3-25-2010

# Branch and Price Solution Approach for Order Acceptance and Capacity Planning in Make-to-Order Operations

Siddharth D. Mestry
*Florida International University*, smest001@fiu.edu

Martha A. Centeno
*Florida International University*, centeno@fiu.edu

Jose A. Faria
*Florida International University*, fariaj@fiu.edu

Purushothaman Damodaran
*Northern Illinois University*, pdamodaran@niu.edu

Chen Chin-Sheng
*Florida International University*, chenc@fiu.edu

**DOI:** 10.25148/etd.FI10041619

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

# BRANCH AND PRICE SOLUTION APPROACH FOR ORDER ACCEPTANCE AND CAPACITY PLANNING IN MAKE-TO-ORDER OPERATIONS

A dissertation submitted in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

in

INDUSTRIAL AND SYSTEMS ENGINEERING

by

Siddharth D. Mestry

2010

To: Dean Amir Mirmiran
College of Engineering and Computing

This dissertation, written by Siddharth D. Mestry, and entitled Branch and Price Solution Approach for Order Acceptance and Capacity Planning in Make-to-Order Operations, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

_____
Martha A. Centeno

_____
José A. Faria

_____
Purushothaman Damodaran, Co-Major Professor

_____
Chin-Sheng Chen, Co-Major Professor

Date of Defense: March 25, 2010

The dissertation of Siddharth D. Mestry is approved.

_____
Dean Amir Mirmiran
College of Engineering and Computing

_____
Interim Dean Kevin O'Shea
University Graduate School

Florida International University, 2010

## DEDICATION

I dedicate this dissertation to my wife Vrunda and to my parents for their tremendous patience and understanding. Without their support, the completion of this work would not have been possible.

And, to my son Arav for giving me the strength to push myself to the limits of my capabilities and potential, to help me see the light at the end of the tunnel, to believe in the future with all our dreams fulfilled.

ACKNOWLEDGMENTS

I am greatly indebted to Dr. Purush Damodaran for being an excellent academic advisor, and a tremendous mentor. He has not only helped me in my scholastic pursuits but has also given me valuable guidance in personal matters. I owe him this dissertation and much more.

I would like to thank Dr. Chin-Sheng Chen for the confidence he showed in my capabilities, for his practical insights and for sharing his limitless knowledge with me. I will forever remember the discussions which we had in his office.

A special thanks to Dr. Martha Centeno for being an excellent teacher and for inculcating in me the qualities to be one. I appreciate Dr. Jose Faria's valuable inputs in improving this Dissertation.

ABSTRACT OF THE DISSERTATION

BRANCH AND PRICE SOLUTION APPROACH FOR ORDER ACCEPTANCE AND CAPACITY PLANNING IN MAKE-TO-ORDER OPERATIONS

by

Siddharth D. Mestry

Florida International University, 2009

Miami, Florida

Professor Chin-Sheng Chen, Co-Major Professor

Professor Purushothaman Damodaran, Co-Major Professor

The increasing emphasis on mass customization, shortened product lifecycles, synchronized supply chains, when coupled with advances in information system, is driving most firms towards make-to-order (MTO) operations. Increasing global competition, lower profit margins, and higher customer expectations force the MTO firms to plan its capacity by managing the effective demand. The goal of this research was to maximize the operational profits of a make-to-order operation by selectively accepting incoming customer orders and simultaneously allocating capacity for them at the sales stage.

For integrating the two decisions, a Mixed-Integer Linear Program (MILP) was formulated which can aid an operations manager in an MTO environment to select a set of potential customer orders such that all the selected orders are fulfilled by their deadline. The proposed model combines order acceptance/rejection decision with detailed scheduling. Experiments with the formulation indicate that for larger problem sizes, the computational time required to determine an optimal solution is prohibitive. This

formulation inherits a block diagonal structure, and can be decomposed into one or more sub-problems (i.e. one sub-problem for each customer order) and a master problem by applying Dantzig-Wolfe's decomposition principles. To efficiently solve the original MILP, an exact Branch-and-Price algorithm was successfully developed. Various approximation algorithms were developed to further improve the runtime. Experiments conducted unequivocally show the efficiency of these algorithms compared to a commercial optimization solver.

The existing literature addresses the static order acceptance problem for a single machine environment having regular capacity with an objective to maximize profits and a penalty for tardiness. This dissertation has solved the order acceptance and capacity planning problem for a job shop environment with multiple resources. Both regular and overtime resources is considered.

The Branch-and-Price algorithms developed in this dissertation are faster and can be incorporated in a decision support system which can be used on a daily basis to help make intelligent decisions in a MTO operation.

TABLE OF CONTENTS

LIST OF TABLES

TABLE PAGE

LIST OF FIGURES

FIGURE PAGE

LIST OF SYMBOLS

| | | |
|---|---|---|
| $\alpha_j$ | : | Dual value of $j^{th}$ convexity constraint representing job $j$ |
| $b_{rts}$ | : | Number of hours resource $r$ is available in source $s$ of time period $t$ |
| $c_{rs}$ | : | Processing cost per hour of resource $r$ in source $s$ |
| $d_j$ | : | Due-date for job $j$ |
| $\varepsilon_p^o$ | : | Precedence error length between operation $o$ and $o$+1 |
| $\gamma$ | : | Optimality tolerance used to prune branch and bound tree |
| $J$ | : | Set of jobs |
| $l_{ts}$ | : | Length of source $s$ in time period $t$ |
| $\lambda_j^k$ | : | Binary variable in the restricted master problem |
| $O_j$ | : | Set of operations in job $j$ |
| $p_{jor}$ | : | Processing time needed for completing operation $o$ of job $j$ on resource $r$ |
| $q_j$ | : | Selling price of job $j$ |
| $R$ | : | Set of resources |
| $S$ | : | Set of sources |
| $T$ | : | Set of time periods |
| $\tau$ | : | smallest time unit for which a operation has to be processed on a resource, if processed at all |
| $\theta_j$ | : | Convex combination of columns for job $j$ |
| $U_j$ | : | Binary decision variable indicating acceptance or rejection of job $j$ |
| $w_{rts}$ | : | Capacity constraint dual value |
| $X_{jorts}$ | : | Continuous decision variable for the number of hours operation $o$ of job $j$ is |

to be processed on resource *r* in source *s* of time period *t*

$Y_{jorts}$ : Binary decision variable indicating if operation *o* of job *j* is processed on resource *r* in source *s* of time period *t*

$Z_{RMP}^{BUB}$ : Best upper bound of all the unexplored nodes in the branch and bound tree for the restricted master problem

$Z_{SP}^{j}$ : Objective function value of the $j^{th}$ sub-problem

$Z_{IP}$ : Objective function value of the best known integer solution

$Z_{RMP}^{LP}$ : Objective function value of the linear relaxation of the restricted master problem

# LIST OF ACRONYMS

| | | |
|---|---|---|
| BeFS | : | Best First Search |
| BPS1 | : | Branch and Price Strategy 1 |
| BPS2 | : | Branch and Price Strategy 2 |
| DC Ratio | : | Demand-to-Capacity Ratio |
| DFS | : | Depth First Search |
| dtu | : | Discretized time unit |
| MTO | : | Make-to-Order |
| MTS | : | Make-to-Stock |
| RMP | : | Restricted Master Problem |
| SP | : | Sub-Problem |

# 1. INTRODUCTION

## 1.1. Background

Manufacturing systems are usually classified along two dimensions: 1) the internal organization of the production system and 2) logistic product/market relations [1]. The former is concerned with the internal structure of the manufacturing and assembly system. The three basic structures are dedicated flow lines, job shops, and on-site manufacturing. In dedicated flow lines, a number of operations have to be carried out on every customer order following the same process route. Job shops on the other hand can manufacture a broad spectrum of products, each product having its own process route. On-site (or fixed location) manufacturing is one in which all the equipments have to be moved to the product manufacturing location. Examples include bridge construction, ship building, etc.

The second classification dimension (i.e. logistic product/market relations) is operational policies for fulfilling customer orders. The three basic operational modes are: 1) make-to-stock, 2) make-to-order, and 3) hybrid [2]. Make-to-stock (MTS) is a philosophy to fill customer orders by stocking finished goods for immediate delivery. MTS is characterized by high-volume production and is normally followed when the firms are product-focused with relatively less component level customization. The second operational mode is make-to-order (MTO). A MTO firm starts working on an order only after it has been placed by the customer. This policy is advantageous when the end product is customer specific, with high level of customization. This policy allows a high degree of flexibility

and the products manufactured are one-of-a-kind or in small batches. Typical examples of MTO system include print shop, semiconductor manufacturing, engineering tooling, special equipments, and large hydraulic pumps, DNA sequencing, laundry service, etc. Most of the MTO firms are process-focused, as the products manufactured share the same kind of operations but differ in the design details. Firms can also have a hybrid of MTS and MTO system for different products that are manufactured at the same production facility. Flow shops and job shops can be operated under MTS or MTO policies or the combination of the above two.

In the current business scenario, customers demand products with a high level of customization. The focus is on innovation and customer satisfaction, leading to shortened product development life cycle. These trends compel the manufacturers to remain agile and flexible, leading to an increase in the appeal and popularity of make-to-order or a hybrid operational philosophy [3]. MTO systems are not only used for unique product manufacturing but are also very efficient in producing greater product variety at lower cost [4].

The major difference between MTO and MTS is the way in which customer demand is handled. By definition, MTS holds finished goods inventory to meet the customer demand. The focus is on anticipating the demand and planning to meet the demand. The main issues that need to be addressed are inventory planning, lot size determination and demand forecasting. Since MTO is characterized by back orders with zero inventories as each customer order is unique and cannot be manufactured in advance, the only way for

managing the effective demand is by holding its capacity in inventory. The production planning focus is on order execution and the competitive priority is shorter delivery lead-time, and adherence to due-dates. Hence the most important operating issue in MTO is the effective and efficient use of available capacity to meet customer demands.

Capacity planning determines the resource requirements of an organization to sustain a given demand over a planning horizon. There are three tiers of capacity planning based on their planning horizon. Long term capacity planning relates primarily to strategic issues involving the firm's major production facilities with a planning horizon anywhere between three to five years. It focuses on determining facility locations, plant capacities, division of new and existing product lines, technology and transferability of process to other products, subject to demand forecast and availability of investment funds. It is used to determine major supplier's plans and their vertical integration; principal operation modes and production methods. The fundamentals of long-term capacity planning are mostly the same for both MTO and MTS operations.

Medium term capacity planning or aggregate planning focuses on setting monthly or quarterly resource requirements for each plant for typically a one-year planning horizon. The process includes developing, analyzing and maintaining a preliminary, approximate schedule of the overall operations of an organization. It decides on workforce level, raw materials and inventory policy by product group and department. Based on sales forecast, it generates production capacity plans for (1) labor-employment level (i.e. lay-offs, hiring, recalls, overtime, and part-timer), (2) inventory policy, (3) utility requirements,

(4) facility modifications, (5) outsourcing, and (6) major material supply contracts. Capacity requirements may vary from period to period in their regular time labor, overtime labor, inventory and subcontracting.

Two conventional aggregate planning approaches for MTS are: (1) matching demand and (2) level capacity. With the matching demand approach, production capacity in each time period varies to exactly match the aggregate demand as forecasted for that time period, by hiring and laying-off workers. With the level capacity approach, production capacity is held constant over the planning horizon; the difference between the constant production rate and the varying demand rate is made up by inventory, backlog, overtime labor, part time labor, temporary labor, and/or sub-contracting. An MTO operation usually adopts a hybrid approach of both. On one hand, it needs to maintain a certain level of production capacity for its core competency. On the other, it cannot leverage on inventory, as every order is a backorder and it requires customization. The common practice is to maintain a minimum level of production capacity, and liberally rely on overtime and subcontracting to adjust capacity and to accommodate demand fluctuation. Aggregate plans serve as foundation for future short term capacity plans.

Short term capacity planning sets a daily or weekly capacity plan for a planning horizon long enough to accommodate each order's lead time. The objective of short-term capacity planning is to ensure an appropriate match between the resources availability and the capacity requirement for a production plan at the work center level [5]. For a MTO operation, it has to specify resources requirement of each labor and machine type for each

customer order at its component level. Each customer order first is translated into internal orders and detailed work orders, which are then summarized into a load schedule (time-phased capacity requirements) by labor and/or equipment, in coordination with materials arrival. A typical MTO operation routinely considers the use of alternative sources such as overtime and outsourcing, in order to meet work order's due date, which is a critical issue for MTO operations.

In most MTO operations, meeting due-dates is considered a hard constraint. With multiple orders and bids competing for common resources, meeting these deadlines is the first criterion used to reject or accept a backorder as due-date feasibility depends on the availability of the time-phased resource capacity. Reliable due-dates require a continuous coordination between marketing/sales and manufacturing departments during the bidding phase. Zijm [1] mentions that it is important to integrate these decisions.

In practice, decisions on order acceptance and production planning are often functionally separated. The objective of the sales department is to bring as much revenue as possible. The sales department will tend to accept all orders, regardless of the available capacity, because their goal is turnover. Production is concerned with limited capacity and it tries to maximize utilization and minimize the number of tardy deliveries. Given these conflicting goals of turnover and tardiness, order acceptance decisions are often made without involving production department or with incomplete information on the available capacity in production department [6]. Accepting too many orders, which is the objective of the sales department, leads to an over-loaded production system, in which lead times

increase and orders are increasingly delivered late. To deal with this short term capacity problems, management may try to use additional non-regular capacity like overtime and outsourcing, thereby increasing the costs significantly. This may lead to lower profit margins or even negative profits. Tardy deliveries may also lead to higher penalty costs, and possibly lead to loss of customer goodwill [6, 7].

While negotiating contracts in MTO environment, the company can either adjust the price or lead time for an order. If the order has tighter deadlines, the MTO enterprise can charge a premium for processing that order as it might have to be expedited with use of non-regular capacity. Recent experiences of firms, such as Amazon.com, indicate that customers may be unwilling to accept dynamic pricing as fair [8]. An alternative to dynamic pricing would be to view the issue as one of allocating capacity between competing orders, making it a capacity allocation problem. When multiple orders, each providing a different profit contribution is present, the capacity allocation problem becomes an order acceptance or refusal problem [9, 10, 11].

### 1.2. Research Problem

A job shop environment is considered to model the make-to-order operation. A MTO firm receives a set of bids or customer orders. A customer order is referred to as jobs in the context of this research. The decision to be made is which customer order is to be accepted and how to schedule it to maximize profit. The decisions should be made simultaneously; otherwise, an order may be accepted but the residual capacity available may not permit timely delivery.

Each customer order has a set of operations to be processed with linear precedence constraint and deterministic processing times, a fixed due-date and a known sales price. No tardy deliveries are allowed. There are multiple types of resources each having one or more machines. Furthermore, job recirculation is allowed, which means that the jobs can visit the same machine more than once. The cost of using a resource in each source is known and is represented in unit cost per hour. The planning horizon is discretized into time buckets of equal length know as a time period. Without loss of generality we assume that each time period is one day. Furthermore each day is divided into sources viz. regular time and overtime. Overtime is usually considered more expensive. The decision of accepting or rejecting the orders is done on day zero.

### 1.3. Research Objective

The goal of this dissertation was to maximize the operational profit of a make-to-order operation by combining the order acceptance decision with capacity planning at the sales stage. The objectives were to mathematically model the problem and solve it within the bounds of practicality.

A mixed-integer linear program was formulated to model the order acceptance and capacity planning problem. Large-scale mixed integer linear programs are difficult to solve because of their combinatorial nature and solving industry-sized problems would take prohibitively long runtimes. Over the last decade, column generation has proven to be one of the most successful approaches for solving large-scale integer programs [12]. Column generation, also known as the Dantzig-Wolfe (DW) decomposition, is

implemented along with branch-and-bound scheme which is collectively known as branch-and-price. The original formulation has to be decomposed into a restricted master problem (RMP) and one or more pricing or sub-problems (SPs). The major components of a branch-and-price scheme are generating columns by solving the sub-problems and branching to restore the integrality constraints for integer variables, as the RMP is solved as a linear relaxation. To solve the mixed-integer linear program in reasonable time we have developed a branch and price algorithm.

Figure 1-1 shows a schematic representation of a typical order acceptance problem in a job shop environment of a MTO operation. For illustration purposes the job shop used has three resources. Resource 1 has two machines of the same type, while resources 2 and 3 each have a single machine. Each resource has fixed operating costs in regular time and overtime. There are three orders, each having a known sales price and a fixed due-date. Each order has a different process route with deterministic processing times. For example, the process route for customer order 1 is Resource 1 → Resource 2 → Resource 3. The model takes these as input parameters and gives a solution which comprises of the orders that have been accepted so as to maximize the operational profits and their corresponding schedules.

For the example shown in Figure 1-1, customer orders 1 and 3 are accepted. Figure 1-2 shows the schedule generated by the model for these orders.

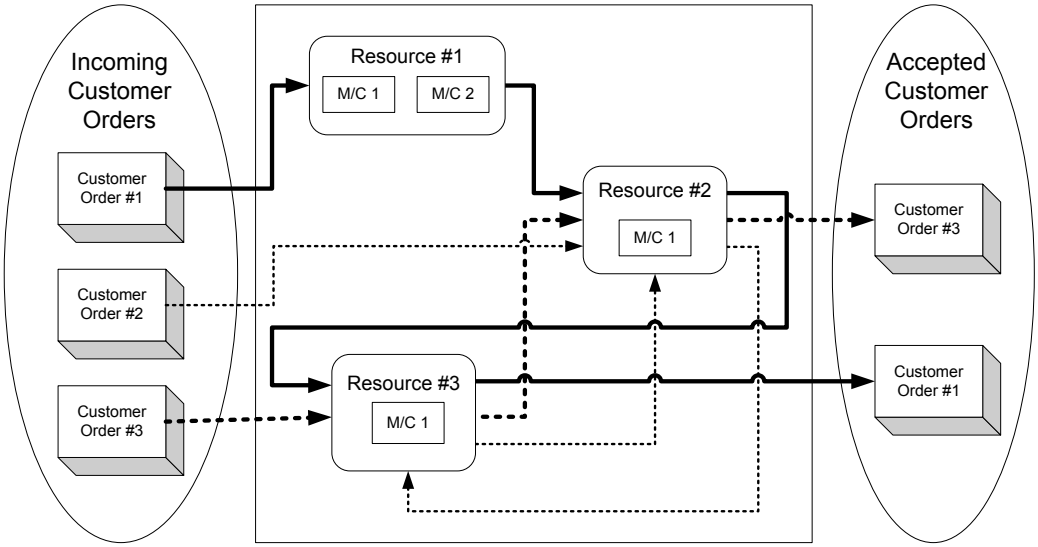



**Figure 1-1. Customer order process route for jobshop-MTO operation**

| Cust. Order (Job) | Opt. # | Res. # | Process. Time | Day 1 | | Day 2 | | Day 3 | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Regular time | Overtime | Regular time | Overtime | Regular time | Overtime |
| 1 | 1 | 1 | 8 | | | | | | |
| | 2 | 2 | 8 | | | | | | |
| | 3 | 3 | 8 | | | | | | |
| 2 | 1 | 2 | 8 | Not Selected | | | | | |
| | 2 | 3 | 5 | | | | | | |
| | 3 | 2 | 10 | | | | | | |
| 3 | 1 | 3 | 10 | | | | | | |
| | 2 | 2 | 10 | | | | | | |
| | | | | Resource 1 | | Resource 2 | | Resource 3 | |

**Figure 1-2. Optimal production plan generated by the mathematical model**

For simplicity, a common due-date of three days was assumed for this example. Each day is divided into regular time and overtime, each eight hours long. All the operations for

order 1 are processed in the regular time of days 1, 2 and 3. Operation 2 of customer order 3 uses two additional hours of overtime on day 3 in order to meet its due-date.

### 1.4. Significance of the Research Problem

The ever increasing emphasis on mass customization, shortened product lifecycles, synchronized supply chains, when coupled with advances in information system, is driving most firms towards make-to-order operations. Increasing global competition, lower profit margins, and higher customer expectations force the MTO firms to effectively manage the capacity to make sustainable profits. Because the main driver in MTO operations is customer orders, coordination of operations and marketing functions for effectively managing capacity by managing the demand placed on the system has been long recognized as vital [10]. This research integrates these two important decisions, by coordinating the order acceptance at the sales stage with a detailed capacity plan at the production level.

The policy of acceptance or rejection of an order based on capacity available is referred to as available-to-promise (ATP), which is common in an Enterprise Resource Planning (ERP) system to search and check resource availability. With most ERP systems, it is a simple search in a database, accompanied by a simple heuristic rule such as first-come-first-serve (FCFS). When an order is accepted, it is usually inserted into the existing master production schedule (MPS). The acceptance/rejection decision in this case is to check whether the available capacity is sufficient to meet the order due-date. ERP systems lack intelligent planning [1] tools in order to maximize the profits based on

selectively accepting incoming orders. Concepts like manufacturing resources planning (MRP), and enterprise resource planning (ERP) were embraced almost immediately after their introduction because of their simplicity and the available computing power. But there are many implementation failures because there are a great number of conditions to be fulfilled before the apparently simple logic can work successfully. Many managers recognized these major drawbacks and are demanding more intelligent solutions. Furthermore, with the recent increase in the computing power Operations Research models have gained a lot of importance.

### 1.5. Dissertation Structure

The rest of the dissertation is organized as follows. In Chapter 2, past and current closely related research is explored. In Chapter 3 the mathematical model is presented along with the assumptions and limitations of this research. Chapter 4 explains the Dantzig-Wolfe decomposition, and an exact branch and price technique. The solution quality from the exact branch and price technique is compared to the results from an optimization solver. In Chapter 5, we present approximation algorithms and comparison between the various solution methodologies. We conclude this dissertation with Chapter 6 by summarizing the contributions of this research and possible extensions.

# 2. LITERATURE REVIEW

Order acceptance in manufacturing is closely related to the principles of revenue management (RM) which is a commonly used in service industry for order acceptance and refusal process, with differential pricing, capacity reallocation and overbooking [9]. There has been a recent interest in applying RM to manufacturing industry in both MTS and MTO operation modes. In MTO, the decisions of order acceptance, lead-time or due-date quotation, pricing and capacity planning are closely related. In the absence of differential pricing, RM becomes a capacity allocation and order acceptance problem. For the purpose of this research we focus on applications of RM to MTO and literature closely related to order acceptance in MTO. Order acceptance in make-to-order can be broadly classified based on static [6, 13, 14, 15, 16, 17] and dynamic arrivals [4, 7, 10, 11, 11, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27] of customer orders. Section 2.1 discusses research done in the area of order acceptance with dynamic customer arrivals while Section 2.2 focuses on the static arrivals. Section 2.3 is dedicated to the applications of column generation technique, especially in the area of scheduling.

## 2.1. Order acceptance with dynamic arrivals

The earliest research in order acceptance was done by Miller [28]. He considered an *n*-server queuing system with *m* customer classes distinguished by the reward associated with serving customers of that class. The objective was to accept or reject customers so as to maximize the expected value of the rewards received over an infinite planning horizon. Poisson arrivals with common exponential service time is considered and formulated as

an infinite horizon continuous time Markov decision problem. Lippman and Ross [29] considered the problem of maximizing the long-run average return in a single server traffic reward system with a customer's offer is a joint distribution of reward and of service time required to earn this reward which is independent of the renewal process which governs customer orders. They formulate it as a semi-Markov decision process with the characterization of accepting the customer if and only if the ratio of his expected reward to his expected service time is larger than $g$, the long-run average return.

Recently, the concepts of revenue management have been applied to manufacturing for tackling the order acceptance problem. Carr and Duenyas [30] address the problem of admission control and sequencing in a production system that produces two classes of products in a hybrid MTO/MTS operation. The first class of products is made-to-stock and the second is make-to-order. The firm has the option to accept or reject a particular MTO customer order. They model the joint admission control / sequencing decision in the context of a single M/M/1 queue with two classes of products.

Webster [31] studies a single stage make-to-order production system to examine policies for adjusting price and capacity in response to periodic and unpredictable shifts in the importance placed by the market on the price and lead-time. He suggests that maintaining a fixed capacity while using lead-time and/or price to absorb changes in the market will be most attractive when stability in throughput and profits are highly valued, but in volatile markets, this stability comes at a cost of low profits.

Balakrishnan et al. [18, 19] and Sridharan and Balakrishnan [32] described how to ration the available capacity when the forecasted demand is higher than the available capacity over a planning horizon, using a decision-theory based approach. Barut and Sridharan [10] extended this research for investigating the effectiveness of a tactical demand-capacity management policy to maximize profit by selectively accepting or rejecting customer orders for multiple product classes. Tardiness is not allowed but earliness is permissible without penalty. They propose a dynamic capacity apportionment procedure (DCAP) to prescribe a static acceptance policy. Ebadian et al. [24] also proposed a decision-making structure for the order entry stage in MTO environments. The aim of the proposed structure is to manage the arriving orders so that the MTO system just proceeds to produce those arriving orders which are feasible and profitable for the system. The appropriate decisions on the arriving orders are taken based on two criteria including price and delivery time. The arriving orders have either fixed or negotiable delivery times. During the first two steps, the new arriving orders either are rejected or appropriate decisions to meet their delivery time are made. At the next step, the optimal prices along with delivery times (if negotiable) of non-rejected orders are determined by a mixed-integer programming model. In the case of the final approval by the customers at the fourth step, another mixed-integer programming model is launched to select a set of suppliers and subcontractors that are able to provide required raw material and workload for the newly accepted orders. They consider a job shop with sub-contracting but without overtime capacity.

Defregger and Heinrich [33] considered the application of revenue management in make-to-order manufacturing company with limited inventory capacity. Orders with different profit margins arrive stochastically over an infinite time horizon and the company has to decide which orders to accept and which to reject. They model the problem as a discrete Markov decision process and propose a heuristic procedure. In numerical tests, they showed the potential benefits of using revenue management compared to a FCFS policy.

Herbots et al. [25] examine the simultaneous dynamic order-acceptance and capacity planning decision. They consider only one resource type, which represents the bottleneck of the company. They consider regular capacity and non-regular capacity.

Charnsirisakskul et al. [20] studied integrated order selection and scheduling decision, where the manufacturer has the flexibility to choose lead-times. They provide a mechanism for coordinating order selection, lead-time and scheduling decisions and to determine under what conditions lead-time flexibility is most useful for increasing the profits. They consider a single machine production system with a bottleneck and no buffers between stations capable of producing multiple products with negligible setup times and preemptions. If the manufacturer cannot complete the order by the latest acceptable due-date tardiness costs are incurred. As an extension to this research, Charnsirisakskul et al. [21] study the simultaneous pricing, order acceptance, scheduling and lead-time decisions, both in the case where the manufacturer has and does not have the flexibility to charge different prices for different customers. They present decision models that can be solved by commercial optimization software and present simple

rounding heuristics that provide initial solution with the objective functions values within 87% of the optimal solution.

Ebben et al. [7] investigate the importance of order acceptance and the benefits of cooperation between the sales and planning function. They develop several workload based order acceptance methods and develop a simulation model of generic MTO job shop, which enables them to simulate the order arrival and production process to test the proposed methods. They use utilization rate and the service level as the performance measures. Nandi and Rogers [34] using simulation demonstrate how to optimally control a manufacturing system under different environments and how main performance measures of a system are affected by using a parameterized order acceptance rule. The order acceptance rule is similar to the path load order review introduced in Philipoom and Fry [35].

Akkan [27] develops heuristics to insert a new work order in the existing schedule in order to minimize the holding costs and the total contribution lost due to rejected work orders in the case where one has to reject the incoming work order because it cannot be fit in the current schedule.

Modarres et al. [36] formulate stochastic capacity allocation problem in a manufacturing system with two classes of “frequent” and “occasional” customers demanding its capacity. The stochastic nature of capacity is caused by machine failures, stops or breakdowns during the operation; and the maintenance duration is random. The price rate

as well as the penalty for order cancellation caused by overbooking is different for each class.

### 2.2. Order acceptance with static arrivals

Within the operational domain of job shop planning with static customer arrivals, job selection has been a topic of growing interest. The problem of selection and ordering of elements from a given set so as to optimize a given objective function was considered by Bahram et al. [17]. They present a generalization of the best-in rule that in many cases can solve the problem while the best-in rule does not. A characterization of such a greedy algorithm has been presented.

Slotnick and Morton [13] examine a set of trade-offs that can arise if a manufacturing facility has more potential work than it can handle easily. They formulate a one-machine model with static arrivals, fixed processing times, due-dates and profits. The objective function maximizes total net profit, which is the sum of the revenues of all jobs minus weighted lateness penalties, by selecting a subset of jobs. Ghosh [16] proves that the Slotnick-Morton [13] version of the job selection problem is NP-Hard. He proposes two dynamic programs that produce the exact solution to the problem.

In an extension to [13] Lewis and Slotnick [14] examine the profitability of job selection decisions over a number of periods when current orders exceed capacity with the objective of maximizing profit and when rejecting a job will result in no future jobs from that customer. The firm processes jobs, over a set number of time periods (stages) within

a given time horizon. The firm has *m* customers at the beginning of the first period; each customer submits one job at each stage, until one of the jobs is rejected. Each job has pre-determined revenue, and the firms pay back a discount to customers whose jobs are completed past a pre-determined due-date; customers are willing to pay a premium for early delivery. Each job has a known processing time and importance. The importance of the job is the weight assigned to it for calculating the lateness penalty. This weight allows the firm to indicate that certain jobs may have importance beyond their immediate profit. The firm has the option of rejecting any job. If a job is rejected, the customer is lost (i.e. never sends another job to be processed within the planning horizon).

In [6], Slotnick and Morton model a manufacturing facility that considers a pool of orders, and chooses for processing the subset that results in the highest profit. In addition to the problem characteristics in [13] they consider customer weight. The objective is to maximize profit, which is the sum of per-job revenues minus total weighted tardiness. They propose two approaches: separation of sequencing and job acceptance decisions, utilizing a property of the problem that is exploited to good advantage in the analogous problem with weighted lateness; and a joint consideration of sequencing and acceptance, using relaxation. They state that the joint approach is far superior to the first. Rom and Slotnick [15] also propose a GA to solve the order acceptance problem with tardiness penalties.

### 2.3. Applications of column generation in scheduling

Decomposition techniques like column generation have been widely used to solve large-scale optimization problems [12, 37]. Column generation has been successfully used in job scheduling for common due date [38], parallel machines [39], and single machines [40, 41]. For a detailed taxonomy of the column generation literature we refer to [12]. Hans [42] developed a branch-and-price loading method that is an exact approach for solving the pre-emptive resource loading problem. The objective is to generate an order schedule for each order, such that the total costs of the required non-regular capacity and the order tardiness penalties are minimized.

### 2.4. Summary

This research considers an order acceptance problem in multi-resource job shop environment with regular and non-regular capacity and static customer arrivals. The only research which tackles a multi-resource job shop problem is by Ebben et al. [7]; but they do not consider non-regular capacity (overtime) and the customer arrivals are dynamic. We solve the problem under study using a branch-and-price algorithm. To the best of our knowledge this approach has never been used for order acceptance; although Hans [42] has developed a branch-and-price resource loading (BPRL) approach for scheduling orders which have already been selected. Ebben et al. [7] use the BPRL technique in their simulation for scheduling the already accepted orders.

Table 2-1 summarizes the literature related to the proposed problem under study. The table compares and contrasts the literature reported on problems similar to the problem

under study. It is evident from this table that the proposed problem and the solution approach are different from what is reported in the literature so far.

**Table 2-1. Summary of relevant research**

| Research | Objective | Order Acceptance | Multiple Resources | Non-regular Capacity | Fixed Due-dates | Solution Approach |
|---|---|---|---|---|---|---|
| [42] | Minimize non-regular capacity costs and tardiness penalties | No | Yes | Yes | No | Branch and Price |
| [13] | Maximize Profit | Yes | No | No | No | Heuristic |
| [6] | Maximize Profit | Yes | No | No | No | Branch and Bound |
| [14] | Maximize Profit | Yes | No | No | No | DP, Heuristic |
| Proposed Research | Maximize Profit | Yes | Yes | Yes | Yes | Branch and Price |

# 3. MATHEMATICAL FORMULATION

This chapter describes the problem environment, proposes a mathematical formulation for the short-term capacity planning problem for the MTO operation environment. The outline of the chapter is as follows. We formally present the problem in Section 3.1. Section 3.3 describes the model developed for solving the MTO order acceptance and scheduling problem. Preliminary results are presented in Section 3.4 that illustrates the usefulness of the model proposed and Section 3.5 provides the motivation for solving this problem by the proposed branch and price technique.

