure that all $Yt_{i,q}$ variables end up being 0 or 1. A good rule of thumb for selecting which decimal $Yt_{i,q}$ to branch upon is using the $type_{1,m,j}$ list of a machine m with the largest demand at a zipcode j. Recall that $type_{1,m,j}$ list provides the sorted likelihood of SSRs to fulfill the demand of machine m at zipcode j. The algorithm checks if $Yt_{i,q}$ is decimal or not, with SSR_i from the top of the $type_{1,m,j}$ list with the most recent q value. The node generation proceeds with the next q, and with SSR_i's in order.

Once one of these constraints are added, an artificial Yt_{dummy}, whose Ct_{dummy} equals to a $BigM$, is introduced to the RMP. This dummy column covers all demands with a cost that is representative of a Super SSR with the salary of all SSRs in the problem along with all necessary skills and trainings, and with maximum travel costs. This ensures that RMP is still feasible, and ColGen is expected to remove Yt_{dummy} from the basis as soon as necessary columns for feasibility and improvement are found in the pricing subproblems. If ColGen cannot find an incoming column with a negative reduced cost while Yt_{dummy} is still in the basis, that means, this node became infeasible after the addition of the most recent cut.

This completes the Cut Generation for removing fractional solutions. In the next section, how these cuts alter the pricing subproblems will be analyzed.

2.5.2 Effects of Branching Cuts in the Pricing Problems

As inequalities (2.46) through (2.57) are added to the RMP at any given node in the B&P tree, the reduced cost definitions change. Since the objective of the pricing subproblems is to find the most negative reduced cost among the implicit variables, the effects of the added inequalities are

explained in this section. The feasible region for the Pricing Subproblem for SSR_i, on the other hand, is not affected by this change at all.

Assume the root node solution is not integer feasible. According to the algorithm design, inequality (2.47) will be first introduced to the RMP with associated dual variable $\eta \geq 0$:

$$\sum_{i \in S} \sum_{q \in itrs} Yt_{i,q} \geq \lceil \alpha \rceil : \eta$$

As it can be seen in (2.47), the coefficient of each Yt is 1, and it is not a parameterized version of the decision variables (mu, ys, yk or x) found in the pricing subproblems. Recall that nonnegative duals are interpreted be rewards for activation, whereas nonpositive duals mean penalties for the activation of SSR_i. Therefore, η will be introduced as a fixed reward in the pricing problem regardless of SSR_i's originating zipcode, as the constraint in the primal domain also tries to increase the number of SSRs in the system. The objective function becomes:

$$\bar{Yt}_{i,q} = Ct_{i,q} - (\sum_{m \in M} \sum_{j \in Z} x_{m,j,i,q}\, \pi_{m,j,q} + \theta_{i,q} + \eta) \tag{2.58}$$

The dual variable for Cut (2.46), however, will be used in the opposite way. Cut (2.46) and its dual variable $\kappa \leq 0$ are:

$$\sum_{i \in S} \sum_{q \in itrs} Yt_{i,q} \leq \lfloor \alpha \rfloor : \kappa$$

which will cause the pricing problem's objective function to become:

$$\bar{Yt}_{i,q} = Ct_{i,q} - (\sum_{m \in M} \sum_{j \in Z} x_{m,j,i,q}\, \pi_{m,j,q} + \theta_{i,q} + \kappa) \tag{2.59}$$

Dual variable κ can be interpreted as a barrier to a SSR becoming activated, since Cut (2.46) also dictates to limit the sum of the activated SSRs to be less than the current solution of the RMP.

For the other 4 primary cuts, the duals and their appearances in the pricing problem are given below.

For (2.48):

$$\sum_{i \in S} \sum_{q \in itrs} (Yt_{i,q} \sum_{k \in Sg} ys_{i,k,q}) \leq \lfloor \beta \rfloor : (\zeta \leq 0) \tag{2.60}$$

This constraint restricts skill upgrades; therefore, ζ will provide a penalty for new skill upgrades and will be incorporated into the pricing subproblem as:

$$\bar{Yt}_{i,q} = Ct_{i,q} - (\sum_{m \in M} \sum_{j \in Z} x_{m,j,i,q}\, \pi_{m,j,q} + \theta_{i,q} + \sum_{k \in Sg} \zeta\, ys_{i,k,q}) \tag{2.61}$$

For (2.49) :

$$\sum_{i \in S} \sum_{q \in itrs} (Yt_{i,q} \sum_{k \in Sg} ys_{i,k,q}) \geq \lceil \beta \rceil : (\nu \geq 0) \tag{2.62}$$

This constraint imposes more skill upgrades; therefore, ν will provide a reward for new skill upgrades and be incorporated into the pricing problem as:

$$\bar{Yt}_{i,q} = Ct_{i,q} - (\sum_{m \in M} \sum_{j \in Z} x_{m,j,i,q}\, \pi_{m,j,q} + \theta_{i,q} + \sum_{k \in Sg} \nu\, ys_{i,k,q}) \tag{2.63}$$

An analogous arguement is valid for machine training, and the relationships are given below directly:

For (2.50) :

$$\sum_{i \in S} \sum_{q \in itrs} (Yt_{i,q} \sum_{m \in M} yk_{i,m,q}) \leq \lfloor \gamma \rfloor : (\varpi \leq 0) \tag{2.64}$$

and the modification in the parantheses in the reduced cost expression (penalty):

$$+ \sum_{m \in M} \varpi \, yk_{i,m,q} \tag{2.65}$$

For (2.51):

$$\sum_{i \in S} \sum_{q \in itrs} (Yt_{i,q} \sum_{m \in M} yk_{i,m,q}) \geq \lceil \gamma \rceil : (\vartheta \geq 0) \tag{2.66}$$

and the modification is a reward:

$$+ \sum_{m \in M} \vartheta \, yk_{i,m,q} \tag{2.67}$$

The second stage cuts, which are specific to individual skills and individual machines, are imposed starting from the skill or machine with the largest demand and have very similar impact on the reduced cost expressions.

For a particular skill k, using Cut (2.52), the pricing problem owing to the dual variable $\zeta_k \leq 0$ is modified with a penalty:

$$+ \zeta_k \, ys_{i,k,q} \tag{2.68}$$

For a particular skill k, using Cut (2.53), the pricing subproblem owing to the dual variable $\nu_k \geq 0$ is modified with a reward:

$$+ \nu_k \, ys_{i,k,q} \tag{2.69}$$

For a particular machine m, using Cut (2.54), the pricing subproblem owing to the dual variable

$\varpi_m \leq 0$ is modified with a penalty:

$$+ \varpi_m \, yk_{i,m,q} \tag{2.70}$$

For a particular machine m, using Cut (2.55), the pricing subproblem owing to the dual variable $\vartheta_m \geq 0$ is modified with a reward:

$$+ \vartheta_m \, yk_{i,m,q} \tag{2.71}$$

Last but not least, the default cuts regarding individual $Yt_{i,q}$ have impact only in the corresponding pricing subproblem for SSR_i.

$$Yt_{i,q} \leq 0 \qquad \text{for a particular } (i,q) : (\psi \leq 0), \text{ impact as } +\psi \text{ (penalty)} \tag{2.72}$$

$$Yt_{i,q} \geq 1 \qquad \text{for a particular } (i,q) : (\xi \geq 0), \text{ impact as } +\xi \text{ (reward)} \tag{2.73}$$

Even though (2.72) translates to a penalty in the corresponding pricing subproblem for SSR_i, this subproblem can propose the same $Yt_{i,q}$ pattern as the best SSR package in some iteration $q + n$. In order to claim that the algorithm terminates in finite number of iterations, this cycling has to be prevented. A simple approach is to ignore when such a proposal is made and use the next best solution with the most negative reduced cost, as suggested in [5]. A more complicated approach would be making sure that such subproblems will never suggest those proposals by adding constraints. We chose the former approach as a practical solution.

2.5.3 Complete Algorithmic Steps for B&P

This B&P Algorithm is designed to progress in the B&P tree by using a depth-first search strategy. A node n is considered for branching or fathomed depending on which of the following 5 conditions it fits:

i. When tailing-off (no UB_n improvement in the last 50 iterations) is detected: branch.

ii. $\frac{|UB_n - LB_n|}{(|UB_n| + 1e^{-10})} \leq 1.0\%$: branch.

iii. All Yt variables obtained from the optimal solution of the RMP at node n is integer feasible; therefore, branching further has no benefit: fathom.

iv. Node n became infeasible after the addition of some cut : fathom.

v. The lower bound of node n (LB_n) is greater than the best integer feasible solution's objective value (UB_{mip}) :fathom

After fathoming a node, backtracking is applied to choose the next node, which is where the most recent branching happened prior to the fathoming. The same type of cut with the opposite sign (the one imposing disjoint feasible region) is introduced.

The Proposed B&P Algorithm:

- **Step 1:** Set $n = 0$. Solve root node with $ColGen_{LoA}$. Obtain LB_n, UB_n and UB^n_{mip}. Update $LB_{mip} = LB_n$ and $UB_{mip} = UB^n_{mip}$
- **Step 2:** Check optimality: if $UB_{mip} = LB_{mip} + \epsilon$, where $\epsilon = 0.001$, terminate.
- **Step 3:** Check branching/fathoming conditions. Backtrack if necessary.
- **Step 4:** Set $n = n + 1$ .Add a new cut from the ordered list in **Section 2.5.1**. Choose the $\geq$ version first.
- **Step 5:** Add an artificial variable Yt_{dummy} and Ct_{dummy} in the RMP to ensure feasibility after addition of the cut. Apply $ColGen_{LoA}$.
- **Step 6:** Obtain LB_n, UB_n and UB^n_{mip}. Update $UB_{mip} = min\{UB^n_{mip}|\forall n\}$ and $LB_{mip} = min\{LB_n|\forall n\}$. Go to step 2.

2.5.4 Certificate of Optimality for B&P

Proposition 2: The Accelerated Branch-and-Price Algorithm, defined above, solves the Workforce Planning Problem with Cross-Training to integer optimality in a finite number of iterations, when every node is solved to optimality.

Proof: Proposition 1 proves that each node of the Branch-and-Price tree can be solved in a finite number of iterations by ColGen Enhanced with LoA. Provided that the input data has finite number of SSRs, zipcodes, machines and skills; the number of Yt variables which corresponds to the extreme points of the subproblems will be countable and finite, since there are a finite number of mixed integer pricing subproblems whose feasible regions are bounded. In the worst case, if there are still integer infeasibilities for a solution of RMP at a node n after adding all possible Cuts (2.46) through (2.55), Cuts (2.72) and (2.73) can be applied as many times as the remaining number of Yt variables with fractional values. Thus, at least a RMP in one of the nodes of the B&P tree will have to yield the integer optimal solution.

2.6 Computational Results for B&P

In this section, computational results for problem sizes of 200x20, 300x30, 400x40, 600x60, 1000x100, 1500x300 (demand points x SSRs) will be presented for the urban scenario setting. These test cases are the same ones used for experimenting with the $Colgen_{LoA}$ on the root node. The table below compares the solution times among CPLEX, Default B&P, and $B\&P_{LoA}$. All solution times are recorded as soon as an integer feasible solution with 1% mipgap is achieved. Default B&P does not use any of the enhancements proposed for ColGen in this thesis, nor does it use a special branching strategy. It branches on the most fractional integer variable. On the other hand, $B\&P_{LoA}$ utilizes LoA heuristic for initialization of the first RMP, and ID+DM+SS for acceleration and stabilization. Moreover, $B\&P_{LoA}$ uses the special inequalities that are introduced in **Section 2.5.1** for branching.

Table 2.5: CPLEX vs Default B&P vs $B\&P_{LoA}$

	CPLEX		Default B&P		$B\&P_{LoA}$	
Size	#vars	Time_1%	Time_1%	Nodes	Time_1%	Nodes
200x20	8,620	65	944	45	79	22
300x30	18,930	369	2,487	124	182	55
400x40	33,240	2,711	3,452	226	296	94
600x60	73,860	8,753	10,433	432	943	196
1000x100	203,100	24,543	34,945	1,040	3,123	423
1500x300	1,209,300	224,884	301,450	3,037	25,902	1,132

According to our experiments, it is shown that the Default B&P is performing very poorly in practice. Although Default B&P can solve the largest test data (1500x300), as opposed to CPLEX not being able to discover a feasible integer solution in 12 hours, more than 300,000 seconds (approximately 84 hours) of run time is required. For the smaller test cases, CPLEX is much faster than Default B&P.

$B\&P_{LoA}$, on the other hand, with its branching logic and enhanced ColGen algorithm, finds 1% mipgap quality solutions 12 times faster than Default B&P and approximately 10 times faster than

CPLEX on the average. The number of nodes explored in the B&P tree is approximately 55% less for $B\&P_{LoA}$.

2.7 Discussion

In this chapter, 3 propositions have been made in order to accelerate and stabilize the ColGen Algorithm for Workforce Planning with Cross-Training Problem. For the exact solution of the problem, a branching strategy is also provided. Numerical results are very promising in terms of convergence speed and the quality of the integer feasible solutions found. These three enhancements, estimating the initial dual vector (virtual void initDualVector(vector<double> & dualVector);), intermediate dual manipulation (virtual void dualManip(map<int, vector<double> > & newDuals);), and subproblem selection (virtual void solveRelaxedWhich(vector<int> & blocksToSolve);) have been added to DIP framework [25] for other ColGen users' benefit.

The workforce planning problem is for strategic/tactical level decision making. The next chapter in this thesis deals with the more detailed shift allocation problem. Since having the shift detail in this problem structure would turn out to be intractable, we will use part of the optimal/high quality solution found here, in setting up the problem in Chapter 3. The Shift Scheduling Problem to be introduced in the next chapter will also be solved by using ColGen with LoA enhancements.

To show the robustness of the enhancements proposed, we will show their applicability, in 3 other capacitated resource management problems. We believe that the concept could also be expanded to other ColGen applications, but, that would be a subject for future research.

Chapter 3

An Accelerated and Stabilized Nested Column Generation Algorithm for Workforce Shift Scheduling Problems with Stochastic Demand

3.1 Introduction

Determining the number of employees with appropriate skill sets in a service company is a strategic decision, and it is a nontrivial problem. Finding the most cost effective shift schedule is even more difficult when the problem at hand deals with a stochastic demand input. The first stage problem introduced in this thesis was Manpower Planning with Cross-Training for a technical service support company. The plan was constructed by using monthly forecast (deterministic) demand data. The output of the first stage reveals the headcount, the necessary skill combinations and the start of the day launch points for the Service Support Representatives (SSRs). The launch point is defined as the zipcode where a SSR starts his day and returns back to this zipcode after every task completion. The

second stage problem is then finding the most cost effective weekly shift assignment while abiding by the work rules, given that the first stage output is used as an input here. Since the demand is not known with certainty but can be predicted for every scenario, a deterministic equivalent of the Stochastic Integer Model will be developed. The objective, hence, becomes finding the shift assignment which covers a sample of all possible demand scenarios in the least costly way.

This chapter is organized in the following way. First, necessary definitions are given to introduce the Workforce Shift Scheduling Problem with Stochastic Demand to the reader. Secondly, scenario generation logic is explained for the creation of rule-based random demand data. Third, the mathematical formulation of the deterministic equivalent of the Workforce Shift Scheduling Problem with Stochastic Demand is provided. Then, the size of the problem is shown to be growing exponentially with the number of scenarios. A custom Likelihood of Assignment (LoA) approach is developed to find the first greedy solution, initialize the Agent-based Tabu Algorithm, and finally stabilize and accelerate the ColGen Algorithm which is to be introduced last in the chapter. Computational results show that the quality of the solutions obtained at the end of the root node does not entail a Branch-and-Price (B&P) algorithm to be invoked. The accomplishment is that the 3-step algorithm (LoA + Tabu + ColGen) finds high quality solutions 25 times faster than CPLEX 12.2 with 30% better mipgaps on average.

3.2 Problem Definition and Stochastic Demand Scenarios

In this problem context, a shift is defined as an 8-hour block. In any given week, since there are 7 days, total number of shifts amount to 21. Every regular employee (SSR) needs to be assigned to at least 5 shifts in a week (40 hours total), or more, depending on the necessity for coverage. Every shift assigned to a SSR on top of his 5 regular shifts is considered as an overtime shift and SSRs get paid 1.5 times their base salary during overtime. Another cost factor arises when a SSR is assigned to 2 shifts consecutively (16 hours straight), which is an undesirable operational outcome, but, at times, it is absolutely required. In such a situation, there is an extra cost, which is 0.5 times the

original pay to further penalize consecutive shift assignment. This penalty expression, in fact, is in place to balance the work life of employees, and to avoid unfair shift assignments.

The rules for shift assignment are as follows. SSRs can at most work total of 3 shifts in any 6 consecutive shift block. This puts an implicit bound on the number of overtime shifts automatically. Moreover, no SSR can work 3 shifts consecutively; but, 2 shifts (16 hours straight) in any 24 hour period is allowed. One last rule is that if a SSR begins to commit himself to a job towards the end of shift sh, but runs out of available time in shift sh, the following shift $sh + 1$ is also assigned to him. The model may then assign new repair jobs during this new shift $sh + 1$ if it is cost effective to do so for the remaining available time. SSRs are restrained in the mathematical model so that they are prevented from beginning a task which would cause them to stay over 2 consecutive shifts.