## 3.1. Problem characteristics

The MTO production system is modeled as a job shop with multiple resources $\{r \in R\}$. Each resource type *r* can have multiple machines. A finite planning horizon is considered which is discretized into equal interval time periods $\{t \in T\}$. Without loss of generality we assume that each time period is one day. Furthermore each day is divided into sources $\{s \in S\}$ viz. regular time and overtime. The length of each source *s* in time period *t* is given by $l_{ts}$ and is eight hours, but can be varied according to the need. The available capacity of resource *r* in source *s* of time period *t* is denoted by $b_{rts}$. The MTO firm receives a set of customer orders or jobs $\{j \in J\}$. Each job *j* has a set of operations $\{o \in O_j\}$ with a processing time $p_{jor}$ on resource *r*, a fixed due date $d_j$ and a sales price $q_j$. Each job can follow different processing route and the operations have a linear precedence relationship. The cost of using a resource *r* in each source *s* is represented in unit cost per hour $c_{rs}$. Overtime is usually considered more expensive. The objective is to

maximize the profit of the MTO operation by selectively accepting the customer orders and planning for their capacity within the planning horizon, such that the accepted orders are completed before their due dates.

### 3.2. Assumptions

For the problem under consideration we assume a single deliverable job without any sub-assemblies. Raw material costs are not considered. The cost of operating machines is same during regular and overtime hours. The operational profit is a function of the sales price of a job and labor costs incurred to process that job in regular time and overtime. The sales price is dictated by the market and hence is considered to be fixed. Preemptions are allowed in scheduling the jobs. Processing times are additive and the amount of processing a preempted job already has received is not lost. Machine failure is not considered. The problem addresses in this research is deterministic.

### 3.3. Mathematical model

The problem is modeled as a mixed-integer linear program. The decision variables used in the model are:

$X_{jorts}$= hours operation $o$ of job $j$ is processed on resource $r$ in source $s$ of period $t$

$$Y_{jorts}=\begin{cases}1, \text{ if operation } o \text{ of job } j \text{ is processed on resource } r \text{ in source } s \text{ of period } t\\ 0, \text{ otherwise}\end{cases}$$

$$U_j=\begin{cases}1, \text{ if job } j \text{ is selected}\\ 0, \text{ otherwise}\end{cases}$$

The mathematical formulation for the problem under study is presented below.

$$\text{Maximize } Z = \sum_{j\in J} q_j U_j - \sum_{j\in J}\sum_{o\in O_j}\sum_{r\in R}\sum_{t\in T}\sum_{s\in S} c_{rs} X_{jorts} \tag{3-1}$$

subject to

$$\sum_{j\in J}\sum_{o\in O_j} X_{jorts} \le b_{rts} \qquad \forall r\in R, t\in T, s\in S \tag{3-2}$$

$$\sum_{s\in S}\sum_{t\in T} X_{jorts} = p_{jor} U_j \qquad \forall j\in J, o\in O_j, r\in R \tag{3-3}$$

$$\sum_{o\in O_j}\sum_{r\in R} X_{jorts} \le l_{ts} \qquad \forall j\in J, t\in T, s\in S \tag{3-4}$$

$$X_{jorts} \ge \tau Y_{jorts} \qquad \forall j\in J, o\in O_j, r\in R, t\in T, s\in S \tag{3-5}$$

$$X_{jorts} \le p_{jor} Y_{jorts} \qquad \forall j\in J, o\in O_j, r\in R, t\in T, s\in S \tag{3-6}$$

$$\sum_{r\in R} tY_{j|O_j|rts} \le d_j U_j \qquad \forall j\in J, t\in T, s\in S \tag{3-7}$$

$$\sum_{s'\in S}\sum_{t'=1}^{t-1} X_{j(o-1)rt's'} + \sum_{s'=1}^{s} X_{j(o-1)rts'} \ge p_{j(o-1)r} \sum_{r'\in R} Y_{jor'ts} \qquad \forall j\in J, o\in O_j\backslash\{1\}, r\in R, t\in T, s\in S\backslash\{|S|\} \tag{3-8}$$

$$\sum_{s\in S}\sum_{t'=1}^{t} X_{j(o-1)rt's} \ge p_{j(o-1)r} \sum_{r'\in R} Y_{jor'|S|t} \qquad \forall j\in J, o\in O_j\backslash\{1\}, r\in R, t\in T \tag{3-9}$$

$$X_{jorts} \ge 0 \qquad \forall j\in J, o\in O_j, r\in R, t\in T, s\in S \tag{3-10}$$

$$Y_{jorts} \in \{0,1\} \qquad \forall j\in J, o\in O_j, r\in R, t\in T, s\in S \tag{3-11}$$

$$U_j \in \{0,1\} \qquad \forall j\in J \tag{3-12}$$

The objective (3-1) is formulated to maximize the total net profit over the planning horizon. The first term in the objective function is the total revenue and the second term is the total processing or labor cost. The constraint set (3-2) ensures that the capacity of resource *r* of source *s* in time period *t* is not violated. Constraint set (3-3) ensures that

adequate resources are allocated to process operation $o$ of job $j$. The total number of hours allocated to process an operation should be equal to its processing time. The equality constraint set (3-3) can be replaced with an inequality ($\geq$) constraint. The second term in the objective function will prevent allocating more resources than what is required.

The constraint set (3-4) ensures that each operation of a job is processed for no more than $l_{ts}$ hours in each source during each time period. If the processing time of operation $o$ is less than $l_{ts}$, then it is possible to start processing the next operation ($o$+1) in the same time period. Since operation ($o$+1) cannot be started before operation $o$, the remaining time available for operation ($o$+1) in period $t$ is only ($l_{ts}$-$p_{jor}$). Consequently, the total time allocated to process job $j$ in any time period cannot exceed $l_{ts}$ hours. The constraint sets (3-5) and (3-6) set the $Y_{jorts}$ decision variables to either 1 or 0. The $Y_{jorts}$ variable is an indicator variable. It takes a value of 1 when $X_{jorts} > 0$, indicating that operation $o$ of job $j$ is scheduled for processing on resource $r$ of source $s$ in time period $t$; otherwise it takes a value of 0. The parameter "$\tau$" in constraint (3-5) indicates that whenever an operation is processed on a resource it should be processed for at least "$\tau$" units of time. The $Y_{jorts}$ variables are used to ensure the precedence relationship. The constraint set (3-7) ensures that when an order for a job is accepted, the completion time of the last operation of that order does not exceed the order due date.

The next two constraints impose precedence restrictions. The constraint set (3-8) ensures that operation $o$ of job $j$ can be processed in period $t$ during regular hours only after

completing operation ($o$-1). The first term in constraint (3-8) represents the total number of hours allotted to process operation ($o$-1) in time periods 1,…,($t$-1). It includes both the regular time and overtime hours allocated to process operation ($o$-1) in each time period up to and including ($t$-1). The second term in constraint (3-8) represents the number of hours allocated to process operation ($o$-1) in time period $t$ during regular hours. Figure 3-1 illustrates how the precedence relationship (i.e., constraint 3-8) will take effect when operation ($o$-1) is completed in time period $t$ during regular hours. The processing of operation ($o$-1) begins during the regular hours of production in time period ($t$-1), continues during the overtime and then to regular hours of production in time period $t$. The processing of operation $o$ can begin either during the regular hours of production or during overtime in time period $t' \geq t$. For illustration purposes, it is assumed that operation ($o$-1) and $o$ require same resource $r$ in Figure 3-1. Figure 3-2 illustrates the case when both the operations require different resources. Operation $o$ can be processed on resource $r$' only after operation ($o$-1) is completely processed on resource $r$ ($r' \neq r$).

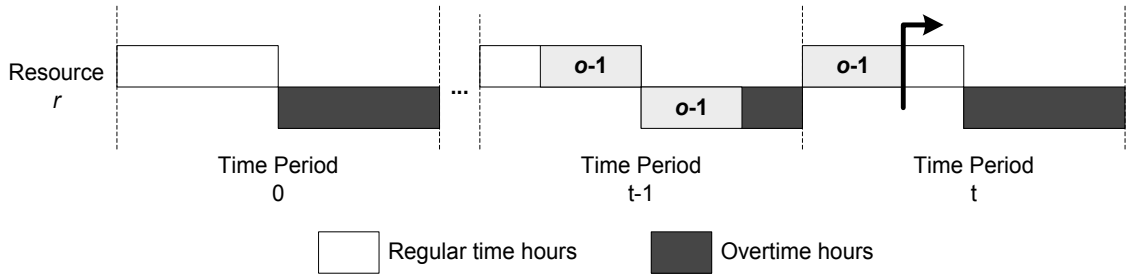



**Figure 3-1. Constraint (3-8) with successive operations on the same resource**

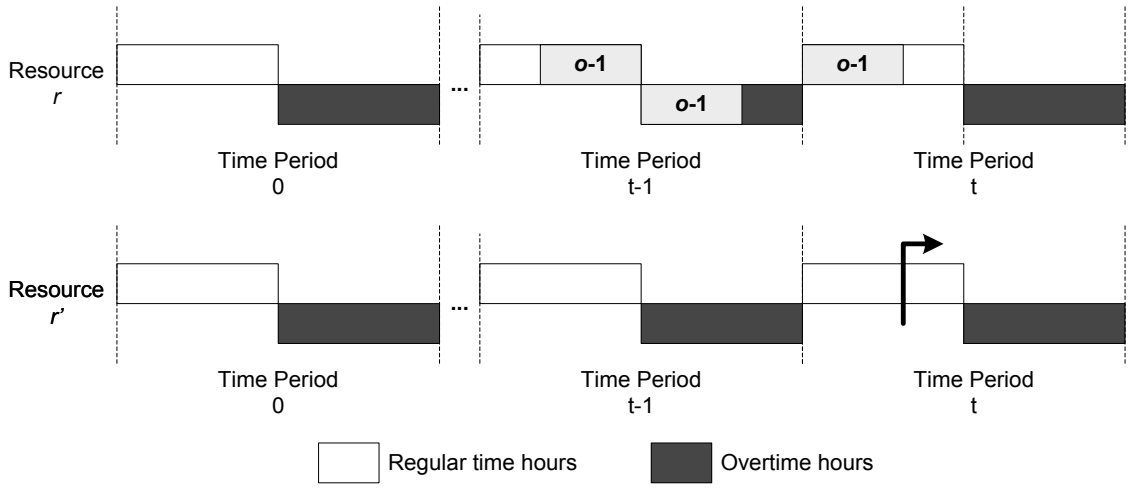



**Figure 3-2. Constraint (3-8) with successive operations on different resources**

The constraint set (3-9) ensures that operation $o$ of job $j$ can be processed in period $t$ during overtime only after completing operation ($o$-1). Figure 3-3 illustrates how the precedence relationship (i.e., constraint 3-9) will take effect when operation ($o$-1) is completed in time period $t$ during overtime. The processing of operation $o$ can begin during the remaining overtime hours in time period ($t$-1) or any source (either regular hours or overtime) in time period $t' > t$.

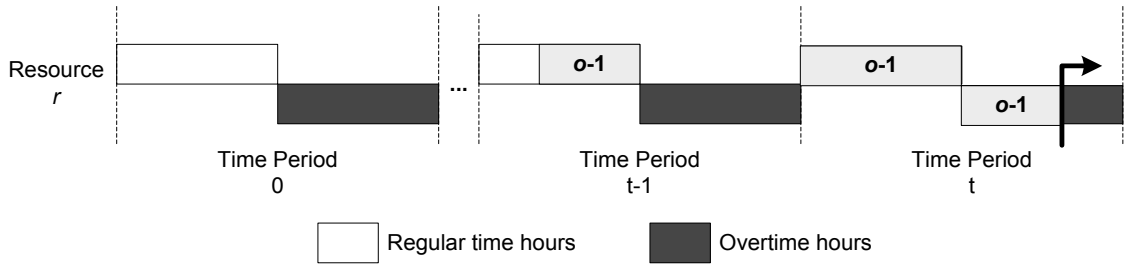



**Figure 3-3. Precedence constraint (3-9)**

The constraint sets (3-10) – (3-12) impose the non-negativity restrictions on the decision variables. In particular, the constraint sets (3-11) and (3-12) impose the binary restrictions on the decision variables $Y$ and $U$.

This model can help the operations manager to determine which subset of incoming customer orders should be selected to maximize profits. The model presents a detailed capacity plan for the accepted orders such that they are completed before their due-dates. This is useful to carefully plan for the resources used in overtime hours. The model can be run at the beginning of each decision period, such that the operations manager can reserve capacity for already accepted orders and determine which new orders to accept. In situation where a particular order or orders have to be selected for strategic reasons, a corresponding subset of orders that will maximize the profits can be known. The model is also useful to reschedule the already accepted orders when new orders have to be accepted. Section 3.3 presents some examples to illustrate the above scenarios that the end user might encounter and to show how the formulation when solved can be used to one's advantage. The mathematical model was solved using ILOG CPLEX 10.1, which is a tool for solving linear optimization problems. ILOG CPLEX can also solve several extensions of linear programs like network flow, quadratic problems, and mixed-integer programs.

### 3.4. "What-if" scenario analysis using the MTO formulation

This section presents some illustrations to show the usefulness of the formulation for making decisions to maximize the revenue in different situations. Table 3-1 through

Table 3-3 give the data for Experiment A for illustrating how a decision maker could use the mathematical formulation proposed. Two kinds of sources are considered, namely regular production time and overtime. It is assumed that the regular production time and overtime is 8 hours each. In each source, three resources are considered. The planning horizon includes three time periods. The decision maker has to decide which jobs to accept and how to schedule the accepted jobs such that they are processed before their due date. The objective is to maximize total profit. At the beginning of the planning horizon, three jobs are available to choose from. The selling price and due date for each job is given.

**Table 3-1. Experiment A to show the usefulness of the model**

| | |
|---|---|
| $j \in J=\{1,2,3\}$ | Three jobs |
| $o \in O_j=\{1,2,3\}$ | Each job comprises of 3 operations |
| $r \in R=\{1,2,3\}$ | Three resources |
| $t \in T=\{1,2,3\}$ | A planning horizon of 3 periods |
| $s \in S=\{1,2\}$ | 1: regular hours of production; 2: overtime |

**Table 3-2. Customer order data for Experiment A**

| Job # | # of operations | Due Date ($d_j$) | Selling Price ($q_j$) | Operation # | Resource # | Process Time ($p_{jor}$) |
|---|---|---|---|---|---|---|
| 1 | 3 | 3 | 10000 | 1 | 3 | 14 |
| | | | | 2 | 1 | 12 |
| | | | | 3 | 2 | 14 |
| 2 | 3 | 3 | 9800 | 1 | 3 | 14 |
| | | | | 2 | 1 | 10 |
| | | | | 3 | 2 | 14 |
| 3 | 3 | 3 | 6500 | 1 | 1 | 8 |
| | | | | 2 | 3 | 6 |
| | | | | 3 | 2 | 8 |

**Table 3-3. Data of resource cost for Experiment A**

| Processing Cost/hr ($c_{rs}$) | | |
|---|---|---|
| Resource ($r$) | Source (s) | |
| | 1 | 2 |
| 1 | 200 | 300 |
| 2 | 200 | 250 |
| 3 | 200 | 200 |

Figure 3-4 shows the Gantt chart for the example instance under consideration. The total profit is $3800 and the orders for jobs 2 and 3 are accepted. The detailed schedule for each job is also shown. Since the cost of using resource 1 during overtime is higher than resources 2 and 3, resource 1 is not used during the overtime hours. The utilization of each resource can be easily computed. The utilization of resources 1, 2, and 3 is 37.5%, 45.83%, and 41.67%, respectively.

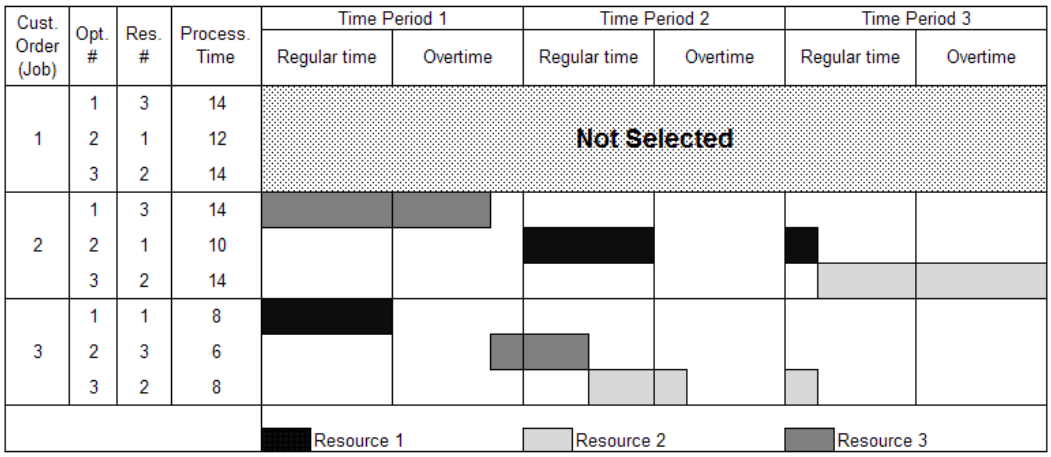



**Figure 3-4. Gantt chart for Experiment A**

Suppose two new jobs (say jobs 4 and 5) become available at the end of the first time period. The decision maker would like to know whether or not to accept these orders as

some of the resources have already been reserved to process orders for jobs 2 and 3. Table 3-4 gives the data associated with jobs 4 and 5.

**Table 3-4. Data for Experiment A extension**

| Job # | # of Operations | Due Date ($d_j$) | Selling Price ($q_j$) | Opt # | Resource # | Process Time ($p_{jor}$) |
|---|---|---|---|---|---|---|
| 4 | 2 | 4 | 5800 | 1 | 1 | 12 |
| | | | | 2 | 3 | 14 |
| 5 | 3 | 4 | 6200 | 1 | 3 | 10 |
| | | | | 2 | 2 | 8 |
| | | | | 3 | 1 | 10 |

When the mathematical model was solved with the new information (fixing the resources already committed to process jobs 2 and 3), job 4 was chosen and the total profit increases to $4000. The model recommends to select job 4 and not to select job 5. The Gantt chart for the new schedule is shown in Figure 3-5.

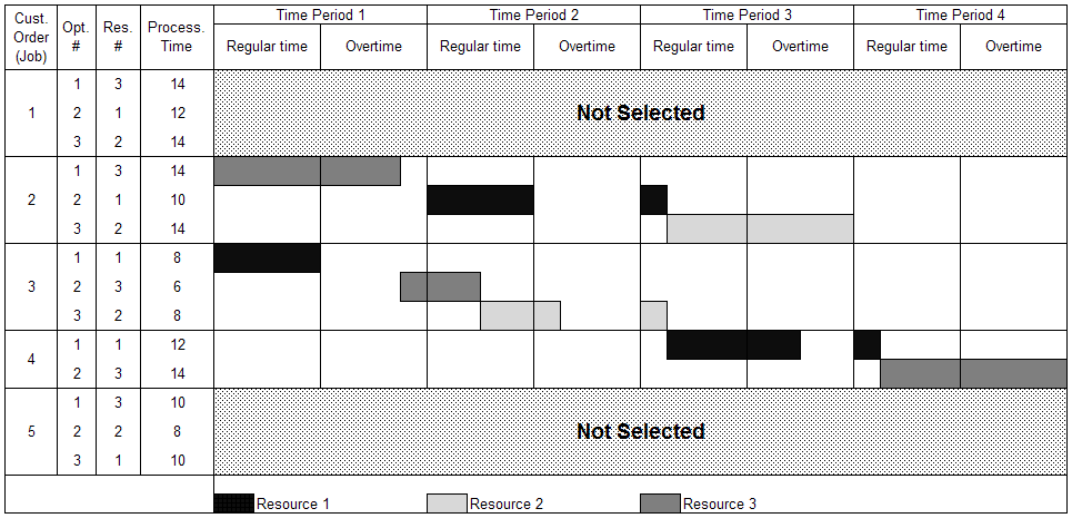



**Figure 3-5. Gantt chart for extension of Experiment A**

In the case where the company policy would have been to maximize capacity utilization both jobs 4 and 5 would have been accepted. The total profit in such a scenario would have been $3900 which is less than the optimal found by solving the model.

Another interesting illustration is the analysis of the effect of the change in the overtime cost with respect to the regular time cost. Four jobs were assumed to be available at the beginning of the planning horizon. Other job specific information is given in Table 3-5. The processing times of the operations were randomly generated from a discrete uniform distribution [3, 12]. The unit cost of all the resources was considered the same and the ratio of the unit cost of the resource in regular time to unit cost of the resource in overtime was varied. The different ratios under which the experiment was run were 1:1, 1:1.2, 1:1.5 and 1:2. The planning horizon was assumed to be five time periods. In this experiment, the combined regular time and overtime hours available is more than the time required to complete all the orders. Consequently, when the ratio was 1:1, the orders for all the jobs were accepted. The results obtained for the various cost structure considered are summarized in Figure 3-6. Even when the ratio was 1:1.2, the orders for all the jobs were accepted. However, the profit reduced due to the additional cost incurred for processing some jobs during the overtime. It can be seen that with the increase in the overtime cost, the number of hours the resources are allocated in overtime is reduced. At a certain point the increase in overtime cost was so high that it prohibited accepting orders for some jobs. For example, when the ratio was 1:1.5, job 1 was not selected. When the ratio was further increased to 1:2, job 4 was not selected.

**Table 3-5. Data for Experiment B**

| Job # | No. of Operations ($O_j$) | Due-date ($d_j$ in days) | Selling Price ($q_j$ in \$) |
|---|---|---|---|
| 1 | 3 | 4 | 3375 |
| 2 | 4 | 4 | 3335 |
| 3 | 3 | 5 | 2990 |
| 4 | 5 | 5 | 4750 |

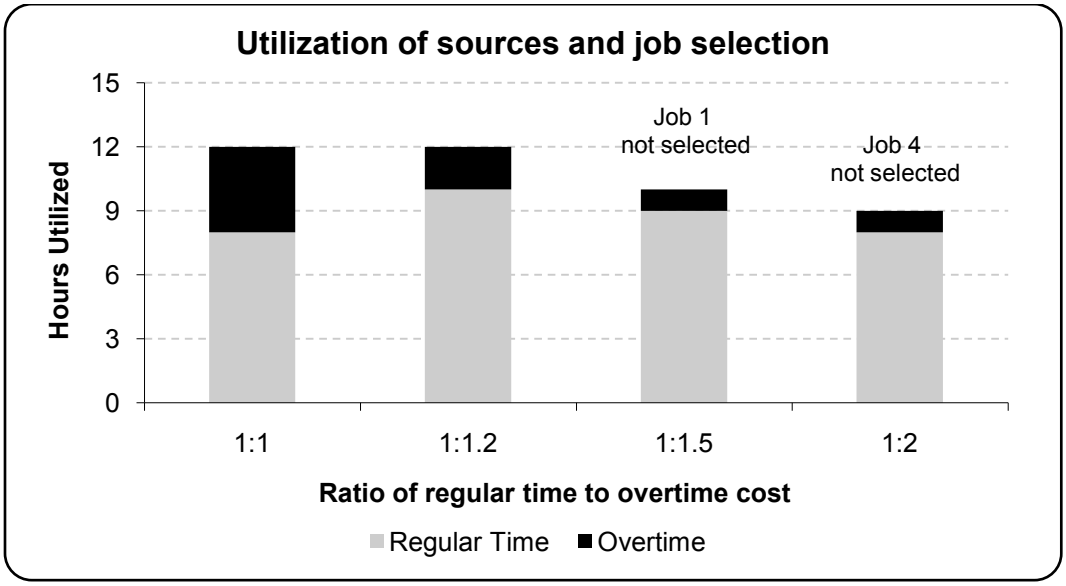



**Figure 3-6. Source utilization and selection of jobs**

In practice, the MTO firms would generally accept an order whenever the capacity is available to fulfill an order. By adopting this policy, the utilization of the resources is maximized. However the next set of experiment (Experiment B) shows that this policy may not be profitable. In addition, by committing all the resources available in the current time period, the more profitable orders which may arise in the future may not be accepted due to lack of capacity. When the cost ratio was 1:1.5, job 1 was not selected. Suppose

the firm decides to accept the orders for all the jobs (note the regular time + overtime capacity available is more than the demand), the profit reduces to $1450. When the cost ratio was 1:2, job 4 was not selected. But when the manager accepts all the orders, the profit reduces to $450. These experiments clearly indicate that even when the available capacity is more than the demand, the cost incurred to operate the resources during the overtime may prohibit accepting all the orders.

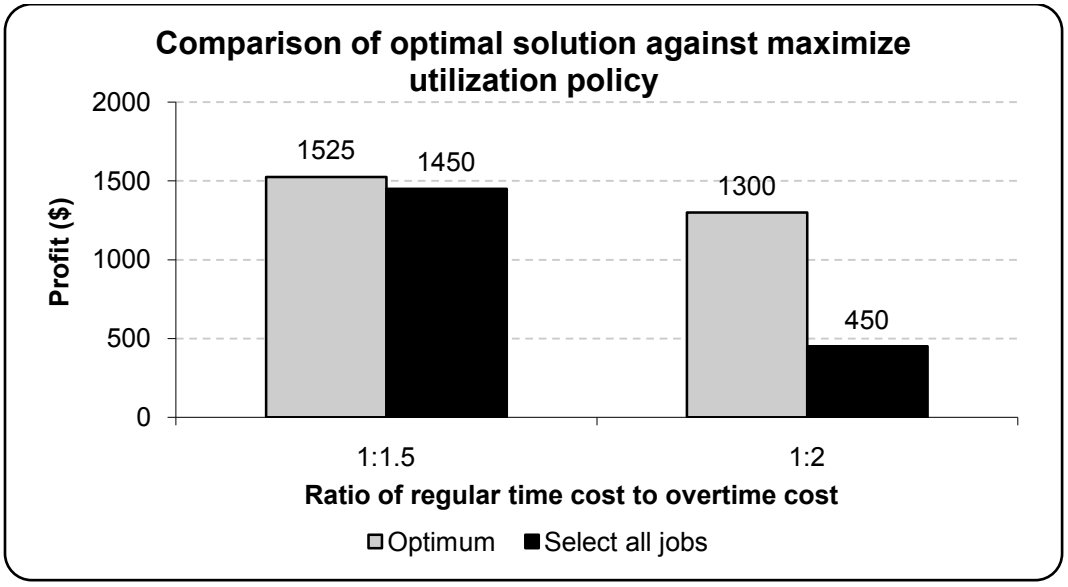



**Figure 3-7. Optimal solution against utilization maximizing policy**

Figure 3-7 shows the reduction in profit when the orders for all the jobs were accepted. These experiments clearly indicate the usefulness of the model proposed. Without the model, the decision maker would incline to accept orders based on the residual resource capacity alone.

### 3.5. Computational runtime analysis

In order to solve a mixed-integer problem, CPLEX uses a branch-and-bound approach to fix the fractional variables to integer values. Consequently, it may not be able to solve problem instances with large number of integer variables in reasonable time. An experimental study (Experiment C) was conducted to determine the effect of problem size on the run-time (computation time) required to find an optimal solution. Various factors determine the size of the problem, namely, the number of customer orders or jobs, number of operations for each job, the number of resources, due-dates for each job and the planning horizon. We introduce a demand-to-capacity ratio (DC ratio) to control the load on the MTO shop-floor. The DC ratio is the ratio of the demand to the regular time capacity available in the MTO operation given by equation (3-13), over the planning horizon with $|T|$ time periods. If we know the total demand and the available resources, we can generate problem instances by computing the number of time periods required for a fixed DC ratio using Equation (3-14).

$$\text{DC Ratio} = \frac{\sum_{j\in J}\sum_{o\in O_j}\sum_{r} p_{jor}}{\sum_{r\in R}\sum_{t\in T} b_{rt,s=1}} \tag{3-13}$$

$$\text{Number of Time Period } (|T|) = \left\lceil \frac{\sum_{j\in J}\sum_{o\in O_j}\sum_{r\in R} p_{jor}}{|R|*l_{t,s=1}*(\text{DC ratio})} \right\rceil \tag{3-14}$$

Table 3-6 presents the data used for Experiment C. Number of jobs and numbers of operations for each job are the two factors which are varied. The levels for the two factors are presented in Table 3-6. The length of each source was fixed to 8 hours. For a

DC ratio of 1.0 with different levels for jobs and operations, the planning horizon varied from 3 to 17 time periods. For each combination of the factor and level, three instances were randomly generated. The due-date for each job was equal to the planning horizon computed for that problem instance. The ratio of regular time to overtime cost was kept constant at 1:1.5. The runtime to solve the mathematical model to optimality was reported.

**Table 3-6. Data for Experiment C (computational runtime analysis)**

| Factors | Levels | |
|---|---|---|
| Number of jobs | 3 | 5 |
| Number of Operations | 3,5, and 8 | |
| Number of resources | 3 | |
| Number of sources | 2 (Regular Time, and overtime) | |
| Processing time | Discrete Uniform (4,16) hours | |
| Demand-to-Capacity Ratio | 1.0 | |

Figure 3-8 shows the runtime in seconds against the number of operations per job for 3 job and 5 job instances. In two instances for 5 jobs with 8 operation problem, CPLEX was not able to find an optimal solution even after a runtime of more than 16 hours (>62000 seconds), hence for those we report the optimality gap in Figure 3-8.

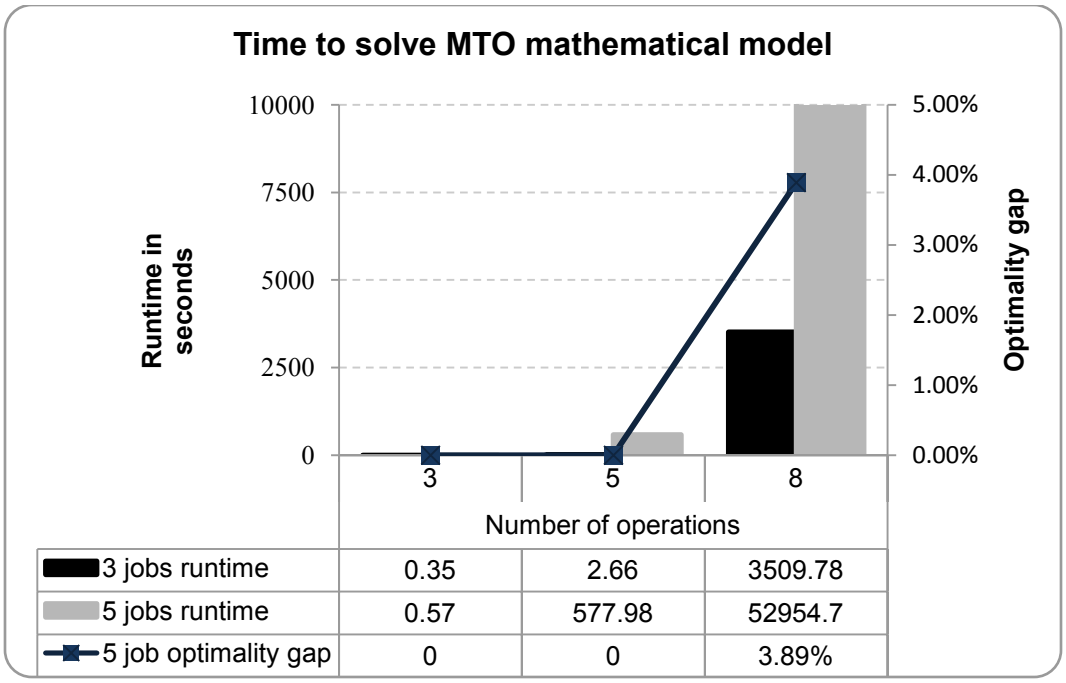



**Figure 3-8. Computation runtime to solve MTO mathematical model to optimality**

For the above problems, the planning horizon was anywhere between 3 to 17 days. We aspire to solve short-term capacity planning problems with a planning horizon up to a month (30 days) and a set of 8 to 10 customer orders, each having more than 5 operations to bid for each day. The bidding process is an iterative negotiation process and hence the sales department may have to consider different scenarios before accepting an order as illustrated in Section 3.4. The decision process in a bidding negotiation is highly time sensitive. Considering the above factors, there is a need to generate solutions to the order acceptance and capacity planning in MTO operations relatively quickly for large problem sizes.

To effectively solve large-scale integer programs column generation has proven to be one of the most successful approaches [12]. Column generation is based on the Dantzig-Wolfe decomposition principle for linear programs. For solving integer programs the column generation procedure is combined with branch-and-bound procedure, which is commonly referred to as branch-and-price.

# 4. BRANCH AND PRICE ALGORITHM

This chapter explains the Dantzig-Wolfe decomposition, column generation and the branch-and-price algorithm. In Section 4.2, the decomposition of the model proposed in Chapter 3 is described.

## 4.1. Theory of Branch-and-Price Algorithm

The decomposition algorithm has an interesting economic interpretation. Consider the case of a large system that is composed of smaller subsystems. Each subsystem has its own objective and constraints, and the objective function of the overall system is the sum of the objective functions of the subsystems. In addition, all the subsystems share a few common resources, and hence, the consumption of these resources by all the subsystems must not exceed the availability. With this in mind the decomposition algorithm can be interpreted as follows. With the current proposals of the subsystems, the total system obtains a set of optimal weights for these proposals and announces a set of prices for using the common resources. These prices are passed down to the subsystem, which modify their proposals according to these new prices [43].

The problem under consideration is well-suited for applying column generation. Each job or customer order can be considered as a subsystem, independent of other job, and a schedule can be generated. Later the individual subsystems should be merged together to obtain a feasible schedule for the entire problem under study.

The decomposition principle is a systematic procedure for solving large-scale linear programs or linear programs that contain specially structured constraints. The constraints are partitioned into two sets: general constraints (or complicating constraints) and constraints having a special structure. Special structure, when available, enhances the efficiency of the procedure.

The strategy of the decomposition procedure is to operate on two separate linear programs: one over the set of general constraints and one over the set of special constraints. Information is passed back and forth between the two linear programs until an optimal solution to the original problem is achieved. The linear program over the general constraints is called the master problem, and the linear program over the special constraints is called the sub-problem. The master problem passes down a continually revised set of cost coefficients to the sub-problem, and receives from the sub-problem a new column (or columns) based on these cost coefficients. For this reason, such a procedure is also known as a column generation technique.

Consider the following linear program, where $X$ is a polyhedral set representing specially structured constraints, $A$ is an $m$ x $n$ matrix, and $b$ is an $m$ vector:

Minimize $\mathbf{cx}$

Subject to $\mathbf{Ax}=\mathbf{b}$

$\mathbf{x}\in X$

To simplify the presentation, assume that $X$ is nonempty and bounded. Since $X$ is a bounded polyhedral set any feasible point $x\in X$ can be represented as a convex

combination of the finite number of extreme points of $X$. Denoting these extreme points by $x_1, x_2, \ldots, x_t$ any $x \in X$ can be represented as:

$$x = \sum_{j=1}^{t} \lambda_j x_j$$

$$\sum_{j=1}^{t} \lambda_j = 1$$

$$\lambda_j \geq 0, \qquad j = 1,2,\ldots,t$$

Substituting for $x$, the foregoing optimization problem can be transformed into the following so-called master problem in the variables $\lambda_1, \lambda_2, \ldots, \lambda_t$.

$$\text{Minimize} \sum_{j=1}^{t} (cx_j)\lambda_j \tag{4-1}$$

Subject to

$$\sum_{j=1}^{t} (\boldsymbol{A}x_j)\lambda_j = \boldsymbol{b} \tag{4-2}$$

$$\sum_{j=1}^{t} \lambda_j = 1 \tag{4-3}$$

$$\lambda_j \geq 0, \qquad j = 1,2,\ldots,t \tag{4-4}$$

Since $t$, the number of extreme points of the set $X$, is usually very large, attempting to explicitly enumerate all the extreme points $x_1, x_2, \ldots, x_t$ and explicitly solving this problem is a very difficult task. In column generation, an extreme point or column is generated as and when required by solving a sub-problem known as the pricing problem. After adding the column to the master problem the LP is re-optimized. This is done iteratively until no

new column can price out of the sub-problem (SP); this being the check for optimality of the LP master problem. The pricing problem is given by,

$$\text{Maximize } (\boldsymbol{wA}-\boldsymbol{c})\boldsymbol{x}+\alpha \tag{4-5}$$

$$\text{Subject to } \boldsymbol{x}\in X \tag{4-6}$$

Where, $\boldsymbol{w}$ and $\alpha$ denote the dual variables for equations (4-2) and (4-3), respectively [43]. The master problem is restricted in the sense that all the columns are not known explicitly and hence it is called as a Restricted Master Problem (RMP).

To start the column generation scheme, an initial feasible solution to the restricted master problem has to be determined. The initial RMP could then pass proper dual information to the pricing problem. The pricing problem or sub-problem plays a critical role through their structure, symmetry, complexity and whether or not they exhibit the Integrality Property. The integer variables in an IP typically become decision variables in one or more SPs. Formulations that have block diagonal structure (i.e. each SP is separable) are attractive because they result in small, independent SPs that are typically more effectively solved. Integer SPs with no special structure are NP-hard and should be avoided. The original IP is NP-hard but need to be solved only once. Solving one or more NP-hard SPs repetitively offers no worst-case advantage and is typically computationally prohibitive. Thus, the ideal SP should have a structure that can be solved effectively since it must be solved repetitively. Experience has shown that NP-hard SPs that can be solved in pseudo-polynomial time satisfy the criterion of being solvable relatively effectively; in addition they avoid the Integrality Property as well [12].

In a maximization linear program, any column with positive reduced cost is a candidate to enter the basis. The pricing problem is to find a column with highest reduced cost. Therefore, if a column with positive reduced cost exists the pricing problem will always identify it. This guarantees that the optimal solution to the linear program will be found. However, it is not necessary to select the column with the highest reduced cost - any column with a positive reduced cost will do. Using this observation can improve the overall efficiency when the pricing problem is computationally intensive. Depending on the pricing problem, it may even be possible to generate more than one column with positive reduced cost per iteration with a large increase in computation time. Such a scheme increases the time per iteration, since a larger RMP has to be solved, but it may decrease the number of iterations.

The second important point to consider while generating columns is the fashion in which the columns are generated and added to the RMP. One strategy would be to solve all sub-problems and select the best improving column to enter the RMP. Instead, all improving columns could be made available to the RMP through column management procedure. A third strategy would be to solve SPs in a round robin fashion, entering each improving column identified and re-optimizing the RMP, solving SPs in a random order. Another method would be to solve SP using a heuristic to generate good solution quickly and in case it fails an optimizing algorithm which takes more runtime can be used to identify an improving column. Preliminary tests may be used to identify the best way to add columns to the RMP [12].

In solving integer programs (a program incorporating integer, and / or binary, or mixed-integers) there is a binary restriction on the $\lambda$ variable in the RMP. The RMP is optimized as a LP but when we have an optimal solution of the LP if those variables do not satisfy the integrality restriction then a branch-and-bound procedure is implemented. Branch-and-price, which is a generalization of branch-and-bound with LP relaxations, allows column generation to be applied throughout the branch-and-bound tree [44]. This is known as Branch & Price (B&P). However, this implementation is not straight forward and there are fundamental difficulties in applying column generation techniques for linear programming in integer programming solution methods [45]. These problems arise because conventional integer programming branching may not be effective as fixing variables can destroy the structure of the sub-problem and solving the relaxed master problem may be inefficient. Furthermore the branching should result in child nodes that represent balanced set of solutions. Balancing is important because it can be expected to result in a tighter bound at each sibling node, facilitating solution. A branching that does not balance solutions defines one sibling that represents just a few of the solutions associated with the parent node and another that represents all remaining solutions. The bound associated with the former node is not likely to be as good as that of the latter. Devising an effective branching strategy may present one of the most difficult challenges to composing an effective B&P algorithm [12]. The three major aspects of implementing branch-and-price algorithm are decomposing the original model, formulating and efficiently solving the sub-problem and lastly determining the branching strategy. The B&P procedure is illustrated in Figure 4-1.

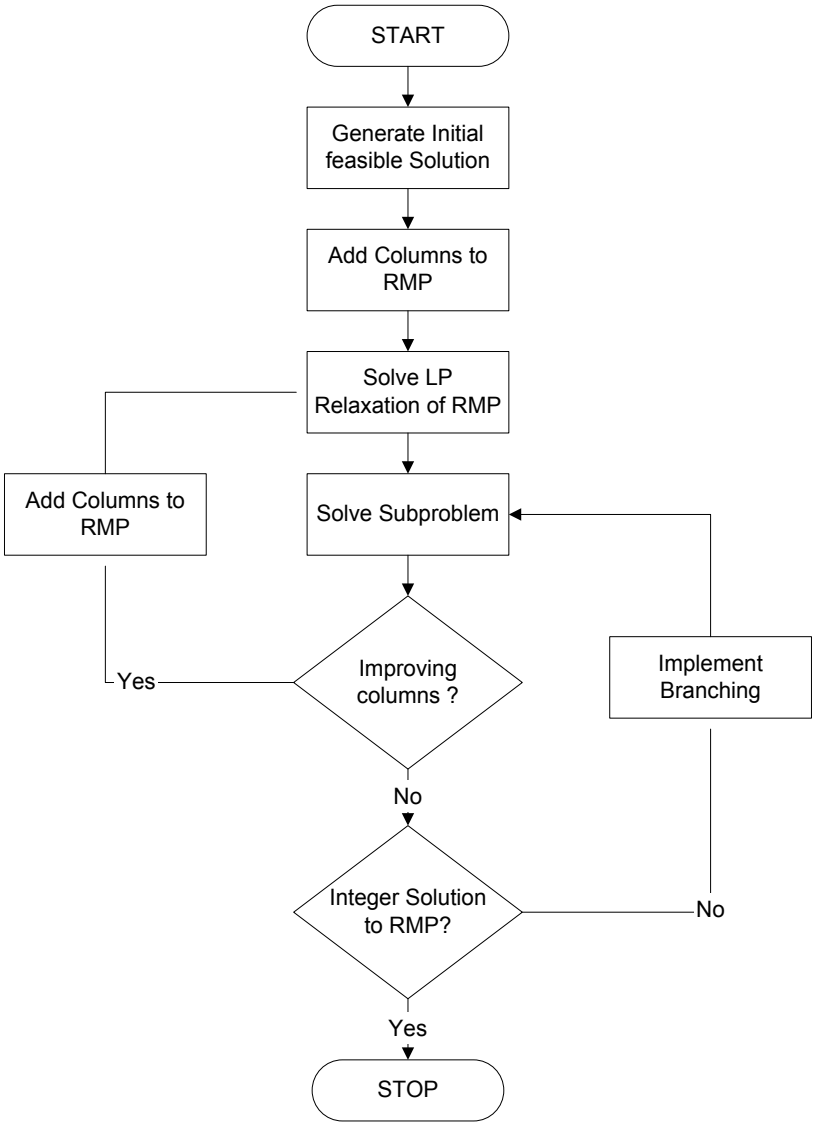



**Figure 4-1. Flowchart for the Branch and Price procedure**

The next sections discuss the implementation of the above with respect to the problem under consideration.

### 4.2. Decomposition of the MTO model

The proposed MTO model has a block diagonal or angular structure as shown in Figure 4-2. This special structure is well suited for applying the decomposition principle. The capacity constraint (3-2) is the binding or complicating constraint. The rest of the constraints can be decomposed into sets of constraints for each job that can go in the sub-problem. The sub-problem solution will generate the schedule for the corresponding job that can be added as a column to the RMP.

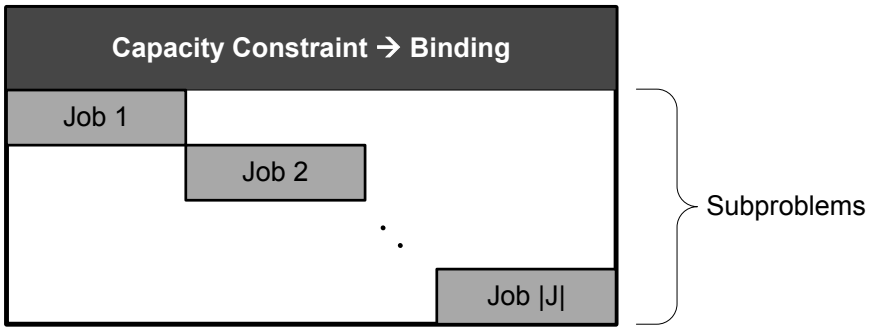



**Figure 4-2. Decomposition of the MTO model (Block-Diagonal Structure)**

The MTO restricted master problem is formulated as follows,

$$\text{Maximize } Z_{RMP}^{LP} = \sum_{j \in J} q_j U_j - \sum_{j \in J} \sum_{o \in O_j} \sum_{r \in R} \sum_{t \in T} \sum_{s \in S} \sum_{k \in K_j} (c_{rs} x_{jorts}^k) \lambda_j^k \tag{4-7}$$

Subject to

$$\sum_{j \in J} \sum_{o \in O_j} \sum_{k \in K_j} x_{jorts}^k \lambda_j^k \le b_{rts} \qquad \forall r \in R, t \in T, s \in S \tag{4-8}$$

$$\sum_{k \in K_j} \lambda_j^k = U_j \qquad \forall j \in J \tag{4-9}$$

$\lambda_j^k \geq 0$ *binary* $\forall j \in J, k \in K_j$ (4-10)

$U_j \geq 0$ *binary* $\forall j \in J$ (4-11)

Where, $K_j$ is the set of columns generated by the sub-problem for job $j$ that are added to the RMP. The column generated by the sub-problem is a feasible schedule for the corresponding job. An initial feasible solution to the RMP is provided by a greedy heuristic presented in Section 4.4.

### 4.3. Solution approach for solving sub-problem

A feasible schedule for job $j$ should satisfy the processing time constraint (3-3), the physical constraint of processing job $j$ for not more than $l_{ts}$ hours in source $s$ of time period $t$, the due-date constraint (3-7) and the precedence constraints (3-8) and (3-9). The corresponding formulation for the sub-problem or pricing problem of job $j$ will consist of the constraint set (3-3) to (3-11) with an objective of minimizing the cost of processing job $j$ by its due-date. The objective function for the pricing problem is formulated as,

$$\text{Minimize } Z_{sp}^j = \sum_{o \in O_j} \sum_{r \in R} \sum_{t \in T} \sum_{s \in S} (c_{rs} + w_{rts}) x_{orts} + \alpha_j \quad (4\text{-}12)$$

Where, $w_{rts}$ and $\alpha$ is the dual variables of constraints (4-8) and (4-9) respectively.

Ideally, the sub-problem should be able to generate columns very quickly as the sub-problem has to be solved many times during the B&P procedure. Solving the sub-problem formulation to optimality using a commercial solver to generate columns is not an efficient way. Figure 4-3 and Figure 4-4 show the average computational time (or

runtime) to solve a sub-problem with 3 jobs and 5 jobs instances to optimality using a commercial solver. It can be seen that the time taken to solve each sub-problem increases as the size of the sub-problem increases. Thus, there is a need to efficiently solve the sub-problem. The data used for this experiment was the same as used in Experiment C (Refer to Section 3.5). The sub-problem size is defined as the product of the number of operations per job and the time periods in the planning horizon, given that the number of resources and sources is kept constant.

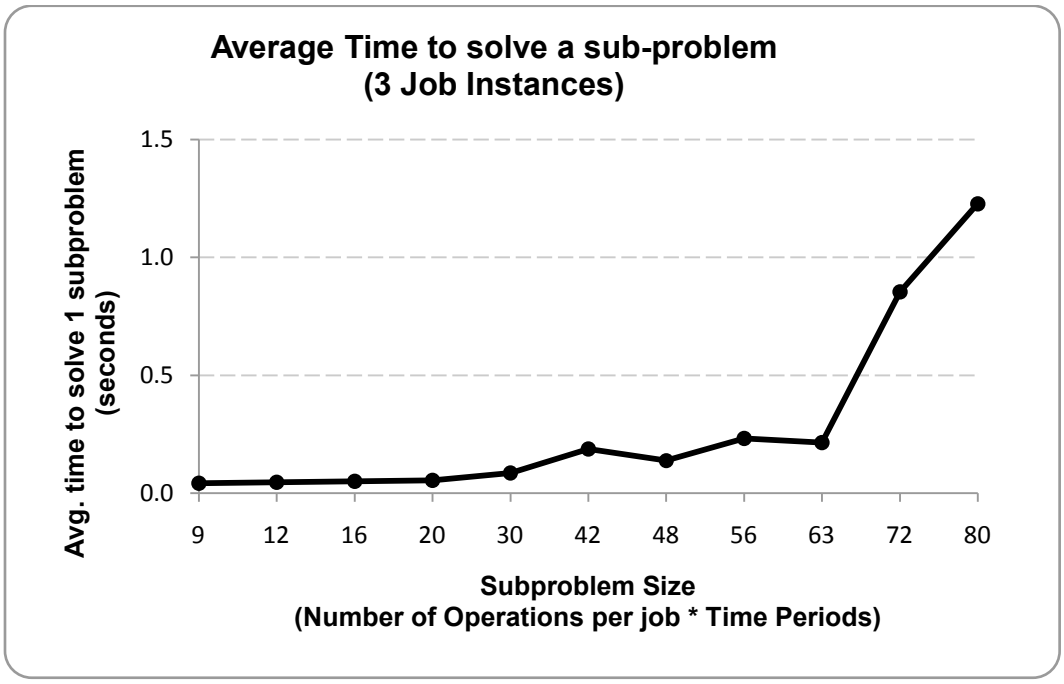



**Figure 4-3. Average time to solve sub-problem formulation for 3 job instances**

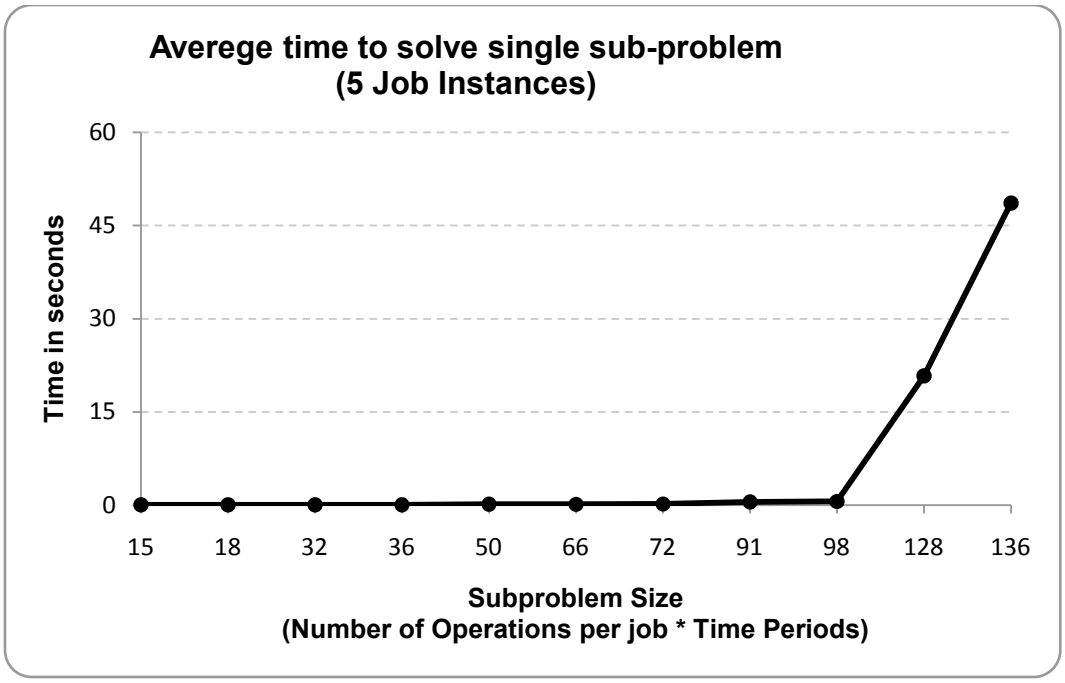



**Figure 4-4. Average time to solve sub-problem formulation for 5 job instances**

We propose an exact procedure to solve the sub-problem to optimality. We represent the sub-problem for job $j$ as a Directed Acyclic Graph (DAG) $G^j=\{N^j,A^j\}$ where $N^j$ denotes the set of nodes and $A^j$ denotes the set of arcs. Each time period is discretized into smaller intervals with equal length denoted by *dtu*. Let the set of discretized time instants for job $j$ from time period one till its due-date $d_j$ be $H^j=\left\{1,2,\ldots,\sum_{t=1}^{d_j}\sum_{s\in S}\frac{l_{ts}}{dtu}\right\}$. Each operation $o$ of job $j$ is split into *dtu* sized operations. Let the set of split operations for all the operations in $j$ be $E^j=\left\{1,\ldots,\frac{\sum_{o\in O_j}p_{jor}}{dtu}\right\}$ where $r$ is the resource type on which operation $o$ of job $j$ needs to be processed and let the set $I_o^j=\left\{1,\ldots,{}^{p_{jor}}/_{dtu}\right\}$ be the set of split operations for operation $o$ of job $j$. The set of nodes consists of three types, an artificial source node, an

artificial sink node, and *OperationTimeNodes*. The nodes in *OperationTimeNodes* set are denoted by a 2-tuple $N\{e \in E^j, h \in H^j\}$ such that we have $|H^j|$ nodes corresponding to each element in $E^j$. We have $l_{ts}/dtu$ nodes in $H^j$ corresponding to each source $s$ in time period $t$. There is a set of secondary attribute for each node represented by a 4-tuple $SA_{e,h}\{o \in O_j, i \in I_o^j, r \in R, t \in T, s \in S\}$. The set of arcs consists of two distinct types, set of idle arcs $\{IArcs \subseteq A^j\}$ and set of processing arcs $\{PArcs \subseteq A^j\}$. An arc is represented by the notation $A_{e,h}^{e'h'}$ , where $(e,h)$ and $(e',h')$ is the tail node and head node respectively. There is a cost associated with each arc denoted by $C_{eh}^{e'h'}$. Idle arcs are connected between two consecutive nodes of the same split operation starting at node $\{e,h\}$ and ending at $\{e,h+1\}$. The processing arc starting from node $\{e,h\}$ goes to node $\{e+1,h+1\}$. This ensures that each discretized operation $e$ is completed before starting discretized operation $e$+1. This structure captures the precedence constraint of the sub-problem. All arc capacities are set to one. A unit flow in the processing arc implies that the split operation $e$ is processed for $dtu$ time units in time instance $h$. A unit flow in the idle arc implies that the split operation $e$ will not be processed for $dtu$ time units in time instance $h$. A unit flow sent from the source node reaching the sink node ensures that all the operations in job $j$ are processed by the due-date $d_j$. The cost of idle arc is zero while the cost of the processing arc is given by $c_{rs}+w_{rts}$, where $r$ is the resource on which operation $o$ of job $j$ needs to be processed in source $s$ of time period $t$. The arc connecting the source node to the first node in the *operationTimeNodes* $N\{1,1\}$ is denoted by $A_{source}^{1,1}$ and cost of that is fixed to zero. All the arcs to the sink node are denoted $A_{e,h}^{sink}$. The shortest path from the source node to the sink node gives us the schedule for job $j$ at the minimum

processing cost. Figure 4-5 shows a general DAG representation of the sub-problem. In the general DAG, it is apparent that there exist nodes which cannot be reached from the source node or nodes whose outbound flow can never reach the sink node and as such they can never be part of the shortest path. Hence we can eliminate such nodes.

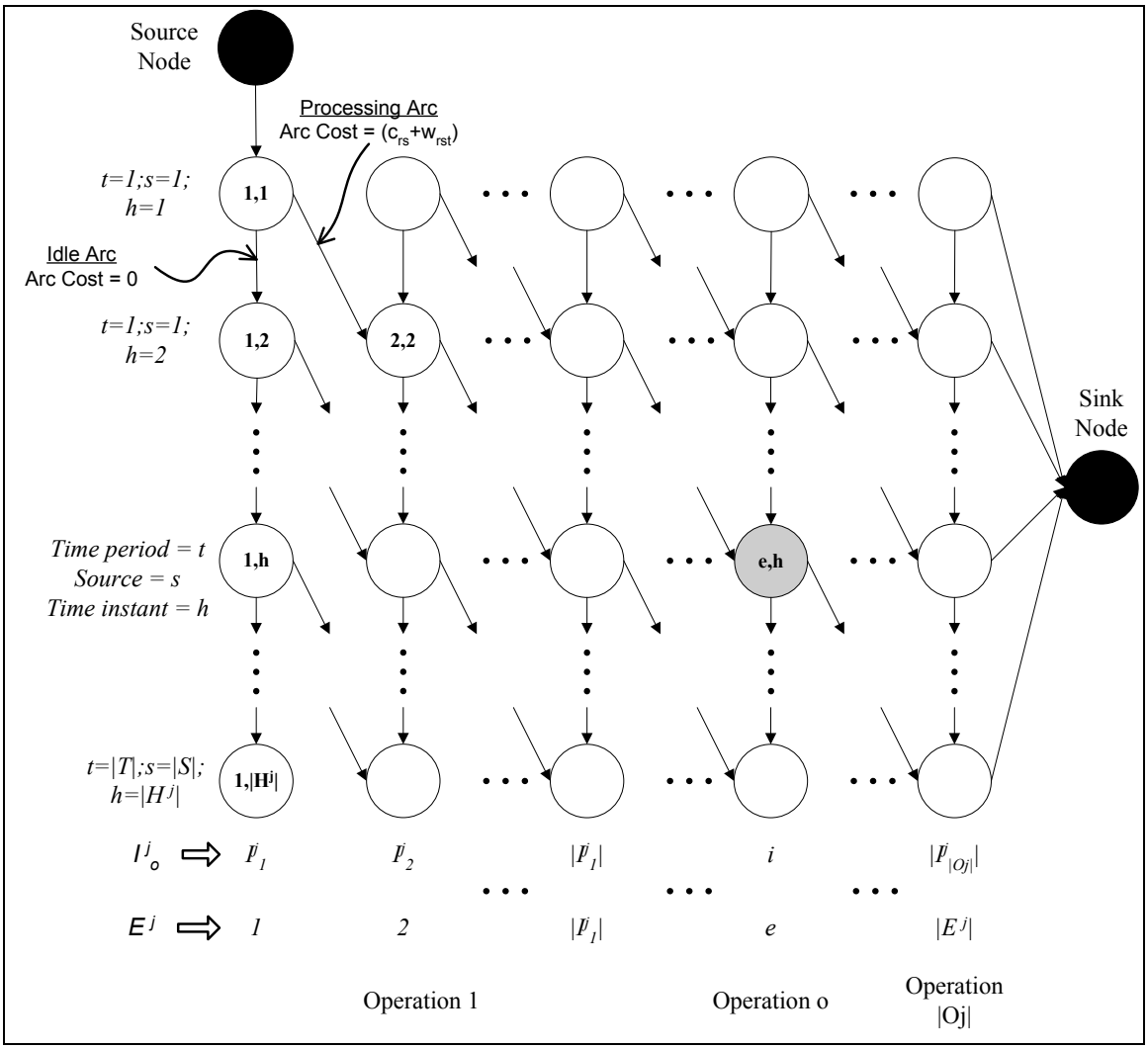



**Figure 4-5. General Directed Acyclic Graph representation of the sub-problem**

To further understand this concept, consider a sub-problem for job $j$ with three operations having processing times 5, 2, and 3 hours, respectively. For simplicity consider that they need to be processed on the same resource. Let the due-date for job $j$ be $d_j$ = 1 day and we have two sources, regular time and over time of 8 hours each. We discretize time in units of one hour. The corresponding graph for the sub-problem is shown in Figure 4-6. The earliest we can process split operation $e$=1 is in time instance 1 corresponding to $h$=1, which implies that the earliest we can process split operation $e'$=e+1=2 is in $h'$=$h$+1=2, and thus all the nodes for $e'$=2 before time instance $h'$=2 can be ignored in $G^j$. For processing job $j$ by its due-date, the latest we can process the split operation $e$=1 is in time instance 7 corresponding to $h$=7, thus the flow from all the nodes {$h$∈8,...,16 ,$e$=1} cannot reach the sink node and thus the corresponding nodes can be ignored in $G^j$. We can extend this logic for all the split operations and time instances $\{e \in E^j,\ h \in H^j\}$ and eliminate the corresponding nodes.

Figure 4-6 shows the shortest path from the source node to the sink node. The nodes visited in the shortest path are shaded in black and the path is represented by thick arrows. In each time period and source we can count for each operation how many processing arcs have been traversed which will give us the number of hours of processing of that operation. For example, for regular time in time period 1, we are processing operation 1 for four hours. Then the processing of operation 1 is continued in overtime for one hour.

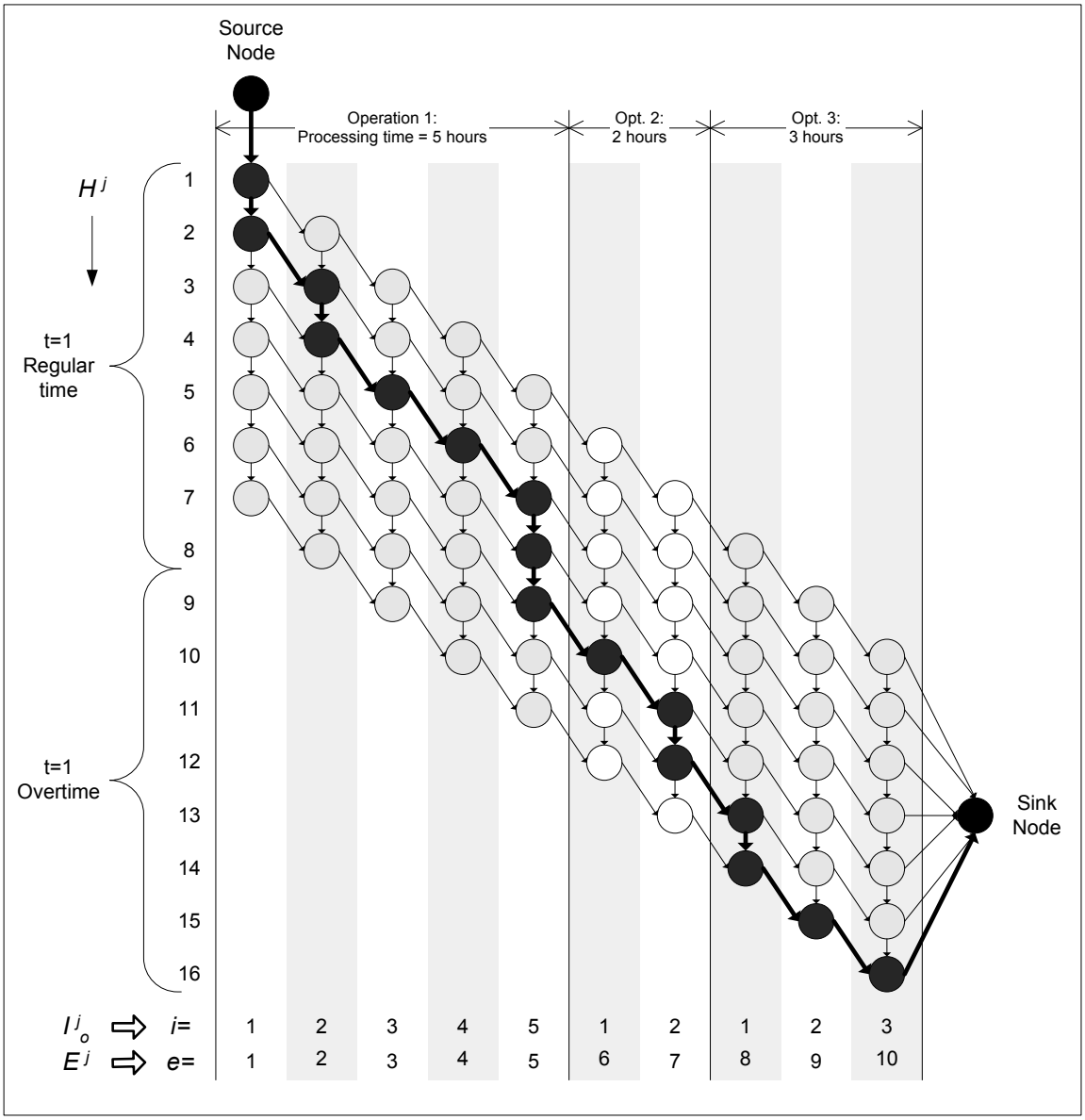



**Figure 4-6. Directed acyclic graph for sub-problem representation**

The pseudo code for finding the shortest path for acyclic digraph [46] and extracting the schedule from the shortest path solution are given below:

Algorithm for finding the shortest path:

*BEGIN*

$v[N\{1,1\}] \leftarrow 0$

$optPathTo[N\{1,1\}] \leftarrow source\ node$

*For* $N\{e',h'\} \in OperationTimeNodes\ \{e \in E^j, h \in H^j\} \backslash N\{1,1\}$

*If N{e',h'} exists then*

*For* $N\{e,h\} \in \{N\{e',h'-1\}, N\{e'-1,h'1-\}\}$

$v[N\{e',h'\}] \leftarrow min\{v[N\{e,h\}] + C_{eh}^{e'h'} : (A_{eh}^{e'h'}\ exists)\}$

*optPathTo[*$N\{e',h'\}$*]*←*node N{e,h} achieving the minimum cost*

*End for*

*End if*

*End for*

*Let v[sink node]*←$min_{\forall o \in O_j}\{v[N\{|E^j|,h\}] + C_{eh}^{sink} : (A_{|E^j|h}^{sink} exists, \forall h \in H^j)\}$

*Let optPathTo[sink node]*←*node N{e,h} achieving the minimum cost*

*END*

Algorithm to extract schedule from the shortest path solution for sub-problem *j*:

*BEGIN*

*Initialize* $x_{jorts}=0\ (\forall o \in O_j, r \in R, t \in T, s \in S)$

*Let N{e,h}*←*optPathTo[sink node]*

*If* $(A_{e,h}^{sink} \in PArcs)$ *then*

*Let* $x_{jorts} \leftarrow x_{jorts}+1\ (\forall o,r,t,s \in SA_{e,h})$

*End if*

*While* $(N\{e,h\} \neq$ *source node)*

*Let* $N\{e',h'\} \leftarrow optPathTo[N\{e,h\}]$

*If* $(A^{e,h}_{e',h'} \in PArcs)$ *then*

*Let* $x_{jorts} \leftarrow x_{jorts} + 1$ $(\forall o,r,t,s \in SA_{e',h'})$

*End if*

*Let* $N\{e,h\} \leftarrow N\{e',h'\}$

*End while*

*END*

### 4.4. Greedy heuristic for initial solution to RMP

In order to start the column generation procedure, a basic feasible solution to the RMP has to be provided. We propose a greedy heuristic to obtain this solution. The set of available jobs *J* are sorted in a non-increasing order of their profit margins, where profit margin is the ratio of the sales price to the cost of processing the job in regular time, and stored in a list. Each job in this list is scheduled one at a time with an objective of minimizing their processing costs. If scheduling a job improves the objective function value, the total profit, then the job is accepted and the residual capacities for the resources are updated along with the total profit yielded by accepting the current job, else the job is rejected. The schedule obtained for the job is added as a column to the RMP. For scheduling each job so that we can minimize its processing costs, we make use of the sub-problem solution approach described in Section 4.3. The dual prices are set to zero

for the capacity constraints and the convexity constraints. The costs of the processing arcs which have been already utilized by previously scheduled jobs are set to infinity, to take care of the residual capacities of the resources in their respective time periods and sources.