The target demand forecast for every machine m, at zipcode j in shift sh is given as an input before setting up this problem. This value is actually found from scaling the monthly demand total to the shift level based on day of the week and then time of day. Suppose the number of service calls ($d_{m,j}$) for machine m at zipcode j is 4.33 per month (which was used in the planning stage). The weekly call rate ($dw_{m,j}$) is calculated by $dw_{m,j} = d_{m,j} * 12/52$. For each of the 21 shifts (set SH), the probability ($Pr\{m, j, sh\}$) of receiving a call for this (m, j) is also known. Therefore, the following call rates per shift is expected as a target for (m, j):

$$d_{m,j,sh} = Pr\{m, j, sh\}\ dw_{m,j} \tag{3.1}$$

where

$$\sum_{sh \in SH} Pr\{m, j, sh\} = 1 \qquad \forall m \in M, j \in Z \tag{3.2}$$

The creation of demand scenarios depends directly on these $dw_{m,j}$ and $d_{m,j,sh}$ values for each (m, j) pair. Since the planning stage used the monthly rate of $dw_{m,j} * 52/12$, in order to maintain feasibility or claim the scheduling problem has sufficient number of SSRs, the following rule is strictly enforced while generating the demand points $d_{m,j,sh,sc}$ for every machine m, at zipcode j,

during shift sh in scenario sc, where $sc \in SC$:

$$\sum_{sh \in SH} d_{m,j,sh,sc} \leq dw_{m,j} \qquad \forall m \in M, j \in Z, sc \in SC \tag{3.3}$$

The pseudocode to generate random scenarios is:

Random Scenario Generation Algorithm:

forall $m \in M$, $j \in J$, $sh \in SH$, $sc \in SC$, initialize $d_{m,j,sh,sc} = 0$.

forall $sc \in SC$.

forall m chosen randomly from set M.

forall j chosen randomly from set J, where $dw_{m,j} > 0$.

forall sh chosen randomly from set SH, where $Pr\{m, j, sh\} > 0$.

if $\sum_{sh \in SH} d_{m,j,sh,sc} + max(d_{\mathbf{m,j,sh,sc}}) \leq dw_{m,j}$:

Generate random $\zeta \sim U[0.0, 1.0]$.

if $d_{m,j,sh} \leq 1.0$:

if $\zeta \leq Pr\{m, j, sh\}$:

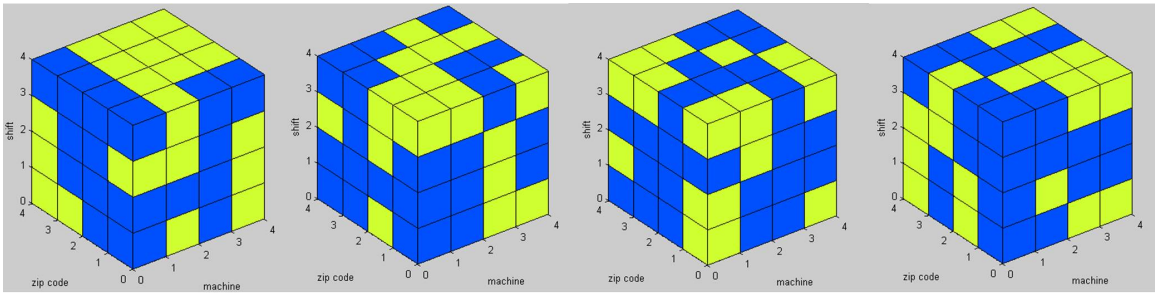
$d_{m,j,sh,sc} = 1$

else: $d_{m,j,sh,sc} = 0$

else: $d_{m,j,sh,sc} = \lceil \zeta * d_{m,j,sh} \rceil$

Figure 3.1, below, shows what the data looks like for a small example in 3 dimensions (4 machines, 4 zipcodes, 4 shifts) for 4 scenarios.

The algorithm for random demand scenario generation, basically, tries to stay faithful to the total weekly target demand ($dw_{m,j}$) and the expectations ($Pr\{m, j, sh\}$) for every (m, j) at shift sh of a weekly scenario sc. Each one of these tables under their corresponding *cubes* is a projection of the cube on the (m, j) plane, by summing the demand amounts that are predicted for all shifts. Every entry in the table belongs to a corresponding (m, j), and as stated before, it is always less than or equal to $dw_{m,j}$ for every scenario generated. The entries, also, behave as an indicator for a



Scenario 1

J4	1	2	2	2
J3	3	1	2	1
J2	4	1	1	0
J1	3	1	4	1
	M1	M2	M3	M4

Scenario 2

J4	3	2	2	3
J3	4	1	2	2
J2	1	3	4	3
J1	3	3	1	2
	M1	M2	M3	M4

Scenario 3

J4	2	2	2	3
J3	3	3	3	2
J2	4	4	4	1
J1	1	2	4	2
	M1	M2	M3	M4

Scenario 4

J4	1	3	2	2
J3	2	3	3	2
J2	2	2	1	0
J1	4	3	2	2
	M1	M2	M3	M4

Figure 3.1: 3 Dimensional Demand Data Representation by Scenario

specific (m, j)'s density in a randomly generated scenario over all shifts.

In our problem setting, every $d_{m,j,sh,sc}$ in every scenario must be either fulfilled in shift sh, or it is ensured that the SSR with appropriate skills starts his trip toward the demand zipcode while the shift sh is still underway. Thus, a particular demand, $d_{m,j,sh,sc}$, is still taken into account in shift sh; however, it is completed in shift $sh + 1$, when the assigned SSR returns to the launch zipcode. This event, on the other hand, entails the consecutive shift $(sh + 1)$ to be activated for this SSR, which is an undesirable assignment.

3.2.1 Deterministic Equivalent of the Model

The Workforce Shift Scheduling Problem with Stochastic Demand can be represented as Mixed Integer Linear Problem, when only a sample from the vast space of possible demand scenarios is considered. The determination of the necessary number of scenarios to be generated will be discussed in the computational results section.

The necessary variables and parameters required in order to mathematically display the problem follow:

Variables:

- $x_{i,sh}$: binary variable, denoting if $SSR_i \in S$ is assigned to shift sh or not.
- $OT_{i,sh}$: binary variable, denoting if shift sh is an overtime shift for SSR_i.
- $cs_{i,sh}$: binary variable, denoting if both shifts sh and $sh+1$ are assigned to SSR_i or not.
- $y_{m,j,i,sh,sc}$: binary variable, denoting if demand for machine m at zipcode j in shift sh in scenario sc is fulfilled by SSR_i or not.
- $ot_{i,sh,sc}$: continuous variable, denoting the extra number of hours necessary to finish the tasks that are started in shift sh, but finished in shift $sh+1$ in scenario sc.

Parameters:

- ch_i: shift salary for SSR_i, whose specific extra skill upgrades are included in this rate, as determined in the strategic planning stage.
- cd : various expenses of operating a service vehicle for 1 hour (gas, tolls, depreciation etc.)
- $t_{j,i}$: total time to travel from SSR_i's launch zipcode to demand zipcode j (round trip)
- $rt_{m,j}$: repair time for a machine of type m at zipcode j
- ξ_{sc}: probability of scenario sc happening(it is by default equal to $1/|SC|$)

The set SH, which has 21 entries in a week, is an ordered and circular set. Namely, if $sh = 21$ and a calculation is necessary for shift $sh+1$, the corresponding shift is 1, not 22. The mathematical formulation is as follows.

Deterministic Equivalent of the Shift Scheduling Problem with Stochastic Demand:

minimize overtime and expected scenario costs:

$$\sum_{i\in S}\sum_{sh\in SH}(1.5\ ch_i\ OT_{i,sh}) + \sum_{i\in S}\sum_{sh\in SH}(0.5\ ch_i\ cs_{i,sh}) +$$
$$\sum_{m\in M}\sum_{j\in Z}\sum_{i\in S}\sum_{sh\in SH}\sum_{sc\in SC}\xi_{sc}(cd\,t_{j,i}\,y_{m,j,i,sh,sc}) \tag{3.4}$$

subject to:

$$\sum_{i\in S} y_{m,j,i,sh,sc} = d_{m,j,sh,sc} \qquad \forall m,j,sh,sc \tag{3.5}$$

$$\sum_{m\in M}\sum_{j\in Z} y_{m,j,i,sh,sc}\,(rt_{m,j}+t_{j,i}) \le 8\ x_{i,sh} - ot_{i,sh-1,sc} + ot_{i,sh,sc} \qquad \forall i,sh,sc \tag{3.6}$$

$$ot_{i,sh,sc} \le 8\ x_{i,sh+1} \qquad \forall i,sh,sc \tag{3.7}$$

$$\sum_{psh=sh}^{sh+1} x_{i,psh} \le cs_{i,sh} + 1 \qquad \forall i,sh \tag{3.8}$$

$$\sum_{sh\in SH} x_{i,sh} \ge 5 \qquad \forall i \tag{3.9}$$

$$\sum_{sh\in SH} 8\ x_{i,sh} \le 40 + \sum_{sh\in SH} 8\ OT_{i,sh} \qquad \forall i \tag{3.10}$$

$$OT_{i,sh} \le x_{i,sh} \qquad \forall i,sh \tag{3.11}$$

$$\sum_{psh=sh}^{sh+2} x_{i,psh} \le 2 \qquad \forall i,sh \tag{3.12}$$

$$\sum_{psh=sh}^{sh+5} x_{i,psh} \le 3 \qquad \forall i,sh \tag{3.13}$$

$$ot_{i,sh,sc} \ge 0 \qquad \forall i,sh,sc \tag{3.14}$$

$$OT_{i,sh},\ x_{i,sh},\ cs_{i,sh},\ y_{m,j,i,sh,sc} \quad binary; \qquad \forall (i,sh),(m,j,i,sh,sc) \tag{3.15}$$

The objective function (3.4) minimizes the total overtime hour payments to SSRs, consecutive

shift assignment penalties and the weighted travelling cost for every scenario in the problem. Demand constraints (3.5) ensure that the demand is fulfilled in every scenario for every shift for every demand point (m, j). Constraint set (3.6) is for asserting that all activities that take place in a particular shift have to be completed in 8 hours; however, if a SSR departs for a job before the shift is over, and cannot complete the job in this shift, he reserves at least $ot_{i,sh}$ hours in the next shift. Thus, shift sh's 8 hour capacity can also be reduced by $ot_{i,sh-1}$ from the previous shift. Both $ot_{i,sh}$ and $ot_{i,sh-1}$ can never be positive together due to constraint set (3.12), which asserts a SSR can work for at most 2 shifts in any given 3 consecutive shifts. Constraints (3.7) indicate that there is an 8 hour limit for the overtime in shift $(sh + 1)$ caused by not being able to complete all jobs started in shift sh. (3.7) also activates the assignment to shift $(sh + 1)$. Constraint (3.8) is in the model to capture the penalty for assigning consecutive shifts to a SSR. Since this problem assumes the number of SSRs and their skills are fixed by the planning stage, they are required to be assigned to at least 5 shifts in a week by (3.9). Any more hours assigned to a SSR count towards overtime, as indicated by (3.10). However, the overtime shifts cannot be arbitrary,; they have to be one of the active shifts, which is ensured by (3.11). Constraint set (3.13) imposes a restriction on the number of total shifts (3 at most) assigned to a SSR in 6 consecutive shifts (48 hours). Constraints (3.14) and (3.15) are for nonnegativity and binary declarations.

Although the planning stage provided the most cost effective number of SSRs with particular skills, the shift scheduling problem is still very difficult to solve in a reasonable amount of time due the number of variables and constraints in the system, even when there are not too many scenarios considered. The largest test case from the planning problem has 1500 (m, j) demand points, and 50 activated SSRs in the optimal solution out of 300 potential SSRs. That means every SSR handles, roughly, 30 of those 1500 calls within a week, and conversely, each (m, j) can be serviced by approximately 20 different SSRs (1000 machine types, 50 SSRs, 4 hour driving distance limit). Later in this chapter, there is an explanation of what happens when the total number of SSRs who can address a repair call for some of the machines at certain zipcodes are very few. The averages are only used in the analysis to estimate a typical problem size, with 100 randomly generated demand

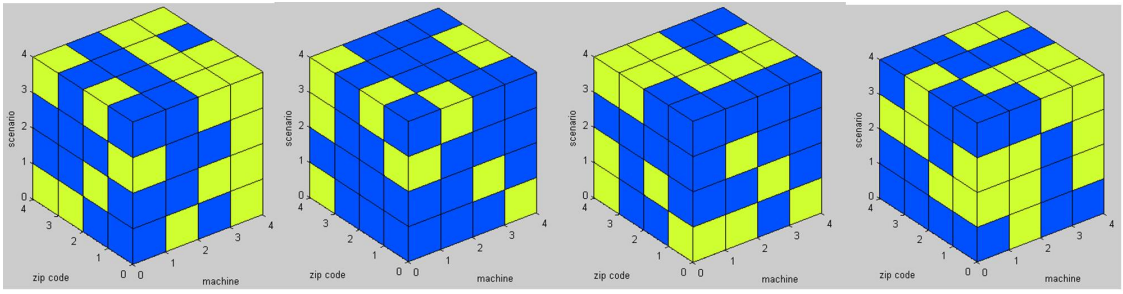
scenarios.

When this same problem setting (1500(m, j) x 50SSRs) is used for the shift scheduling stage, the shift layer is added in the demand data for each of the 100 scenarios. Based on the input data from a real setting given for $Pr\{m, j, sh\}$, for every (m, j), $Pr\{m, j, sh\}$ is non-zero for only 1 sh out of every 7 shifts (i.e., 3 out of 21) on the average. This results in a problem with more than 9 million $y_{m,j,i,sh,sc}$ variables and approximately 664,300 constraints. AMPL + CPLEX runs out of memory even while loading the problem.

The enormous size of the problem motivated us to develop a Likelihood of Assignment (LoA) approach to be used for reducing the search space for better solutions.

3.3 Likelihood of Assignment for the Stochastic Scheduling Problem

There are too many associations among SSRs and shifts and scenarios, and most have very low potential to appear in the optimal solution of this problem. LoA could help make the decision whether such an association is a beneficial one or a poor one.



Shift 1

J4	2	3	1	3
J3	3	1	3	2
J2	2	3	3	0
J1	3	3	2	0
	M1	M2	M3	M4

Shift 2

J4	1	3	4	2
J3	4	2	4	2
J2	3	3	3	1
J1	3	3	3	3
	M1	M2	M3	M4

Shift 3

J4	1	1	2	3
J3	3	3	2	1
J2	3	3	2	2
J1	3	2	3	3
	M1	M2	M3	M4

Shift 4

J4	3	2	1	2
J3	2	2	1	2
J2	3	1	2	1
J1	2	1	3	1
	M1	M2	M3	M4

Figure 3.2: 3-D Representation of Demand for Machine-Zipcode by Shifts

When Figure 3.2 is carefully inspected, it is a little different from Figure 3.1. The z-axis is the

scenarios instead of shifts. The corresponding tables under each cube shows the demand density for every (m, j) over all scenarios. As the magnitude of a particular entry increases in the table for a particular shift sh, more of this (m, j) demand is expected to be seen in this shift sh over all scenarios. LoA will use this observation to assign the best SSR to this shift sh for this demand point (m, j).

As was done similarly in the planning stage, a likelihood penalty can be assigned between every SSR and demand point (m, j). The penalty will be very low for beneficial associations, e.g., for a SSR and (m, j) that are very close to each other ($factor1$). The less time is spent during driving, the smaller are the travelling costs. Another beneficial association will be made between some particular (m, j) points and some specific SSRs who actually have the unique skills to complete the repair task ($factor2$). The intuitive reason is that if there are only 2 SSRs who can fix a very specific skill-requiring machine, the optimal solution will never use these 2 technicians for other tasks during the same time period that the calls arrive. Otherwise, the problem becomes infeasible, as this model does not have the luxury to add new people in the workforce, and it has to make the best of the available input skill set.

Thus, one can obtain 2 sorted lists based on the Likelihood penalties; i.e., list of best SSRs for every (m, j) (recall the $type1_{m,j}$ list) and list of best (m, j)'s for every SSR_i ($type2_i$ list). Next, the shift associations need to be figured out. The tables in Figure 3.2 help in that regard. For each SSR_i, the list of best shifts ($type3_i$ list) are found by adding the entries for the top 10 (m, j)'s in the $type2_i$ list.

To see the impact of $factor1$ versus $factor2$ in the optimal solutions, 20 instances of problem size (200(m,j)x20SSRx10scenarios) are solved to optimality with CPLEX. The assignments that appeared in the optimal solutions showed that if the machines required rare skills among the SSRs, the distance between the SSR launchzip and the demand zipcode ($factor1$) is the secondary factor compared to $factor2$. The ratio of assignments that are made to the closest SSR versus the SSR with the unique skill is 1 to 2. Therefore, in the likelihood penalty calculations w_f for $factor2$ is

chosen as 2, and w_f for $factor1$ is 1. Hence, the formula is given:

$$wR_{m,j,i} = \frac{\sum_f (100\, w_f\ f\_value_{f,m,j,i})}{|\sum_f w_f|}, \qquad \forall m, j, i \tag{3.16}$$

where f_value for $factor2$ is the number of SSRs who have the skill to fix this machine and are within 4-hour driving distance. f_value for $factor1$ is the distance from between the SSR launchzip and the demand zipcode.

To obtain the penalties for associations among the SSRs, (m, j)'s and the shifts, this last manipulation is made:

$$wR_{m,j,i,sh} = wR_{m,j,i} * (\text{1-percentile of } sh \text{ in } type3_i \text{ list}), \qquad \forall m, j, i, sh \tag{3.17}$$

After all $wR_{m,j,i,sh}$'s are calculated, a sorted SSR list ($type1_{m,j,sh,sc}$) in increasing $wR_{m,j,i,sh}$ values is constructed for each demand point (m, j) in shift sh for every scenario sc; i.e., ($d_{m,j,sh,sc}$). The SSRs at the top of these lists are the best candidates for task assignments by LoA.