### 4.5. Branching in Branch and Price algorithm

In Section 4.5.1 we define an integer feasible solution to the original problem. In Section 4.5.2 we discuss the proposed branching strategies. In Sections 4.5.3 and 4.5.4 we derive the Lagrangean bounds for fathoming the nodes and node selection for exploring the branch and bound tree respectively.

#### 4.5.1. Definition of an integer feasible solution to RMP

We formally define a feasible integer solution to the RMP.

Definition 4.1: Consider a set of columns $k \in K_j$ for job $j$ represented by the basic variables $\lambda_j^{k \in K_j}$ in the RMP, such that $\sum_{k \in K_j} \lambda_j^k = 1$, which implies $U_j$=1 (from constraint (4-9)). The convex combination $\theta_j = \sum_{k \in K_j} x_{jorts}^k \lambda_j^k$ is a feasible integer solution to the RMP for job $j$ if for any pair of operations $(o_i, o_{i+1})\{i=1,\ldots,|O_j|-1\}$ in $\theta_j$, there is no precedence violation. Since there is no restriction over $x_{jorts}$ to be integer in the original problem, $\theta_j$ is a feasible solution for the original problem.

Consider job $j$ with 3 operations having processing times 6, 10 and 4 hours, respectively. In the RMP, suppose we have two schedules corresponding to the basic variables,

$\lambda_j^1$=0.45 and $\lambda_j^2$=0.55. For an intuitive representation of a schedule we represent it as a matrix, where the rows denote the time period *t* and source *s* while the columns denote the operations. Then the schedules corresponding to the basic variables are as follows:

$$\lambda_j^1 \Rightarrow x_{jorts}^1 \Rightarrow \begin{bmatrix} 4 & 0 & 0 \\ 0 & 0 & 0 \\ 2 & 6 & 0 \\ 0 & 0 & 0 \\ 0 & 4 & 4 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{matrix} t=1,s=1 \\ t=1,s=2 \\ t=2,\ s=1 \\ t=2,s=2 \\ \\ \vdots \\ \\ t=4,\ s=2 \end{matrix}$$

$$\lambda_j^2 \Rightarrow x_{jorts}^2 \Rightarrow \begin{bmatrix} 0 & 0 & 0 \\ 6 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 2 & 2 \\ 0 & 0 & 2 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

The convex combination $\theta_j$ is given by,

$$\theta_j = \sum_{k=1}^{2} x_{jorts}^k \lambda_j^k \Rightarrow \begin{bmatrix} 1.8 & 0.0 & 0.0 \\ 3.3 & 0.0 & 0.0 \\ 0.9 & 2.7 & 0.0 \\ 0.0 & 4.4 & 0.0 \\ 0.0 & 2.9 & 2.9 \\ 0.0 & 0.0 & 1.1 \\ 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \end{bmatrix}$$

In $\theta_j$, none of the adjacent operation pairs have a precedence violation; the processing time constraint (3-3), physical constraint (3-4) and due-date constraint (3-7) are satisfied and hence $\theta_j$ is an integer feasible solution to RMP for job *j*.

Now consider another basic column $\lambda_j^3$ with a corresponding schedule given by $x_{jorts}^3$ and the new solution to RMP is $\lambda_j^1$=0.15, $\lambda_j^2$=0.35, and $\lambda_j^3$=0.5.

$$\lambda_j^3 \Rightarrow x_{jorts}^3 \Rightarrow \begin{bmatrix} 6 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 4 & 4 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

The convex combination $\theta'_j$ for the above RMP for job *j* is,

$$\theta'_j = \sum_{k=1}^{3} x_{jorts}^k \lambda_j^k \Rightarrow \begin{bmatrix} 3.6 & 0.0 & 0.0 \\ 2.1 & 3.0 & 0.0 \\ 0.3 & 2.9 & 2.0 \\ 0.0 & 2.8 & 0.0 \\ 0.0 & 1.3 & 1.3 \\ 0.0 & 0.0 & 0.7 \\ 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \end{bmatrix}$$

In $\theta'_j$, operation 1 ends in time period 2, source 1, while operation 2 starts in time period 1, source 2. Thus for operation pair (1,2) there is a precedence violation. Similarly for

operation pair (2,3) the precedence constraint is violated. Thus $\theta'_j$, is an infeasible integer solution to the RMP for job $j$.

### 4.5.2. Branching strategies

In our original formulation, we have two binary variables $Y$ and $U$. While branching in B&P algorithm the literature suggests to branch on the original variables, instead of branching on the variable $\lambda$ in the RMP. From the Definition 4.1, we know that, to get an integer feasible solution to the RMP for job $j$, $U_j$ should be exactly equal to 1, implying that job $j$ is selected and the processing time requirements of it are satisfied. Hence at any node in the branch and bound tree if we find $U_j$ to be fractional, we branch on $U_j$, setting $U_j$ to 0 in the first (left) child node and $U_j$ to1 in its twin (right) node. If at any node in the branch and bound tree, if we find $U_j$ $\{j \in J\}$ to be either 0 or 1, and all the corresponding $\theta_j$ $\{j \in J\}$ to be integer feasible solutions as per Definition 4.1, then we report the corresponding integer solution.

In the case that we find $U$ to be binary, but $\theta_j$ $\{j \in J\}$ to be integer infeasible, then we have to fix the original variable $Y$. Fixing $Y$ variables is not as straight forward as fixing $U$, since we do not have them as explicit variables in the RMP. In the original formulation $Y$ is an indicator variable used for ensuring that the linear precedence amongst operations is maintained. Whenever we have $\theta_j$ as fractional, all the constraints except the precedence constraint are satisfied. In such a case we try to restore the precedence between the pair of adjacent operations violating this constraint such that the solution subsets are equally

divided so that we get a balanced branch and bound tree. We propose a branching strategy to accomplish this as explained in the rest of the section.

A pair of operations ($o$,$o$+1) has a precedence violation if operation $o$+1 starts before the completion of operation $o$. We define this violation in absolute terms as the precedence error length $\varepsilon_p^o$ for job pair ($o$,$o$+1) given by the difference in the end time of operation $o$ and start time of operation $o$+1. Equation (4-13) computes the precedence error length between the violating adjacent pair of operations.

$$\varepsilon_p^o = 2\left(et_o - st_{o+1}\right) + (es_o - ss_{o+1}) \tag{4-13}$$

Where, $et_o$ and $es_o$ is the time period and source respectively in which operation $o$ is completed, while $st_{o+1}$ and $ss_{o+1}$ is respectively the time period and source in which operation $o$+1 starts. For the purposes of this research we refer to a particular time period and source combination as time-source instance.

Let $\theta_j'$ be an integer infeasible solution at node $n$ in the branch and bound tree and all $U_j$ $\{\forall j \in J\}$ be binary. We try to restore the precedence amongst the violating operation pair with maximum precedence error length. Let this pair be denoted by ($o$,$o$+1).

Consider the illustration in Section 4.5.1., with schedule $\theta_j'$ being an integer infeasible solution at some node $n$ in the branch and bound tree and all $U_j$ $\{\forall j \in J\}$ are binary. Figure

4-7 shows the Gantt chart for the schedule generated from this convex combination. Operation pair (2,3) has the maximum precedence error length ($\varepsilon_p^2$) of 2 units. Hence, we select this pair to restore precedence feasibility.

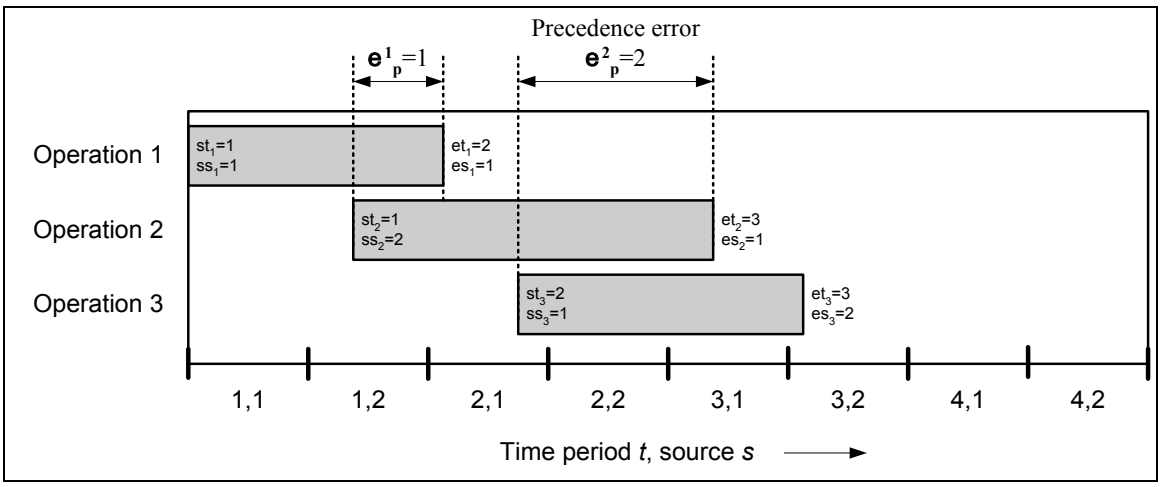



**Figure 4-7. Gantt chart for the schedule obtained from the convex combination**

In the B&B tree at node *n*, we form two child nodes. In branching strategy 1 (BPS1) for the first child node we place the restriction that operation *o* cannot be scheduled in the time period ($et_o$) and source ($es_o$) in which it had finished its processing in schedule $\theta_j'$. In the second child node we place the restriction that operation *o*+1 cannot be scheduled in the time period ($st_{o+1}$) and source ($ss_{o+1}$) when it began its processing in schedule $\theta_j'$. There are no other restrictions on scheduling either these operations or other operations.

Continuing the discussion on the illustration from Section 4.5.1, if we want to implement BPS1 for this example then in child node 1 we place the restriction that operation 2 is not

allowed to be scheduled in regular time ($s$=1) of the third time period, while in child node 2 we do not allow operation 3 to be scheduled during the regular time ($s$=1) of the second time period. Figure 4-8 shows the branching with the restrictions on each node. The black colored box shows that scheduling during those time periods and sources is not allowed.

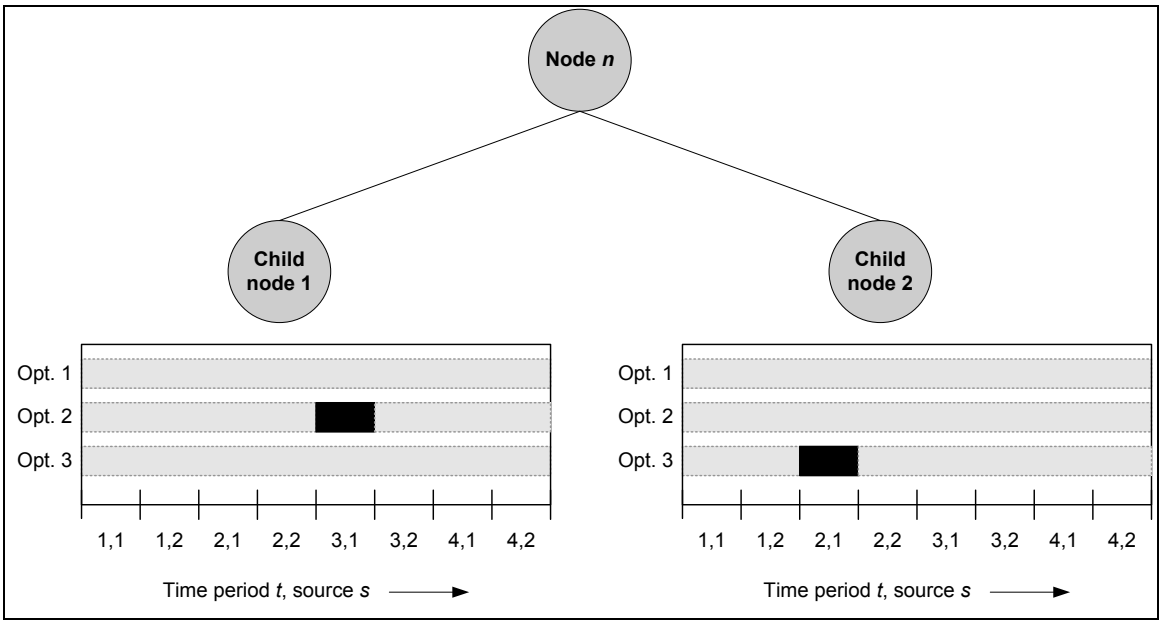



**Figure 4-8. Branching strategy 1 (BPS1)**

### 4.5.3. Lagrangian bounds

Column generation process carries out many iterations with very small improvements to objective function value of the RMP. Thus it takes relatively longer times to prove optimality of the current solution. This is called the "tailing-off" effect. We can reduce this effect by stopping the column generation procedure earlier by proving optimality of the current solution. To achieve this we provide an upper bound (since the original

problem is a maximization problem). If the upper bound is less than the best known integer solution value then we can terminate the column generation procedure at the node and fathom the corresponding node without risk of missing the optimum.

Lasdon [47] provides a lower bound calculation for the master problem from the current objective value and the reduced costs obtained by solving the sub-problems. We follow a similar method, but unlike Lasdon our RMP is a maximization problem and hence the bound which we get is in fact an upper bound to the Master Problem (MP). Also, we have an additional variable $U_j$ which is non-decomposable; as such it is not a part of the sub-problem solution. We now discuss the computation of the upper bound.

Proposition 4-1: Given that $Z_{RMP}^{LP}$ is the current objective function value of the RMP at optimality, then the upper bound to the optimal objective value for the MP is given by $Z_{RMP}^{LP}$- $\left[min\left(\sum_j\left(min\ Z_{SP}^{j}\right),\ 0\right)\right]$

Proof:

The proof is presented in matrix notations. We use bold-face capital letters to represent variables and bold-face lower-case letters to represent parameters and dual values.

$$\boldsymbol{qU - cx\lambda - wb = qU - cx\lambda - wx\lambda - \alpha(\lambda - U)} \tag{4-14}$$

$$\boldsymbol{qU - cx\lambda - wb = qU - \lambda[(c + w)x + \alpha\ ] + \alpha U} \tag{4-15}$$

From Equation (4-12) we know $(\boldsymbol{c}+\boldsymbol{w})\boldsymbol{x}+\boldsymbol{\alpha} = \sum_j(min\ Z_{SP}^j)$. Since $min\ Z_{SP}^j$ is the reduced cost of $j^{th}$ sub-problem, we consider only those that will improve the objective function value of RMP, hence $(\boldsymbol{c}+\boldsymbol{w})\boldsymbol{x}+\boldsymbol{\alpha}$ can be replaced by $min\left(\sum_j\left(min\ Z_{SP}^j\right),0\right)$.

$$\boldsymbol{qU}-\boldsymbol{cx\lambda}-\boldsymbol{wb} \leq \boldsymbol{qU}-\sum_{j\in J}\sum_{k\in K_j}\lambda_j^k\left[min\left(\sum_j\left(min\ Z_{SP}^j\right),0\right)\right]+\boldsymbol{\alpha U} \tag{4-16}$$

Rearranging the terms in Equation (4-16), we get,

$$\boldsymbol{qU}-\boldsymbol{cx\lambda} \leq \boldsymbol{qU}+\boldsymbol{wb}+\boldsymbol{\alpha U}-\sum_{j\in J}\sum_{k\in K_j}\lambda_j^k\left[min\left(\sum_j\left(min\ Z_{SP}^j\right),0\right)\right] \tag{4-17}$$

The dual objective function value of the RMP is given by $\boldsymbol{qU}+\boldsymbol{wb}+\boldsymbol{\alpha U}$, which is equal to the objective function value of the primal RMP at optimality. We can re-write equation (4-17) as,

$$\mathbf{qU}-\mathbf{cx\lambda} \leq Z_{RMP}^{LP}-\sum_{j\in J}\sum_{k\in K_j}\lambda_j^k\left[min\left(\sum_j\left(min\ Z_{SP}^j\right),0\right)\right] \tag{4-18}$$

### 4.5.4. Node Selection

For searching the branch and bound tree we use three strategies, namely, Depth first Search (DFS), Best First Search (BeFS) and a combination of depth first and best first strategy which we denote by (DFS+BeFS). In DFS strategy when exploring a particular node, we form two child nodes and select the node with the best bound for exploration. We continue this till we find an integer solution and then backtrack to the nodes which

are unexplored. In BeFS strategy we search for the node with the best bound in the complete B&B tree for exploration. In DFS+BeFS strategy we try to combine the first and second strategy. We begin with DFS strategy and after finding an integer solution we implement BeFS so as to select an unexplored node having the best bound in the B&B tree and thereafter continue with DFS.

### 4.6. Experimentation

The research problem under consideration does not have benchmark problems and solutions. Hence, we compare the results from Branch & Price algorithm to the best known integer solution provided by a commercial solver for the original formulation. For it is seen from some initial experimentation with the original formulation that the solution time depends on the number of jobs, number of operations in each job, and number of time periods considered in the planning horizon. The length of the planning horizon and the number of jobs and operations determine the complexity of the problem. As mentioned earlier, if we consider a very long planning horizon and relatively few jobs, the problem is trivial to solve, because we will have enough capacity to accept and schedule all the available jobs in regular time. Thus, we control the length of the planning horizon indirectly by introducing the demand-to-capacity ratio (DC ratio) explained in Section 3.5. The objective is to obtain an experimental range for the DC ratio for which the commercial solver yields relatively good results in the stipulated time. The pilot experiment which we conducted for this purpose is explained in Section 4.6.1.

### 4.6.1. Pilot experiment

The experimental setup is provided in Table 4-1. We consider a full factorial experimental design. The number of operations per job, and the number of resources is kept constant at 5 and 3 respectively. The processing times are generated randomly for each replication from a discrete uniform distribution. We have two replications for each factor-level combination. The DC ratio is varied from 0.2 to 2.2 with a step size of 0.4 as shown in the experimental setup. The time periods obtained by using these DC ratios varied from 2 to 144. The due-date for each job is randomly generated within 60% to 100% of the planning horizon. The regular time cost for each resource is randomly generated from a uniform distribution between 20 and 80. The ratio of regular time to over time cost is 1:1.5. The sales price for each job is decided using Equation 4-19.

**Table 4-1. Pilot Experiment**

| Factors | Levels |
|---|---|
| Number of jobs | 3, 10 and 15 |
| Number of operations | 5 |
| DC ratio | 0.2, 0.6, 1.0, 1.4, 1.8, and 2.2 |
| Number of resources | 3 |
| Processing time distribution | DU[2,8] |

$$q_j = \sum_{o \in O_J} p_{jor} c_{r,s=1} * (1 + (0.7 * U[0,1])) \quad (4\text{-}19)$$

The problem instances are solved for the original formulation by using a commercial solver CPLEX 10.1. Each problem instance is allowed to be run either till the optimality

of the solution is proved or for 1800 seconds, whichever is the first. We report the time taken to solve the problem, and the absolute and relative gap. The relative gap is defined by CPLEX as given in Equation (4-20), where $Z_{IP}$ is the best known integer solution and the Bound is the best bound in the branch and bound tree.

$$\text{Relative gap in \%} = \left|\frac{Z_{IP}\text{-LP Bound}}{Z_{IP}}\right| * 100 \quad (4\text{-}20)$$

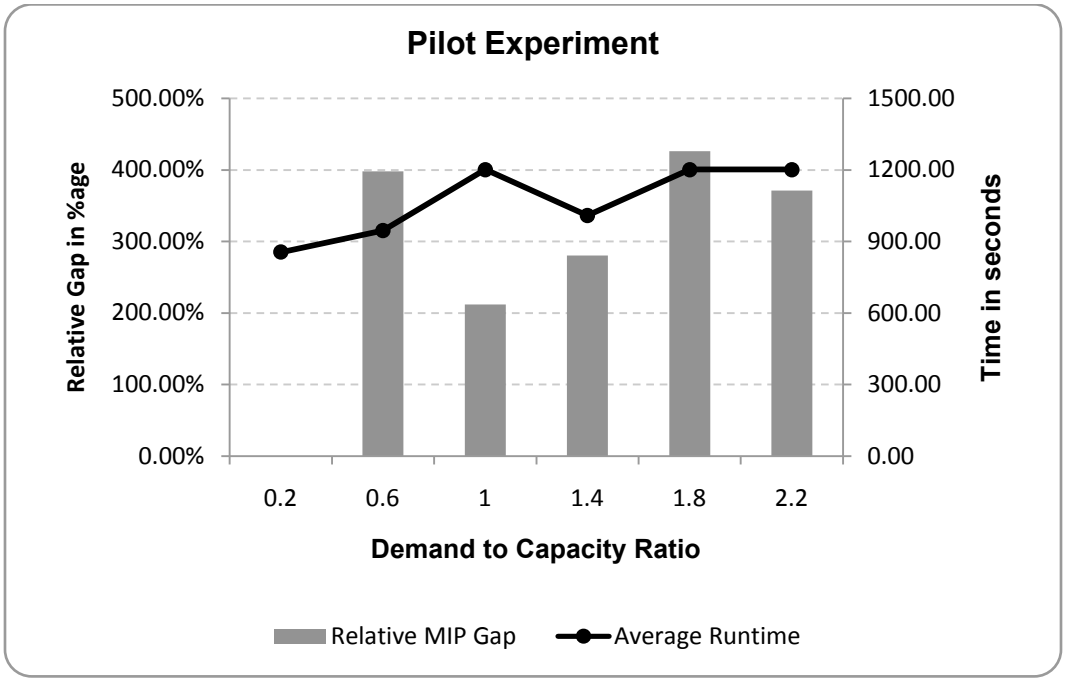



**Figure 4-9. Summary of results for Pilot Experiment**

When DC ratio is 0.2, we have enough capacity to accept all the orders and schedule them in regular time, hence the problem is trivial to solve. It is seen that the solution quality of CPLEX is relatively better for problem instances with a DC ratio of 1.0. Hence

for further experimentation to assess the solution quality of the B&P algorithm we generate a set of problem instances with a DC ratio as 0.8, 1.0 and 1.2, by comparing them with CPLEX.

### 4.6.2. Experimental setup

As discussed in Section 4.6.1, we have three levels for the DC ratio, namely 0.8, 1.0 and 1.2. The complete experimental setup (Experiment D) to assess the solution quality of the B&P algorithm is presented in Table 4-1. We conduct a full factorial experiment for the different factors and their respective levels with three replications. The number of resources is fixed to 3 for problem instances with 3 and 5 operations per job, and to 5 for instances with 8 and 10 operations per job. The due-dates for each job, the regular time and overtime costs and the sales price are generated as discussed in Section 4.6.1.

**Table 4-2. Experiment D setup for assessing the quality of B&P Algorithm**

| Factors | Levels |
|---|---|
| Number of jobs | 3, 5, 8 and 10 |
| Number of operations | 3, 5, 8 and 10 |
| DC ratio | 0.8, 1.0 and 1.2 |
| Processing time distribution (hours) | DU[4,16] |

### 4.6.3. Solution quality of Branch and Price Algorithm

In this section we first show that column generation provides tighter bounds than LP relaxation of the original problem at root node. Then we discuss the relative performance of the three node selection strategies implemented in BPS1 with CPLEX results as the

benchmark. CPLEX was not able to provide feasible solutions to all the problem instances; hence we report the number of instances that CPELX could provide a feasible solution and the number instances solved to optimality. We also present the relative gap reported by CPLEX at the end of 1800 seconds and the average runtime in seconds. We empirically show that BPS1 with DFS+BeFS as the node selection strategy proves to be the best solution approach with solution quality and runtime as the measures of performance. Finally we show the efficiency of the sub-problem solution strategy, which is a very important factor in successfully implementing a branch-and-price algorithm.

Figure 4-10 graphically shows the percentage gap between the bounds obtained by LP relaxation of the original problem and bounds obtained from column generation at the root node of the branch and bound tree. The X-axis consists of two rows, the first row gives the number of operations for the number of jobs in the second row. The Y-axis gives the percentage gap between the two bounds. A positive gap means that the objective value obtained from the column generation was lesser than LP relaxation. For a maximization problem this means that we found tighter bounds using column generation. On an average we find that the percentage gap between the two bounds is anywhere from 2% to 10%. In some instances especially with 3 and 5 jobs having 8 and 10 operations each, the gaps are relatively larger. This can be attributed to the length of the planning horizon where very few jobs or none could be accepted in reality because of the due-date constraint which is implicit in column generation but yields a highly fractional solution in LP relaxation. A specific example is for the instance with 3 jobs, 10 operations and planning horizon of 6 days. The LP relaxation yields an objective function value of

1109.797 with all jobs being selected, but in the integer solution we find that none of the jobs could be processed because they violated their due-dates and the integer solution had an objective function value of 0.

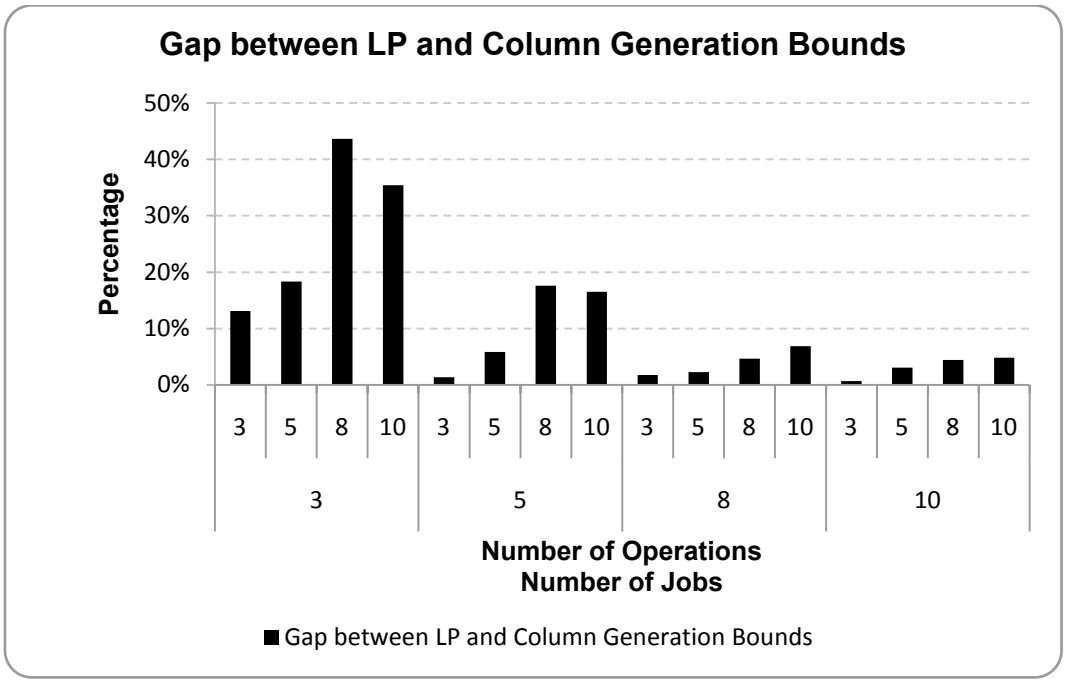



**Figure 4-10. Percentage gap between LP and column generation bounds**

Figure 4-11 shows the time taken to reach the root node solution for the LP relaxation of the original problem and column generation. We can easily verify from this graph that column generation takes comparatively lesser time to determine the root node solution of the branch and bound tree as compared to the LP relaxation of the original problem, as the problem size increases.

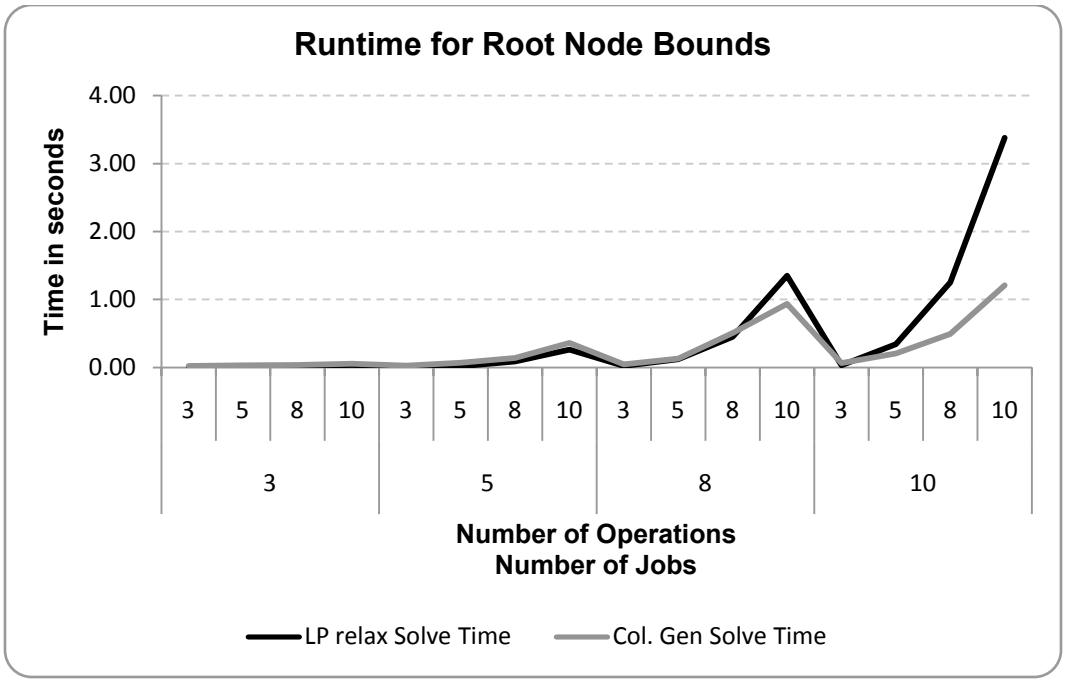



**Figure 4-11. Time to reach root node in LP relaxation and column generation**

For BPS1 procedure we experimented with three different node selection strategies as explained in Section 4.5.4. In Table 4-3, we report the number of optimal solutions (A) obtained by CPLEX and B&P. We also report the relative gap in percent (B) computed using Equation 4-21 between the best known integer solution $Z_{IP}$ provided by the solution method and the best bound computed in the branch and bound tree $Z_{RMP}^{BUB}$. We allow BPS1 to run till it proves optimality or a maximum of 900 seconds, whichever comes first. The improvement made by BPS1 over CPLEX is shown in Table 4-4. For problem instances with less than or equal to 5 jobs CPLEX yields marginally better results than BPS1. But as the number of jobs increase, BPS1 shows huge improvements over CPLEX. From these results we can see that CPLEX cannot provide feasible solutions for some problem instances with high number of jobs and operations. As the number of operations increase,

the problem complexity increases, and hence CPLEX cannot provide optimal solutions in the stipulated 1800 seconds. Also the relative gap increases with this increase in number of operations. A side-by-side comparison between the relative gap of the four solution strategies shows that BPS1 with DFS and DFS+BeFS consistently provides low relative gap and relatively more number of optimal solutions as compared to CPLEX. BPS1(BeFS) yields poor quality solutions for problems with more than 3 jobs as compared to the other node selection strategies of BPS1. On an average the relative gap for BPS1 with DFS and DFS+BeFS is always less than 4%; which means that the solutions provided by these two strategies is within 4% of the optimum. On the other hand, the solutions obtained from CPLEX can be on an average as far as 200% and up to 1200% away from the optimum.

$$\text{Relative gap in \%} = \frac{Z_{IP} - Z_{RMP}^{BUB}}{Z_{IP}} * 100 \qquad (4\text{-}21)$$

The improvements made by BPS1 over CPLEX are shown in Table 4-4. For problem instances with less than or equal to 5 jobs CPLEX yields marginally better results than BPS1. But as the number of jobs increase, BPS1 shows huge improvements over CPLEX.

**Table 4-3. Number of optimal solutions obtained and relative gap**

| | | | (A) Number of Optimal Solutions<br>(B) Relative gap in % | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Jobs | Opts. | # of Inst. solved by CPLEX | CPLEX | | BPS1 | | | | | |
| | | | | | DFS | | BeFS | | DFS +BeFS | |
| | | | A | B | A | B | A | B | A | B |
| 3 | 3 | 9 | 9 | 0.00 | 9 | 0.00 | 9 | 0.00 | 9 | 0.00 |
| | 5 | 9 | 9 | 0.01 | 8 | 0.22 | 9 | 0.00 | 7 | 0.27 |
| | 8 | 9 | 9 | 0.00 | 9 | 0.00 | 9 | 0.00 | 8 | 0.14 |
| | 10 | 9 | 7 | 3.94 | 8 | 2.31 | 9 | 7.83 | 6 | 0.63 |
| Average | | | | 0.99 | | 0.63 | | 1.96 | | 0.26 |
| 5 | 3 | 9 | 9 | 0.00 | 9 | 0.00 | 9 | 0.00 | 8 | 0.02 |
| | 5 | 9 | 4 | 4.39 | 6 | 0.60 | 6 | 4.25 | 6 | 0.41 |
| | 8 | 9 | 0 | 14.51 | 2 | 10.3 | 4 | 20.89 | 0 | 9.03 |
| | 10 | 9 | 0 | 22.86 | 1 | 3.06 | 1 | 77.71 | 0 | 2.69 |
| Average | | | | 10.44 | | 3.42 | | 25.71 | | 3.04 |
| 8 | 3 | 9 | 5 | 1.52 | 7 | 0.48 | 5 | 6.00 | 9 | 0.00 |
| | 5 | 9 | 0 | 5.24 | 3 | 0.96 | 0 | 35.66 | 4 | 0.58 |
| | 8 | 9 | 0 | 29.62 | 0 | 4.51 | 0 | 29.81 | 0 | 3.96 |
| | 10 | 8 | 0 | 176.86 | 0 | 4.71 | 0 | 39.27 | 0 | 2.73 |
| Average | | | | 49.76 | | 2.61 | | 27.35 | | 1.79 |
| 10 | 3 | 9 | 6 | 0.52 | 8 | 0.05 | 1 | 21.63 | 8 | 0.05 |
| | 5 | 9 | 1 | 5.02 | 5 | 0.37 | 0 | 22.31 | 5 | 0.16 |
| | 8 | 9 | 0 | 60.79 | 0 | 2.37 | 0 | 30.61 | 0 | 1.54 |
| | 10 | 5 | 0 | 1269.25 | 0 | 3.82 | 0 | 25.8 | 0 | 3.40 |
| Average | | | | 216.98 | | 1.38 | | 24.89 | | 1.02 |
| Grand Average | | | | 65.44 | | 2.02 | | 19.78 | | 1.54 |

It can be seen from Table 4-3 and Table 4-4 that for problems with less number of jobs, DFS and BeFS performs better than DFS+BeFS. But as the number of jobs increase there

is no significant difference between DFS and DFS+BeFS, while BeFS provides no optimal solutions and on an average performs very poorly as compared to the other two strategies.

**Table 4-4. BPS1 solution improvement over CPLEX**

| | | Improvement over CPLEX in % | | |
|---|---|---|---|---|
| | | BPS1 | | |
| Jobs | # of Opts. | DFS | BeFS | DFS +BeFS |
| 3 | 3 | 0.00 | 0.00 | 0.00 |
| | 5 | -0.05 | 0.00 | 0.00 |
| | 8 | 0.00 | 0.00 | 0.00 |
| | 10 | -1.51 | -4.44 | -0.04 |
| Average | | -0.39 | -1.11 | -0.01 |
| 5 | 3 | 0.00 | 0.00 | 0.00 |
| | 5 | -0.34 | -3.56 | -0.13 |
| | 8 | -4.18 | -11.82 | -3.24 |
| | 10 | 4.16 | -29.56 | 4.43 |
| Average | | -0.09 | -11.23 | 0.26 |
| 8 | 3 | -0.46 | -4.89 | 0.00 |
| | 5 | 0.80 | -20.73 | 1.18 |
| | 8 | 20.54 | -0.73 | 20.99 |
| | 10 | 139.84 | 84.59 | 144.34 |
| Average | | 37.33 | 12.56 | 38.69 |
| 10 | 3 | 0.00 | -16.36 | 0.00 |
| | 5 | 1.13 | -14.85 | 1.35 |
| | 8 | 50.79 | 23.46 | 52.13 |
| | 10 | 1175.60 | 1032.22 | 1178.07 |
| Average | | 198.29 | 159.10 | 199.11 |
| Grand Avg. | | 54.93 | 36.59 | 55.65 |

MILP problems are hard to solve using conventional branch and bound procedures. This is evident from our previous results in Sections 3.5, 4.6.1 and also from Table 4-5 where we show the average runtimes for different problem instances. We can observe that BPS1 is comparatively faster than CPLEX and at the same time provides better quality solutions. The DFS and DFS+BeFS strategies are comparable in terms of improvement over CPLEX, but on an average DFS+BeFS terminates earlier than DFS.

It is typical for a decomposition solution approach to find good quality feasible solutions early on in the solution process. This is true for the problem under consideration where the best integer solution was found much earlier than it took to terminate the B&P procedure. Table 4-6 summarizes the average time taken to find the best solution, after which no improvement was made. These results also suggest that BPS1(DFS+BeFS) finds solutions of comparable quality as BPS1(DFS) but quite early in the B&P process. Furthermore, BPS1(BeFS) makes no improvement over the initial solution for larger problem instances. Since BPS1(DFS+BeFS) performs the best with respect to the solution quality and runtime, for the rest of the chapter we will restrict our discussion to this strategy.

**Table 4-5. Average runtime for CPLEX and BPS1**

| | | Average runtime in seconds | | | |
|---|---|---|---|---|---|
| | | | BPS1 | | |
| Jobs | Opts. | CPLEX | DFS | BeFS | DFS +BeFS |
| 3 | 3 | 0.04 | 0.03 | 0.03 | 0.33 |
| | 5 | 8.34 | 100.44 | 1.00 | 4.33 |
| | 8 | 81.79 | 1.43 | 1.51 | 0.54 |
| | 10 | 537.74 | 133.12 | 105.39 | 102.08 |
| Average | | 156.98 | 58.75 | 26.98 | 26.82 |
| 5 | 3 | 0.69 | 23.52 | 8.68 | 0.32 |
| | 5 | 1181.86 | 319.15 | 458.18 | 210.50 |
| | 8 | 1800.59 | 708.87 | 542.98 | 483.20 |
| | 10 | 1800.23 | 805.60 | 801.41 | 800.67 |
| Average | | 1195.84 | 464.28 | 452.81 | 373.67 |
| 8 | 3 | 955.33 | 200.63 | 443.33 | 0.89 |
| | 5 | 1800.15 | 602.08 | 839.96 | 502.71 |
| | 8 | 1800.14 | 900.46 | 900.35 | 900.45 |
| | 10 | 1800.05 | 900.38 | 900.33 | 900.38 |
| Average | | 1582.88 | 643.76 | 767.30 | 566.84 |
| 10 | 3 | 665.43 | 103.84 | 792.90 | 101.87 |
| | 5 | 1783.29 | 408.87 | 892.04 | 408.61 |
| | 8 | 1800.04 | 900.40 | 900.19 | 900.43 |
| | 10 | 1800.12 | 900.23 | 900.44 | 900.76 |
| Average | | 1476.23 | 538.10 | 867.76 | 537.56 |
| Grand Avg. | | 1088.79 | 421.44 | 512.24 | 370.21 |

**Table 4-6. Time to Best Integer Solution for BPS1 (Experiment)**

| | | (A)Avg. time to Best Integer Solution for BPS1 in seconds<br>(B) Improvement over Initial Solution in % | | | | | |
|---|---|---|---|---|---|---|---|
| | | DFS | | BeFS | | DFS+BeFS | |
| Jobs | Opts. | A | B | A | B | A | B |
| 3 | 3 | 0.03 | 31.67 | 0.02 | 31.67 | 0.32 | 31.67 |
| | 5 | 19.64 | 16.39 | 0.89 | 16.44 | 4.19 | 16.44 |
| | 8 | 1.23 | 137.52 | 1.51 | 137.52 | 0.45 | 137.52 |
| | 10 | 38.07 | 46.05 | 2.79 | 41.18 | 2.09 | 48.49 |
| Average | | 14.74 | 57.91 | 1.3 | 56.7 | 1.76 | 58.53 |
| 5 | 3 | 22.55 | 21.3 | 8.67 | 21.3 | 0.15 | 21.3 |
| | 5 | 44.12 | 26.65 | 156.69 | 22.91 | 9.52 | 26.86 |
| | 8 | 112.51 | 52.16 | 23.67 | 40.74 | 131.57 | 53.51 |
| | 10 | 158.62 | 84.49 | 0.89 | 10.36 | 126.63 | 84.84 |
| Average | | 84.45 | 46.15 | 47.48 | 23.83 | 66.97 | 46.63 |
| 8 | 3 | 5.25 | 19.81 | 43.28 | 14.36 | 0.89 | 20.37 |
| | 5 | 184.11 | 34.51 | 0 | 0 | 83.44 | 34.96 |
| | 8 | 333.64 | 24.59 | 0 | 0 | 142.51 | 25.16 |
| | 10 | 210.79 | 33.04 | 0 | 0 | 211.23 | 35.59 |
| Average | | 182.66 | 27.84 | 11.13 | 3.69 | 106.61 | 28.83 |
| 10 | 3 | 60.57 | 22.53 | 71.19 | 14.36 | 2.09 | 22.53 |
| | 5 | 70.36 | 21.88 | 0 | 0 | 36.25 | 22.12 |
| | 8 | 183.91 | 27.58 | 0 | 0 | 258.33 | 28.15 |
| | 10 | 463.43 | 20.56 | 0 | 0 | 492.07 | 21.17 |
| Average | | 160.96 | 23.46 | 20.02 | 0.27 | 160.32 | 23.88 |
| Grand Average | | 108.74 | 39.36 | 20.05 | 21.85 | 81.55 | 39.99 |

Figure 4-12 graphically presents the comparison of total time taken to the time taken to find the best solution for BPS1 with DFS+BeFS strategy.

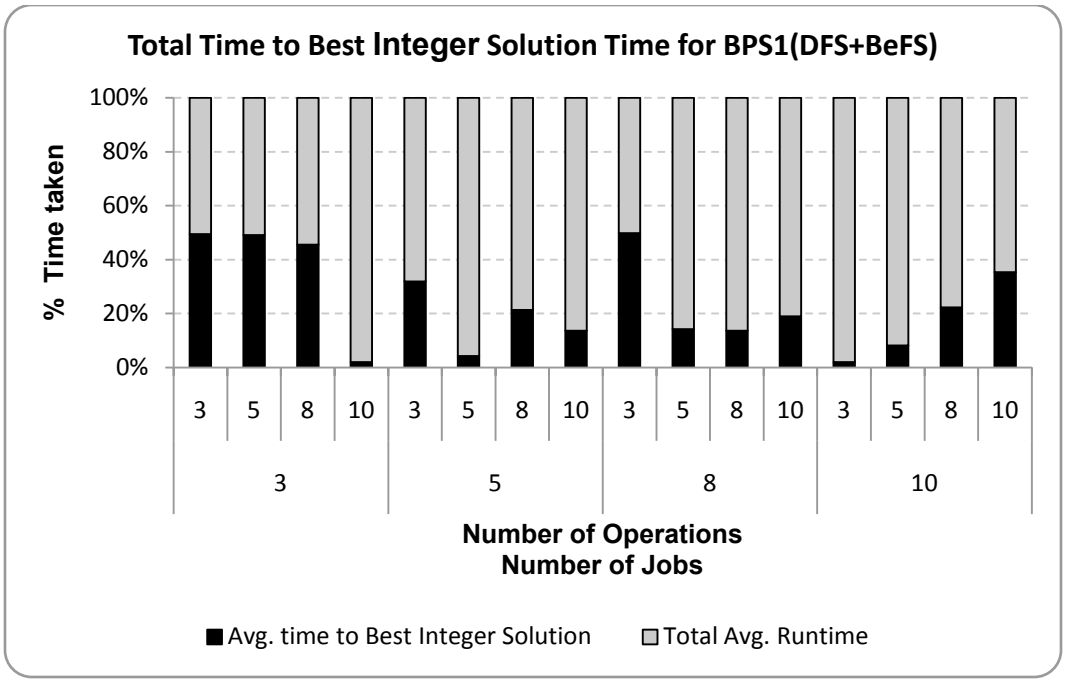



**Figure 4-12. Comparison of total time and time taken to find best integer solution**

These results are summarized graphically in Figure 4-13. We can see that for problems with lesser number of jobs and operations, CPLEX can provide optimal results and hence the improvements are zero. In some instance CPLEX actually performs better than BPS1 in solution quality, as BPS1 is unable to find optimal solution in the allotted 900 seconds, although the average runtime of CPLEX is much higher that BPS1 as we terminate CPLEX after 1800 seconds. Typically, CPLEX can solve most of the 3 operation instances with higher number of jobs to optimality. As the number of jobs and number of operations per job increase BPS1 consistently provides better results at a much lesser computational time. This is corroborated by the main effects plot and the interaction plots. In 10 job-10 operation instance BPS1 shows an improvement over CPLEX of close to 1178%. The improvement made by BPS1 decreases as the DC ratio increases from 0.8

to 1.2. This corroborates the results from Section 4.6.1 where we found that it was more difficult to solve problems with DC ratio of 0.6 than those with 1.0.

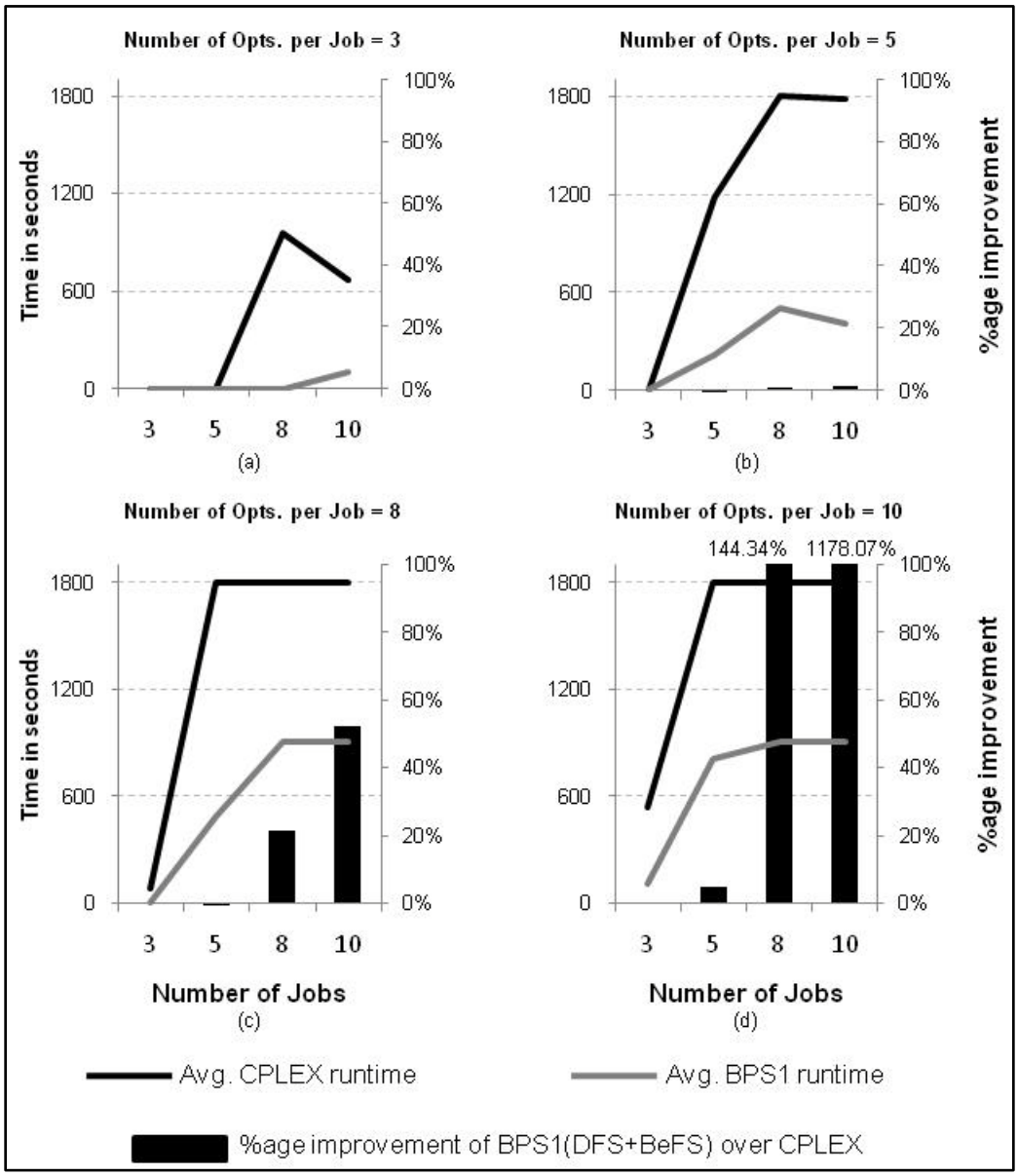



**Figure 4-13. Comparative graphs of BPS1 with CPLEX**

Another important aspect of column generation is the number of times a subproblem needs to be solved and the efficiency in solving it. Table 4-7 shows maximum times a sub-problem was solved for each combination of job and operation. We see that for a sub-problem size of 10 operations with a planning horizon of 30 days and on an average 100 hours of processing time per job take at most 0.00395 seconds to solve the sub-problem. Table 4-8 shows the average maximum time to solve a sub-problem and average percentage time spent in solving the sub-problems for a problem instance with respect to the number of operations and the DC ratio.

**Table 4-7. Maximum number of times sub-problem solved**

| | Number of Operations per job | | | |
|---|---|---|---|---|
| Number of Jobs | 3 | 5 | 8 | 10 |
| 3 | 94 | 108580 | 7389 | 95042 |
| 5 | 80828 | 136190 | 156880 | 138331 |
| 8 | 203608 | 209408 | 157189 | 142188 |
| 10 | 234384 | 260582 | 180230 | 124804 |

**Table 4-8. Runtime analysis for sub-problem solution**

| | Average Maximum time to solve a single sub-problem (sec.) | | | % time spent in solving sub-problems | | |
|---|---|---|---|---|---|---|
| | DC Ratio | | | DC Ratio | | |
| Operations | 0.8 | 1 | 1.2 | 0.8 | 1 | 1.2 |
| 3 | 0.00046 | 0.00042 | 0.00034 | 44.41 | 47.75 | 28.84 |
| 5 | 0.00192 | 0.00116 | 0.00097 | 56.32 | 38.83 | 30.34 |
| 8 | 0.00255 | 0.00192 | 0.00178 | 34.89 | 27.57 | 25.12 |
| 10 | 0.00395 | 0.00321 | 0.00246 | 36.04 | 33.70 | 29.11 |

# 5. BRANCH AND PRICE HEURISTIC & APPROXIMATION ALGORITHMS

BPS1 guarantees an optimal solution. However, it takes a long time to prove optimality, which is common to decomposition procedures. Furthermore, since we are fixing one original variable at a time in BPS1, the branch and bound tree can grow exponentially. To overcome this problem we propose a branch and price heuristic in Section 5.1. In Section 5.2 we propose approximation algorithms for the proposed branch and price algorithms. In Section 5.3 we present comparative analysis for the various branch and price strategies discussed.

## 5.1. Branch and Price Strategy 2 (BPS2)

For BPS2, unlike in BPS1 instead of fixing a single time period and source in each child node, we introduce time windows, during which operations $o$ and $o$+1 are not allowed to be scheduled. This proposed method for fixing original variables gives us an approximate solution; but is intended to reduce the computational time. In $\theta_j'$, we can look at the violation as two mutually exclusive events. The first is keeping the start time of operation $o$+1 as is. In that case, operation $o$ has to be completely processed by the time period and source in which operation $o$+1 has started in $\theta_j'$. The second event is that we keep the end time of operation $o$ as is, so in such a case the earliest we can start processing operation $o$+1 is from the time operation $o$ ends. Thus we can create the time windows during which we cannot schedule the two operations. In the first child node we place the restriction that operation $o$ cannot be processed after source ($ss_{o+1}$) in time period ($st_{o+1}$), since this is the start time of operation $o$+1. Also, since we want to keep this start time as

is, we can have an additional restriction that we cannot process operation $o$+1 before this time/source instance. In the other child node we place a restriction that operation $o$ cannot be scheduled after time period ($et_o$) and source ($es_o$) onwards along with operation $o$+1 not to be scheduled before this time period and source. Figure 5-1 shows the way in which we can branch for $\theta_j'$ in the previous example (Section 4.5.1) using BPS2.

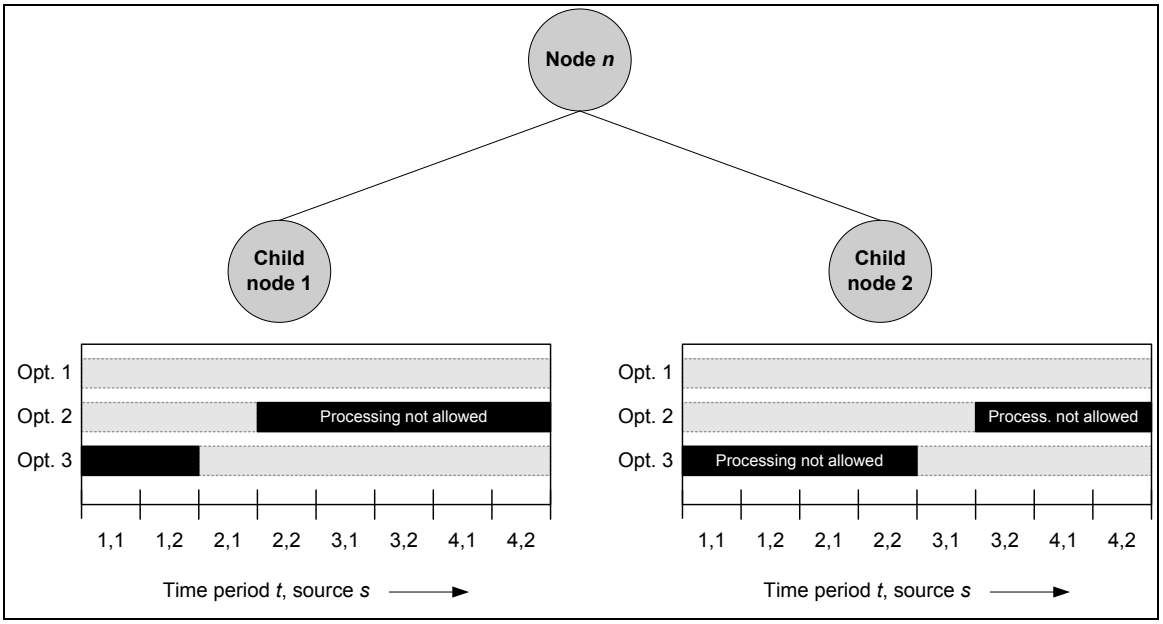



**Figure 5-1. Branching in BPS2**

We implement these restrictions of disallowing operations to be processed during certain time periods and sources in the sub-problem network by fixing the processing arc costs for the corresponding time periods and sources to $+\infty$.

### 5.2. Approximation Algorithms for Branch & Price

In decomposition algorithms, we have tighter linear programming bounds as compared to the linear relaxations of the aggregated or non-decomposed formulation, and we usually find good feasible solutions early on in the solution process. Due to this reason, truncated tree search algorithms may provide very good approximation algorithms. In truncated tree search algorithms the number of nodes evaluated in the solution process is reduced according to some pre-specified scheme [48].

In the approximation algorithm, which we propose, we introduce a optimality tolerance $\gamma$, such that a node is fathomed if $Z_{RMP}^{LP} \leq (1+\gamma) Z_{IP}$, where $Z_{IP}$ is the value of the best known integer solution.

We implement the approximation algorithms for both BPS1 and BPS2 with $\gamma$ value of 0.01 and 0.05. For $\gamma$ value of 0.0 we get the original BPS1 and BPS2 strategies. For being concise, we represent the name of the branching strategy followed by the optimality tolerance within round brackets. For example, BPS2(0.01) represents branch strategy BPS2 with an optimality tolerance $\gamma = 0.01$. We follow this convention in the following sections. Section 5.3 presents a comprehensive analysis of the different branch and price strategies along with the results for the approximation algorithms.

### 5.3. Comparative Analysis

In this section we present a comprehensive analysis of the different branch and price strategies along with the results for the approximation algorithms. In this section we show

a paired *t*-test analysis between BPS1 and BPS2 with $\gamma$ value of 0.00, and prove that there is no statistical difference between the solution qualities of the two strategies. Then we summarize the improvements made over CPLEX by the various B&P strategies in Table 5-3. In Table 5-4 we present the computational runtime required to solve the problem instances and in Table 5-5 we summarize the reductions in runtime by using BPS2 and the various approximation algorithms instead of BPS1(0.0). Finally we give some statistics pertaining to the size of the branch and bound tree, and number of columns generated for the various strategies. For all the above implementations we use Lagrangean bounds discussed in Section 4.5.3 to fathom the nodes in the branch and bound tree and DFS+BeFS strategy for node selection.

### 5.3.1. Comparing solution quality of BPS2 against BPS1

A paired *t*-test is conducted to conclude whether there is any statistical difference between the solution quality from BPS1 and BPS2. The null hypothesis is that there is no statistical difference in the results obtained by the two strategies at a 95% significance level. The null and alternate hypothesis can be states as follows:

Null Hypothesis $H_0$: $\mu_{BPS1(0.0)}=\mu_{BPS2(0.0)}$

Alternate Hypothesis $H_1$: $\mu_{BPS1(0.0)}\neq\mu_{BPS2(0.0)}$

**Table 5-1. Results of the paired-t test**

| | N | Mean | St. Dev | SE Mean |
|---|---|---|---|---|
| BPS1(0.0) | 139 | 5408 | 4171 | 354 |
| BPS2(0.0) | 139 | 5415 | 4164 | 353 |
| Difference | 139 | -7.4 | 259.9 | 22.0 |
| 95% CI for mean difference: (-51.0, 36.2) | | | | |
| T-Test of mean difference = 0 (vs not = 0): T-Value = -0.34 P-Value = 0.738 | | | | |

Since p-value is 0.738, we fail to reject the null hypothesis with a 95% confidence level. A test to check for equal variances was conducted. The 95% Bonferroni confidence intervals for standard deviations and the result from the Levene's Test is shown in Table 5-2. The null hypothesis is testing for equal variances are that they are the same. In the Levene's Test we get a p-value of 0.971, and hence we fail to reject the null hypothesis at a 0.05 significance level. We can infer the variances for BPS1(0.0) and BPS2(0.0) are equal

**Table 5-2. Test for Equal Variances**

| | N | Lower | St. Dev | Upper |
|---|---|---|---|---|
| BPS1(0.0) | 139 | 3674.03 | 4171 | 4171.30 |
| BPS2(0.0) | 139 | 3667.18 | 4164 | 4808.32 |
| Levene's Test (Any Continuous Distribution) | Test statistic = 0.00<br>p-value = 0.971 | | | |

From the above two tests we can conclude that there is no statistical difference between the objective function values of BPS1(0.0) and BPS2(0.0). This implies that the solution quality obtained from the two strategies is the same.

In Table 5-3, we show the percentage improvement made by the various B&P strategies over CPLEX.

**Table 5-3. Improvement over CPLEX**

| Jobs | Opts. | %age improvement over CPLEX | | | | | |
|---|---|---|---|---|---|---|---|
| | | BPS1 | | | BPS2 | | |
| | | $\gamma = 0.00$ | 0.01 | 0.05 | 0.00 | 0.01 | 0.05 |
| 3 | 3 | 0.00 | 0.00 | 0.00 | -0.12 | -0.12 | -0.12 |
| | 5 | 0.00 | -0.05 | -0.81 | -0.28 | -0.36 | -0.52 |
| | 8 | 0.00 | -0.04 | -0.83 | -0.13 | -0.13 | -0.92 |
| | 10 | -0.04 | -0.10 | -0.14 | -0.09 | -0.14 | -0.23 |
| Average | | -0.01 | -0.05 | -0.44 | -0.15 | -0.19 | -0.45 |
| 5 | 3 | 0.00 | 0.00 | -0.24 | -0.07 | -0.16 | -0.34 |
| | 5 | -0.13 | -0.35 | -1.24 | -0.18 | -0.38 | -0.80 |
| | 8 | -3.24 | -3.28 | -3.28 | 0.36 | 0.31 | -0.26 |
| | 10 | 4.43 | 4.16 | 3.96 | 2.36 | 2.22 | 1.70 |
| Average | | 0.26 | 0.13 | -0.20 | 0.62 | 0.50 | 0.07 |
| 8 | 3 | 0.00 | -0.04 | -1.69 | 0.00 | -0.07 | -1.65 |
| | 5 | 1.18 | 1.09 | -0.36 | 1.48 | 1.27 | -0.23 |
| | 8 | 20.99 | 20.94 | 20.51 | 21.65 | 21.14 | 19.52 |
| | 10 | 144.34 | 140.65 | 139.81 | 141.76 | 141.03 | 139.44 |
| Average | | 38.69 | 37.80 | 36.70 | 38.35 | 37.98 | 36.41 |
| 10 | 3 | 0.00 | -0.07 | -0.69 | 0.00 | -0.05 | -0.76 |
| | 5 | 1.35 | 1.28 | 0.12 | 1.47 | 1.46 | -0.22 |
| | 8 | 52.13 | 51.70 | 50.58 | 51.78 | 51.64 | 50.91 |
| | 10 | 1178.07 | 1178.07 | 1175.39 | 1200.37 | 1199.62 | 1170.61 |
| Average | | 199.11 | 198.95 | 197.72 | 202.53 | 202.36 | 196.95 |

For instances with 3 jobs we see that CPLEX performs marginally better than the B&P strategies. We see marginal improvement in the solution quality of B&P as the number of jobs increase. For 8 job and 10 job instances we see an average improvement of 35% and 200% respectively.

In Table 5-4 we show the computation runtime to solve the various problem instances by the different B&P strategies.

**Table 5-4. Runtime analysis of Branch and Price**

| Jobs | Opts. | Runtime in seconds | | | | | |
|---|---|---|---|---|---|---|---|
| | | BPS1 | | | BPS2 | | |
| | | $\gamma = 0.00$ | 0.01 | 0.05 | 0.00 | 0.01 | 0.05 |
| 3 | 3 | 0.33 | 0.03 | 0.03 | 0.03 | 0.03 | 0.03 |
| | 5 | 4.33 | 0.87 | 0.16 | 0.15 | 0.13 | 0.10 |
| | 8 | 0.54 | 0.34 | 0.30 | 0.18 | 0.14 | 0.09 |
| | 10 | 102.08 | 100.68 | 0.90 | 0.94 | 0.71 | 0.60 |
| Average | | 26.82 | 25.48 | 0.35 | 0.32 | 0.25 | 0.20 |
| 5 | 3 | 0.32 | 0.16 | 0.16 | 0.10 | 0.09 | 0.08 |
| | 5 | 210.50 | 102.33 | 0.70 | 0.56 | 0.41 | 0.31 |
| | 8 | 483.20 | 403.18 | 102.94 | 14.02 | 5.83 | 1.69 |
| | 10 | 800.67 | 633.56 | 205.96 | 360.15 | 308.88 | 110.18 |
| Average | | 373.67 | 284.81 | 77.44 | 93.71 | 78.80 | 28.07 |
| 8 | 3 | 0.89 | 0.82 | 0.31 | 0.34 | 0.29 | 0.16 |
| | 5 | 502.71 | 204.83 | 4.32 | 122.79 | 102.27 | 1.74 |
| | 8 | 900.45 | 802.12 | 145.02 | 807.37 | 711.64 | 155.66 |
| | 10 | 900.38 | 900.94 | 301.34 | 900.64 | 755.10 | 254.47 |
| Average | | 566.84 | 465.07 | 107.36 | 445.13 | 381.96 | 98.68 |
| 10 | 3 | 101.87 | 101.88 | 1.62 | 0.94 | 0.65 | 0.41 |
| | 5 | 408.61 | 33.88 | 16.63 | 21.73 | 7.40 | 4.28 |
| | 8 | 900.43 | 639.28 | 180.69 | 900.34 | 805.50 | 81.14 |
| | 10 | 900.76 | 780.15 | 487.35 | 900.81 | 900.95 | 391.71 |
| Average | | 537.56 | 339.88 | 132.10 | 400.34 | 369.59 | 85.35 |

BPS1(0.0) takes the most time to solve the problems. When the number of operations are less (3 and 5) BPS2(0.0) is faster than BPS1(0.0) and BPS1(0.01), but as the number of operations increase BPS1(0.01) is faster than BPS1 (0.0). The approximation algorithms with optimality tolerance of 0.05 are much faster than any other, while BPS2(0.05) performs the best in terms of runtime. Since BPS1(0.0) is slowest, we compute the reduction in runtime achieved by using the other B&P strategies. The results are shown in Table 5-5.