3.4 Agent-Based Tabu Search Algorithm

An Agent-Based Tabu Search Algorithm is most convenient when there is a relatively good solution already available and when the neighborhood search for adjacent feasible solutions is not expensive [51]. The algorithm looks for the best improving feasible solution in the local neighborhood of the current solution. Since this is a local search approach, the algorithm can easily get stuck at a local optimal point. Therefore, defining the size of the local neighborhood is critical for continuous improvement. The algorithm to be proposed is able to evade local optimal solutions by moving towards directions which may seem to be poor at first; however, better quality solutions can be obtained after jumping to these seemingly bad search spaces. Aside from its computational ease, the Tabu Search Algorithm does not require the problem specific parameters to be heavily tuned as

in other metaheuristics such as Genetic Algorithm and Simulated Annealing.

Our algorithm is similar to Sabar et. al.'s approach [53]. The algorithm is based on the "swap" of shift assignments among agents (SSRs in this context). Every agent in the system, regardless of considering others, tries to minimize his cost as given in (3.4). His cost to the system is comprised of overtime shift payments, consecutive shift penalties, and the weighted (by scenario probabilities) sum of all travelling costs in all scenarios. An agent would first want to eliminate consecutive shift penalties, and then overtime shift payments, and finally, he would try to minimize travelling costs by exchanging job assignments within the same shift with another agent. The System Coordinator confirms the swaps or proposes to move towards tabu directions.

The benefits of this algorithm are two-fold. First, the feasible solutions found throughout will be used to initialize the Restricted Master Problems to be introduced for the Nested Column Generation Algorithm later in this chapter. Secondly, this heuristic can quickly respond to disruptions in the system, e.g., a SSR calling in sick for the week or taking a vacation, or when a critical contract from a client is lost, etc.

3.4.1 Initializing the Heuristic

The Agent-based Tabu Heuristic requires an initial solution (not necessarily feasible) to begin to look for improving solutions in the local neighborhood of this solution. A greedy initial solution can be constructed using the $wR_{m,j,i,sh}$ penalties found by the LoA approach. $UsedCap_{i,sh,sc}$ is the total time spent (repair or travel) by SSR_i during shift sh in scenario sc. The pseudocode for initialization is given as follows:

The Initialization Algorithm:

forall $i \in S$, $sh \in SH$, $sc \in SC$, initialize $UsedCap_{i,sh,sc} = 0$.

forall $sc \in SC$.

forall $sh \in SH$

forall (m,j), find and iterate over (m, j)'s with $m \in M$, $j \in J$, $d_{m,j,sh,sc} > 0$, whose

$type1_{m,j,sh,sc}$ has the least number of SSRs. Select the SSR_i at the top of the list.

If $UsedCap_{i,sh,sc} + (rt_{m,j} + t_{j,i}) \leq 8$ and (3.12) and (3.13) hold:

Set $y_{m,j,i,sh,sc} = 1$. Set $d_{m,j,sh,sc} = d_{m,j,sh,sc} - 1$. Set $x_{i,sh} = 1$.

Remove SSR_i from $type1_{m,j,sh,sc}$.

If $\sum_{sh} x_{i,sh} > 5$: set $OT_{i,sh} = 1$.

If $x_{i,sh-1} = 1$: set $cs_{i,sh-1} = 1$.

If $UsedCap_{i,sh,sc} + (rt_{m,j} + t_{j,i}) > 8$ and (3.12) and (3.13) hold:

Set $ot_{i,sh,sc} = UsedCap_{i,sh,sc} + y_{m,j,i,sh,sc}\,(rt_{m,j} + t_{j,i}) - 8$.

Set $UsedCap_{i,sh+1,sc} = UsedCap_{i,sh+1,sc} + ot_{i,sh,sc}$.

If $UsedCap_{i,sh+1,sc} \leq 8$:

Set $UsedCap_{i,sh,sc} = 8$.

Set $y_{m,j,i,sh,sc} = 1$. Set $d_{m,j,sh,sc} = d_{m,j,sh,sc} - 1$. Set $x_{i,sh} = 1$.

Remove SSR_i from $type1_{m,j,sh,sc}$.

If $\sum_{sh} x_{i,sh} > 5$: set $OT_{i,sh} = 1$.

Set $x_{i,sh+1} = 1$.

If $\sum_{sh} x_{i,sh} > 5$: set $OT_{i,sh+1} = 1$.

Set $cs_{i,sh} = 1$.

Else:

Revert $UsedCap_{i,sh+1,sc}$ and $ot_{i,sh,sc}$ to their previous values.

Continue selecting SSRs from $type1_{m,j,sh,sc}$ list until $d_{m,j,sh,sc} = 0$.

According to our experiments, this greedy assignment algorithm ends up with a feasible solution 96% of the time when tested on 50 instances of (1500x300x100) problem. When it is not feasible, there are still some demands ($d_{m,j,sh,sc}$) not assigned to any SSRs. This situation is easily overcome by using a BigM value, assuming a dummy SSR is assigned to it. The system coordinator proposes these unassigned demands to be taken care of even if it adds a consecutive shift or overtime shift to an already existing agent's shift schedule.

3.4.2 Defining the Tabus

A Tabu Swap is defined as a legal, but prohibited shift or assignment swap between agents. Tabu swap rules are vital for the algorithm's performance. If the tabus are too strict, the search space in the local neighborhood becomes too small. When the tabus are too relaxed, the algorithm might navigate in the wrong direction as opposed to the best direction. Tabu swap lists are also updated as the algorithm proceeds to avoid cycles.

Before introducing the tabu swaps in our algorithm, the following definitions need to be made:

- $C_{i,q}$: cost of Agent i to the system at iteration q, and equals to:

$$C_{i,q} = \sum_{sh \in SH} (1.5\ ch_i\ OT_{i,sh,q}) + \sum_{sh \in SH} (0.5\ ch_i\ cs_{i,sh,q}) + \sum_{m \in M} \sum_{j \in Z} \sum_{sh \in SH} \sum_{sc \in SC} \xi_{sc} (cd\, t_{j,i}\, y_{m,j,i,sh,sc,q}) \tag{3.18}$$

- $ST_q = \sum_{i \in S} C_{i,q}$: Total system operating cost in iteration q:

$$ST_q = \sum_{i \in S} \sum_{sh \in SH} (1.5\ ch_i\ OT_{i,sh,q}) + \sum_{i \in S} \sum_{sh \in SH} (0.5\ ch_i\ cs_{i,sh,q}) + \sum_{i \in S} \sum_{m \in M} \sum_{j \in Z} \sum_{sh \in SH} \sum_{sc \in SC} \xi_{sc} (cd\, t_{j,i}\, y_{m,j,i,sh,sc,q}) \tag{3.19}$$

- $\Delta^{+}_{i,sh}$: the added cost in $C_{i,q}$, if Agent i accepts this new shift assignment sh and all jobs in this shift that his capacity can allow.

- $\Delta^{-}_{i,sh}$: the cost removed from $C_{i,q}$, if some other Agent agrees to take on all responsibility for all the tasks in this shift sh.

- $\Delta^{+}_{i,sh,m,j,sc}$: the added cost in $C_{i,q}$, if Agent i agrees to fulfill a specific (m, j) demand in shift sh in which he is already active and has capacity for, in scenario sc.

- $csw(i, sh_1, k, sh_2) =$

$\Delta^+_{i,sh_2} + \Delta^-_{k,sh_1} - \Delta^-_{i,sh_1} - \Delta^-_{k,sh_2} + \sum_{v,m,j,sc} \Delta^+_{v,sh_1,m,j,sc} + \sum_{v,m,j,sc} \Delta^+_{v,sh_2,m,j,sc}$: the marginal benefit/cost of swapping that takes place between Agent i's shift sh_1 and Agent k's shift sh_2 and the cost of fulfilling the unmet demands caused by this swap, by other agents.

- $cNosw(i, sh_1, k) = \Delta^+_{k,sh_1} - \Delta^-_{i,sh_1} + \sum_{v,m,j,sc} \Delta^+_{v,sh_1,m,j,sc}$: the marginal benefit/cost of Agent k being assigned to shift sh_1 along with other agents, forcibly by the System Coordinator (shift transfer). This happens when $cNosw(i, sh_1, k) < 0$, or the heuristic initializes with a Dummy demand, or when the System Coordinator calls for a tabu step.

There are 3 types of tabu swaps in our algorithm:

- Considering $csw(i, sh_1, k, sh_2)$, when sh_1 is not in the top 33% of $type3_k$ list or when sh_2 is not in the top 33% of $type3_i$ or both. The percentile is given in order to limit the search space for feasible swaps and avoid the algorithm to discover another local optimal.
- A shift transfer occasion where $cNosw(i, sh_1, k) > 0$.
- After a swap takes place, e.g., (i, sh_1, k, sh_2), the reverse swap (i, sh_2, k, sh_1) is put in the tabu list for the next 50 iterations.

Tabu swaps are invoked by the System Coordinator when there are no $csw(i, sh_1, k, sh_2)$ swaps with negative value in iteration q, to jump to a new neighborhood with a worse ST_{q+1} with the intention of discovering a $ST_{q+n} < ST_q$ after n more iterations.

3.4.3 Steps for the Agent-based Tabu Algorithm

The Agent-based Tabu Algorithm is a heuristic which monotonically improves the objective function value until a tabu swap is invoked. However, the drawback of the algorithm is that, at any moment in the algorithm, there is no known lower bound. Therefore, the quality of the solution cannot be compared against a benchmark. Due to the huge number of possible tabu swaps that the System Coordinator can invoke, the algorithm has a 30 minute time limit. However, for relatively

small problems such as (200x20x10), the algorithm happens to stumble upon the integer optimal solution much faster than CPLEX (usually under 30 minutes). The algorithmic steps are given in the following:

Agent-based Tabu Algorithm:

- **Step 1:** Set $q = 0$. Start with the greedy heuristic guided by LoA. If the initial solution is infeasible, add dummy agents with $C_{dummy,q} = BigM$.

- **Step 2:** Calculate $C_{i,q}$ for all other agents, ST_q for the whole system, including dummy agents. Set $q = q + 1$.

- **Step 3:** If there are dummy agents who are fulfilling demand (m, j, sh, sc), search for the best agents using the $type1_{m,j,sh,sc}$ lists to fulfill those demands. Find Agents with the highest swap benefit $cNosw(dummy, sh_1, k)$. If there is no feasible and improving swap, generate a tabu swap to evade this region in the feasible domain. Go to Step 2.
  ElseIf: No active dummy agents: Go to Step 4.

- **Step 4:** Search for the SSR with the most $OT_{i,sh}$ shifts active. Break ties by comparing $C_{i,q}$ with the agent with higher $C_{i,q}$ chosen.
  If $cs_{i,sh}$ is also active, $x_{i,sh+1}$ is active. Let $sh_2 = sh + 1$. To avoid the consecutive shift penalty, Agent i offers to swap shift sh_2.
  Else: find the shift with the greatest Δ^-_{i,sh_1}. Agent i offers to swap shift sh_1.

- **Step 5:** Based on the shift sh_i proposed by Agent i, search for an Agent k whose $type3_k$ list has shift sh_i in the top 33%. If no Agent k is found, Go to Step 8.

- **Step 6:** Enumerate all possible swap offers from Agent k. sh_k must be in the top 33% of $type3_i$ list as well. Check for feasibility (fulfilling all demands with shift capacity) as it is done in the initialization heuristic. If there is any unmet demand, search for the best SSR using the $type1_{m,j,sh,sc}$ lists, enumerate from the top of the list for SSRs who are already active and have capacity in the shift in which that unmet demand occurs in that particular

scenario. Calculate Δ^{+}_{i,sh_k}, Δ^{-}_{i,sh_i}, Δ^{+}_{k,sh_i}, Δ^{-}_{k,sh_k}, $\sum_{v,m,j,sc} \Delta^{+}_{v,sh_k,m,j,sc}$, $\sum_{v,m,j,sc} \Delta^{+}_{v,sh_i,m,j,sc}$ and hence, $csw(i, sh_i, k, sh_k)$ for all swaps.

- **Step 7:** If there is at least one swap, not in the tabu list, with $csw(i, sh_i, k, sh_k) < 0$, execute the swap that is the most beneficial.

  Else: Go to Step 8.

- **Step 8:** Evaluate the tabu options. To avoid cycling do not consider a swap that took place within 50 iterations of q. Execute the tabu swap which causes the largest increase in the ST_{q-1}.

- **Step 9:** Update the tabu list with the most recent swap.

  Terminate after 30 minutes and report the best solution found so far.

  Else: go to Step 2.

The numerical results for this algorithm are provided in the Computational Results section of this chapter. The quality of the solutions are assessed with the best lower bounds found by the Nested ColGen Algorithm. There is an article by Mason et. al. [43], which describes a Nested Column Generator for solving deterministic rostering problems. Their master and subproblem structures are much simpler than what is presented in this research; however, it may provide additional insight to the reader.

3.5 Accelerated Nested Column Generation Algorithm

Although the Agent-based Tabu algorithm discovers significantly better solutions very fast, there is no information about the mipgap of the solution. AMPL + CPLEX, on the other hand cannot even load problems larger than (400(m,j) x 40SSRs x 100Scenarios). ColGen, however, is a very convenient approach when handled with care, to find valid lower bounds for large scale problems.

The shift scheduling problem with stochastic demand, unfortunately, does not have a simple structure to use a ColGen algorithm with 1 RMP and a set of subproblems. A 3-layer decomposition

is necessary to be able to utilize ColGen efficiently. A Set Covering formulation will be introduced first, where SSRs proposed in the lower layers will try to cover the demand in the top layer Restricted Master Problem. At every level, all of the subproblems' feasible domains are strictly bounded; therefore, there will be no mention of extreme rays throughout this chapter.

3.5.1 Set Covering Reformulation

Let binary variable $\mu_{i,r}$ represent the proposal for SSR_i found in iteration r of main ColGen or the r^{th} feasible solution that is used to initialize the procedure. Then, one can obtain the cost of this SSR to the problem by:

$$Ct_{\mu_{i,r}} = \sum_{sh \in SH} (1.5\ ch_i\ OT_{i,sh,r}) + \sum_{sh \in SH} (0.5\ ch_i\ cs_{i,sh,r}) + \sum_{m \in M} \sum_{j \in Z} \sum_{sh \in SH} \sum_{sc \in SC} \xi_{sc} (cd\, t_{j,i}\, y^r_{m,j,i,sh,sc}) \tag{3.20}$$

Assume we have sufficient number of SSRs to fulfill all demands that are to happen during all shifts in all scenarios, and further assume the proposed SSRs are already meeting the constraints for business rules, capacity, and overtime restrictions. The following model is equivalent to the Deterministic Equivalent of the Shift Scheduling Problem with Stochastic Demand using the Resolution Theorem [54] and Dantzig-Wolfe Decomposition [14]:

Problem RMP_equivalent:

minimize the total cost for employing SSRs:

$$\sum_{i \in S} \sum_{r \in itrs1} Ct_{\mu_{i,r}} \mu_{i,r} \tag{3.21}$$

subject to:

$$\sum_{i \in S} \sum_{r \in itrs1} \mu_{i,r}\ y^r_{m,j,i,sh,sc} \geq d_{m,j,sh,sc} \qquad \forall m, j, sh, sc \tag{3.22}$$

$$\sum_{r \in itrs1} \mu_{i,r} \leq 1 \qquad \forall i \tag{3.23}$$

$$\mu_{i,r} \quad binary \qquad \forall i, r \tag{3.24}$$

where $r \in itrs1$ is the set of all columns found, and $y^r_{m,j,i,sh,sc}$ is the task assignment givens for the r^{th} proposal of SSR_i. The only issue with this reformulation is that there are too many constraints (approximately 450K for 1500x300x100 case) which prevents this from becoming an efficient SSR proposal evaluator and dual vector generator.

However, when the contents of $Ct_{\mu_{i,r}}$ are carefully examined, that expression (3.20) can be decomposable by shifts. The demand constraints can also be written for every shift sh without depending on other shifts. Unfortunately, the convexity constraints (3.23) are not decomposable by shifts, but can be relaxed. The following is the relaxation proposed for (3.23) when Problem RMP_equivalent is decomposed by shifts:

$$\sum_{r \in itrs1} \mu_{i,r,sh} \leq 1 \qquad \forall i, sh \tag{3.25}$$

This constraint set allows the same SSR to be active on one shift and chosen to be not active on another, despite the fact that at the time of the column generation, he might have been actually scheduled to work in both. If (3.23) was retained, only 1 proposal for every SSR could have been activated. Thus, we obtain relaxed, but, solvable RMP subproblems by shift (see Figure 3.3 for

better understanding):

Problem RMP Level1 by shift sh in iteration q:

minimize the total cost for employing SSRs in shift sh:

$$\sum_{i \in S} \sum_{r \in itrs1} Ct_{\mu_{i,r,sh}} \mu_{i,r,sh} \tag{3.26}$$

subject to:

$$\sum_{i \in S} \sum_{r \in itrs1} \mu_{i,r,sh}\, y^{r}_{m,j,i,sh,sc} \geq d_{m,j,sh,sc} \qquad (\pi_{L1,m,j,sh,sc,q}) \qquad \forall m, j, sc \tag{3.27}$$

$$\sum_{r \in itrs1} \mu_{i,r,sh} \leq 1 \qquad (\theta_{L1,i,sh,q}) \qquad \forall i \tag{3.28}$$

$$\mu_{i,r,sh} \quad binary \qquad \forall i, r \tag{3.29}$$

where $\pi_{L1,m,j,sh,sc,q}$ is the dual vector for the demand constraints, which is going to be used throughout the algorithm as a reward for meeting demand. Dual vector $\theta_{L1,i,sh,q}$ on the other hand can be considered as an extra cost in front of a SSR-shift assignment.