**Table 5-5. Reduction in runtime**

| Jobs | Opts. | %age reduction in runtime | | | | |
|---|---|---|---|---|---|---|
| | | BPS1 | | BPS2 | | |
| | | 0.01 | 0.05 | 0.00 | 0.01 | 0.05 |
| 3 | 3 | 22.74 | 22.81 | 30.35 | 25.17 | 13.27 |
| | 5 | 13.68 | 28.77 | 24.52 | 29.80 | 40.39 |
| | 8 | -40.30 | 7.29 | 10.64 | 15.22 | 14.33 |
| | 10 | 21.74 | 34.38 | 31.52 | 32.51 | 35.52 |
| Average | | 4.46 | 23.31 | 24.26 | 25.67 | 25.88 |
| 5 | 3 | 6.04 | 1.92 | 37.03 | 41.05 | 49.65 |
| | 5 | 25.25 | 40.97 | 62.46 | 66.31 | 66.00 |
| | 8 | 30.43 | 70.58 | 88.68 | 90.31 | 91.33 |
| | 10 | 24.34 | 71.89 | 50.86 | 59.82 | 84.45 |
| Average | | 21.52 | 46.34 | 59.76 | 64.37 | 72.86 |
| 8 | 3 | 6.81 | 47.95 | 48.31 | 54.39 | 67.85 |
| | 5 | 32.78 | 69.73 | 67.94 | 70.33 | 87.30 |
| | 8 | 10.92 | 83.89 | 10.34 | 20.97 | 82.71 |
| | 10 | -0.06 | 66.53 | -0.03 | 16.13 | 71.74 |
| Average | | 12.98 | 67.04 | 32.55 | 41.15 | 77.56 |
| 10 | 3 | -0.13 | 26.91 | 70.54 | 66.53 | 81.05 |
| | 5 | 41.58 | 54.30 | 79.57 | 81.27 | 87.19 |
| | 8 | 29.00 | 79.93 | 0.01 | 10.54 | 90.99 |
| | 10 | 13.39 | 45.90 | -0.01 | -0.02 | 56.50 |
| Average | | 21.91 | 52.49 | 42.22 | 44.53 | 81.74 |

For 10 job problems BPS2(0.05) can show on an average 81% reduction in runtime over BPS1(0.0). As seen from the improvements made over CPLEX BPS2(0.05) makes 196% improvement as opposed to BPS2(0.0) which makes 202.53% but at a much lesser computation overhead. This shows that the approximation algorithms are a viable alternative to the exact procedure.

Finally we present the maximum number of columns generated, maximum nodes formed and number of times the sub-problem is solved for the different number of jobs and operations. This information is helpful to analyze the size of the branch and bound tree, the effectiveness of the sub-problem solution approach and the memory requirements for the solution approach. It can be seen from Table 5-6 that there is no clear pattern as regards to the columns generated and the problem size. But for high number of jobs and operations per each job, the number of columns generated by BPS2 are more than BPS1. The maximum number of columns generated were for a 5 job 8 operation problem by BPS2(0.01).

The number of nodes in the branch and bound tree, as shown in Table 5-7 are lesser in BPS2 as compared to BPS1, which makes intuitive sense because in BPS2, we are setting a set of original variables to zero based on the concept of time windows, as compared to BPS1, where we are setting the value of a single original variable to zero. Hence, BPS1 takes more time because the search space is much bigger. Also the overhead of traversing the branch and bound tree to set the columns at each node in the branch to zero which

violates the current restrictions is much more when the number of nodes in the branch is more and hence the increase in computational time.

**Table 5-6. Number of columns generated in B&P**

| Jobs | Opts. | Maximum number of columns generated | | | | | |
|---|---|---|---|---|---|---|---|
| | | BPS | | | BPS2 | | |
| | | $\gamma = 0.0$ | 0.01 | 0.05 | 0 | 0.01 | 0.05 |
| 3 | 3 | 41 | 41 | 41 | 81 | 81 | 81 |
| | 5 | 1713 | 396 | 348 | 316 | 231 | 268 |
| | 8 | 851 | 851 | 376 | 498 | 447 | 253 |
| | 10 | 50546 | 49085 | 1064 | 2631 | 1350 | 400 |
| 5 | 3 | 9370 | 2972 | 388 | 437 | 379 | 191 |
| | 5 | 61797 | 68180 | 808 | 1197 | 500 | 412 |
| | 8 | 64480 | 62893 | 2904 | 61562 | 83342 | 2423 |
| | 10 | 64747 | 15366 | 7588 | 20085 | 6646 | 3851 |
| 8 | 3 | 1891 | 1172 | 1083 | 642 | 447 | 287 |
| | 5 | 64786 | 66269 | 48435 | 16900 | 7873 | 1614 |
| | 8 | 67009 | 65031 | 51689 | 74524 | 77517 | 71755 |
| | 10 | 63420 | 55750 | 44805 | 73850 | 76483 | 47446 |
| 10 | 3 | 51436 | 53432 | 1560 | 2321 | 1874 | 1536 |
| | 5 | 61202 | 63623 | 65702 | 67678 | 68219 | 70360 |
| | 8 | 59340 | 62094 | 54651 | 66367 | 65360 | 67664 |
| | 10 | 50688 | 52075 | 27938 | 61918 | 65165 | 63451 |

**Table 5-7. Size of branch and bound tree in B&P**

| Jobs | Opts. | Maximum nodes formed in the branch and bund tree | | | | | |
|---|---|---|---|---|---|---|---|
| | | BPS | | | BPS2 | | |
| | | $\gamma = 0.0$ | 0.01 | 0.05 | 0 | 0.01 | 0.05 |
| 3 | 3 | 15 | 15 | 15 | 19 | 19 | 19 |
| | 5 | 2327 | 691 | 85 | 49 | 37 | 19 |
| | 8 | 213 | 111 | 103 | 67 | 43 | 23 |
| | 10 | 6239 | 6097 | 135 | 161 | 127 | 99 |
| 5 | 3 | 325 | 75 | 95 | 43 | 33 | 35 |
| | 5 | 14627 | 6541 | 109 | 89 | 43 | 27 |
| | 8 | 9111 | 9057 | 9105 | 809 | 353 | 97 |
| | 10 | 6589 | 5245 | 4283 | 3557 | 3065 | 2027 |
| 8 | 3 | 261 | 261 | 161 | 67 | 57 | 31 |
| | 5 | 10845 | 11225 | 389 | 4161 | 4699 | 143 |
| | 8 | 5455 | 4179 | 2797 | 4725 | 2861 | 1541 |
| | 10 | 3939 | 3081 | 3047 | 3055 | 2865 | 1665 |
| 10 | 3 | 13921 | 13969 | 319 | 253 | 141 | 41 |
| | 5 | 9411 | 1771 | 849 | 959 | 321 | 209 |
| | 8 | 4915 | 5163 | 2555 | 2997 | 2903 | 1253 |
| | 10 | 3377 | 2233 | 2211 | 1981 | 1649 | 1061 |

# 6. CONCLUSIONS AND FUTURE WORK

This dissertation is concluded by summarizing the problem and the solution approach in Section 6.1. Section 6.2 states the contributions of this research followed by their significance. Lastly in Section 6.3 possible extensions of this research are outlined.

## 6.1. Summary

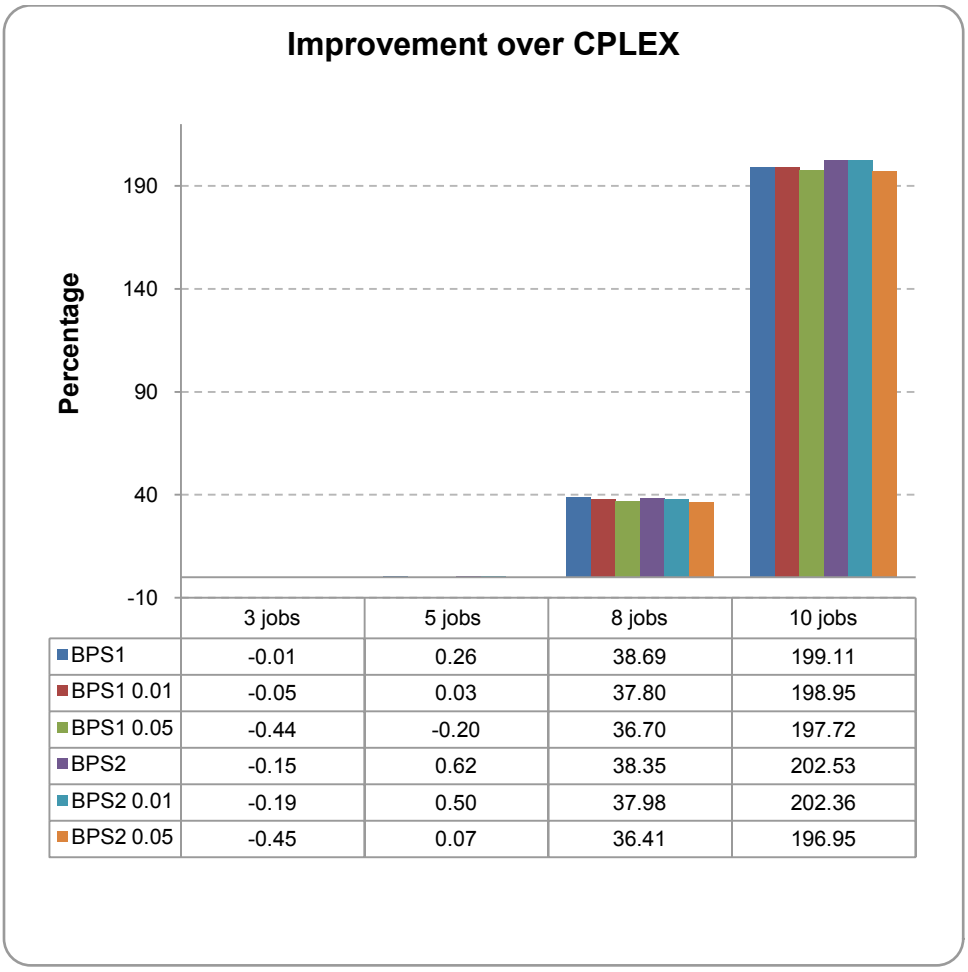
Integrating order acceptance and capacity planning provides tremendous opportunities to maximize the operational profits of make-to-order operations. This is done by selectively accepting jobs from the available pool of customer orders and simultaneously planning for their capacity. This integrated problem is difficult to solve and many researchers have tried to simplify the problem by planning for the bottleneck machines and solving the problem as a single machine problem. But in reality, the bottleneck is floating as it depends on the orders which are selected. Furthermore, capacity is not fixed since you can extend your capacities by adding overtime and outsourcing, which might be beneficial for improving the profits. Non-regular capacity has not been considered in any of the previous work done in the area of MTO order acceptance problem. In this dissertation we model the MTO operation as a job shop with multiple resources and recirculation. We consider regular capacity (regular shift) and non-regular capacity (overtime shift). The MTO operation receives customer orders or jobs each with a number of operations having linear precedence relationship. Typically order acceptance problems are solved on daily basis for short term capacity planning with a rolling planning horizon of 3 to 4 weeks. Hence the solution approach to this integrated problem

should be quick such that the decision maker can use it recursively not only to find the optimal set of orders and to allocate capacity but also to explore various other scenarios that would help in negotiating order due-dates, prices and be aligned with the organization's long-term business strategy. The goal of this dissertation is to develop such a tool.

In order to achieve this goal we first proposed a Mixed-Integer Linear Program (MILP) to model the research problem under consideration. Using the model we illustrate that integrating the two decisions of order acceptance and capacity planning can achieve our goal to maximize the operational profits. Since the proposed MILP had a block diagonal structure, and column generation can solve models with this structure efficiently, we proposed an exact branch and price algorithm (BPS1). We also develop approximate branching scheme (BPS2) and various approximation algorithms for the exact and approximate branching schemes.

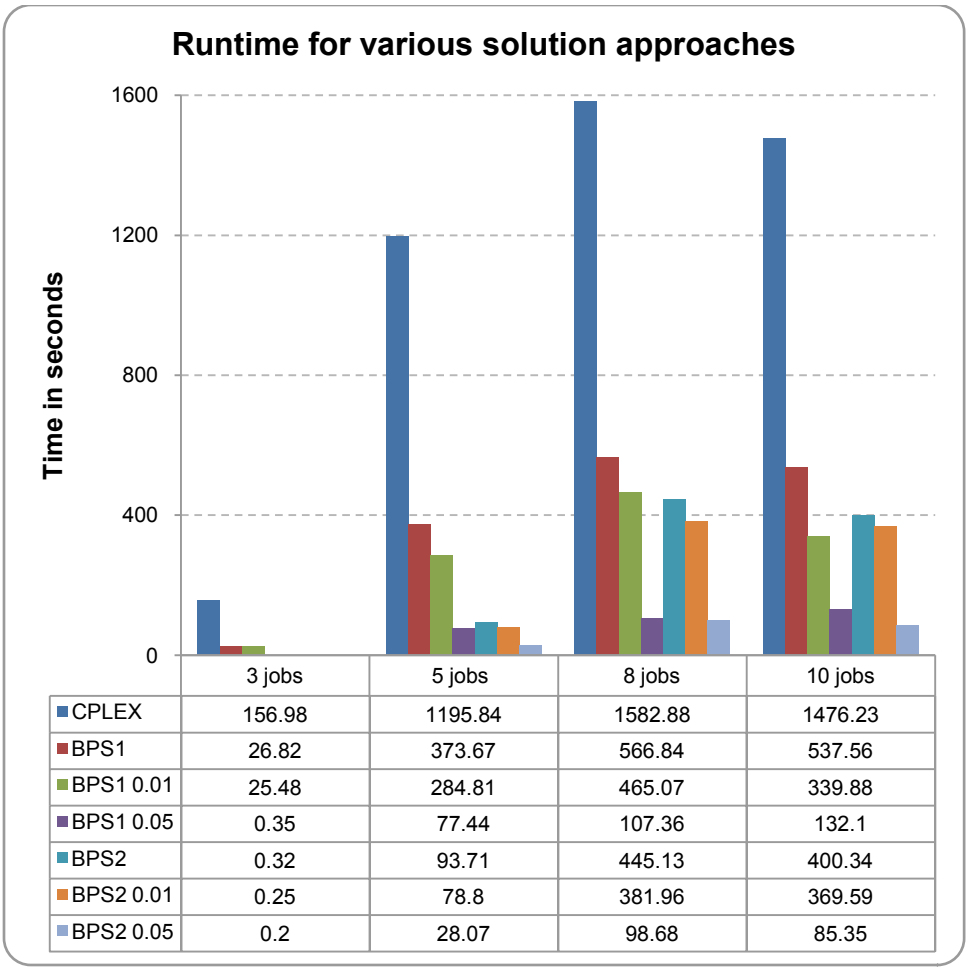
We show through experiments that the BPS1 and other approximation schemes perform better than the solution provided by the commercial solver, and can solve problems of sizes which can be typically found in the real-life applications. Figure 6-1 and Figure 6-2 graphically summarize the improvements made by B&P algorithms and the computational runtime of various solution approaches discussed in this dissertation. We observe that B&P performs 200% better than the results obtained from solving the MILP at a much lesser computational overhead as compared to a commercial solver CPLEX. BPS2(0.05) can solve, on an average, 10 jobs problems in 85 seconds and making 196%

improvements over CPLEX. B&P algorithms are significantly faster and solve problems in reasonable time, and thus can be utilized in a decision support system used on a daily basis to help make intelligent decisions in a MTO operation.




| | 3 jobs | 5 jobs | 8 jobs | 10 jobs |
|---|---|---|---|---|
| BPS1 | -0.01 | 0.26 | 38.69 | 199.11 |
| BPS1 0.01 | -0.05 | 0.03 | 37.80 | 198.95 |
| BPS1 0.05 | -0.44 | -0.20 | 36.70 | 197.72 |
| BPS2 | -0.15 | 0.62 | 38.35 | 202.53 |
| BPS2 0.01 | -0.19 | 0.50 | 37.98 | 202.36 |
| BPS2 0.05 | -0.45 | 0.07 | 36.41 | 196.95 |

**Figure 6-1. Average improvement in solution quality**




| | 3 jobs | 5 jobs | 8 jobs | 10 jobs |
|---|---|---|---|---|
| CPLEX | 156.98 | 1195.84 | 1582.88 | 1476.23 |
| BPS1 | 26.82 | 373.67 | 566.84 | 537.56 |
| BPS1 0.01 | 25.48 | 284.81 | 465.07 | 339.88 |
| BPS1 0.05 | 0.35 | 77.44 | 107.36 | 132.1 |
| BPS2 | 0.32 | 93.71 | 445.13 | 400.34 |
| BPS2 0.01 | 0.25 | 78.8 | 381.96 | 369.59 |
| BPS2 0.05 | 0.2 | 28.07 | 98.68 | 85.35 |

**Figure 6-2. Runtime for various solution approaches**

## 6.2. Contributions & Significance

The contributions of this dissertation are:

- Development of a mathematical model for maximizing the operational profits of a make-to-order operation by combining order acceptance with capacity planning at the sales stage in job-shop environment having multiple resources with regular time and overtime capacity.

- Development of a Branch and Price solution approach along with various approximation algorithms were proposed for solving this model within practical considerations of time-limit.

Existing literature on MTO with static job arrivals only considers order acceptance for single resource cases with regular capacity. However, most MTO operations have multiple resources in a job shop environment with regular and non-regular capacity like overtime and outsourcing available to them. These have not been addressed in the past research efforts. Hans [42] has developed a B&P solution approach for capacity planning in job shop with multiple resources and non-regular capacity, but he does not consider order acceptance. Ebben [7] has studied resource loading based order acceptance using simulation. In one of the approaches they used BPRL approach proposed by Hans [42] to schedule already accepted orders. This dissertation has made significant contribution to the scientific community by studying and modeling the MTO production system as a job shop with multiple resources having regular and overtime capacity. The branch and algorithm developed and implemented in this dissertation can solve larger problem sizes, typically found in MTO operations within practical time limits.

### 6.3. Future Work

Column generation is a decomposition approach and is efficient when the formulation exhibits a block diagonal structure. The MILP proposed in this dissertation exhibited this special structure and hence column generation was adopted to solve the MTO problem. There are other decomposition techniques in literature like Lagrangian relaxation and Benders' decomposition which are commonly used in solving large-scale optimization

problems. It could be worthwhile to explore the Lagrangian relaxation approach to solve the MTO order acceptance problem. Since the capacity constraint is the complicating or binding constraint, it could be relaxed and put in the objective function. Once this constraint is relaxed, a solution to the problem can be found out by generating schedules for individual jobs by using the sub-problem solution approach proposed in this dissertation. The Lagrange multipliers can be updated by looking at the capacity violations, and these multipliers can be used to find the new schedules from the sub-problem.

The problem under consideration is typically found in many practical applications. It lays the foundation for other complex problems of practical interest having variations to this basic problem. For example, we consider non-regular capacity as overtime, but many make-to-order operations have outsourcing options. Integrating outsourcing is another important variation to the problem we have considered. The solution approach proposed in this dissertation can be extended to cases where pre-emption is not allowed. In many other operations, instead of a single deliverable job, we may have a bigger assemblies made of smaller sub-assemblies. Hence, the precedence relations are much more complex although each sub-assembly will have linear precedence amongst their sub-operations or tasks. The solution approach proposed in this research forms the foundation for solving such complex problems.

One of the assumptions made here in this research is that all the parameters are known with certainty, however, in real-life there may be some uncertainties. In such case a

stochastic formulation is necessary, and hence, the solution approaches may have to be modified. There are B&P implementations for stochastic MILP's for other applications. Consequently, a stochastic version of the proposed B&P algorithm may have to be implemented. A scenario tree approach is commonly applied in the literature to tackle uncertainties. The size of the model (constraints, and decision variables) will increase as a result. A goal programming approach can be devised to relax the due date constraints to capture the capability to negotiate due dates for different customer orders.

# LIST OF REFERENCES

[1] Zijm, W. H. M., 2000, "Towards intelligent manufacturing," OR Spektrum, 22, 313-345.

[2] Chen, Chin-Sheng, 2006, "Concurrent engineering-to-order operation in the manufacturing engineering contracting industries," International Journal of Industrial and Systems Engineering, 1(1), 37-58.

[3] Jalora, Anshu. , August 2006, "Order acceptance and scheduling at a make-to-order system using revenue management," Dissertation/Thesis, Unpublished, Texas A&M University.

[4] Gallien, Jeremie, Tallec, Yann Le and Schoenmeyr, Tor, "A Model for Make-To-Order Revenue Management," Unpublished Material.

[5] Vollmann, T. E., Berry, W. L. and Whybark, D. C., 1997, "Manufacturing planning and control systems," Mc-Graw Hill.

[6] Slotnick, Susan A. and Morton, Thomas E., 2007, "Order acceptance with weighted tardiness," Computers & Operations Research, 34, 3029-3042.

[7] Ebben, M. J. R., Hans, E. W. and Weghuis, Olde F. M., 2005, "Workload based order acceptance in job shop environments," OR Spectrum, 27, 107-122.

[8] Streitfeld, D., 2000, "Amazon pays a price for marketing test," Los Angeles Times, October 2.

[9] Harris, Frederick H., deB. and Pinder, Jonathan P., 1995, "A revenue management approach to demand management and order booking in assemble-to-order manufacturing," Journal of Operations Management, 13, 299-309.

[10] Mehmet, Barut and Sridharan, V., 2005, "Revenue management in order-driven production systems," Decision Sciences, 36(2), 287-316.

[11] Barut, M. and Sridharan, V., 2004, "Design and evaluation of a dynamic capacity apportionment procedure," European Journal of Operations Research, 155(1), 112-133.

[12] Wilhelm, Wilbert E., 2001, "A technical review of column generation in integer programming," Optimization and Engineering, 2, 159-200.

[13] Slotnick, Susan A. and Morton, Thomas E., 1996, "Selecting jobs for heavily loaded shop with lateness penalties," Computers & Operations Research, 23(2), 131-140.

[14] Lewis, Herbert F. and Slotnick, Susan A., 2002, "Multi-period job selection:Planning work loads to maximize profit," Computers & Operations Research, 29, 1081-1098.

[15] Rom, Walter O. and Slotnick, Susan A., 2009, "Order acceptance using genetic algorithms," Computers & Operations Research, 36(6), 1758-1767.

[16] Ghosh, Jay B., 1997, "Job selection in a heavily loaded shop," Computers Ops. Res., 24(2), 141-145.

[17] Alidaee, Bahram, Kochenberger, Gary A. and Amini, Mohammad M., 2001, "Greedy solutions of selection and ordering problems," European Journal of Operational Research, 134(1), 203-215.

[18] Balakrishnan, N., Sridharan, V. and Patterson, J. W., 1996, "Rationing capacity between two product classes," Decision Sciences, 27(2), 185-214.

[19] Balakrishnan, N., Patterson, J. W. and Sridharan, V., 1999, "Robustness of capacity rationing policies," European Journal of Operational Research, 115, 324-338.

[20] Charnsirisakskul, Kasarin, Griffin, Paul M. and Keskinocak, Pinar, 2004, "Order selection and scheduling with leadtime flexibility," IIE Transactions, 36(7), 697(11).

[21] Charnsirisakskul, Kasarin, Griffin, Paul M. and Keskinocak, Pınar, 2006, "Pricing and scheduling decisions with leadtime flexibility," European Journal of Operational Research, 171(1), 153-169.

[22] Duenyas, Izak, 1995, "Single facility due date setting with multiple customer classes," Management Science, 41(4), 608-619.

[23] Easton, Fred F. and Moodie, Douglas R., 1999, "Pricing and lead time decisions for make-to-order firms with contingent orders," European Journal of Operational Research, 116(2), 305-318.

[24] Ebadian, M., Rabbani, M., Jolai, F., Torabi, S. A. and Tavakkoli-Moghaddam, R., 2008, "A new decision-making structure for the order entry stage in make-to-order environments," International Journal of Production Economics, 111(2), 351-367.

[25] Herbots, Jade, Herroelen, Willy and Leus, Roel, "Dynamic order acceptance and capacity planning within a multi-project environment,"

[26] Akkan, Can, 1996, "Overtime scheduling: An application in finite-capacity real-time scheduling," The Journal of OPerational Research Society, 47(9), 1137-1149.

[27] Akkan, Can, 1997, "Finite-capacity scheduling-based planning for revenue-based capacity management," European Journal of Operational Research, 100(1), 170-179.

[28] Miller, Bruce L., 1969, "A queuing reward system with several customer classes," Management Science, 16(3, Theory Series), 234-245.

[29] Lippman, Steven A. and Sheldon M. Ross, 1971, "The streetwalker's dilemma: A job shop model," SIAM J Appl Math, 20(3), 336-342.

[30] Carr, Scott and Duenyas, Izak, 2000, "Optimal admission control and sequencing in a make-to-Stock/Make-to-order production system," Operations Research, 48(5), 709-720.

[31] Webster, Scott, 2002, "Dynamic pricing and lead-time policies for make-to-order systems," Decision Sciences, 33(4), 579-599.

[32] Sridharan, V. and Balakrishnan, N., 1996, "Capacity rationing in multi-period planning environments," in Proceedings of the Decision Sciences Institute, Orlando, FL.

[33] Defregger, Florian and Kuhn, Heinrich, 2007, "Revenue management for a make-to-order company with limited inventory capacity," OR Spectrum, 29, 137-156.

[34] Nandi, Amitava and Rogers, Paul, "Optimal control of make-to-order manufacturing systems via selected order acceptance," in Winter Simulation Conference

[35] Philipoom, P. R. and Fry, T. D., 1992, "Capacity-based order review/release strategies to improve manufacturing perofrmance," International Journal of Production Research, 30(11), 2559-2572.

[36] Modarres, Mohammad and Sharifyazdi, Mehdi, "Revenue management approach to stochastic capacity allocation problem," European Journal of Operational Research, In Press, Corrected Proof

[37] Faria, Jose A. , 2005, "Multiobjective optimization models and solution methods for planning land development using minimum spanning trees, lagrangian relaxation and decomposition techniques," Dissertation/Thesis, Unpublished, University of Maryland.

[38] Van den Akker, J. M., Hoogeveen, H. and van de Velde, S., 1997, "A column generation algorithm for common due date scheduling,"

[39] Van den Akker, J. M., Hoogeveen, H. and van de Velde, S., 1999, "Parallel machine scheduling by column generation," Operations Research, 47(6), 862-872.

[40] Van den Akker, J. M., Hurkens, C. A. J. and Savelsbergh, M. W. P., 2000, "A time-indexed formulation for single-machine scheduling problems: Column generation," INFORMS Journal of computing, 12(2), 111-124.

[41] Van den Akker, J. M., Van Hoesel, C. P. M. and Savelsbergh, M. W. P., 1999, "A polyhedral approach to single-machine scheduling problems," Mathematical Programming, 85(3), 541-572.

[42] Hans, Erwin. , 2001, "Resource loading by branch-and-price techniques," Dissertation/Thesis, Unpublished, University of Twente.

[43] Bazarra, Mokhtar S., Jarvis, John J. and Sherali, Hanif D., 2004, "Linear programming and network flows," Wiley.

[44] Barnhart, Cynthia, Johnson, Ellis L., Nemhauser, George L., Savelsbergh, Martin W. P. and Vance, Pamela H., 1998, "Branch-and-price: Column generation for solving huge integer programs," Operations Research, 46(3), 316-329.

[45] Appelgren, L. H., 1969, "A column generation algorithm for a ship scheduling problem," Transportation Science, 3, 53-68.

[46] Rardin, Ronald L.,1998, "Optimization in operations research," Prentice Hall.

[47] Lasdon, L. S.,1970, "Optimization theory for large systems," Macmillan Publishing Co., Inc.

[48] Savelsbergh, Martin, 1997, "A branch-and-price algorithm for the generalized assignment problem," Oper.Res., 45(6), 831-841.

# APPENDICES

APPENDIX A: Experimental results (Experiment C) for showing empirically the exponential increase in runtime of the Mixed-Integer Linear Program.

APPENDIX B: Experimental results to show the exponential increase in solving the Sub-problem using a mathematical optimization model.

APPENDIX C: CPLEX results for solving the Mixed-Integer Linear Program using Experimental setup D

APPENDIX D: Results for the Root Node solution (Column Generation) for Experimental setup D

APPENDIX E: Experimental results for Branch & Price Strategy 1 (BPS1) with Depth First (DFS) node selection.

APPENDIX F: Experimental results for Branch & Price Strategy 1 (BPS1) with combination of Depth First and Best First (DFS+BeFS) node selection.

APPENDIX G: Experimental results for Branch & Price Strategy 1 (BPS1) with Best First (BeFS) node selection.

APPENDIX H: Experimental results for approximation algorithm employing BPS1 and DFS+BeFS with optimality tolerance $\alpha$=0.01

APPENDIX I: Experimental results for approximation algorithm employing BPS1 and DFS+BeFS with optimality tolerance $\alpha$=0.05

APPENDIX J: Experimental results for Branch & Price Strategy 2 (BPS2) with combination of Depth First and Best First (DFS+BeFS) node selection.