Not being able to obtain guaranteed feasible solutions, hence MIP upper bounds, by directly solving RMP Level1 by shifts, seems like a drawback, but with some post-processing one can still find quality integer solutions. This will be explained while introducing the formal algorithm steps.

The LP relaxation of Problem RMP Level1 by shift requires initial SSR proposals to start providing dual vectors for the Pricing Subproblem Level1. The duty of the Pricing Subproblem Level1 is to provide best shift assignment for a given set of job assignments in various scenarios. Fortunately, Pricing Subproblem Level1 can be decomposed by SSRs, but they are still very difficult to solve when there are more than 100 scenarios in the problem. Therefore, even the Pricing Subproblem Level1 decomposed by SSRs will have to be solved by column generation. The structure of the Pricing Subproblem Level2 allows the problem to be decomposed by scenarios; thus, this huge problem with 9 Million variables can be attacked by solving a series of very small subproblems.

Pricing Subproblem Level1 should then find the minimum reduced cost for $\mu_{i,q}$ in order to bring

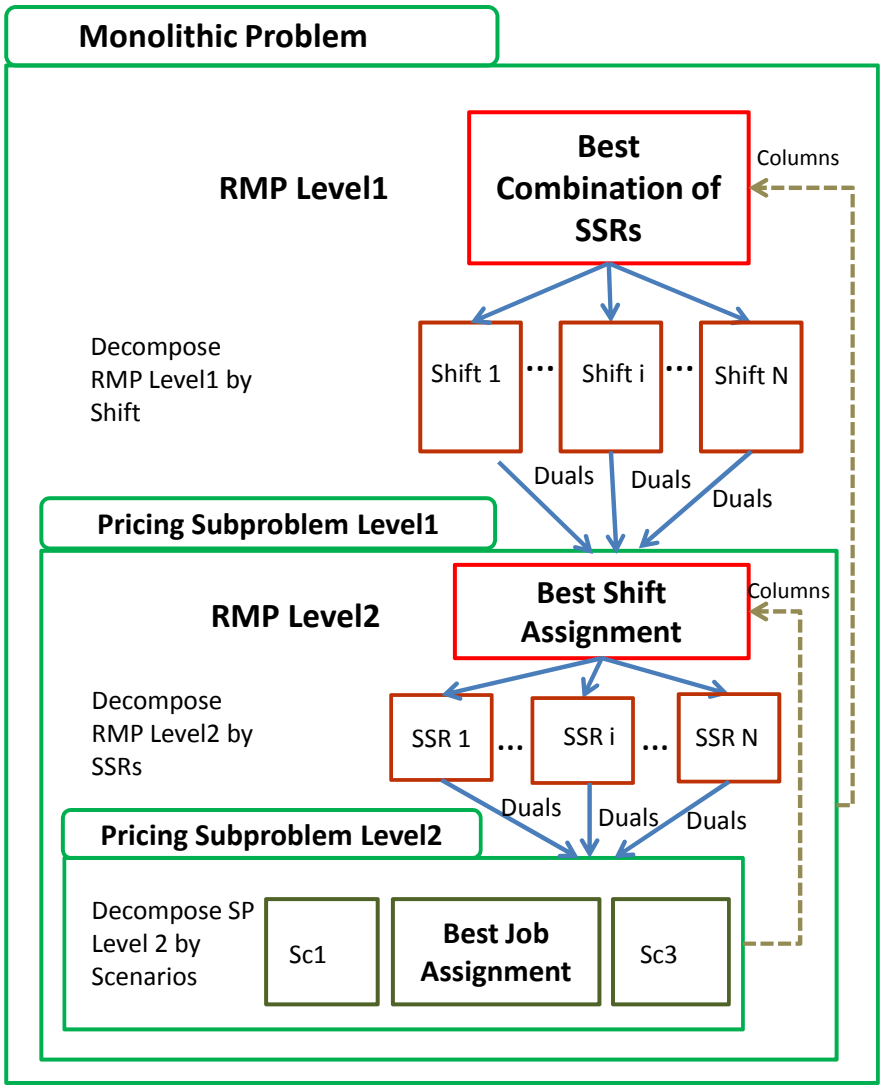


Figure 3.3: Nested Column Generation Algo Flowchart

in the most beneficial proposal for a SSR as a whole package. The reduced cost for $\mu_{i,q}$ is given in the following formula using the duals from iteration q of RMP Level1:

$$\bar{\mu_{i,q}} = Ct_{\mu_{i,q}} - (\sum_{m \in M} \sum_{j \in Z} \sum_{sh \in SH} \sum_{sc \in SC} \pi_{L1,m,j,sh,sc,q}\, y^{q}_{m,j,i,sh,sc} + \sum_{sh \in SH} \theta_{L1,i,sh,q}) \quad \forall i \quad (3.30)$$

(3.31)

where

$$Ct_{\mu_{i,q}} = \sum_{sh \in SH} (1.5\ ch_i\ OT_{i,sh}) + \sum_{sh \in SH} (0.5\ ch_i\ cs_{i,sh}) + \sum_{m \in M} \sum_{j \in Z} \sum_{sh \in SH} \sum_{sc \in SC} \xi_{sc} (cd\, t_{j,i}\, y^{q}_{m,j,i,sh,sc}) \quad \forall i \tag{3.32}$$

As mentioned before, Pricing Subproblem Level1 decomposed by SSRs, is designed to produce shift assignment proposals for each SSR considering all scenario proposals. The layer below, Pricing Subproblem Level2 decomposed by scenarios, produces scenario based proposals $\lambda_{i,sc,p}$, for every SSR, separately. But, it is Pricing Subproblem Level1's job to accept or reject scenario proposals (by finding the best values for $\lambda_{i,sc,p}$; therefore, Pricing Subproblem Level1 has to be written accordingly and renamed as RMP Level2:

Problem RMP Level2 by SSR_i in iteration q of Level1 and p of Level2:

min $\bar{\mu_{i,q}}$

$$= Ct_{\mu_{i,q}} - (\sum_{m\in M}\sum_{j\in Z}\sum_{sh\in SH}\sum_{sc\in SC}\sum_{p\in itrs2}\pi_{L1,m,j,sh,sc,q}\,\lambda_{i,sc,p}\,y^q_{m,j,i,sh,sc,p}+ \sum_{sh\in SH}\theta_{L1,i,sh,q})$$

$$= \sum_{sh\in SH}(1.5\,ch_i\,OT_{i,sh}) + \sum_{sh\in SH}(0.5\,ch_i\,cs_{i,sh})+ \sum_{m\in M}\sum_{j\in Z}\sum_{sh\in SH}\sum_{sc\in SC}\sum_{p\in itrs2}\xi_{sc}(cd\,t_{j,i}\lambda_{i,sc,p}\,y^q_{m,j,i,sh,sc,p})-$$

$$(\sum_{m\in M}\sum_{j\in Z}\sum_{sh\in SH}\sum_{sc\in SC}\sum_{p\in itrs2}\pi_{L1,m,j,sh,sc,q}\,\lambda_{i,sc,p}\,y^q_{m,j,i,sh,sc,p}+ \sum_{sh\in SH}\theta_{L1,i,sh,q}) \tag{3.33}$$

subject to:

$$\sum_{m\in M}\sum_{j\in Z}\sum_{p\in itrs2}\lambda_{i,sc,p}\,y^q_{m,j,i,sh,sc,p}\,(rt_{m,j}+t_{j,i}) \leq$$

$$8\,x_{i,sh} - \sum_{p\in itrs2}\lambda_{i,sc,p}ot_{i,sh-1,sc,p} + \sum_{p\in itrs2}\lambda_{i,sc,p}ot_{i,sh,sc,p} \quad (\beta_{L2,i,sc,p}) \qquad \forall sh, sc \tag{3.34}$$

$$\sum_{p\in itrs2}\lambda_{i,sc,p}ot_{i,sh,sc,p} \leq 8\,x_{i,sh+1} \quad (\gamma_{L2,i,sc,p}) \qquad \forall sh, sc \tag{3.35}$$

$$\sum_{p\in itrs2}\lambda_{i,sc,p} \leq 1 \quad (\theta_{L2,i,sc,p}) \qquad \forall i \tag{3.36}$$

$$OT_{i,sh},\ x_{i,sh},\ cs_{i,sh}, \lambda_{i,sc,p} \quad binary \qquad \forall sh, sc, p \tag{3.37}$$

$$3.8, 3.9, 3.10, 3.11, 3.12, 3.13 \qquad \text{for this } i \tag{3.38}$$

In the model above, only the dual variables corresponding to the constraints that are directly relevant to $\lambda_{i,sc,p}$ are given. $\beta_{L2} \leq 0$, $\gamma_{L2} \leq 0$ and $\theta_{L2} \leq 0$ will be used in the reduced cost calculation of $\lambda_{i,sc,p}$.

Assuming there are sufficiently many scenario proposals, i.e., valid and feasible inputs for $y^q_{m,j,i,sh,sc,p}$, $ot_{i,sh-1,sc,p}$ and $ot_{i,sh,sc,p}$, when this model is solved to integer optimality, the solution to each Problem RMP Level2 by SSR yields improving columns to the Problem RMP Level1. However, since the LP relaxations of RMPs are needed to be solved in order to obtain valid dual vectors, the integer problem is solved with classic Branch-and-Bound using the columns found up until the time Pricing Subproblem Level2 stops offering new columns.

Variables $\lambda_{i,sc,p}$ are the proposals to be obtained from Pricing Subproblem Level2, decomposed by scenarios, which is the best feasible task assignments that a SSR_i can have in a given scenario. To find the best proposal for a scenario, the reduced cost for $\lambda_{i,sc,p}$ in iteration p of RMP Level2 and iteration q of RMP Level1 is used as the objective function value.

Pricing Subproblem Level2 by sc in iteration q of Level1 and p of Level2:

$$\begin{aligned} min\ \bar{\lambda}_{i,sc,p} = (&\sum_{m\in M}\sum_{j\in Z}\sum_{sh\in SH}\xi_{sc}(cd\,t_{j,i}\,y^{q}_{m,j,i,sh,sc,p}) - \\ &\sum_{m\in M}\sum_{j\in Z}\sum_{sh\in SH}\pi_{L1,m,j,sh,sc,q}\,y^{q}_{m,j,i,sh,sc,p}) - \\ (\beta_{L2,i,sc,p}&\left(\sum_{m\in M}\sum_{j\in Z}\sum_{sh\in SH}y^{q}_{m,j,i,sh,sc,p}(rt_{m,j}+t_{j,i}) + \sum_{sh\in SH}ot_{i,sh-1,sc,p} - \sum_{sh\in SH}ot_{i,sh,sc,p}\right) + \\ &\sum_{sh\in SH}\gamma_{L2,i,sc,p}\,ot_{i,sh,sc,p} + \theta_{L2,i,sc,p}) \end{aligned} \tag{3.39}$$

subject to:

$$\sum_{m\in M}\sum_{j\in Z}y^{q}_{m,j,i,sh,sc,p}\,(rt_{m,j}+t_{j,i}) \le 8\,x_{i,sh} - ot_{i,sh-1,sc,p} + ot_{i,sh,sc,p} \qquad \forall sh \tag{3.40}$$

$$ot_{i,sh,sc,p} \le 8\,x_{i,sh+1} \qquad \forall sh \tag{3.41}$$

$$y^{q}_{m,j,i,sh,sc,p}, OT_{i,sh},\ x_{i,sh},\ cs_{i,sh}\ binary; \quad ot_{i,sh,sc,p} \ge 0 \qquad \forall sh \tag{3.42}$$

$$3.8, 3.9, 3.10, 3.11, 3.12, 3.13 \qquad \text{for this } i \tag{3.43}$$

The SSR-Scenario proposals conform to every constraint in the original monolithic problem except the demand constraints. Having introduced all the RMPs and Pricing Subproblems, another look at Figure 3.3 will help to see the big picture. Going from the bottom of the figure to the top might be more provide a more intuitive understanding. While every green box needs to yield an integer solution to be acceptable for the next stage, the red boxes decompose into brown boxes which evaluate those offerings from green boxes by using a LP. The brown boxes also provide the direction for the incoming columns by supplying them with dual vectors. The dark green boxes are the scenario subproblems for each SSR.

As mentioned earlier, ColGen is designed for this problem to have a benchmark (a valid lower bound) for the obtained MIP upper bounds, and as an extra, the incoming columns may improve the integer feasible solution along the way. However, our tests showed that without the acceleration and stabilization techniques introduced in Chapter 2, the performance is very poor, even for problems of size (200x20x100).

3.5.2 Acceleration and Stabilization

Computational experiments have shown that attempts at solving this huge problem using classical ColGen steps fail. A generic ColGen algorithm with no customization fails in providing a valid (positive) lower bound for a very long time. Progress in improving the lower bound is also unacceptably slow. Including the initialization of the RMP, 4 proposals are made for acceleration and stabilization of this Nested ColGen Algorithm.

Initializing the RMP

Every RMP in the algorithm decomposition structure is initialized by the relavant piece of information in the feasible solutions proposed by the Agent-based Tabu Heuristic. Including the best solution found at the termination of the heuristic, a total of 20 solutions will be fed to the RMP Level1 and RMP Level2. To reduce the possible impact of degeneracy, not the most recent 20 feasible solutions upon termination, but the best solutions that have been found after the most recent 20 tabu swaps will be used. Recall that the tabu swaps change the previous solutions drastically. Thus, the $y^q_{m,j,i,sh,sc}$ variables in RMP Level1, and $y^q_{m,j,i,sh,sc,p}$, and $ot_{i,sh-1,sc,p}$ and $ot_{i,sh,sc,p}$ variables in RMP Level2 are initialized by these solutions.

For the largest test case, 20 solutions translate to $20 * 50 = 1000$ SSR columns. Our computational results suggest that starting less than 20 solutions cause the dual of the RMP Level1 to be too relaxed; therefore, the dual vectors obtained would be misleading. More than 20 solutions to begin with, on the other hand, makes the solution of the RMP harder and hinders algorithmic performance.

Initial Dual Vector Estimation

Initial dual vector estimation is critical in order to obtain a good lower bound, even in the first iteration. The lower bound (LB_q) will be computed during the main iteration calls, where all subproblems for all levels are solved. The lower bound is calculated in any main iteration q of the algorithm and is given in the following formula. Let z^q be the sum of the objective functions for the primal feasible solutions to the LP of RMP Level1 over all shifts. Let z_i^q be the objective function for the RMP Level2 for every SSR_i.

$$LB_q = z^q - \sum_{i \in S} z_i^q \tag{3.44}$$

In order to ensure that all reduced costs for all the columns that are used in the initialization of RMP Level1 are nonnegative (thus, the dual vector to be estimated is dual feasible for the first RMP Level1), the following algorithm shows the necessary steps to achieve this objective. Recall, dual feasibility has to hold for all the corresponding columns that are used to initialize the system:

$$\begin{aligned}
&\sum_{sh \in SH} (1.5\ ch_i\ OT_{i,sh}) + \sum_{sh \in SH} (0.5\ ch_i\ cs_{i,sh}) + \\
&\sum_{m \in M} \sum_{j \in Z} \sum_{sh \in SH} \sum_{sc \in SC} \sum_{p \in itrs2} \xi_{sc} (cd\, t_{j,i} \lambda_{i,sc,p}\, y^q_{m,j,i,sh,sc,p}) - \\
&(\sum_{m \in M} \sum_{j \in Z} \sum_{sh \in SH} \sum_{sc \in SC} \sum_{p \in itrs2} \pi_{L1,m,j,sh,sc,q}\ \lambda_{i,sc,p}\ y^q_{m,j,i,sh,sc,p} + \\
&\sum_{sh \in SH} \theta_{L1,i,sh,q}) \geq 0 \qquad (3.45)
\end{aligned}$$

$$\pi_{L1} \geq 0 \tag{3.46}$$

$$\theta_{L1} \leq 0 \tag{3.47}$$

If θ_{L1}, which is the vector for all dual entries of $\theta_{L1,i,sh,q}$, is set to 0, the only unknown becomes π_{L1}, the dual vector for all $\pi_{L1,m,j,sh,sc,q}$ in the expression, above. Remember that $\lambda_{i,sc,p}$ is also given for the initialization stage. Let $q^* = 1$, for the rest of this section. Then, the upper bound for

every π_{L1} can be calculated by using the assignments for every SSR_i:

$$\pi_{L1,m^*,j^*,sh^*,sc^*,q^*} \leq (\sum_{sh \in SH} (1.5\ ch_i\ OT_{i,sh}) + \sum_{sh \in SH} (0.5\ ch_i\ cs_{i,sh}) + \sum_{m \in M} \sum_{j \in Z} \sum_{sh \in SH} \sum_{sc \in SC} \sum_{p \in itrs2} \xi_{sc}(cd\, t_{j,i} \lambda_{i,sc,p}\, y^q_{m,j,i,sh,sc,p}) - (\sum_{m \in M2} \sum_{j \in Z2} \sum_{sh \in SH2} \sum_{sc \in SC2} \sum_{p \in itrs2b} \pi_{L1,m,j,sh,sc,q}\ \lambda_{i,sc,p}\ y^q_{m,j,i,sh,sc,p}))/(\lambda_{i,sc^*,p}\ y^q_{m^*,j^*,i,sh^*,sc^*,p^*}) \tag{3.48}$$

where

$$\lambda_{i,sc^*,p}\ y^q_{m^*,j^*,i,sh^*,sc^*,p^*} > 0 \tag{3.49}$$

where $m^*, j^*, sh^*, sc^*, p^*$ are excluded from sets $M2$, $Z2$, $SH2$, $SC2$, and $itrs2b$, respectively. Let $\pi_{L1,avg}$ be the average of all positive π_{L1} after the solution of RMP Level1.

i. Set all $\theta_{L1} = 0$.

ii. Set all $\pi_{L1,m,j,sh,sc,q} > \pi_{L1,avg}$ to $\pi_{L1,avg}$. Remove these m, j, sh, sc from sets $M2$, $Z2$, $SH2$, $SC2$, and the relevant p^*s from $itrs2b$. Set all entries in π_{L1} for sets $M2$, $Z2$, $SH2$, $SC2$ to 0.

iii. Iterate over all m,j,sh,sc,p in Sets $M2$, $Z2$, $SH2$, $SC2$: Set the $\pi_{L1,m^*,j^*,sh^*,sc^*,q^*}$ to the minimum upper bound found in (3.48).

iv. Remove $m^*, j^*, sh^*, sc^*, p^*$ from Sets $M2$, $Z2$, $SH2$, $SC2$, and $itrs2b$, respectively.

The initial dual vector π_{L1} thus becomes balanced in the sense that the dual entries have similar magnitudes, and zero value duals are eliminated, as much as possible.

Dual Manipulation

As explained in Chapter 2, using the same dual vector for all subproblems in a ColGen procedure has detrimental effects. It causes unwanted columns to be brought into the RMP and those astray columns help neither RMP optimality nor integer optimality. Furthermore, if the duals are not stabilized, rapid jumps in the lower bound calculations are observed, and the progress in lower bounds is very erratic. To mitigate the hindering effects of this phenomena, intermediary Dual Manipulation is proposed.

The dual vector $\pi_{L1,m,j,sh,sc,q}$ obtained from RMP Level1 is used both in Pricing Subproblems Level1 and Level2. While the Pricing Subproblems Level1 choose the best shift assignments for every SSR by the reward provided by $\pi_{L1,m,j,sh,sc,q}$, Pricing Subproblems Level2 determine the activities that take place in every scenario. Unwanted assignments at Pricing Subproblems Level2 have cascading effects in RMP Level2, and then in RMP Level1. Moreover, $\beta_{L2,i,sc,p}$ found in RMP Level2 could be further manipulated to guide the SSRs to concentrate on critical scenarios.

The manipulation procedure proceeds in 2 steps. For every RMP Level2 belonging to each SSR, $\pi_{L1,m,j,sh,sc,q}$ is manipulated in the following way in iteration q:

$$\pi^{new}_{L1,m,j,sh,sc,q} = \pi_{L1,m,j,sh,sc,q} * \frac{\text{\# of SSRs in } type1_{m,j,sh,sc} \text{ - ranking of } SSR_i}{\text{\# of SSRs in } type1_{m,j,sh,sc} \text{ list}} \tag{3.50}$$

This is going to ensure that not the same reward is given for the same demand to every SSR.

For $\beta_{L2,i,sc,p}$, the manipulation is done in the following way, in iteration p of RMP Level2. Let $dp_{i,sc}$ be the total quantity of $d(m, j, sh, sc)$ summed over all shifts in a scenario, for the (m, j) which are in the top 20% of the $type2_i$ list (the preferred demand points list for every SSR). Then, the preferred order of scenarios for every SSR can be obtained in a list called $type4_i$. Recall that $\beta_{L2,i,sc,p}$ behaves like a penalty between SSR_i and scenario sc. The objective, here, is to reduce the penalties for preferred scenarios:

$$\beta^{new}_{L2,i,sc,p} = \beta_{L2,i,sc,p} * \frac{\text{ranking of scenario } sc \text{ in } type4_i}{\text{\# of scenarios}} \tag{3.51}$$

The final touch on the duals is applied by Wentges' smoothing. Let $\Omega_{q,p}^{new}$ represent all dual vectors (π and β) in consideration in iteration q of RMP Level1 and iteration p of RMP Level2. Also, let $\Omega_{qq,pp}^{*}$ be the best lower bound yielding representative dual vector found in the corresponding iteration for qq and pp. Then, the smoothing is done with a chosen $0 < \alpha < 1$:

$$\Omega_{q,p}^{sm} = \alpha\Omega_{qq,pp}^{*} + (1-\alpha)\Omega_{q,p}^{new} \tag{3.52}$$

α is chosen to be 0.5, in order to not wander too far away from the best lower bound yielding dual vector, but also, to stay open for new directions. According to our computational experiments, α values less than 0.33 slows down the lower bound progress by staying close to the last best dual vector, and α values greater than 0.66 reduce the desired stability in the dual domain.

Subproblem Selection

Subproblem Selection is vital for the fast convergence of the algorithm. RMP Level1's pricing subproblems are decomposed by SSRs. Therefore, if the SSR pricing subproblem that would bring the best incoming column for RMP Level1 can be estimated, the algorithm does not have to solve for all SSRs. Similarly, the Pricing Subproblem Level 2 is decomposed by scenarios for every SSR. There are particular scenarios that every SSR can make changes in, and therefore, it would impact the shift assignment in RMP Level2 significantly. Therefore, if the best set of pricing subproblems for scenarios are chosen, then, the algorithm avoids wasting time on solving potentially non-beneficial ones.

When the dual vector $\pi_{L1,m,j,sh,sc,q}$ is obtained from RMP Level1, the entries of the vector are sorted in decreasing order ($\pi_{L1,\check{sorted}}$). At the top of the list lies the most rewarding demand point (m, j, sh, sc).

i. Starting from the top of this list, find the best and the 75^{th} percentile SSR using the $type1_{m,j,sh,sc}$ lists.

ii. Continue with the next demand point in the sorted list for duals.

iii. Form the subset of subproblems to be solved with at least 10% of the total number of SSRs in the problem.

The scenario subproblem selection for SSRs is conducted likewise. The dual vector $\beta_{L2,i,sc,p}$ acts as a penalty for activities that take place in scenario sc by SSR_i (see objective function for Pricing Subproblem Level2 (3.39)).

i. Sort the entries in $\beta_{L2,i,sc,p}$ in increasing order. At the top of the list is the scenario with the least penalty for SSR_i.

ii. Pick the scenarios in the top 10 percentile for every SSR.

Steps of the Nested ColGen Algorithm

Although the Nested ColGen starts with a very good quality initial feasible solution, incoming columns can still improve the MIP upper bound. Recall that the Nested ColGen is now applied to a relaxation of the original problem. Therefore, solving RMP Level1 to integer optimality with the columns found so far may not result in an integer feasible solution for the monolithic problem. The issue is that, $SSR_{i,p}$ who is proposed in RMP Level2 in iteration p could be activated along with $SSR_{i,p2}$, which may be assigned to completely different shifts. The reason is that RMP Level1 problems are decomposed by shifts; therefore, feasibility checks do not go beyond that particular shift in that subproblem.

To be able to use the incoming columns for improving MIP upper bounds, a simple heuristic is designed. When there are more than 1 proposals offered for the same SSR_i, the heuristic removes the columns which have worse $Ct_{i,q}$ than the smallest one, leaving only 1 proposal for this SSR and fix $\mu_{i,r,sh} = 1$, $\forall sh$ in all RMP Level1 shift subproblems. After the removals, if the whole problem becomes infeasible, no update in MIP upper bound is made.

The steps of the ColGen algorithm is given below. To avoid tailing-off, a 1% duality gap is adopted to claim that the LP relaxation for that RMP is successfully solved.

- **Step 0:** Initialize the system with 20 solutions as suggested in the Initializing the RMPs section. RMP Level1 iteration $q = 0$ and RMP Level2 iteration $p = 0$.

- **Step 1:** Solve the initial RMP Level1. Obtain the first dual vector to estimate the duals to be used in the pricing subproblems below and for computing LB_q.

- **Step 2:** Set RMP Level1 iteration $q = q + 1$.

- **Step 3:** If $q\ mod\ 50 = 0$ or $q = 1$: Solve RMP Level1 to integer optimality. Apply the feasible integer solution heuristic described above. Update MIP upper bound if a better solution is found.

  Solve all RMP Level1s. Obtain UB^q, which is the sum of all objective functions of RMP Level1s. If $q \neq 1$: Obtain duals.

  If $q\ mod\ 50 \neq 0$: Manipulate Duals and select which SSRs will be used for RMP Level2.

- **Step 4:** Set RMP Level2 iteration $p = p + 1$.

- **Step 5:** If $q\ mod\ 50 \neq 0$ or $q \neq 1$: Solve the chosen RMP Level2 Problems. Obtain duals. Manipulate Duals. Select which scenarios will be used for Pricing Subproblem Level2. Go to Step 6.

  Else: Solve all RMP Level2 Problems for all SSRs. Obtain $UB_{L2,i}$. Calculate LB_q. Update $LB_q = max\{LB_{qq},\ \forall qq\}$. Obtain duals. Solve all RMP Level2 Problems to integer optimality with the existing columns.

  Check for optimality: $UB^q \leq LB_q + \epsilon$, where $epsilon = 0.001$. Terminate if optimal or close to optimal ($\leq 1\%$ duality gap for Level1) Go to Step 2.

- **Step 6:** If $p\ mod\ 50 \neq 0$ or $p \neq 1$: Solve the chosen Pricing Subproblem Level2s to integer optimality. Go to Step 4.

  Else: Solve all Pricing Subproblem Level2s for all scenarios. Obtain $LB^p_{L2,i}$.

  Update $LB_{L2,i} = max\{LB^{pp}_{L2,i},\ \forall pp\}$

  Check for optimality: $UB^p_{L2,i} \leq LB_{L2,i} + \epsilon$, where $epsilon = 0.001$. Terminate if optimal

or close to optimal ($\leq 1\%$ duality gap for Level2 for SSR_i)

Go to Step 4.

Certificate of Valid Lower Bounds

Proposition1: The Nested Column Generation Algorithm defined above, provides valid lower bounds to the original monolithic problem for the Deterministic Equivalent of the Workforce Shift Scheduling Problem with Stochastic Demand.

Proof: The Nested Column Generation Algorithm with LoA enhancements terminates in finite number of steps, because the main iteration call is made with a known frequency. Repetition of columns cannot happen during the main iterations owing to the dual vector being feasible to the most recent RMP Level1's dual counterpart.

Then, let LB^*_{mono} be the integer optimal solution to the monolithic problem, LB^*_{rlx} be the integer optimal solution to the problem when constraint set (3.23) is relaxed and converted to (3.28). Also, let LB_{rlx_LP} be the optimal solution to the LP relaxation of the relaxed problem and let LB^*_q be the lower bound obtained at iteration q of RMP Level1 where no more columns with negative reduced cost are found. It is proved in [40] that for any integer problem with block-angular structure, $LB^*_q \geq LB_{rlx\_LP}$ due to the integer solutions coming from the subproblems. The algorithm described above calculates LB^*_q only during the main iterations where all subproblems are solved. Thus, the following statement is true when RMP Level1 concludes with a main iteration and no incoming columns are detected:

$$LB_{rlx\_LP} \leq LB^*_q \leq LB^*_{rlx} \leq LB^*_{mono} \tag{3.53}$$

3.6 Computational Results

We were only able to compare our solutions with CPLEX for problems with fewer than 50 scenarios and 400(m, j)x40SSRs combinations or less. However, results for larger size problems will also be

displayed. Both the Tabu and the Nested ColGen algorithm are coded in Python programming language. ColGen uses AMPL as the mathematical modelling language and CPLEX 12.2 is the IP and LP solver. The solution times are obtained on a computer with the following properties: 4 GB RAM and Pentium Xeon 3.0 GHz(x2) CPU.

The pleasing outcome is that the Agent-based Tabu Algorithm reported very high quality solutions when allowed to operate for 30 minutes. They were so good that the improvement gained from the Nested ColGen is very minor compared to the initial solutions. Figure 3.4 shows the progress of the MIP upper bound versus time, for a (400x40x100) size problem during the heuristic phase. The spikes in the figure for the heuristic performance are due to the tabu swaps accepted, which worsen the current best solution.

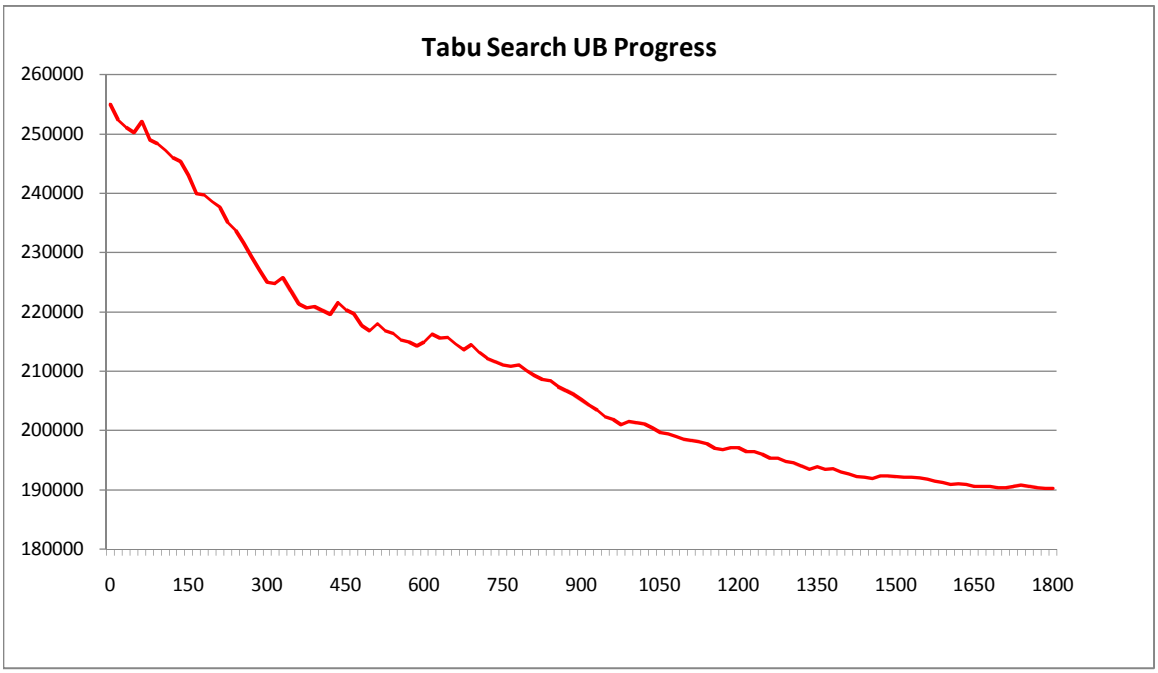


Figure 3.4: Tabu Search Algorithm Upper Bound Progress

The number of scenarios to generate for each problem was another decision to make while conducting the computational experiments. Throughout this chapter, the number ”100” was alluded to, as if it was the default choice. However, if this solution is to be presented to the field manager at a service company, we would choose to run our experiments with 125 iterations. Table 3.1 shows

the percentage of solutions that are feasible among 1,000 randomly generated scenarios, while the solutions were found by using 25 through 150 scenarios. In assuming feasibility in every scenario, it is meant that in the solution found, there is a sufficient number of SSRs with specific skills assigned to every shift with positive demand. The (1500x300) size problems do not benefit much from increasing the number of scenarios; the last significant step is from 100 to 125. Therefore, all results for (1500x300) size problems are run with 125 scenarios. For all other problem sizes, 100 scenarios are used for computational testing.

Table 3.1: Impact of Number of Scenarios on the % of Solutions that are Feasible

#scenarios	25	50	75	100	125	150
200x20	90.1%	95.2%	98.2%	99.2%	99.5%	99.9%
400x40	75.2%	83.5%	92.9%	96.2%	97.3%	98.1%
1000x100	69.4%	81.9%	88.9%	94.9%	96.9%	97.2%
1500x300	49.2%	65.7%	80.4%	90.9%	94.6%	95.1%

Figure 3.5 provides evidence that the chosen range of 125 to 150 scenarios is appropriate for (1500x300) size problems. While the number of scenarios increase from 25, the number of overtime and consecutive shift assignments increase as well. However, the selected shift assignments begin to become sufficient to cover the distinct scenario demands after 125 scenarios. Thus, the objective function value of the problem levels out after 125 scenarios.

Table 3.2 presents the average solution times and tabu swaps taken for 10 instances of each problem size. The quality of the solutions are also calculated by the best lower bound found after Nested ColGen concluded. The quality of the solutions are good enough that designing a Branch-and-Price approach could be considered unnecessary. The number of tabu swaps is much larger for the smaller problems, due to intermediate solutions quickly finding local optimal points. As the problem size grows, there are more opportunities for valid swaps, therefore, the number of tabu calls is lower. For all instances of 200x20 to 600x60, the heuristic terminates before the 30 minute time limit, due to not being able to find a valid swap at that particular local optimal point.