APPENDIX K: Experimental results for approximation algorithm employing BPS2 and DFS+BeFS with optimality tolerance $\alpha$=0.01

APPENDIX L: Experimental results for approximation algorithm employing BPS2 and DFS+BeFS with optimality tolerance $\alpha$=0.05

# APPENDIX A

| Number of Jobs | Number Of Operations per job | Time Period (Days) | Random Instance | CPLEX Runtime (seconds) | Absolute MIP Gap | Relative MIP Gap |
|---|---|---|---|---|---|---|
| 3 | 3 | 3 | 1 | 0.047 | | |
| 3 | 3 | 4 | 2 | 0.063 | | |
| 3 | 3 | 4 | 3 | 0.938 | | |
| 3 | 4 | 4 | 1 | 0.938 | | |
| 3 | 4 | 4 | 2 | 0.141 | | |
| 3 | 4 | 5 | 3 | 0.156 | | |
| 3 | 5 | 6 | 1 | 0.266 | | |
| 3 | 5 | 6 | 2 | 0.578 | | |
| 3 | 5 | 6 | 3 | 7.141 | | |
| 3 | 6 | 7 | 1 | 66.063 | | |
| 3 | 6 | 7 | 2 | 0.859 | | |
| 3 | 6 | 8 | 3 | 0.625 | | |
| 3 | 7 | 8 | 1 | 9.250 | | |
| 3 | 7 | 9 | 2 | 17.391 | | |
| 3 | 7 | 9 | 3 | 16.484 | | |
| 3 | 8 | 9 | 1 | 385.516 | | |
| 3 | 8 | 10 | 2 | 10135.500 | | |
| 3 | 8 | 10 | 3 | 8.328 | | |
| 5 | 3 | 6 | 1 | 0.313 | | |
| 5 | 3 | 5 | 2 | 1.219 | | |
| 5 | 3 | 5 | 3 | 0.172 | | |
| 5 | 4 | 8 | 1 | 0.641 | | |
| 5 | 4 | 8 | 2 | 1.469 | | |
| 5 | 4 | 9 | 3 | 0.859 | | |
| 5 | 5 | 10 | 1 | 9.109 | | |
| 5 | 5 | 10 | 2 | 1234.860 | | |
| 5 | 5 | 10 | 3 | 489.969 | | |
| 5 | 6 | 12 | 1 | 346.938 | | |
| 5 | 6 | 12 | 2 | 47896.600 | 68 | 2.81% |
| 5 | 6 | 11 | 3 | 11.188 | | |
| 5 | 7 | 13 | 1 | 24.563 | | |
| 5 | 7 | 14 | 2 | 23.234 | | |
| 5 | 7 | 14 | 3 | 1917.980 | | |
| 5 | 8 | 17 | 1 | 62186.400 | 22 | 0.42% |
| 5 | 8 | 16 | 2 | 61971.800 | 28 | 0.03% |
| 5 | 8 | 16 | 3 | 34705.900 | 552 | 11.22% |

# APPENDIX B

| Number Of Jobs | Number of Operations per job | Time Period (Days) | Random Instance | No. of Sub-Problems Solved | Total Sub-Problem Solve Time (seconds) | Average Sub-Problem Solve Time (seconds) |
|---|---|---|---|---|---|---|
| 3 | 3 | 3 | 1 | 18 | 0.766 | 0.043 |
| 3 | 3 | 4 | 2 | 15 | 0.750 | 0.050 |
| 3 | 3 | 4 | 3 | 21 | 0.922 | 0.044 |
| 3 | 4 | 4 | 1 | 21 | 1.031 | 0.049 |
| 3 | 4 | 4 | 2 | 18 | 0.953 | 0.053 |
| 3 | 4 | 5 | 3 | 21 | 1.156 | 0.055 |
| 3 | 5 | 6 | 1 | 33 | 2.625 | 0.080 |
| 3 | 5 | 6 | 2 | 30 | 2.250 | 0.075 |
| 3 | 5 | 6 | 3 | 54 | 5.625 | 0.104 |
| 3 | 6 | 7 | 1 | 60 | 11.047 | 0.184 |
| 3 | 6 | 7 | 2 | 54 | 10.359 | 0.192 |
| 3 | 6 | 8 | 3 | 30 | 4.156 | 0.139 |
| 3 | 7 | 8 | 1 | 51 | 11.875 | 0.233 |
| 3 | 7 | 9 | 2 | 36 | 6.797 | 0.189 |
| 3 | 7 | 9 | 3 | 27 | 6.531 | 0.242 |
| 3 | 8 | 9 | 1 | 42 | 35.875 | 0.854 |
| 3 | 8 | 10 | 2 | 63 | 109.938 | 1.745 |
| 3 | 8 | 10 | 3 | 21 | 14.922 | 0.711 |
| 5 | 3 | 6 | 1 | 40 | 1.859 | 0.046 |
| 5 | 3 | 5 | 2 | 30 | 1.453 | 0.048 |
| 5 | 3 | 5 | 3 | 30 | 1.406 | 0.047 |
| 5 | 4 | 8 | 1 | 45 | 2.938 | 0.065 |
| 5 | 4 | 8 | 2 | 35 | 2.297 | 0.066 |
| 5 | 4 | 9 | 3 | 35 | 2.422 | 0.069 |
| 5 | 5 | 10 | 1 | 60 | 8.016 | 0.134 |
| 5 | 5 | 10 | 2 | 120 | 15.031 | 0.125 |
| 5 | 5 | 10 | 3 | 50 | 11.016 | 0.220 |
| 5 | 6 | 12 | 1 | 45 | 9.109 | 0.202 |
| 5 | 6 | 12 | 2 | 40 | 6.813 | 0.170 |
| 5 | 6 | 11 | 3 | 45 | 5.900 | 0.131 |
| 5 | 7 | 13 | 1 | 35 | 18.000 | 0.514 |
| 5 | 7 | 14 | 2 | 55 | 43.297 | 0.787 |
| 5 | 7 | 14 | 3 | 60 | 25.078 | 0.418 |
| 5 | 8 | 17 | 1 | 150 | 7294.660 | 48.631 |
| 5 | 8 | 16 | 2 | 80 | 770.562 | 9.632 |
| 5 | 8 | 16 | 3 | 115 | 3675.120 | 31.958 |

# APPENDIX C

| Instance | LP Relaxation | Time to Root Node | CPLEX Integer Solution | CPLEX Best Bound | CPLEX Runtime | Absolute MIP Gap | Relative MIP Gap (%) |
|---|---|---|---|---|---|---|---|
| j3o3r3lr0.8t4_i1 | 906.00 | 0 | 794 | 794 | 0.06 | 0.000 | 0.000 |
| j3o3r3lr0.8t5_i2 | 1163.06 | 0 | 1135 | 1135 | 0.03 | 0.000 | 0.000 |
| j3o3r3lr0.8t5_i3 | 1890.00 | 0 | 1807 | 1807 | 0.14 | 0.000 | 0.000 |
| j3o3r3lr1t4_i1 | 1161.06 | 0 | 1120 | 1120 | 0.02 | 0.000 | 0.000 |
| j3o3r3lr1t4_i2 | 1169.00 | 0 | 1169 | 1169 | 0.02 | 0.000 | 0.000 |
| j3o3r3lr1t3_i3 | 787.51 | 0.02 | 729 | 729 | 0.06 | 0.000 | 0.000 |
| j3o3r3lr1.2t2_i1 | 257.13 | 0 | 92 | 92 | 0.02 | 0.000 | 0.000 |
| j3o3r3lr1.2t3_i2 | 377.95 | 0 | 274 | 274 | 0.02 | 0.000 | 0.000 |
| j3o3r3lr1.2t3_i3 | 1333.88 | 0.02 | 1202 | 1202 | 0.03 | 0.000 | 0.000 |
| j3o5r3lr0.8t8_i1 | 3009.00 | 0 | 2781 | 2781.225 | 15.53 | 0.225 | 0.010 |
| j3o5r3lr0.8t6_i2 | 3546.00 | 0 | 3416 | 3416.294 | 6.34 | 0.294 | 0.010 |
| j3o5r3lr0.8t7_i3 | 2271.73 | 0 | 2157 | 2157 | 1.53 | 0.000 | 0.000 |
| j3o5r3lr1t6_i1 | 2474.00 | 0.01 | 2134 | 2134.138 | 11.06 | 0.138 | 0.010 |
| j3o5r3lr1t6_i2 | 4090.85 | 0 | 3661 | 3661.327 | 12.27 | 0.327 | 0.010 |
| j3o5r3lr1t6_i3 | 2651.00 | 0.01 | 2377 | 2377.2 | 27.19 | 0.204 | 0.010 |
| j3o5r3lr1.2t5_i1 | 2949.61 | 0 | 2282 | 2282 | 0.39 | 0.000 | 0.000 |
| j3o5r3lr1.2t6_i2 | 834.64 | 0.02 | 164 | 164 | 0.67 | 0.000 | 0.000 |
| j3o5r3lr1.2t5_i3 | 1335.12 | 0 | 988 | 988 | 0.08 | 0.000 | 0.000 |
| j3o8r5lr0.8t8_i1 | 3180.41 | 0.01 | 2235 | 2235.214 | 57.36 | 0.214 | 0.010 |
| j3o8r5lr0.8t7_i2 | 2589.25 | 0.02 | 1889 | 1889 | 1.27 | 0.000 | 0.000 |
| j3o8r5lr0.8t8_i3 | 3605.25 | 0.03 | 3034 | 3034.3 | 673.94 | 0.300 | 0.010 |
| j3o8r5lr1t6_i1 | 1158.91 | 0.02 | 266 | 266 | 0.27 | 0.000 | 0.000 |
| j3o8r5lr1t5_i2 | 1842.81 | 0 | 1175 | 1175 | 0.77 | 0.000 | 0.000 |
| j3o8r5lr1t6_i3 | 3216.30 | 0.02 | 1610 | 1610 | 1.92 | 0.000 | 0.000 |
| j3o8r5lr1.2t4_i1 | 2010.64 | 0.02 | 649 | 649 | 0.08 | 0.000 | 0.000 |
| j3o8r5lr1.2t4_i2 | 2043.22 | 0.02 | 1102 | 1102 | 0.05 | 0.000 | 0.000 |
| j3o8r5lr1.2t5_i3 | 4064.47 | 0.02 | 1806 | 1806 | 0.45 | 0.000 | 0.000 |
| j3o10r5lr0.8t9_i1 | 2433.76 | 0.03 | 1620 | 1620.147 | 133.39 | 0.147 | 0.010 |
| j3o10r5lr0.8t9_i2 | 4628.15 | 0.06 | 3954 | 4420.984 | 1800 | 466.984 | 11.810 |
| j3o10r5lr0.8t9_i3 | 3693.22 | 0.06 | 2563 | 3169.34 | 1800 | 606.340 | 23.660 |
| j3o10r5lr1t8_i1 | 7709.53 | 0.06 | 6645 | 6645.637 | 1092.74 | 0.637 | 0.010 |
| j3o10r5lr1t7_i2 | 1020.29 | 0.03 | 861 | 861 | 0.58 | 0.000 | 0.000 |
| j3o10r5lr1t7_i3 | 3717.05 | 0.01 | 3171 | 3171 | 12.34 | 0.000 | 0.000 |
| j3o10r5lr1.2t6_i1 | 1109.80 | 0.03 | 1E-07 | 0 | 0.09 | 0.000 | 0.000 |
| j3o10r5lr1.2t5_i2 | 3583.72 | 0.02 | 3004 | 3004 | 0.2 | 0.000 | 0.000 |
| j3o10r5lr1.2t6_i3 | 1109.90 | 0.03 | 137 | 137 | 0.28 | 0.000 | 0.000 |
| j5o3r3lr0.8t7_i1 | 1775.46 | 0 | 1729 | 1729 | 0.16 | 0.000 | 0.000 |
| j5o3r3lr0.8t8_i2 | 2259.00 | 0.01 | 2259 | 2259 | 3.31 | 0.000 | 0.000 |
| j5o3r3lr0.8t8_i3 | 2793.23 | 0.02 | 2627 | 2627 | 1.2 | 0.000 | 0.000 |
| j5o3r3lr1t6_i1 | 3691.67 | 0.02 | 3621 | 3621 | 0.25 | 0.000 | 0.000 |
| j5o3r3lr1t6_i2 | 3187.28 | 0.02 | 3031 | 3031 | 0.39 | 0.000 | 0.000 |
| j5o3r3lr1t5_i3 | 3041.00 | 0 | 3041 | 3041 | 0.44 | 0.000 | 0.000 |
| j5o3r3lr1.2t4_i1 | 1319.17 | 0 | 1312 | 1312 | 0.05 | 0.000 | 0.000 |
| j5o3r3lr1.2t5_i2 | 1470.32 | 0 | 1321 | 1321 | 0.11 | 0.000 | 0.000 |
| j5o3r3lr1.2t5_i3 | 2751.36 | 0 | 2670 | 2670 | 0.34 | 0.000 | 0.000 |
| j5o5r3lr0.8t10_i1 | 2905.00 | 0.01 | 2657 | 2781.752 | 1800.28 | 124.752 | 4.700 |
| j5o5r3lr0.8t14_i2 | 3566.74 | 0.03 | 3404 | 3543 | 1800.02 | 139.000 | 4.080 |
| j5o5r3lr0.8t12_i3 | 7187.00 | 0.03 | 6891 | 7187 | 1801.24 | 296.000 | 4.300 |
| j5o5r3lr1t10_i1 | 3989.59 | 0.02 | 3862 | 3862 | 3.63 | 0.000 | 0.000 |
| j5o5r3lr1t10_i2 | 4003.40 | 0.02 | 3188 | 3790 | 1800.01 | 602.000 | 18.880 |
| j5o5r3lr1t11_i3 | 6443.00 | 0.03 | 5992 | 6443 | 1800.58 | 451.000 | 7.530 |

| Instance | LP Relaxation | Time to Root Node | CPLEX Integer Solution | CPLEX Best Bound | CPLEX Runtime | Absolute MIP Gap | Relative MIP Gap (%) |
|---|---|---|---|---|---|---|---|
| j5o5r3lr1.2t8_i1 | 1538.37 | 0.02 | 1182 | 1182.111 | 1377.17 | 0.111 | 0.010 |
| j5o5r3lr1.2t9_i2 | 3264.17 | 0.01 | 3114 | 3114.275 | 251.77 | 0.275 | 0.010 |
| j5o5r3lr1.2t8_i3 | 1906.20 | 0 | 1700 | 1700 | 2 | 0.000 | 0.000 |
| j5o8r5lr0.8t11_i1 | 3057.00 | 0.09 | 2640 | 2982.154 | 1800.19 | 342.154 | 12.960 |
| j5o8r5lr0.8t13_i2 | 9445.00 | 0.09 | 8657.5 | 9414 | 1800 | 756.500 | 8.740 |
| j5o8r5lr0.8t12_i3 | 11458.83 | 0.06 | 11091.5 | 11444 | 1803.53 | 352.500 | 3.180 |
| j5o8r5lr1t9_i1 | 2920.34 | 0.09 | 1559 | 2039.745 | 1800.8 | 480.745 | 30.840 |
| j5o8r5lr1t9_i2 | 4275.43 | 0.08 | 3667 | 4062.136 | 1800.03 | 395.136 | 10.780 |
| j5o8r5lr1t9_i3 | 3907.58 | 0.09 | 2626 | 3310 | 1800.44 | 684.000 | 26.050 |
| j5o8r5lr1.2t9_i1 | 4487.86 | 0.11 | 3752 | 3964.069 | 1800 | 212.069 | 5.650 |
| j5o8r5lr1.2t8_i2 | 3417.08 | 0.13 | 2018 | 2332.823 | 1800.03 | 314.823 | 15.600 |
| j5o8r5lr1.2t8_i3 | 8365.71 | 0.08 | 6833 | 7982.636 | 1800.3 | 1149.640 | 16.820 |
| j5o10r5lr0.8t15_i1 | 7765.44 | 0.37 | 6664 | 7695 | 1800.05 | 1031.000 | 15.470 |
| j5o10r5lr0.8t14_i2 | 3795.69 | 0.24 | 2967 | 3611.158 | 1800.03 | 644.158 | 21.710 |
| j5o10r5lr0.8t16_i3 | 11333.81 | 0.3 | 7667 | 11056 | 1800.02 | 3389.000 | 44.200 |
| j5o10r5lr1t12_i1 | 7116.60 | 0.25 | 5661 | 6964.841 | 1800.05 | 1303.840 | 23.030 |
| j5o10r5lr1t12_i2 | 7509.24 | 0.23 | 6274 | 7313 | 1800 | 1039.000 | 16.560 |
| j5o10r5lr1t12_i3 | 7930.32 | 0.3 | 5438 | 7596.897 | 1800.02 | 2158.900 | 39.700 |
| j5o10r5lr1.2t10_i1 | 5238.88 | 0.25 | 3615 | 4511.766 | 1801.88 | 896.766 | 24.810 |
| j5o10r5lr1.2t10_i2 | 6544.65 | 0.23 | 5676 | 6246.1 | 1800.02 | 570.100 | 10.040 |
| j5o10r5lr1.2t10_i3 | 5508.28 | 0.2 | 4401 | 4851.961 | 1800.02 | 450.961 | 10.250 |
| j8o3r3lr0.8t13_i1 | 2391.00 | 0.03 | 2295 | 2391 | 1801.34 | 96.000 | 4.180 |
| j8o3r3lr0.8t13_i2 | 4341.00 | 0.03 | 4341 | 4341 | 3.89 | 0.000 | 0.000 |
| j8o3r3lr0.8t11_i3 | 4365.53 | 0.02 | 4193 | 4299 | 1800.19 | 106.000 | 2.530 |
| j8o3r3lr1t9_i1 | 2953.62 | 0.02 | 2889 | 2889 | 1.02 | 0.000 | 0.000 |
| j8o3r3lr1t9_i2 | 2082.87 | 0.02 | 2069 | 2069 | 0.27 | 0.000 | 0.000 |
| j8o3r3lr1t10_i3 | 3285.82 | 0.01 | 3218 | 3266 | 1800.02 | 48.000 | 1.490 |
| j8o3r3lr1.2t8_i1 | 3109.27 | 0.02 | 2963 | 2963.295 | 1389.91 | 0.295 | 0.010 |
| j8o3r3lr1.2t8_i2 | 5439.52 | 0.02 | 5436 | 5436 | 1.31 | 0.000 | 0.000 |
| j8o3r3lr1.2t8_i3 | 3492.07 | 0.02 | 3202 | 3377 | 1800.01 | 175.000 | 5.470 |
| j8o5r3lr0.8t20_i1 | 6205.00 | 0.17 | 5340 | 6205 | 1800.03 | 865.000 | 16.200 |
| j8o5r3lr0.8t20_i2 | 6039.13 | 0.14 | 5848 | 6010 | 1800 | 162.000 | 2.770 |
| j8o5r3lr0.8t20_i3 | 7039.50 | 0.13 | 6924 | 7028 | 1801.03 | 104.000 | 1.500 |
| j8o5r3lr1t17_i1 | 7072.25 | 0.14 | 6838 | 7072 | 1800.05 | 234.000 | 3.420 |
| j8o5r3lr1t14_i2 | 6045.00 | 0.11 | 5724 | 6034 | 1800.17 | 310.000 | 5.420 |
| j8o5r3lr1t16_i3 | 4183.03 | 0.14 | 4009 | 4177 | 1800.03 | 168.000 | 4.190 |
| j8o5r3lr1.2t14_i1 | 5739.75 | 0.09 | 5369 | 5685 | 1800.03 | 316.000 | 5.890 |
| j8o5r3lr1.2t14_i2 | 5966.38 | 0.11 | 5807 | 5887 | 1800.03 | 80.000 | 1.380 |
| j8o5r3lr1.2t12_i3 | 4336.10 | 0.08 | 4003 | 4259 | 1800.02 | 256.000 | 6.400 |
| j8o8r5lr0.8t21_i1 | 7346.00 | 0.63 | 4129 | 7346 | 1800.05 | 3217.000 | 77.910 |
| j8o8r5lr0.8t18_i2 | 6931.50 | 0.2 | 3305 | 6803 | 1800.06 | 3498.000 | 105.840 |
| j8o8r5lr0.8t19_i3 | 15558.67 | 0.74 | 13522 | 15452 | 1800.01 | 1930.000 | 14.270 |
| j8o8r5lr1t15_i1 | 17591.00 | 0.47 | 15757 | 17549.47 | 1800.03 | 1792.470 | 11.380 |
| j8o8r5lr1t16_i2 | 11858.60 | 0.31 | 10583 | 11713 | 1800.06 | 1130.000 | 10.680 |
| j8o8r5lr1t16_i3 | 11831.00 | 0.56 | 9902 | 11831 | 1800.03 | 1929.000 | 19.480 |
| j8o8r5lr1.2t13_i1 | 11489.89 | 0.5 | 10102 | 11090.85 | 1800.88 | 988.854 | 9.790 |
| j8o8r5lr1.2t13_i2 | 7388.77 | 0.33 | 6889 | 7357 | 1800.09 | 468.000 | 6.790 |
| j8o8r5lr1.2t13_i3 | 4850.94 | 0.31 | 4309 | 4757 | 1800.01 | 448.000 | 10.400 |
| j8o10r5lr0.8t26_i1 | 18082.60 | 2.45 | 0 | 18001 | 1800.05 | 18001.000 | infinity |
| j8o10r5lr0.8t23_i2 | 11581.00 | 1.84 | 6441 | 11581 | 1800.02 | 5140.000 | 79.800 |
| j8o10r5lr0.8t23_i3 | 18384.00 | 1.56 | 11813 | 18384 | 1800.03 | 6571.000 | 55.630 |
| j8o10r5lr1t19_i1 | 13919.71 | 1.23 | 8399 | 13528 | 1800.08 | 5129.000 | 61.070 |
| j8o10r5lr1t17_i2 | 11238.00 | 0.94 | 2985 | 11238 | 1800.08 | 8253.000 | 276.480 |
| j8o10r5lr1t19_i3 | 9858.08 | 1.09 | 2836 | 9850 | 1800.05 | 7014.000 | 247.320 |

| Instance | LP Relaxation | Time to Root Node | CPLEX Integer Solution | CPLEX Best Bound | CPLEX Runtime | Absolute MIP Gap | Relative MIP Gap (%) |
|---|---|---|---|---|---|---|---|
| j8o10r5lr1.2t15_i1 | 9478.00 | 0.66 | 3457 | 9478 | 1800.03 | 6021.000 | 174.170 |
| j8o10r5lr1.2t16_i2 | 16972.46 | 1.23 | 7713 | 16737 | 1800.02 | 9024.000 | 117.000 |
| j8o10r5lr1.2t16_i3 | 12323.25 | 1.13 | 2430 | 12220.57 | 1800.05 | 9790.570 | 402.900 |
| j10o3r3lr0.8t15_i1 | 7366.00 | 0.03 | 7366 | 7366 | 2.17 | 0.000 | 0.000 |
| j10o3r3lr0.8t15_i2 | 3593.58 | 0.05 | 3529 | 3592 | 2.63 | 0.000 | 0.000 |
| j10o3r3lr0.8t14_i3 | 4795.27 | 0.03 | 4676 | 4676 | 1.53 | 0.000 | 0.000 |
| j10o3r3lr1t12_i1 | 3719.00 | 0.03 | 3639 | 3719 | 1800 | 90.000 | 2.200 |
| j10o3r3lr1t12_i2 | 5296.00 | 0.05 | 5218 | 5266 | 1800.14 | 48.000 | 0.920 |
| j10o3r3lr1t11_i3 | 4669.00 | 0.02 | 4658 | 4658 | 5.11 | 0.000 | 0.000 |
| j10o3r3lr1.2t10_i1 | 4683.00 | 0.03 | 4683 | 4683 | 2.53 | 0.000 | 0.000 |
| j10o3r3lr1.2t9_i2 | 4464.71 | 0.03 | 4373 | 4373.433 | 574.72 | 0.433 | 0.010 |
| j10o3r3lr1.2t10_i3 | 4476.02 | 0.03 | 4306 | 4372 | 1800.02 | 66.000 | 1.530 |
| j10o5r3lr0.8t26_i1 | 9833.00 | 0.34 | 9833 | 9833 | 1648.53 | 0.000 | 0.000 |
| j10o5r3lr0.8t28_i2 | 12311.29 | 0.59 | 11425 | 12049 | 1800 | 624.000 | 5.460 |
| j10o5r3lr0.8t25_i3 | 9519.00 | 0.34 | 9169 | 9519 | 1800.03 | 350.000 | 3.820 |
| j10o5r3lr1t20_i1 | 4709.46 | 0.34 | 4672 | 4703 | 1800.01 | 31.000 | 0.660 |
| j10o5r3lr1t19_i2 | 6191.70 | 0.41 | 6156 | 6178 | 1800.02 | 22.000 | 0.360 |
| j10o5r3lr1t19_i3 | 7237.36 | 0.25 | 6967 | 7215 | 1800.03 | 248.000 | 3.560 |
| j10o5r3lr1.2t16_i1 | 4071.03 | 0.22 | 3896 | 4018 | 1800.01 | 122.000 | 3.130 |
| j10o5r3lr1.2t18_i2 | 6851.82 | 0.31 | 6123 | 6782 | 1800.92 | 659.000 | 10.760 |
| j10o5r3lr1.2t18_i3 | 8083.24 | 0.3 | 6880 | 8082 | 1800.05 | 1202.000 | 17.470 |
| j10o8r5lr0.8t24_i1 | 13565.00 | 1.58 | 7206 | 13565 | 1800.03 | 6359.000 | 88.250 |
| j10o8r5lr0.8t24_i2 | 10602.00 | 1.31 | 8169 | 10602 | 1800.06 | 2433.000 | 29.780 |
| j10o8r5lr0.8t22_i3 | 11857.94 | 1.55 | 4377 | 11804 | 1800.06 | 7427.000 | 169.680 |
| j10o8r5lr1t17_i1 | 12802.59 | 0.81 | 10782 | 12750 | 1800.05 | 1968.000 | 18.250 |
| j10o8r5lr1t18_i2 | 9899.41 | 1.39 | 6798 | 9783 | 1800.03 | 2985.000 | 43.910 |
| j10o8r5lr1t19_i3 | 9671.39 | 1.22 | 5408 | 9581 | 1800.03 | 4173.000 | 77.160 |
| j10o8r5lr1.2t16_i1 | 15029.52 | 1.01 | 9231 | 14717 | 1800.02 | 5486.000 | 59.430 |
| j10o8r5lr1.2t16_i2 | 13955.67 | 1.2 | 11403 | 13785 | 1800.08 | 2382.000 | 20.890 |
| j10o8r5lr1.2t16_i3 | 10523.54 | 1.17 | 7518 | 10510 | 1800.03 | 2992.000 | 39.800 |
| j10o10r5lr0.8t29_i1 | 18526.50 | 3.23 | 4.97E-14 | 18491 | 1800.09 | 18491.000 | infinity |
| j10o10r5lr0.8t29_i2 | 12322.00 | 4.06 | 3474 | 12322 | 1800.09 | 8848.000 | 254.690 |
| j10o10r5lr0.8t30_i3 | 17822.00 | 4.23 | 358 | 17822 | 1800.22 | 17464.000 | 4900.000 |
| j10o10r5lr1t24_i1 | 11457.00 | 4.02 | 1737 | 11397 | 1800.17 | 9660.000 | 556.130 |
| j10o10r5lr1t23_i2 | 18395.00 | 3.14 | 0 | 18395 | 1800.03 | 18395.000 | infinity |
| j10o10r5lr1t25_i3 | 17716.31 | 3.72 | 4.75E-13 | 17643 | 1800.13 | 17643.000 | infinity |
| j10o10r5lr1.2t19_i1 | 19037.58 | 2.66 | 7304 | 18841 | 1800.08 | 11537.000 | 157.950 |
| j10o10r5lr1.2t21_i2 | 14230.88 | 3.06 | -3.2E-14 | 14201 | 1800.13 | 14201.000 | infinity |
| j10o10r5lr1.2t19_i3 | 16799.00 | 2.28 | 2909 | 16799 | 1800.03 | 13890.000 | 477.480 |

# APPENDIX D

| Instance | Fractional $U_j$ | Fractional $\lambda_j$ | Initial Solution | Max. Initial Integer Profit | Root Node Solution | Columns Added @ Root Node | Time to Root Node (seconds) |
|---|---|---|---|---|---|---|---|
| j3o3r3lr0.8t4_i1 | 1 | 5 | 716 | 716 | 902 | 17 | 0.031 |
| j3o3r3lr0.8t5_i2 | 0 | 9 | 960 | 960 | 1135 | 18 | 0.031 |
| j3o3r3lr0.8t5_i3 | 0 | 2 | 1181 | 1181 | 1807 | 19 | 0.031 |
| j3o3r3lr1t4_i1 | 0 | 7 | 1070 | 1070 | 1120 | 14 | 0.031 |
| j3o3r3lr1t4_i2 | 0 | 0 | 420 | 420 | 1169 | 19 | 0.016 |
| j3o3r3lr1t3_i3 | 1 | 7 | 655 | 655 | 742.818 | 16 | 0.016 |
| j3o3r3lr1.2t2_i1 | 0 | 0 | 92 | 92 | 92 | 1 | 0.031 |
| j3o3r3lr1.2t3_i2 | 0 | 0 | 274 | 274 | 274 | 3 | 0.016 |
| j3o3r3lr1.2t3_i3 | 0 | 5 | 1107 | 1107 | 1202 | 12 | 0.016 |
| j3o5r3lr0.8t8_i1 | 0 | 13 | 2518 | 2518 | 2839.54 | 46 | 0.047 |
| j3o5r3lr0.8t6_i2 | 0 | 9 | 3366 | 3366 | 3416 | 33 | 0.031 |
| j3o5r3lr0.8t7_i3 | 0 | 9 | 1624 | 1624 | 2170 | 34 | 0.047 |
| j3o5r3lr1t6_i1 | 1 | 10 | 1679 | 1679 | 2145.4 | 29 | 0.031 |
| j3o5r3lr1t6_i2 | 1 | 9 | 3223 | 3223 | 3670.27 | 45 | 0.047 |
| j3o5r3lr1t6_i3 | 0 | 11 | 2225 | 2225 | 2413.7 | 36 | 0.031 |
| j3o5r3lr1.2t5_i1 | 1 | 7 | 1466 | 1466 | 2399 | 30 | 0.031 |
| j3o5r3lr1.2t6_i2 | 1 | 1 | 164 | 164 | 217.5 | 5 | 0.031 |
| j3o5r3lr1.2t5_i3 | 0 | 0 | 988 | 988 | 988 | 1 | 0.016 |
| j3o8r5lr0.8t8_i1 | 0 | 3 | 2195 | 2195 | 2258.4 | 14 | 0.046 |
| j3o8r5lr0.8t7_i2 | 1 | 5 | 284 | 284 | 1900.18 | 11 | 0.046 |
| j3o8r5lr0.8t8_i3 | 1 | 14 | 1324 | 1324 | 3125.5 | 49 | 0.078 |
| j3o8r5lr1t6_i1 | 0 | 0 | 203 | 203 | 266 | 7 | 0.031 |
| j3o8r5lr1t5_i2 | 2 | 3 | 1148 | 1148 | 1294.21 | 8 | 0.031 |
| j3o8r5lr1t6_i3 | 2 | 8 | 1550 | 1550 | 1675.68 | 19 | 0.031 |
| j3o8r5lr1.2t4_i1 | 0 | 0 | 649 | 649 | 649 | 1 | 0.031 |
| j3o8r5lr1.2t4_i2 | 0 | 0 | 284 | 284 | 1102 | 3 | 0.031 |
| j3o8r5lr1.2t5_i3 | 0 | 2 | 571 | 571 | 1806 | 16 | 0.031 |
| j3o10r5lr0.8t9_i1 | 0 | 6 | 1208 | 1208 | 1621.71 | 18 | 0.063 |
| j3o10r5lr0.8t9_i2 | 0 | 16 | 2375 | 2375 | 4154.04 | 49 | 0.094 |
| j3o10r5lr0.8t9_i3 | 0 | 7 | 2297 | 2297 | 2634.36 | 25 | 0.063 |
| j3o10r5lr1t8_i1 | 0 | 18 | 3686 | 3686 | 6706.5 | 73 | 0.125 |
| j3o10r5lr1t7_i2 | 0 | 0 | 861 | 861 | 861 | 1 | 0.016 |
| j3o10r5lr1t7_i3 | 0 | 5 | 1795 | 1795 | 3197.31 | 32 | 0.078 |
| j3o10r5lr1.2t6_i1 | 0 | 0 | 0 | 0 | 0 | 0 | 0.016 |
| j3o10r5lr1.2t5_i2 | 0 | 0 | 1121 | 1121 | 3004 | 10 | 0.046 |
| j3o10r5lr1.2t6_i3 | 0 | 0 | 137 | 137 | 137 | 1 | 0.016 |
| j5o3r3lr0.8t7_i1 | 1 | 11 | 1449 | 1449 | 1775.46 | 27 | 0.031 |
| j5o3r3lr0.8t8_i2 | 0 | 7 | 2113 | 2113 | 2259 | 31 | 0.031 |
| j5o3r3lr0.8t8_i3 | 2 | 18 | 2341 | 2341 | 2786.08 | 37 | 0.031 |
| j5o3r3lr1t6_i1 | 0 | 12 | 2883 | 2883 | 3621 | 31 | 0.031 |
| j5o3r3lr1t6_i2 | 2 | 14 | 2555 | 2555 | 3081.58 | 35 | 0.031 |
| j5o3r3lr1t5_i3 | 0 | 14 | 2843 | 2843 | 3041 | 35 | 0.031 |
| j5o3r3lr1.2t4_i1 | 1 | 9 | 773 | 773 | 1319.17 | 18 | 0.016 |
| j5o3r3lr1.2t5_i2 | 2 | 12 | 1206 | 1206 | 1408.43 | 33 | 0.031 |
| j5o3r3lr1.2t5_i3 | 0 | 16 | 2174 | 2174 | 2674.13 | 44 | 0.031 |
| j5o5r3lr0.8t10_i1 | 1 | 18 | 2119 | 2119 | 2688.8 | 68 | 0.063 |
| j5o5r3lr0.8t14_i2 | 1 | 25 | 2882 | 2882 | 3470.11 | 60 | 0.094 |
| j5o5r3lr0.8t12_i3 | 0 | 17 | 6440 | 6440 | 6891 | 30 | 0.047 |
| j5o5r3lr1t10_i1 | 1 | 14 | 3411 | 3411 | 3989.59 | 51 | 0.063 |
| j5o5r3lr1t10_i2 | 1 | 21 | 2658 | 2658 | 3424.76 | 84 | 0.078 |
| j5o5r3lr1t11_i3 | 0 | 20 | 5767 | 5767 | 5992 | 60 | 0.078 |
| j5o5r3lr1.2t8_i1 | 1 | 11 | 647 | 647 | 1380.03 | 45 | 0.063 |