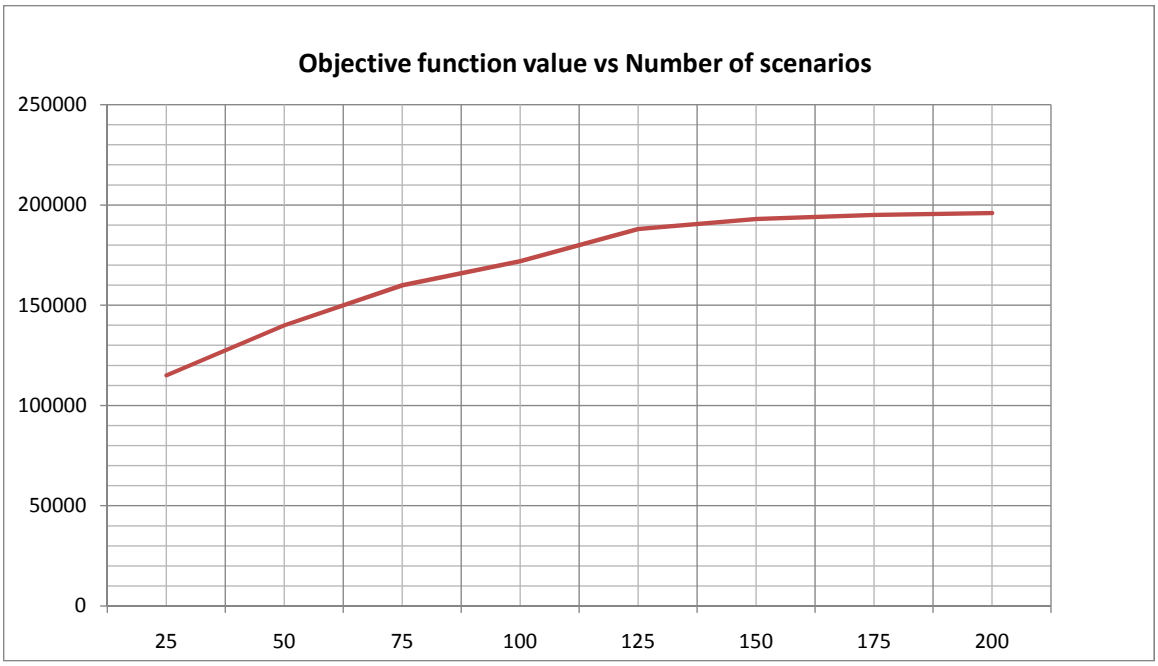


Figure 3.5: Objective Function Values changing with Number of Scenarios

Table 3.2: Agent-based Tabu Heuristic Computational Results

Size	#vars	Time	Gap	Tabu Swaps
200x20	1.2M	1,002	2.4%	1,308
300x30	1.9M	1,211	2.1%	1,121
400x40	2.5M	1,238	4.3%	804
600x60	3.7M	1,459	3.9%	503
1000x100	6.1M	1,719	5.1%	450
1500x300	11.3M	1,800	7.6%	312

The next table shows the solution time comparisons among CPLEX (when it can load the problem), ColGen, and LoA+Tabu enhanced ColGen for 10 instances of each problem size. LoA+Tabu enhanced ColGen solution time includes the time spent during the Agent-based Tabu Algorithm. The number of RMP Level2 iterations is its. CPLEX was given a 12 hour time limit with default solver options. Unfortunately, CPLEX could not provide an integer feasible solution after 12 hours

(43,200 seconds) for 300x30, and problems greater than 300x30, they were not even loadable. The Default Colgen did not perform up to its expectations from the textbooks. The LoA+Tabu enhanced ColGen, on the other hand, starts with already high quality integer feasible solutions and with good lower bounds; therefore, it quickly converges to the root node optimality. Thanks to the Subproblem Selection logic, more column proposals (SSR-Scenario) are made to RMP Level2, compared to the Default ColGen.

Table 3.3: CPLEX vs Default ColGen vs Colgen with (LoA+ID+DM+SS)

	CPLEX		Default ColGen			Tabu+LoA+ID+DM+SS		
Size	Time	Gap	Gap	Time	its	Gap	Time	its
200x20	43,200	97.4%	25.4%	20,211	4,634	1.9%	3,592	14,210
300x30	43,200	n/a	48.3%	29,004	6,217	2.0%	4,240	17,790
400x40	n/a	n/a	111.1%	38,743	6,912	4.1%	5,892	20,123
600x60	n/a	n/a	86.7%	43,200	9,034	3.1%	8,089	30,631
1000x100	n/a	n/a	89.3%	43,200	12,256	4.2%	11,430	36,555
1500x300	n/a	n/a	169.8%	43,200	11,413	6.5%	16,354	48,215

3.7 Discussion

Shift scheduling is a complex operational level decision making problem, which should be carefully done in a small amount of time to react to the dynamic nature of service businesses. In this chapter, a mixture of heuristics and a formal ColGen algorithm are utilized to successfully solve such a problem with intractable number of variables, in a reasonable amount of time.

The heuristic proposed can respond to perturbations in the system very fast due to the cheap computational effort for swap benefit calculations; however, the actual optimal solution is very difficult to attain. The Nested ColGen Algorithm is not only utilized to find high quality lower bounds, but also, it produces columns that improve the solution. The issue is, since RMP Level1

is not 1-to-1 equivalent of the original monolithic problem (recall that it is a relaxation), Branch-and-Price is not pursued. Thus, the optimal solution, if ever obtained, during the root node solve of RMP Level1, would be purely coincidental.

Chapter 4

Applications on other Capacitated Resource Management Problems and Guidelines for Algorithmic Setup

4.1 Applicability of LoA on Various Problems

This chapter is dedicated to the demonstration of possible uses of LoA on Large Scale Capacitated Resource Management Problems. While Chapter 2 and 3 showed the actual implementation of the algorithm and provided numerical results, this chapter will focus on how LoA could be applied to several well-known research problems in the literature. An interested reader is invited to follow the instructions and guidance to successfully implement Column Generation for the particular problem type under consideration.

4.1.1 Train Routing Problems

A very challenging problem in the Railway Business is to make the most cost effective decisions regarding the routing of trains, utilization of their capacities, and managing the places of pick-up

and set-out events. A pick-up event occurs when additional cars are attached to an existing train (for supply purposes) and a set-out event happens when the train arrives at the customer destination to unload the commodity it is carrying. This problem is most commonly realized with trains assigned to move agricultural products. The reason for the complexity arises from the fact that there are numerous places on the railway network where a train can pick-up cars and there are more demand destinations to set-out. Since starting a new train requires crew and locomotive and has a fixed cost, an efficient railway system cannot run a separate train for every distinct origin-destination (supply-demand) pair. Therefore, a good plan is necessary to dictate where the trains will originate and stop at which customer destination terminals to set-out its cars.

Assumptions made before building the mathematical model are:

- The railroad network $G = (N, e)$ has finite number of nodes in set N and every node is accessible from another node using the edges e.
- The shortest path is already known for every origin-destination node pair using the edges.
- Single commodity (for instance grains) is carried.
- Sum of all the demands along the route of the train is loaded at the origin of the train.
- The total demand (number of cars) at every node is known: $Unload_j$, where $j \in D$
- Trains in set T can only be generated at the nodes with infinite positive supply (e.g. huge grain depots): $i \in S$
- Total demand can always be fulfilled from the supply nodes.
- The terminals have sufficient outbound and inbound capacities for departing and incoming trains, respectively.
- The railway network has sufficient parallel tracks and sidings to accommodate the number of trains necessary in any given day.

- The final node of every feasible route is a temporary gathering and distribution terminal on the network N.

The objective of the study is, then, to minimize the costs regarding the operation of the activated trains while conforming to the flow preservation constraints and train capacities. A train with more than 140 cars is considered to be too large owing to the fact that it requires more locomotives for power, and consumes more fuel.

Thus, the mathematical model resembles that of a Vehicle Routing Problem as stated in [34]:

Problem Train_Routing:

minimize operational costs:

$$\sum_{t \in T} f_t \, y_t + \sum_{t \in T} \sum_{i \in N} \sum_{j \in N} c_{i,j,t} \, x_{i,j,t} \tag{4.1}$$

subject to :

$$\sum_{t \in T} \sum_{i \in N} x_{i,j,t} \geq 1 \qquad \forall j \in D \tag{4.2}$$

$$\sum_{t \in T} \sum_{j \in N} x_{i,j,t} \geq 1 \qquad \forall i \in S \tag{4.3}$$

$$\sum_{i \in N} \sum_{j \in D} Unload_j \, x_{i,j,t} \leq cap_t \, y_t \qquad \forall t \in T \tag{4.4}$$

$$\sum_{i \in N} x_{i,h,t} - \sum_{j \in N} x_{h,j,t} = 0 \qquad \forall t \in T, h \in D \tag{4.5}$$

$$x_{i,j,t} \quad binary \qquad \forall t \in T, i \in N, j \in N \tag{4.6}$$

$$y_t \quad binary \qquad \forall t \in T \tag{4.7}$$

The objective function (4.1) penalizes every train creation y_t by a fixed cost f_t and considers the costs $c_{i,j,t}$ for traversing on the arc (i, j) by train t, denoted by $x_{i,j,t}$. Constraint (4.2) ensures that at least 1 train stops at every demand node j, similarly (4.3) guarantees that there is at least 1 train departure from supply node i. Constraint (4.4) restrains the number of cars that can be picked up by

train t, based on the demand along its route. Flow conservation constraints (4.5) indicate that trains will have to leave for another node after stopping for a set-out at the customer terminal. (4.6) and (4.7) are binary declarations for the arc assignments and train activations, respectively.

The transshipment model introduced above has an intractable number of variables because of the number of routes a train can take even in a modest size network. A Column Generation based approach, therefore, is an appropriate method to attack this problem. We claim, however, that utilizing LoA can make the ColGen algorithm even more attractive.

The decomposition of the problem into the master and the pricing subproblems is carried out by keeping the coupling constraints (4.2 and 4.3) in the master problem, and putting the train specific capacity (4.4), flow conservation (4.5), and binary declarations (4.6 and 4.7) in the subproblems for each train. Very similar to the Workforce Planning Model introduced in Chapter 1, the subproblems are expected to propose trains with distinct and feasible routes for supply and demand nodes, while abiding by the capacity constraints. The issue is that with an unbalanced initial dual vector, and with the lack of smoothing of the duals based on the likelihood of a particular train attached to a route, the subproblems will try to offer useless incoming trains to the Restricted Master Problem. Recall that the RMP decides which of the proposed trains from the incoming columns are to appear in the optimal network. Therefore, it is critical that only trains with beneficial routing assignments are brought in.

To alleviate this matter, the LoA approach could be adopted. LoA can leverage the fact that there is a distinct cost associated with starting a train on every unique node. That is owing to the track grade which causes different numbers of locomotives and fuel requirements. Based on the distance between the origination node of the train and the demand nodes j on the network, LoA could penalize arcs (i, j) and routes which are clearly not in the shortest path of this particular train. The size of the demand will also be a factor in the LoA calculation. The nodes closer to the origin with larger demand quantities will be more preferable in the optimal solution.

Thus, using a similar logic as in Chapter 1, a penalty vector $wR_{i,j,t}$ for every arc (i, j) and train t, can be calculated based on the factors denoted above. Right after the creation of the wR vector,

the likelihood of assignment (preferred train) list for each arc (i, j) can be obtained. A greedy heuristic scheme can easily assign every arc to its best train until the capacity constraint is violated. The route can then be constructed, when which demand nodes (j) are serviced by this train t are known. The first pass provides the initial feasible solution to the RMP, but, randomizing wR with $Uniform \sim [0.90, 1.10]$ and applying the greedy scheme, repeatedly, would provide diverse a set of diverse bases for the RMP, so that the dual of the RMP is not degenerate. Degeneracy mitigation is critical for rapid lower bound and upper bound progress.

The initial dual vector obtained from the RMP is then utilized for estimating the first dual vector to be used in the pricing subproblems as shown in **Section 2.3.1**. The Dual Manipulation will be necessary for better incoming columns using the train preference list created for each arc (i, j). The Subproblem Selection would also be similarly performed, by sorting the dual values retrieved for every demand constraint, and then selecting the best train subproblem in the hopes for finding a good incoming column with a high negative reduced cost value.

4.1.2 Capacitated Facility Location Problems

Although Capacitated Facility Location Problems (CFLP) are very difficult problems by their nature, there are numerous heuristics and algorithms developed in order to solve them. The articles published which use column generation as the basis for an algorithmic solution show that for even modest size problems, they encounter convergence and performance issues; see ([47], [39], [56], [35]).

A generic CFLP is modeled in the following way:

Problem CFLP:

minimize Facility Opening and Distribution Costs:

$$\sum_{j \in J} f_j \, y_j + \sum_{k \in K} \sum_{j \in J} c_{k,j} \, x_{k,j} \tag{4.8}$$

subject to :

$$\sum_{j \in J} x_{k,j} = 1 \qquad \forall k \in K \tag{4.9}$$

$$\sum_{k \in K} d_k \, x_{k,j} \geq s_j \, y_j \qquad \forall j \in J \tag{4.10}$$

$$\sum_{j \in J} s_j \, y_j \geq d(K) \tag{4.11}$$

$$x_{k,j} - y_j \leq 0 \qquad \forall k \in K, j \in J \tag{4.12}$$

$$0 \leq x_{k,j} \leq 1 \qquad \forall k \in K, j \in J \tag{4.13}$$

$$y_j \quad binary \qquad \forall j \in J \tag{4.14}$$

In the formulation above, J is the potential facility locations, K is the set of customers, $c_{k,j}$ is the distribution cost of all the demand d_k by customer k from facility j, which has capacity s_j. f_j is the facility opening cost which is activated when the binary variable y_j is; i.e., it is decided to be open the facility. $d(K)$ is the sum of all demands. Finally, $x_{k,j}$ indicates the fraction of the demand customer k is asking from facility j. The objective function tries to minimize the cost of facilities being opened along with the total distribution costs. Constraint (4.9) ensures that all demands are fulfilled, while constraint (4.10) guarantees that the capacity is not exceeded for any of the facilities. The aggregate capacity constraint (4.11) and activation constraints (4.12) are in place for strengthening the formulation.

A ColGen Algorithm would only keep the demand and aggregate capacity constraints (4.9 and 4.11) in the Master Problem in order to decompose the problem by every facility. The individual capacity constraint (4.10), and the activation constraints would then be placed in the pricing

subproblems, along with the fractional $x_{k,j}$ and binary y_j declarations.

The solutions of the subproblems are specific facilities with customer assignments and supply fractions for those particular customer demands. The pricing subproblems, by their design, try to find the best combination of customers and supply amounts, while looking for the most negative reduced cost. The reduced cost is calculated similar to the formula given in (2.22), where the greatest impact is due to the dual variables from the demand constraint (4.9), aggregate capacity constraint (4.11) and the convexity constraint for each subproblem.

LoA methodology can again be shown to prove useful in this problem setting. When the ColGen is not properly initialized, the subproblems will most likely propose facilities with rather bad customer assignments and supply amounts. The primary decision behind opening a facility is the fixed cost of actually opening the facility. This is going to be factor number 1 while constructing the penalty vector $wR_{k,j}$ for each assignment $x_{k,j}$. The next important factor is the cost of distributing goods from supplier j to customer k. The size of the demand is also vital in establishing cheap distribution assignments.

Once penalty vector wR is found using the factors above, the preferred facilities list for every customer can be constructed. While abiding by the capacity constraint for each facility and ensuring all demand is met, the assignments which are unlikely to exist in the optimal solution are eliminated. Then, Problem Integer_Yielding_LP_cflp could be used to initialize the RMP, by randomizing the wR vector.

Problem Integer_Yielding_LP_cflp:

minimize overall penalties:

$$\sum_{k \in K} \sum_{j \in J} wR_{k,j}\, x_{k,j} \tag{4.15}$$

subject to :

$$\sum_{j \in J} x_{k,j} = 1 \qquad \forall k \in K \tag{4.16}$$

$$\sum_{k \in K} d_k\, x_{k,j} \geq s_j \qquad \forall j \in J \tag{4.17}$$

$$0 \leq x_{k,j} \leq 1 \qquad \forall k \in K, j \in J \tag{4.18}$$

The solution out of Problem Integer_Yielding_LP_cflp is post-processed to obtain actual y_j activations. After the RMP is initialized with the solutions fed from this simple LP, the estimation of the initial dual vector is carried out for balancing the first dual vector to be used in the pricing subproblems. In order to avoid useless incoming columns, Dual Manipulation is invoked to smooth the reward for unwanted assignments. The Subproblem Selection is again called to accelerate the ColGen process by choosing the facilities which, potentially, would yield the most negative column. This is achieved by sorting the dual values obtained for (4.9) and choosing the best facility in the preferred list for that customer.

The flexibility and robustness of the procedure arises from the fact that, LoA is readily applicable to any specialized version of CFLP, even the multi-commodity case without loss of generality, where only certain facilities can provide service to specific customers.

4.1.3 Aircrew Pairing and Rostering Problems

In spite of the fact that providing detailed model description for the Aircrew Pairing and Rostering Problems is beyond the scope of this thesis, the reader is encouraged to consider using LoA while attacking these type of problems. It is recommended to use a 2-stage decomposition approach, which

is utilized to solve the Manpower Planning and Scheduling Problem. While the first stage problem finds the best headcount for pilots and crew along with their appropriate aircraft assignments for a given schedule, the second stage problem finalizes the shift scheduling and flight assignments, as we also propose for SSRs. Of course, the restrictions from the Federal Aviation Administration are far more strict and complicated than for manpower scheduling in a technical service support company. These being pilot/crew specific restrictions, however, cause the subproblems to have more restrictive feasible domain, but still solvable with efficient network algorithms.

LoA becomes valuable when there are millions of different pairings between crew and aircrafts, and millions of possible shift assignments. Based on the factors such as crew's seniority/skills, crew's preference, crew's current rest amount, idle/rest hours between consecutive flights, and length of the total route, a penalty value can practically be assigned to every crew to aircraft to shift assignment. After initializing the RMP (by a greedy heuristic using the penalties for assignments) for ColGen, preference lists created by this penalty vector can be used to manipulate the duals for incoming columns, and help to select the next batch of (crew) subproblems to solve.

4.1.4 Discussion

Applicability of LoA has only been shown on a few of the most prominent research problems in this section. However, we believe that instead of finding Likelihood of Assignment of a capacitated resource to a task, the variables in any block-angular problem that is suitable for column generation could be assessed for their potential to appear in the actual optimal solution. That in turn would provide the sorted lists that are utilized to activate the stabilization and acceleration methods introduced earlier. The effectiveness of LoA on other problem structures has only been speculated upon and its real utility in the proposed concepts is a subject for future research.

4.2 Parameter Tuning Suggestions for $Colgen_{LoA}$

This section of the thesis is prepared to showcase the flexibility and robustness of the Likelihood of Assignment Heuristic. Although the best combination of parameters were used while providing the numerical results for Workforce Planning with Cross Training Problem in Chapter 2, and for the Stochastic Scheduling Problem in Chapter 3, by changing various algorithmic parameters, users of the heuristic can work towards their particular objectives. The different settings can let them achieve a higher quality solution or a better lower bound at first. Another user may only be interested in generating different integer feasible solutions to see their potential action alternatives.

4.2.1 # of Variables to Eliminate using LoA

Recall that in Chapter 1, approximately only 10% of all SSRs are retained in the problem for each demand point (m, j). As this percentage increases, it is guaranteed that the solution obtained will be at least as good as the solution from the 10% setting. On the other hand, decreasing the number of eliminated variables causes the size of Problem Integer_Yielding_LP to grow proportionately. The following table shows the solve times for different percentage settings for variable retention, and the quality of the solution obtained against the best known lower bound provided by BP_{LoA}:

Table 4.1: LoA Retention Sensitivity for Problem Size (1500x300)

%	#vars_LoA	LoA_Gap	LoA_T
10%	91,699	11.8%	75
20%	183,399	9.2%	224
25%	229,277	8.3%	311
33%	302,727	6.4%	454
50%	458,780	6.0%	657
100%	917,560	5.9%	1,528

Table 4.1 is a good display of the value of effective variable elimination by LoA. There are

1,209,300 total variables in the original problem. When only half of the ($x_{m,j,i}$) variables are retained, the solutions obtained from the test cases are not bad at all, with the average mipgap being 6.0%. When Problem Integer_Yielding_LP is solved without any $x_{m,j,i}$ variables getting eliminated, solution quality does not get more than 0.1% better than the solutions obtained from 50% retention rate. This is a strong indication that the LoA sorting and elimination logic is doing its job, and almost all of the critical variables are kept in the top 33% of the sorted lists of beneficial Likelihoods of Assignments. The reason why the actual optimal solution cannot be obtained even though all $x_{m,j,i}$ variables are retained is that, Problem Integer_Yielding_LP is only trying to estimate the effects of the binary associations from the original model.

The mipgap of 6%, however, can only be known when there is a valid lower bound provided, meaning, it is an "after the fact" value. If BP_{LoA} or CPLEX had not provided a lower bound for the mipgap calculation, the algorithm would have been in its 12^{th} minute, with an integer feasible solution, but, against no value to compare to. Repeating the Problem Integer_Yielding_LP with this slow setting is not desirable. Despite the fact that only one initial feasible solution is sufficient to invoke ColGen, the corresponding dual vector obtained would be useless since this dual vector would be nowhere close to the optimal RMP_dual solution. This would also cause a very problematic lower bound calculation as explained in **Section 2.3.1**.

The trade-off here is to choose between a setting that yields higher quality solutions versus the one which solves fast enough to be repeated during the initialization of ColGen. We recommend using the smallest useful number of variables (e.g. 10%, since it is very difficult to have a feasible solution with less than 10% retention). With the help of the lower bound acceleration and new integer feasible solution finding techniques introduced in this research, it is preferred to initialize the ColGen algorithm as quick as possible and then expect to obtain better quality integer feasible solutions as the algorithm progresses.

4.2.2 Randomization of wR

In order to initialize the feasible basis for the first RMP of ColGen, Problem Integer_Yielding_LP is solved repeatedly with randomized values of the wR penalty vector. This subsection shows the impact on the quality of the first integer solution obtained when randomization is tightened from 10%, which is the value recommended in Chapter 2, and also displays the outcome in the opposite case where randomization boundaries are loosened.

Table 4.2: Impact of wR Randomization for Problem Size (1500x300)

$U \sim$	$\pm 1\%$	$\pm 2.5\%$	$\pm 5\%$	$\pm 10\%$	$\pm 25\%$	$\pm 33\%$	$\pm 50\%$
avg_mipgap	11.9%	13.5%	14.3%	16.1%	24.3%	28.4%	43.4%
$\#root\_itrs$	62,244	57,498	45,889	31,313	28,229	38,904	59,982

$\#root\_itrs$ is the average number of iterations it took ColGen to solve the root node for 20 instances of (1500x300). avg_mipgap in Table 4.2 refers to the average mipgap that is calculated after solving the recommended number of (20) instances of Problem Integer_Yielding_LP, using Uniform distribution with the specified bounds. The lower bound used in the mipgap calculation is the best known lower bound (or the optimal solution). $\pm 1\%$ means every entry in the original wR vector will either be reduced at most by 1% or increased by 1%, depending on the random number generated by the computer. Although the best integer feasible solution from Problem Integer_Yielding_LP might have a much better mipgap than the avg_mipgap, there are also rather bad quality integer solutions obtained from these randomized cases. Recall that the best solution we presented in Chapter 2 by using LoA has 11.8% mipgap for the 1500x300 problem, but the avg_mipgap is computed to be 16.1%, as can be seen under the $\pm 10\%$ column.

Inspection of the avg_mipgap values for smaller range randomization reveal that the new solutions do not vary a lot from the original wR vector's solution. Despite the fact that this statement sounds good, it actually adversely affects the ColGen performance. Remember that ColGen's primary objective is to solve the RMP, not the integer optimization problem. Therefore, when too many

similar/parallel columns in the primal domain coexist, they also cause a very degenerate dual domain. This causes iterations with no lower or upper bound improvements recorded. That is why the number of ColGen iterations at the root node is almost twice as much when very similar columns are used to initialize the first RMP. On the opposite side, when the randomization is larger than $\pm 10\%$, the diversity of incoming columns is pleasing. In fact, the number of iterations for the root node is even less when randomization is at $\pm 25\%$. The issue is, even if ColGen converges for the root node, the integer feasible solution is poorer than when $\pm 10\%$ is used.

More computational experiments may be conducted with distinct demand input data in order to find the best range for randomization. However, the poor quality integer solutions retrieved when any value of randomization larger than $\pm 10\%$ is used, supports our claim that the wR calculation initially proposed in Chapter 1 is a quality result.

4.2.3 Number of Columns to Initialize the RMP

Another critical decision is to determine the number of columns to initialize the first RMP of the ColGen procedure. RMP, in theory, can be initialized with only 1 feasible solution, which is composed of some number of columns. For the largest test instance (1500x300), the optimal solution to Problem Integer_Yielding_LP provides 1 feasible solution (approximately 50 SSRs, hence columns) every time it is invoked. The rationale behind initializing the RMP with more than one feasible solution is to make sure the dual of the RMP has sufficiently many constraints and therefore is tight. When the feasible region for the dual of the RMP is not tight at all, the dual vectors obtained from such a RMP are both unbalanced and too distant from the actual optimal dual vector. Conversely, when the RMP is initialized with too many feasible solutions from Problem Integer_Yielding_LP, there is a greater risk for degeneracy in the dual domain, because similar columns in the primal domain would transform to parallel constraints in the dual problem. Moreover, as the number of columns increase in the RMP, it gets harder and takes longer to solve the RMP, and solving RMP_integer becomes much more difficult. Therefore, column management needs to be carefully designed from the beginning of the algorithm.

4.2. PARAMETER TUNING SUGGESTIONS FOR $COLGEN_{LOA}$

Many experiments have been carried out to find the "magical" number of times to solve the Problem Integer_Yielding_LP. The following table summarizes the findings and helps to deduce a rule:

Table 4.3: Number of Solutions to Initialize the RMP

Size/Sols	10	15	**20**	33	66	100
200x20	49	41	35	33	29	36
300x30	98	92	85	75	78	84
400x40	108	99	95	94	111	124
600x60	294	288	270	273	301	338
1000x100	1,021	820	775	780	824	893
1500x300	9,582	9,356	8,994	9,274	9,890	11,217

Table 4.3 displays the average time required for 20 instances of the root node of the specified size problems when Problem Integer_Yielding_LP is invoked 10, 15, 20, 33, 66, 100 times with randomized ($\pm 10\%$) wR for initializing the first RMP. For (1500x300) size problem, 100 solutions corresponds to approximately $100 * 50 = 5,000$ columns for the initial RMP. The number of columns can be obtained for other problem sizes similarly. The numerical experiments suggest that when the original problem size is small (i.e. less than 35,000 variables or smaller than 400x40 problem setting), increasing the number of calls to the LoA heuristic is favorable (up to 66 calls), and it improves the solution times. However, as the problem size grows, starting with too many columns hurts the performance of the algorithm, due to the fact that RMP and RMP_integer becoming more difficult to solve rather early in the procedure. Starting with fewer solutions less than 20 turned out to be never favorable.

In the algorithm declaration of $Colgen_{LoA}$, 20 is proposed as the appropriate number of solutions to initialize the RMP for this problem structure, regardless of the problem size. The reason why 20 is the recommended number for even smaller problem sizes is that Table 4.3 shows the averages for each problem size as a group. However, every instance is recorded separately. After

checking the maximum solution times for each problem size group, the solution time for the most difficult instance is found to be longer when the initialization is done with more than 20 solutions.

4.2.4 Modifications to the Problem Integer_Yielding_LP

The main benefit of using Problem Integer_Yielding_LP is that it is an LP and it solves fast. Moreover, the assignment variables can easily be post-processed to get the values for the original binary variables. However, the model in which the variable eliminations are to be activated does not have to be a LP. When only SSR activation binary variables ($mu_i, \forall i \in S$) are kept in the model as opposed to the skill upgrade variables $ys_{i,k}$ and machine training variables $yk_{i,m}$, the model still solves in a reasonable amount of time for a heuristic and provides much better mipgaps than Problem Integer_Yielding_LP. The reduced model then becomes:
Problem Reduced_Binaries:

minimize LoA penalties + SSR base salary:

$$\sum_{i \in S} (125\, ch_i\, mu_i) + \sum_{m \in M} \sum_{j \in Z} \sum_{i \in S} x_{m,j,i} wR_{m,j,i} \tag{4.19}$$

subject to :

$$\sum_{i \in S} x_{m,j,i} = d_{m,j} \qquad \forall m, j \tag{4.20}$$

$$\sum_{m \in M} \sum_{j \in Z} x_{m,j,i} (rt_{m,j} + t_{j,i}) \leq 125 mu_i \qquad \forall i \tag{4.21}$$

$$x_{m,j,i} >= 0 \qquad \forall m, j, i \tag{4.22}$$

$$mu_i \qquad binary \qquad \forall i \tag{4.23}$$

Problem Reduced_Binaries produce very encouraging results in terms of solution quality as can be seen in Table 4.4. However, solution speed is not fast enough to be used in ColGen, beyond problem sizes of (600x60). This model could be used by practitioners to visualize a possible good solution for a problem where finding a *good* feasible solution is not trivial.

Table 4.4: Integer_Yielding_LP vs Reduced_Binaries

	CPLEX		Integer_Yielding_LP			Reduced_Binaries		
Size	#vars	Time_1%	#vars	Gap	Time	#vars	Gap	Time
200x20	8,620	65	749	5.4%	12	769	2.2%	19
300x30	18,930	369	1498	6.1%	13	1,528	3.8%	28
400x40	33,240	2,711	2,568	7.5%	30	2,608	6.6%	65
600x60	73,860	8,753	5,671	9.9%	46	5,731	6.9%	88
1000x100	203,100	24,543	15,408	10.8%	66	15,508	7.1%	352
1500x300	1,209,300	224,844	91,699	11.8%	85	91,999	7.4%	9,445

4.2.5 Varying the Intensity of Intermediate Dual Vector Manipulation

At every iteration of the $ColGen_{LoA}$, the dual vector obtained from the solution of RMP is manipulated in 2 ways. First, the dual vector entries for the demand constraints, $\pi_{m,j}$, are altered depending on which pricing problem they will be used in. This ensures that astray columns, which would be beneficial neither in solving the RMP or the actual integer optimization problem, are avoided. A second manipulation is applied in addition to the first one, where a convex combination of the dual vector that yielded the best lower bound so far and the manipulated most recent dual vector is constructed. The second manipulation is called Wentges' Smoothing and used to control the lower bound's zigzag behavior.

Recall that after which subproblems to be solved at iteration q are determined the first stage dual manipulation takes place using the following formula:

$$\pi_{m,j,q_i} = \pi_{m,j,q} \frac{\text{number of SSRs} - \text{ranking in the } type_{1,m,j} \text{ list}}{\text{number of SSRs}} \tag{4.24}$$

What equation (4.24) asserts is that the new values of the duals for each SSR_i will be based on their percentile in the $type_{1,m,j}$ list for the demand point (m, j). Even though, Dual Manipulation (DM)

with Wentges' Smoothing has a significant performance impact with the above setting, practitioners may adopt a more aggressive approach if they would like to discover better integer feasible solutions in the early iterations by only keeping the dual entries positive for the top 33% of the $type_{1,m,j}$ list, and set all other entries to 0. In **Section 4.2.1**, it is shown that very few critical variables remain outside the top 33% of such lists. The drawback, here, is that the incoming columns will be primarily helping the problem RMP_integer, but not the RMP itself. This is because too many columns with similar properties will be introduced, which causes severe dual degeneracies. It is the practitioner's preference to obtain different quality integer solutions earlier versus improving the lower bound as fast as possible. It is recommended that keeping the DM formula as it is, is the good choice for general optimization audience, where achieving high quality lower bounds rapidly is also vital.

Selecting the smoothing parameter for Wentges' method is also critical for algorithmic performance. Recall the formula:

$$\pi^{new}_{m,j,q_i} = \alpha\, \pi_{m,j,q_{best}} + (1-\alpha)\, \pi_{m,j,q_i} \qquad \forall m, j \tag{4.25}$$

$$\theta^{new}_{i,q} = \alpha\, \theta_{i,q_{best}} + (1-\alpha)\, \theta_{i,q} \qquad \forall i \tag{4.26}$$

If a large α is used (e.g. ≥ 0.66), then it becomes very difficult to move away from the initial lower bound found. On the contrary, when α is less than 0.33, the stabilization begins to lose power. That is why in this thesis, the default value is chosen as 0.5. However, a good recommendation to a practitioner who already knows his lower bound is not good at the beginning, would be the following: He should use $\alpha \leq 0.33$ to explore non-parallel incoming columns and try to move away from the current lower bound with fresh dual vectors. Conversely, if he knows from previous knowledge that his lower bound is already good enough, he should choose $\alpha \geq 0.66$ in order not to waste iterations on discovering incoming columns which would transform to non-beneficial corresponding constraints in the dual domain.

4.2.6 Selection of the Subproblem Subset Size and Frequency of Solving All Subproblems

In order to accelerate $ColGen_{LoA}$, picking which subproblems to be solved at every iteration is a very critical decision. To be able to claim that a good subset is chosen using the method given in **Section 2.3.3** versus a subset chosen arbitrarily (e.g randomly picked from the set of all subproblems), the next table will present the solution times obtained for the root node solution of size 1500x300 problem, while the subset size is varying:

Table 4.5: Effects of Random Selection vs SS Method for Varying Subset Sizes

SS	5%	r_5%	10%	r_10%	25%	r_25%	33%	r_33%	50%	r_50%
Time	18,998	46,042	8,994	36,490	14,223	28,473	23,994	27,338	37,453	39,443

The experiments illustrate why 10% of the subproblems are to be chosen by SS. Using 5% turns out to be too few in order to bring in columns which need to improve both the lower bound and the upper bound. Subsets greater than 10% on the other hand results in rapidly growing RMP, which becomes harder to solve at every iteration. Also, arbitrary selection of subproblems is not a good idea, as can be observed in Table 4.5. While 10% subset size chosen by SS quickly converges to the optimal solution (or 1% duality gap for that matter) for the root node, randomly picked 10% subset performs very poorly due to the fact that there is no guarantee whether the randomly picked SSR Subproblems are for SSRs who actually appear in the optimal solution or not.

Table 4.6 presents the impact of changing the frequency of solving all subproblems and Problem RMP_integer (which is referred to as a *main* iteration) during the solution of root node to 1% mipgap.

Table 4.6: Changing the Frequency of *Main* Iteration Calls

Freq	1	25	33	50	75	100
Time	23,443	15,887	11,231	8,994	14,459	22,245

Recall from Table 2.3 that if SS is never used, that means only initDualVector and DualManip

functions are activated in $ColGen_{LoA}$. This corresponds to calling the *main* iteration at every iteration (see Table 4.6, entry for the column corresponding to interval "1"). As the interval for calling the *main* iteration increases, the solution time dramatically reduces until interval is every 50 iterations. The reason is the less frequent computation of RMP_integer and fewer columns that are brought into the RMP by solving all subproblems less frequently. The solution time begins to worsen beyond the interval of 50 iterations, and the explanation is that the $\pi_{m,j,q_{best}}$ becomes very stale when the interval is too large; therefore, the lower bound does not improve, as desired.

4.2.7 Impact of Optimality Tolerance on B&P Performance

The absolute value of the reduced costs obtained from the pricing subproblems at the beginning of ColGen algorithms are much larger than the ones obtained later on. In particular, when the duality gap between UB_n^q and LB_n^q is smaller than 2%, the phenomenon called "Tailing-off" recurs, where the improvements in the upper bound UB_n^q become numerically insignificant. It is a very common practice among ColGen practitioners that if the mipgap for the integer solution is not already at the desired level, further ColGen iterations are stopped for this node and branching is activated. This is to ensure that no more valuable time is wasted on bringing columns into the RMP which will yield numerically insignificant improvement, but, rather tighter lower bounds and better integer solutions are searched for in new branches.

Throughout this thesis, as a default value, 1% duality gap is used for branching on a node. The following table shows the solution times for the whole BP_{LoA} algorithm when different duality gap values are tested:

Table 4.7: Handling Tailing-off with Different duality gaps

DualityGap	5%	2.5%	2%	1%	0.5%
Time	38,240	31,042	28,445	25,902	55,676
Nodes	9,321	6,304	3,355	1,132	903

Although, the duality gap at the large end (5%) causes the algorithm to converge more slowly,

this setting enables traversing through more nodes more quickly. This also means that without solving RMP_integer, the RMPs with added branching constraints begin to yield integer feasible solutions naturally. At the lower end (0.5%), ColGen spends too much time finding columns that have significant improvement.

Chapter 5

Summary and Future Research

5.1 Summary

The results obtained in this thesis are significant in the sense that the Operations Research community can once again tap into the considerable power of the Column Generation Algorithm using the techniques introduced in this research. The shortcomings of the default textbook version Column Generation algorithms are avoided altogether for a broad class of problems. The proposed acceleration and stabilization techniques are robust, and do not depend on the input data. Namely, the approach does not entail too much parameter tweaking for performance improvements. Initial Dual Estimation provides a guaranteed valid lower bound; whereas, there was none available at the beginning. Dual Manipulation brings higher quality and diverse set of columns to reduce degeneracy in the dual domain. It also causes columns to be generated with MIP objective in mind, rather than only with intentions of solving the RMP. Subproblem selection step reduces the time spent during periods of non-beneficial subproblems and continuously growing RMP. Guidance for a general algorithm setup is also given for interested readers who are willing to utilize the approach.

Chapter 3 is also the first implementation of Nested Column Generation in Stochastic Workforce Shift Scheduling Problems. The Likelihood of Assignment's applicability and impact is shown here, once again.

Another contribution is the general framework used to attack the monolithic problem and how it is decomposed into 2 stages. Many real life problems have a similar problem structure like the one in this research; therefore, the way we defined the hierarchical decomposition and the relationship between the stages can be helpful while analyzing a host of related problems.

5.2 Future Research

In this thesis, it has been shown how to make Column Generation effective for solving large-scale capacitated resource management problems with block angular structure. In addition to that, being able to demonstrate that the Likelihood of Assignment initialization, estimating the initial dual vector, intermediate dual manipulation, and LoA based subproblem selection are useful for various classes of additional large-scale problems would be another significant achievement. We believe that the techniques, above, should improve the Column Generation performance in general after the necessary adjustments are made, depending on the problem structure. Meet-Pass Planning and Block-Route-Train Assignment Problems are two specific problem types under consideration, which are very challenging in the railway business context. The analysis of these problems would require significant amount of time for robust algorithm design and computational studies. We are hoping to release our findings as we attack these special problem structures in the near future.

In the approach proposed for solving the Workforce Planning and Scheduling Problem, the problem is decomposed into 2 primary stages; however, there is only 1 way information flow allowed. In the future, 2 way communication may increase the approach's flexibility and robustness. Thus, the planning stage can in fact have some information on what is happening in the shift scheduling stage. Designing the interaction between the shift scheduling and planning as a feedback loop is beyond the scope of this thesis. However, we believe an automated system can be devised to trigger to rerun the planning model, based on some observed events in the shift scheduling stage. This could include not being able to meet the demand on a specific shift anymore (headcount and machine training), or failing to provide on time service for specific zipcodes (launchzip).

We have proposed a very efficient heuristic for the shift scheduling stage, but an exact algorithm is not provided for finding the optimal solution. A Branch-and-Price approach on top of what we currently have proposed, could increase the chances of coming closer to the optimal solution; however, more research is required to start the ColGen with an exact equivalent of the original problem which, in turn, must be suitable for decomposition.

Bibliography

[1] Abernathy, W., Baloff, N., Hershey, J., Wandel, S. (1973) A Three-stage Manpower Planning and Scheduling Model: A Service Sector Example. *Operations Research*, Vol. 22, pp. 693-711.

[2] Agnihothri, S., Mishra, A. K. (2004) Cross-training Decisions in Field Services with Three Job Types and Server-Job Mismatch. *Decision Sciences*, Vol. 35, pp. 239-257.

[3] Anbil, R., Forrest, J. J., Pulleybank, W. R. (1998) Column Generation and Airline Crew Pairing Problem. *Documenta Mathematica*, Vol. Extra, pp. 677-686.

[4] Barahona, F., Chudak, F. (2005) Near-optimal Solutions to Large Scale Facility Location Problems. *Discrete Optimization*, Vol. 2, pp. 35-50.

[5] Barnhart, C., Johnson, E., Nemhauser, G., Savelsbergh, M., Vance, P. (1998) Branch-and-Price: Column Generation for Solving Huge Integer Programs. *Operations Research*, Vol. 46, pp. 316-329.

[6] Ben Amor, H., Desrosiers, J., de Carvalho, J. M. V. (2006) Dual-Optimal Inequalities for Stabilized Column Generation. *Operations Research*, Vol. 54, pp. 454-463.

[7] Benders, J. F. (1962) Partitioning Procedures for Solving Mixed-variables Programming Problems. *Numerische Mathematik*, Vol. 4, pp. 238-252.

[8] Brusco, M. J., Jacobs, L. W., Bongiorno, R. J., Lyons, D. V. (1995) Improving Personnel Scheduling at Airline Stations. *Operations Research*, Vol. 43, pp. 741-751.

[9] Cai, X., Li, K. N. (2000) A Genetic Algorithm for Scheduling Staff of Mixed Skills under Multi-criteria. *European Journal of Operational Research*, Vol. 125, pp. 359-369.

[10] Caprara, A., Focacci, F., Lamma, E., Mello, P., Milano, M., Toth, P., Vigo, D. (1998) Integrating Constraint Logic Programming and Operations Research Techniques for the Crew Rostering Problem. *Software Practice and Experience*, Vol. 28, pp. 49-76.

[11] Caroe, C. C., Schults, R. (1999) Dual Decomposition in Stochastic Integer Programming. *Operations Research Letters*, Vol. 24, pp. 37-45.

[12] Crowder, H., Johnson, E. L., and Padberg, M. W. (1983) Solving Large-scale Zero-one Linear Programming Problems. *Operations Research*, Vol. 31, pp. 803-834.

[13] Damodaran, P., Wilhelm, W. (2004) Branch-and-Price Methods for Prescribing Profitable Upgrades of High-technology Products with Stochastic Demands. *Decision Sciences*, Vol. 35, pp. 55-81.

[14] Dantzig, G. B., Wolfe, P. (1960) Decomposition Principle for Linear Programs. *Operations Research*, Vol. 8, pp. 101-111.

[15] Dantzig, G. (1954) A Comment on Edie's Traffic Delay at Toll booths. *Operations Research*, Vol. 2, pp. 339-341.

[16] de Carvalho, J. M. V. (2005) Using Extra Dual Cuts to Accelerate Column Generation. *Informs Journal on Computing*, Vol. 17, pp. 175-182.

[17] Dell Amico, M., Righini, G. Salani, M. (2006) A Branch-and-Price Approach to the Vehicle Routing Problem with Simultaneous Distribution and Collection. *Transportation Science*, Vol. 40, pp. 235-247.

[18] Desaulniers, G., Desrosiers, J., Solomon, M. M. (2005). *Column Generation*. Springer, New York, USA.

[19] Desaulniers, G. (2007) Managing Large Fixed Costs in Vehicle Routing and Crew Scheduling Problems Solved by Column Generation. *Computers & Operations Research*, Vol. 34, pp. 1221-1239.

[20] Desrochers, M., Desrosiers, J., Solomon, M. (1992) A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows. *Operations Research*, Vol. 40, pp. 342-354.

[21] Desrosiers, J., Dumas, Y., Solomon, M., Soumis, F. (1995) Time Constrained Routing and Scheduling, in: *Networks and Distribution*, Handbooks in Operations Research and Management Science, Vol. 8, pp. 35-139.

[22] du Merle, O., Villeneuve, D. , Desrosiers, J., Hansen, P. (1999) Stabilized Column Generation. *Discrete Mathematics*, Vol. 194, pp. 229-237.

[23] Ernst, A. T., Jiang, H., Krishnamoorthy, M., Sier, D. (2004) Staff Scheduling and Rostering: A Review of Applications, Methods and Models. *European Journal of Operational Research*, Vol. 153, pp. 3-27.

[24] Falkner, J., Ryan, D. (1987) A Bus Crew Scheduling System Using a Set Partitioning Model. *Asia-Pacific Journal of Operational Research*, Vol. 4, pp. 39-56.

[25] Galati, M. V., Ralphs, T. K. (2009) DIP - Decomposition for Integer Programming. *https://projects.coin-or.org/Dip*.

[26] Gamache, M., Soumis, F., Marquis, G., Desrosiers, J. (1999) A Column Generation Approach for Large-scale Aircrew Rostering Problems. *Operations Research*, Vol. 47, pp. 247-263.

[27] Gamache, M., Soumis, F. (1998) A Method for Optimally Solving the Rostering Problem, in: G. Yu (Ed.), *OR in Airline Industry*, Kluwer Academic Publishers, Boston, USA. pp. 124-157.

[28] Geoffrion, A., Me Bride, R. (1978) Lagrangean Relaxation Applied to Capacitated Facility Location Problems. *IIE Transactions*, Vol. 10, pp. 40-47.

[29] Graves, G., McBride, R., Gershkoff, I., Anderson, D., Mahidhara, D. (1993) Flight Crew Scheduling. *Management Science*,Vol. 39, pp. 736-745.

[30] Guerinik, N., van Caneghem, M. (1995) Solving Crew Scheduling Problems by Constraint Programming, in: *Lecture Notes in Computer Science*, Proceedings of the 1st International Conference on Principles and Practice of Constraint Programming, pp. 481-498.

[31] Gustafsson, T. (1999) A Heuristic Approach to Column Generation for Airline Crew Scheduling. *Ph.D Thesis*, Chalmers University of Technology.

[32] Hoffman, K., Padberg, M. (1993) Solving Airline Crew Scheduling Problems by Branch-and-Cut. *Management Science*, Vol. 39, pp. 657-682.

[33] IBM ILOG CPLEX Division. (2010) User's Manual v12.2.

[34] Kallehauge, B., Larsen J., Madsen O. B. G., Solomon M. M. (2005) Vehicle Routing Problem with Time Windows. In: Desaulniers G., Desrosiers J., Solomon M. M., editors. *Column generation, GERAD 25th Anniversary Series*. Springer, New York, USA. pp. 67-98.

[35] Klose, A., Gortz, S. (2007) A Branch-and-Price Algorithm for the Capacitated Facility Location Problem. *European Journal of Operational Research*, Vol. 179, pp. 1109-1125.

[36] Laporte, G., Louveaux, F. V. (1993) The Integer L-shaped Method for Stochastic Integer Programs with Complete Recourse. *Operations Research Letters*, Vol. 13, pp. 133-142.

[37] Lavoie, S., Minoux, M., Odier, E. (1988) A New Approach for Crew Pairing Problems by Column Generation with an Application to Air Transportation. *European Journal of Operational Research*, Vol. 35, pp. 45-58.

[38] Lokketangen, A., Woodruff, D. L. (1996) Progressive Hedging and Tabu Search Applied to Mixed Integer (0,1) Multistage Stochastic Programming. *Journal of Heuristics*, Vol. 2, pp. 111-128.

[39] Lorena, L., Senne, E. (2004) A Column Generation Approach to the Capacitated p-median Problems. *Computers & Operations Research*, Vol. 31, pp. 863-876.

[40] Lubbecke, M. E., Desrosiers, J. (2005) Selected Topics in Column Generation. *Operations Research*, Vol. 53, pp. 1007-1023.

[41] Lulli, G., Sen, S. (2004) A Branch-and-Price Algorithm for Multistage Stochastic Integer Programming with Application to Stochastic Batchsizing Problems. *Management Science*, Vol. 50, pp. 786-796.

[42] Marsten, R. E., Hogan, W. W., Blankenship, J. W. (1975) The Boxstep Method for Large-scale Optimization. *Operations Research*, Vol. 23, pp. 389-405.

[43] Mason, A. J., Smith, M. C. (1998). A Nested Column Generator for Solving Rostering Problems with Integer Programming. In Caccetta, L., Teo, K. L., Siew, P. F., Leung, Y. H., Jennings, L. S., Rehbock V., (Ed.), *International Conference on Optimisation: Techniques and Applications*, pp. 827-834.

[44] Morgado, E., Martins, J. (1992) Scheduling and Managing Crew in the Portuguese Railways. *Expert Systems with Applications*, Vol. 5, pp. 301-321.

[45] Morgado, E., Martins, J. (1993) An AI-based Approach to Crew Scheduling, in: *Proceedings of the Ninth Conference on Artificial Intelligence for Applications*, IEEE Computer Society Press, California, pp. 71-77.

[46] Mourgaya, M., Vanderbeck, F. (2007) Column Generation Based Heuristic for Tactical Planning in Multi-period Vehicle Routing. *European Journal of Operational Research*, Vol. 183, pp. 1028-1041.

[47] Neebe, A. W., Rao, M. R. (1983) An Algorithm for the Fixed-Charge Assigning Users to Sources Problem. *Journal of Operational Research Society*, Vol. 34, pp. 1107-1113.

[48] Norkin, V. I., Pflug, G. C., Ruszczynski, A. (1998) A Branch and Bound Method for Stochastic Global Optimization. *Mathematical Programming*, Vol. 83, pp. 425-450.

[49] Osman, I., Laporte, G. (1996) Metaheuristics: A Bibliography. *Annals of Operations Research*, Vol. 63, pp. 513-623.

[50] Oukila, A., Ben Amor, H., Desrosiers, J., El Gueddaria, H. (2007) Stabilized Column Generation for Highly Degenerate Multiple-depot Vehicle Scheduling Problems. *Computers & Operations Research*, Vol. 34, pp. 817-834.

[51] Rayward-Smith, V., Osman, I., Reeves, C., Smith, G. (1996) *Modern Heuristic Search Methods*, John Wiley and Sons, Chichester, UK.

[52] Rousseau, L. M., Gendreau, M., Feillet, D. (2003) Interior Point Stabilization for Column Generation. *Technical Report*, LIA - Universite d'Avignon.

[53] Sabar, M., Montreuil, B., Frayret, J. M. (2009) A Multi-agent-based Approach for Personnel Scheduling in Assembly Centers. *Engineering Applications of Artificial Intelligence*, Vol. 22, pp. 1080-1088.

[54] Schrijver, A. (1986) *Theory of Linear and Integer Programming*. John Wiley and Sons, Chichester, UK.

[55] Seitman, D. (1994) In-house Medical Personnel Scheduler – A Computerized On-call Scheduling Program. *International Journal of Clinical Monitoring and Computing*, Vol. 11, pp. 7-10.

[56] Senne, E. L. F., Lorena, L. A. N., Pereira, M. A. (2005) A branch-and-Price Approach to p-median Location Problems. *Computers & Operations Research*, Vol. 32, pp. 1655-1664.

[57] Shiina, T., Birge, J. R. (2004) Stochastic Unit Commitment Problem. *International Transactions in Operational Research*, Vol. 11, pp. 19-32.

[58] Silva, E. F., Wood, R. K. (2006) Solving a Class of Stochastic Mixed Integer Programs with Branch and Price. *Mathematical Programming*, Vol. 108, pp. 395-418.

[59] Singh, K. J., Philpott, A. B., Wood, R. K. (2009) Dantzig-Wolfe Decomposition for Solving Multistage Stochastic Capacity-Planning Problems. *Operations Research*, Vol. 57, pp. 1271-1286.

[60] Van Roy, T. (1986) A Cross Decomposition Algorithm for Capacitated Facility Location. *Operations Research*, Vol. 34, pp. 145-163.

[61] Vanderbeck, F. (1994) Decomposition and Column Generation for Integer Programs. *Ph.D. Thesis*, Universit Catholique de Louvain, Louvainla-Neuve, Belgium.

[62] Wentges, P. (1997) Weighted Dantzig-Wolfe Decomposition for Linear Mixed-Integer Programming. *International Transactions in Operational Research*, Vol. 4, pp. 151-162.

Author Vita

Alper Uygur has a B.S. degree in Industrial Engineering from Middle East Technical University, Ankara, Turkey. He received his Master of Science degree in Management Science from the I&SE Department at Lehigh University.

He recently began working at BNSF Railway as a Sr. Operations Research Specialist. He is currently designing decision making tools for large-scale optimization problems regarding Train Scheduling and Timetabling Problems to support the Service Design Department. From 2005 to 2011, he was working as a Sr. Development Engineer at IBM, Maintenance and Technical Support Subdivision, Service Delivery Planning Department. Throughout his career at IBM, he supported the Business Process Reengineering Team for cost effective management of Maintenance and Technical Support Subdivision, by providing exact and heuristic approaches for strategic, tactical and operational level decisions.

His research interests emphasize on decomposition methods for large-scale mixed integer problems and developing efficient solution strategies for such.

You can email him at Alper.Uygur@BNSF.com if you have any questions or comments regarding his thesis or research interests.