| Instance | Fractional $U_j$ | Fractional $\lambda_j$ | Initial Solution | Max. Initial Integer Profit | Root Node Solution | Columns Added @ Root Node | Time to Root Node (seconds) |
|---|---|---|---|---|---|---|---|
| j5o5r3lr1.2t9_i2 | 2 | 18 | 2503 | 2503 | 3170.5 | 65 | 0.078 |
| j5o5r3lr1.2t8_i3 | 2 | 16 | 1144 | 1144 | 1833.44 | 46 | 0.062 |
| j5o8r5lr0.8t11_i1 | 1 | 16 | 1995 | 1995 | 2802.2 | 54 | 0.11 |
| j5o8r5lr0.8t13_i2 | 1 | 29 | 7905 | 7905 | 8788.49 | 136 | 0.25 |
| j5o8r5lr0.8t12_i3 | 0 | 23 | 10094 | 10094 | 11234 | 85 | 0.188 |
| j5o8r5lr1t9_i1 | 3 | 18 | 1146 | 1146 | 1705.34 | 63 | 0.11 |
| j5o8r5lr1t9_i2 | 1 | 18 | 2127 | 2127 | 3809.49 | 55 | 0.11 |
| j5o8r5lr1t9_i3 | 0 | 14 | 1057 | 1057 | 2818.31 | 50 | 0.11 |
| j5o8r5lr1.2t9_i1 | 1 | 16 | 3091 | 3091 | 4026.43 | 58 | 0.141 |
| j5o8r5lr1.2t8_i2 | 1 | 18 | 869 | 869 | 2161.05 | 55 | 0.094 |
| j5o8r5lr1.2t8_i3 | 1 | 28 | 4229 | 4229 | 7212.8 | 119 | 0.156 |
| j5o10r5lr0.8t15_i1 | 1 | 30 | 5419 | 5419 | 6965.13 | 160 | 0.469 |
| j5o10r5lr0.8t14_i2 | 2 | 25 | 755 | 755 | 3074.46 | 115 | 0.391 |
| j5o10r5lr0.8t16_i3 | 2 | 34 | 7241 | 7241 | 10019.8 | 172 | 0.5 |
| j5o10r5lr1t12_i1 | 0 | 44 | 4123 | 4123 | 6025.39 | 223 | 0.5 |
| j5o10r5lr1t12_i2 | 1 | 25 | 2828 | 2828 | 6388.75 | 102 | 0.328 |
| j5o10r5lr1t12_i3 | 1 | 35 | 4177 | 4177 | 6327.77 | 173 | 0.391 |
| j5o10r5lr1.2t10_i1 | 2 | 20 | 1871 | 1871 | 3677.49 | 75 | 0.219 |
| j5o10r5lr1.2t10_i2 | 1 | 29 | 5019 | 5019 | 5877.61 | 113 | 0.266 |
| j5o10r5lr1.2t10_i3 | 2 | 24 | 2185 | 2185 | 4556.57 | 91 | 0.188 |
| j8o3r3lr0.8t13_i1 | 1 | 23 | 2263 | 2263 | 2349.3 | 48 | 0.063 |
| j8o3r3lr0.8t13_i2 | 0 | 27 | 4181 | 4181 | 4341 | 53 | 0.063 |
| j8o3r3lr0.8t11_i3 | 1 | 25 | 4057 | 4057 | 4240.05 | 48 | 0.046 |
| j8o3r3lr1t9_i1 | 1 | 22 | 2305 | 2305 | 2953.62 | 53 | 0.046 |
| j8o3r3lr1t9_i2 | 1 | 14 | 1616 | 1616 | 2082.87 | 52 | 0.046 |
| j8o3r3lr1t10_i3 | 1 | 25 | 2473 | 2473 | 3248.93 | 56 | 0.047 |
| j8o3r3lr1.2t8_i1 | 2 | 21 | 2133 | 2133 | 2996.59 | 69 | 0.046 |
| j8o3r3lr1.2t8_i2 | 1 | 24 | 4147 | 4147 | 5439.24 | 69 | 0.047 |
| j8o3r3lr1.2t8_i3 | 2 | 21 | 2641 | 2641 | 3269.25 | 56 | 0.047 |
| j8o5r3lr0.8t20_i1 | 1 | 31 | 3993 | 3993 | 5964.56 | 77 | 0.125 |
| j8o5r3lr0.8t20_i2 | 1 | 44 | 5681 | 5681 | 6039.13 | 74 | 0.125 |
| j8o5r3lr0.8t20_i3 | 1 | 34 | 6630 | 6630 | 6962.5 | 81 | 0.141 |
| j8o5r3lr1t17_i1 | 1 | 36 | 5833 | 5833 | 6946 | 93 | 0.156 |
| j8o5r3lr1t14_i2 | 2 | 38 | 4989 | 4989 | 5903.37 | 80 | 0.094 |
| j8o5r3lr1t16_i3 | 1 | 27 | 1698 | 1698 | 4082.04 | 61 | 0.11 |
| j8o5r3lr1.2t14_i1 | 3 | 44 | 4338 | 4338 | 5567.58 | 128 | 0.156 |
| j8o5r3lr1.2t14_i2 | 2 | 38 | 4040 | 4040 | 5835.14 | 106 | 0.156 |
| j8o5r3lr1.2t12_i3 | 2 | 34 | 3224 | 3224 | 4166.66 | 137 | 0.141 |
| j8o8r5lr0.8t21_i1 | 1 | 48 | 6360 | 6360 | 7280.75 | 112 | 0.375 |
| j8o8r5lr0.8t18_i2 | 1 | 34 | 5514 | 5514 | 6857.5 | 97 | 0.281 |
| j8o8r5lr0.8t19_i3 | 0 | 49 | 13046 | 13046 | 15035.8 | 293 | 0.922 |
| j8o8r5lr1t15_i1 | 1 | 56 | 14347 | 14347 | 16285.8 | 328 | 0.75 |
| j8o8r5lr1t16_i2 | 2 | 54 | 8145 | 8145 | 11080.2 | 184 | 0.453 |
| j8o8r5lr1t16_i3 | 0 | 51 | 9819 | 9819 | 10996 | 160 | 0.375 |
| j8o8r5lr1.2t13_i1 | 1 | 43 | 6443 | 6443 | 10429.2 | 218 | 0.469 |
| j8o8r5lr1.2t13_i2 | 1 | 47 | 5481 | 5481 | 7350.56 | 243 | 0.547 |
| j8o8r5lr1.2t13_i3 | 1 | 31 | 2824 | 2824 | 4564.68 | 142 | 0.39 |
| j8o10r5lr0.8t26_i1 | 1 | 52 | 15038 | 15038 | 17460.8 | 184 | 0.843 |
| j8o10r5lr0.8t23_i2 | 2 | 57 | 8724 | 8724 | 11333.7 | 168 | 0.594 |
| j8o10r5lr0.8t23_i3 | 1 | 47 | 16541 | 16541 | 18279.8 | 162 | 0.688 |
| j8o10r5lr1t19_i1 | 1 | 51 | 7217 | 7217 | 13069.8 | 326 | 1.266 |
| j8o10r5lr1t17_i2 | 1 | 54 | 7382 | 7382 | 9356.48 | 280 | 0.844 |
| j8o10r5lr1t19_i3 | 1 | 44 | 5850 | 5850 | 8981.05 | 202 | 0.735 |
| j8o10r5lr1.2t15_i1 | 1 | 56 | 5440 | 5440 | 9242.17 | 339 | 1.032 |

| Instance | Fractional $U_j$ | Fractional $\lambda_j$ | Initial Solution | Max. Initial Integer Profit | Root Node Solution | Columns Added @ Root Node | Time to Root Node (seconds) |
|---|---|---|---|---|---|---|---|
| j8o10r5lr1.2t16_i2 | 4 | 67 | 13092 | 13092 | 15463.8 | 490 | 1.438 |
| j8o10r5lr1.2t16_i3 | 2 | 57 | 8131 | 8131 | 10754.3 | 328 | 1 |
| j10o3r3lr0.8t15_i1 | 0 | 35 | 6757 | 6757 | 7366 | 70 | 0.078 |
| j10o3r3lr0.8t15_i2 | 1 | 34 | 3247 | 3247 | 3593.58 | 86 | 0.078 |
| j10o3r3lr0.8t14_i3 | 1 | 34 | 4532 | 4532 | 4795.27 | 70 | 0.078 |
| j10o3r3lr1t12_i1 | 0 | 35 | 2813 | 2813 | 3639 | 86 | 0.078 |
| j10o3r3lr1t12_i2 | 1 | 38 | 4781 | 4781 | 5280 | 89 | 0.078 |
| j10o3r3lr1t11_i3 | 1 | 25 | 3663 | 3663 | 4669 | 65 | 0.047 |
| j10o3r3lr1.2t10_i1 | 0 | 31 | 3542 | 3542 | 4683 | 79 | 0.063 |
| j10o3r3lr1.2t9_i2 | 2 | 26 | 3125 | 3125 | 4384.69 | 63 | 0.046 |
| j10o3r3lr1.2t10_i3 | 1 | 34 | 2989 | 2989 | 4382.75 | 89 | 0.062 |
| j10o5r3lr0.8t26_i1 | 0 | 46 | 9405 | 9405 | 9833 | 110 | 0.25 |
| j10o5r3lr0.8t28_i2 | 1 | 50 | 10474 | 10474 | 12037 | 109 | 0.312 |
| j10o5r3lr0.8t25_i3 | 1 | 45 | 8585 | 8585 | 9245.9 | 110 | 0.234 |
| j10o5r3lr1t20_i1 | 1 | 43 | 4215 | 4215 | 4684.4 | 109 | 0.188 |
| j10o5r3lr1t19_i2 | 0 | 44 | 5903 | 5903 | 6156 | 108 | 0.172 |
| j10o5r3lr1t19_i3 | 2 | 47 | 6083 | 6083 | 7184.96 | 100 | 0.187 |
| j10o5r3lr1.2t16_i1 | 2 | 38 | 3062 | 3062 | 3970.99 | 105 | 0.14 |
| j10o5r3lr1.2t18_i2 | 1 | 28 | 4564 | 4564 | 6226.5 | 97 | 0.172 |
| j10o5r3lr1.2t18_i3 | 1 | 34 | 4050 | 4050 | 7350 | 136 | 0.234 |
| j10o8r5lr0.8t24_i1 | 0 | 62 | 12568 | 12568 | 13293 | 126 | 0.438 |
| j10o8r5lr0.8t24_i2 | 0 | 70 | 9930 | 9930 | 10589.1 | 141 | 0.5 |
| j10o8r5lr0.8t22_i3 | 1 | 58 | 10337 | 10337 | 11319 | 118 | 0.344 |
| j10o8r5lr1t17_i1 | 1 | 66 | 10408 | 10408 | 12341.1 | 188 | 0.454 |
| j10o8r5lr1t18_i2 | 3 | 45 | 6189 | 6189 | 8824.09 | 205 | 0.454 |
| j10o8r5lr1t19_i3 | 1 | 42 | 6268 | 6268 | 9378.57 | 95 | 0.297 |
| j10o8r5lr1.2t16_i1 | 0 | 44 | 7276 | 7276 | 13790 | 154 | 0.391 |
| j10o8r5lr1.2t16_i2 | 2 | 54 | 10204 | 10204 | 13313.9 | 265 | 0.563 |
| j10o8r5lr1.2t16_i3 | 1 | 72 | 8061 | 8061 | 10207 | 398 | 1 |
| j10o10r5lr0.8t29_i1 | 1 | 70 | 15839 | 15839 | 18500.4 | 196 | 0.985 |
| j10o10r5lr0.8t29_i2 | 1 | 83 | 10919 | 10919 | 12169.9 | 161 | 0.766 |
| j10o10r5lr0.8t30_i3 | 1 | 73 | 15273 | 15273 | 17592.9 | 291 | 1.438 |
| j10o10r5lr1t24_i1 | 0 | 91 | 9028 | 9028 | 11023.9 | 332 | 1.437 |
| j10o10r5lr1t23_i2 | 2 | 59 | 7497 | 7497 | 16441 | 223 | 0.828 |
| j10o10r5lr1t25_i3 | 2 | 87 | 11700 | 11700 | 16344.9 | 541 | 2.531 |
| j10o10r5lr1.2t19_i1 | 1 | 72 | 13871 | 13871 | 18232.6 | 336 | 1.218 |
| j10o10r5lr1.2t21_i2 | 2 | 47 | 6483 | 6483 | 12910.3 | 219 | 0.938 |
| j10o10r5lr1.2t19_i3 | 1 | 69 | 10791 | 10791 | 15929.2 | 205 | 0.734 |

# APPENDIX E

| Instance | Best Integer Solution | Best Integer @ node | Time to Best Integer Solution (seconds) | Number of Columns | Best Bound | Runtime (seconds) | Total Nodes |
|---|---|---|---|---|---|---|---|
| j3o3r3lr0.8t4_i1 | 794 | 14 | 0.046 | 41 | 794 | 0.046 | 15 |
| j3o3r3lr0.8t5_i2 | 1135 | 3 | 0.031 | 24 | 1135 | 0.031 | 5 |
| j3o3r3lr0.8t5_i3 | 1807 | 1 | 0.031 | 24 | 1807 | 0.031 | 3 |
| j3o3r3lr1t4_i1 | 1120 | 1 | 0.031 | 23 | 1120 | 0.031 | 3 |
| j3o3r3lr1t4_i2 | 1169 | 3 | 0.031 | 26 | 1169 | 0.031 | 5 |
| j3o3r3lr1t3_i3 | 729 | 13 | 0.031 | 55 | 729 | 0.031 | 15 |
| j3o3r3lr1.2t2_i1 | 92 | 0 | 0 | 1 | 92 | 0.031 | 3 |
| j3o3r3lr1.2t3_i2 | 274 | 0 | 0 | 3 | 274 | 0.016 | 3 |
| j3o3r3lr1.2t3_i3 | 1202 | 1 | 0.031 | 24 | 1202 | 0.031 | 3 |
| j3o5r3lr0.8t8_i1 | 2781 | 23 | 0.297 | 2113 | 2781 | 3.157 | 247 |
| j3o5r3lr0.8t6_i2 | 3416 | 15 | 0.094 | 92 | 3416 | 0.094 | 17 |
| j3o5r3lr0.8t7_i3 | 2157 | 7 | 0.094 | 150 | 2157.89 | 0.156 | 17 |
| j3o5r3lr1t6_i1 | 2134 | 3 | 0.063 | 73 | 2134 | 0.063 | 5 |
| j3o5r3lr1t6_i2 | 3661 | 7 | 0.078 | 79 | 3661 | 0.078 | 9 |
| j3o5r3lr1t6_i3 | 2367 | 5649 | 175.87 | 50259 | 2413.7 | 900.046 | 13167 |
| j3o5r3lr1.2t5_i1 | 2282 | 39 | 0.25 | 275 | 2282 | 0.328 | 55 |
| j3o5r3lr1.2t6_i2 | 164 | 0 | 0 | 20 | 164 | 0.046 | 5 |
| j3o5r3lr1.2t5_i3 | 988 | 0 | 0 | 1 | 988 | 0.016 | 3 |
| j3o8r5lr0.8t8_i1 | 2235 | 4 | 0.062 | 24 | 2235 | 0.078 | 7 |
| j3o8r5lr0.8t7_i2 | 1889 | 1 | 0.046 | 19 | 1889 | 0.046 | 3 |
| j3o8r5lr0.8t8_i3 | 3034 | 498 | 10.704 | 4268 | 3034.12 | 12.423 | 563 |
| j3o8r5lr1t6_i1 | 266 | 1 | 0.031 | 7 | 266 | 0.031 | 3 |
| j3o8r5lr1t5_i2 | 1175 | 6 | 0.047 | 22 | 1175 | 0.047 | 7 |
| j3o8r5lr1t6_i3 | 1610 | 9 | 0.11 | 79 | 1610 | 0.125 | 11 |
| j3o8r5lr1.2t4_i1 | 649 | 0 | 0 | 1 | 649 | 0.031 | 3 |
| j3o8r5lr1.2t4_i2 | 1102 | 1 | 0.031 | 3 | 1102 | 0.031 | 3 |
| j3o8r5lr1.2t5_i3 | 1806 | 0 | 0.031 | 16 | 1806 | 0.031 | 1 |
| j3o10r5lr0.8t9_i1 | 1620 | 3 | 0.094 | 39 | 1620 | 0.125 | 5 |
| j3o10r5lr0.8t9_i2 | 3417 | 2587 | 130.891 | 51012 | 4126.9 | 900.183 | 8073 |
| j3o10r5lr0.8t9_i3 | 2563 | 409 | 5.735 | 2006 | 2563.49 | 6.141 | 435 |
| j3o10r5lr1t8_i1 | 6645 | 5101 | 205.615 | 24045 | 6645 | 291.264 | 5879 |
| j3o10r5lr1t7_i2 | 861 | 0 | 0 | 1 | 861 | 0.031 | 3 |
| j3o10r5lr1t7_i3 | 3171 | 8 | 0.234 | 104 | 3171 | 0.234 | 9 |
| j3o10r5lr1.2t6_i1 | 1E-07 | 0 | 0 | 0 | 1E-07 | 0.031 | 3 |
| j3o10r5lr1.2t5_i2 | 3004 | 1 | 0.046 | 10 | 3004 | 0.046 | 3 |
| j3o10r5lr1.2t6_i3 | 137 | 0 | 0 | 1 | 137 | 0.016 | 3 |
| j5o3r3lr0.8t7_i1 | 1729 | 10 | 0.062 | 58 | 1729 | 0.062 | 11 |
| j5o3r3lr0.8t8_i2 | 2259 | 30 | 0.141 | 98 | 2259 | 0.141 | 31 |
| j5o3r3lr0.8t8_i3 | 2627 | 71 | 0.234 | 155 | 2627 | 0.234 | 73 |
| j5o3r3lr1t6_i1 | 3621 | 15 | 0.094 | 122 | 3621 | 0.094 | 17 |
| j5o3r3lr1t6_i2 | 3031 | 55 | 0.187 | 197 | 3031 | 0.187 | 57 |
| j5o3r3lr1t5_i3 | 3041 | 8412 | 201.895 | 23426 | 3041 | 201.911 | 8413 |
| j5o3r3lr1.2t4_i1 | 1312 | 22 | 0.063 | 97 | 1312 | 0.063 | 23 |
| j5o3r3lr1.2t5_i2 | 1321 | 25 | 0.093 | 105 | 1321 | 0.109 | 29 |
| j5o3r3lr1.2t5_i3 | 2670 | 49 | 0.172 | 5016 | 2670 | 8.845 | 1001 |
| j5o5r3lr0.8t10_i1 | 2635 | 8051 | 250.322 | 52274 | 2684.97 | 900.079 | 14773 |
| j5o5r3lr0.8t14_i2 | 3404 | 87 | 0.969 | 450 | 3404 | 0.985 | 89 |
| j5o5r3lr0.8t12_i3 | 6891 | 107 | 0.797 | 326 | 6891 | 0.813 | 109 |
| j5o5r3lr1t10_i1 | 3862 | 62 | 0.578 | 334 | 3862 | 0.578 | 63 |
| j5o5r3lr1t10_i2 | 3176 | 2117 | 86.46 | 64930 | 3226.57 | 900.016 | 7971 |
| j5o5r3lr1t11_i3 | 5881 | 791 | 19.392 | 64901 | 5992 | 900.129 | 8167 |

| Instance | Best Integer Solution | Best Integer @ node | Time to Best Integer Solution (seconds) | Number of Columns | Best Bound | Runtime (seconds) | Total Nodes |
|---|---|---|---|---|---|---|---|
| j5o5r3lr1.2t8_i1 | 1182 | 65 | 0.516 | 342 | 1182 | 0.531 | 67 |
| j5o5r3lr1.2t9_i2 | 3114 | 1294 | 37.94 | 28121 | 3114 | 169.149 | 3059 |
| j5o5r3lr1.2t8_i3 | 1700 | 14 | 0.109 | 71 | 1700 | 0.109 | 15 |
| j5o8r5lr0.8t11_i1 | 2725 | 19 | 0.422 | 666 | 2725 | 1.735 | 89 |
| j5o8r5lr0.8t13_i2 | 8600 | 1044 | 64.597 | 64339 | 8776.36 | 900.011 | 5469 |
| j5o8r5lr0.8t12_i3 | 11074 | 2991 | 257.855 | 62414 | 11227.6 | 900.318 | 6119 |
| j5o8r5lr1t9_i1 | 1565 | 104 | 1.891 | 55125 | 1604.94 | 900.422 | 8725 |
| j5o8r5lr1t9_i2 | 3653 | 268 | 7.453 | 58247 | 3756.73 | 900.088 | 7523 |
| j5o8r5lr1t9_i3 | 2710 | 6895 | 657.063 | 54945 | 2805.72 | 900.13 | 8167 |
| j5o8r5lr1.2t9_i1 | 3484.5 | 928 | 20.469 | 51105 | 3817.4 | 900.187 | 9271 |
| j5o8r5lr1.2t8_i2 | 2059 | 143 | 2.828 | 14025 | 2059 | 76.845 | 1851 |
| j5o8r5lr1.2t8_i3 | 4229 | 0 | 0 | 48497 | 7121.13 | 900.09 | 9109 |
| j5o10r5lr0.8t15_i1 | 6785 | 316 | 23.188 | 51005 | 6922.13 | 900.527 | 5647 |
| j5o10r5lr0.8t14_i2 | 2967 | 2865 | 308.488 | 61307 | 3047.61 | 900.479 | 5287 |
| j5o10r5lr0.8t16_i3 | 9715 | 201 | 19.672 | 56125 | 9967.47 | 901.318 | 3431 |
| j5o10r5lr1t12_i1 | 5693 | 3465 | 570.098 | 51686 | 6024.62 | 900.834 | 4607 |
| j5o10r5lr1t12_i2 | 6277.33 | 72 | 4.719 | 61371 | 6355.08 | 900.334 | 3983 |
| j5o10r5lr1t12_i3 | 5995 | 1617 | 233.861 | 55398 | 6326.13 | 901.068 | 3799 |
| j5o10r5lr1.2t10_i1 | 3615 | 889 | 37.578 | 8815 | 3615.53 | 44.5 | 1007 |
| j5o10r5lr1.2t10_i2 | 5628 | 2075 | 221.892 | 58146 | 5858.59 | 900.756 | 4933 |
| j5o10r5lr1.2t10_i3 | 4351 | 194 | 8.063 | 51928 | 4504.98 | 900.579 | 6879 |
| j8o3r3lr0.8t13_i1 | 2295 | 231 | 2.172 | 851 | 2295 | 2.188 | 233 |
| j8o3r3lr0.8t13_i2 | 4341 | 109 | 0.813 | 312 | 4341 | 0.828 | 111 |
| j8o3r3lr0.8t11_i3 | 4193 | 57 | 0.343 | 217 | 4193 | 0.359 | 59 |
| j8o3r3lr1t9_i1 | 2889 | 65 | 0.296 | 167 | 2889 | 0.312 | 69 |
| j8o3r3lr1t9_i2 | 2069 | 276 | 1.343 | 594 | 2069 | 1.343 | 277 |
| j8o3r3lr1t10_i3 | 3218 | 50 | 0.25 | 149 | 3218 | 0.25 | 51 |
| j8o3r3lr1.2t8_i1 | 2963 | 44 | 0.296 | 248 | 2963 | 0.296 | 45 |
| j8o3r3lr1.2t8_i2 | 5417 | 75 | 0.438 | 54257 | 5436 | 900 | 13913 |
| j8o3r3lr1.2t8_i3 | 3081 | 2402 | 41.266 | 53880 | 3202 | 900.094 | 13375 |
| j8o5r3lr0.8t20_i1 | 5940 | 155 | 2.484 | 613 | 5940 | 2.5 | 157 |
| j8o5r3lr0.8t20_i2 | 5941 | 624 | 13.734 | 1885 | 5941 | 13.734 | 625 |
| j8o5r3lr0.8t20_i3 | 6942 | 9562 | 322.906 | 32755 | 6954 | 900.203 | 18503 |
| j8o5r3lr1t17_i1 | 6739 | 4033 | 356.578 | 65076 | 6915.17 | 900.515 | 6897 |
| j8o5r3lr1t14_i2 | 5652 | 5070 | 390.344 | 62914 | 5724 | 900.219 | 7945 |
| j8o5r3lr1t16_i3 | 4009 | 128 | 1.328 | 429 | 4009 | 1.328 | 129 |
| j8o5r3lr1.2t14_i1 | 5369 | 2622 | 119.641 | 62272 | 5405.5 | 900 | 8201 |
| j8o5r3lr1.2t14_i2 | 5775 | 8591 | 430.859 | 47246 | 5807 | 900.234 | 12819 |
| j8o5r3lr1.2t12_i3 | 3899 | 731 | 19.11 | 58654 | 4029 | 900.031 | 9127 |
| j8o8r5lr0.8t21_i1 | 7046 | 852 | 191.563 | 52036 | 7252 | 900.313 | 2857 |
| j8o8r5lr0.8t18_i2 | 6748 | 599 | 37.172 | 63811 | 6802 | 900.531 | 4921 |
| j8o8r5lr0.8t19_i3 | 14660 | 3768 | 865.219 | 55034 | 15033.3 | 900.187 | 3885 |
| j8o8r5lr1t15_i1 | 15751 | 354 | 49.531 | 60840 | 16230.2 | 900.031 | 3659 |
| j8o8r5lr1t16_i2 | 9146 | 3139 | 761 | 52870 | 11032.9 | 900.515 | 3497 |
| j8o8r5lr1t16_i3 | 10741 | 404 | 47.406 | 54057 | 10996 | 900.203 | 4047 |
| j8o8r5lr1.2t13_i1 | 9986 | 338 | 35.359 | 61866 | 10375.1 | 900.609 | 4223 |
| j8o8r5lr1.2t13_i2 | 7154 | 3311 | 611.25 | 64948 | 7347.76 | 900.203 | 4155 |
| j8o8r5lr1.2t13_i3 | 4484 | 2146 | 404.25 | 70409 | 4557.27 | 901.515 | 3305 |
| j8o10r5lr0.8t26_i1 | 17003 | 1639 | 463.922 | 38406 | 17432 | 900.578 | 2999 |
| j8o10r5lr0.8t23_i2 | 11073 | 1840 | 537.61 | 48132 | 11258 | 900.547 | 2673 |
| j8o10r5lr0.8t23_i3 | 18021 | 753 | 127.391 | 55007 | 18213 | 900.344 | 3051 |
| j8o10r5lr1t19_i1 | 12758 | 388 | 77.704 | 51462 | 13067.4 | 900.219 | 3663 |
| j8o10r5lr1t17_i2 | 9056 | 469 | 66.172 | 65335 | 9354.93 | 900.359 | 3093 |

| Instance | Best Integer Solution | Best Integer @ node | Time to Best Integer Solution (seconds) | Number of Columns | Best Bound | Runtime (seconds) | Total Nodes |
|---|---|---|---|---|---|---|---|
| j8o10r5lr1t19_i3 | 8645 | 312 | 44.844 | 62333 | 8870.84 | 900.657 | 3475 |
| j8o10r5lr1.2t15_i1 | 7899 | 441 | 76.813 | 53938 | 9229.92 | 900.047 | 3619 |
| j8o10r5lr1.2t16_i2 | 14621 | 876 | 237.25 | 56144 | 15248.3 | 900.672 | 2729 |
| j8o10r5lr1.2t16_i3 | 10111.3 | 2278 | 518.516 | 52221 | 10667.7 | 900.172 | 3389 |
| j10o3r3lr0.8t15_i1 | 7366 | 205 | 2.281 | 595 | 7366 | 2.297 | 207 |
| j10o3r3lr0.8t15_i2 | 3529 | 317 | 3.797 | 1064 | 3529 | 3.828 | 319 |
| j10o3r3lr0.8t14_i3 | 4676 | 303 | 2.766 | 644 | 4676 | 2.781 | 305 |
| j10o3r3lr1t12_i1 | 3639 | 186 | 1.421 | 388 | 3639 | 1.437 | 187 |
| j10o3r3lr1t12_i2 | 5218 | 198 | 2.453 | 933 | 5218 | 2.453 | 199 |
| j10o3r3lr1t11_i3 | 4658 | 166 | 1.297 | 556 | 4658 | 1.297 | 167 |
| j10o3r3lr1.2t10_i1 | 4683 | 197 | 1.469 | 569 | 4683 | 1.485 | 199 |
| j10o3r3lr1.2t9_i2 | 4373 | 1546 | 19 | 5423 | 4373 | 19 | 1547 |
| j10o3r3lr1.2t10_i3 | 4306 | 8542 | 510.687 | 56346 | 4323.5 | 900 | 11481 |
| j10o5r3lr0.8t26_i1 | 9833 | 744 | 29.938 | 2439 | 9833 | 29.938 | 745 |
| j10o5r3lr0.8t28_i2 | 11643 | 847 | 85.656 | 47689 | 11817 | 901.094 | 5199 |
| j10o5r3lr0.8t25_i3 | 9189 | 499 | 15.797 | 1575 | 9189 | 15.922 | 501 |
| j10o5r3lr1t20_i1 | 4672 | 481 | 10.407 | 1284 | 4672 | 10.438 | 483 |
| j10o5r3lr1t19_i2 | 6104 | 5354 | 445.625 | 56265 | 6156 | 900.235 | 8235 |
| j10o5r3lr1t19_i3 | 7072 | 399 | 12.922 | 2446 | 7072 | 15.172 | 465 |
| j10o5r3lr1.2t16_i1 | 3874 | 794 | 20.812 | 61149 | 3896 | 900.047 | 8421 |
| j10o5r3lr1.2t18_i2 | 6193 | 200 | 5.187 | 40741 | 6221 | 900.046 | 11637 |
| j10o5r3lr1.2t18_i3 | 7350 | 272 | 6.906 | 1379 | 7350 | 6.906 | 273 |
| j10o8r5lr0.8t24_i1 | 13219 | 900 | 125.579 | 47423 | 13293 | 900.594 | 3271 |
| j10o8r5lr0.8t24_i2 | 10424 | 1063 | 213.109 | 45801 | 10589.1 | 900.656 | 2865 |
| j10o8r5lr0.8t22_i3 | 11032 | 1151 | 154.375 | 47225 | 11289 | 900.329 | 4117 |
| j10o8r5lr1t17_i1 | 11909 | 785 | 130.266 | 55772 | 12294 | 900 | 3785 |
| j10o8r5lr1t18_i2 | 8552 | 1601 | 315.797 | 57191 | 8749.66 | 900.282 | 2969 |
| j10o8r5lr1t19_i3 | 8978 | 640 | 70.75 | 54967 | 9346 | 900.281 | 4349 |
| j10o8r5lr1.2t16_i1 | 13490 | 2199 | 421.938 | 56042 | 13790 | 900.219 | 3635 |
| j10o8r5lr1.2t16_i2 | 12987.5 | 1150 | 125.813 | 58582 | 13293 | 900.453 | 4397 |
| j10o8r5lr1.2t16_i3 | 9903 | 628 | 97.594 | 51449 | 10160.5 | 900.829 | 4195 |
| j10o10r5lr0.8t29_i1 | 18435 | 884 | 146.438 | 31466 | 18491 | 900.907 | 3503 |
| j10o10r5lr0.8t29_i2 | 11507 | 1767 | 781.094 | 30290 | 12091 | 900.266 | 1983 |
| j10o10r5lr0.8t30_i3 | 16690.5 | 1447 | 661.907 | 31413 | 17464 | 900.407 | 1915 |
| j10o10r5lr1t24_i1 | 10682 | 1127 | 546.922 | 33994 | 11023.9 | 900.343 | 1761 |
| j10o10r5lr1t23_i2 | 15578 | 1731 | 458.781 | 45250 | 16190.3 | 900.297 | 3123 |
| j10o10r5lr1t25_i3 | 15369 | 754 | 204.937 | 45808 | 16095.9 | 901.047 | 2277 |
| j10o10r5lr1.2t19_i1 | 17710 | 637 | 171.484 | 49363 | 18177.4 | 900.031 | 2707 |
| j10o10r5lr1.2t21_i2 | 12650 | 590 | 220.703 | 54019 | 12867.1 | 900.875 | 1671 |
| j10o10r5lr1.2t19_i3 | 15336 | 679 | 155.765 | 54988 | 15882.9 | 900.078 | 2765 |

# APPENDIX F

| Instance | Best Integer Solution | Node of Best Integer Solution | Time to Best Integer Solution (seconds) | Columns Generated | Best Bound | Runtime (seconds) | Total Nodes |
|---|---|---|---|---|---|---|---|
| j3o3r3lr0.8t4_i1 | 794 | 14 | 2.719 | 41 | 794 | 2.719 | 15 |
| j3o3r3lr0.8t5_i2 | 1135 | 3 | 0.031 | 24 | 1135 | 0.031 | 5 |
| j3o3r3lr0.8t5_i3 | 1807 | 1 | 0.031 | 24 | 1807 | 0.046 | 3 |
| j3o3r3lr1t4_i1 | 1120 | 1 | 0.031 | 23 | 1120 | 0.031 | 3 |
| j3o3r3lr1t4_i2 | 1169 | 3 | 0.031 | 26 | 1169 | 0.031 | 5 |
| j3o3r3lr1t3_i3 | 729 | 8 | 0.046 | 40 | 729 | 0.046 | 9 |
| j3o3r3lr1.2t2_i1 | 92 | 0 | 0 | 1 | 92 | 0.015 | 3 |
| j3o3r3lr1.2t3_i2 | 274 | 0 | 0 | 3 | 274 | 0.031 | 3 |
| j3o3r3lr1.2t3_i3 | 1202 | 1 | 0.031 | 24 | 1202 | 0.031 | 3 |
| j3o5r3lr0.8t8_i1 | 2781 | 23 | 0.312 | 976 | 2825.36 | 1.25 | 105 |
| j3o5r3lr0.8t6_i2 | 3416 | 15 | 0.078 | 92 | 3416 | 0.078 | 17 |
| j3o5r3lr0.8t7_i3 | 2157 | 7 | 0.109 | 150 | 2157.89 | 0.156 | 17 |
| j3o5r3lr1t6_i1 | 2134 | 3 | 0.078 | 73 | 2134 | 0.078 | 5 |
| j3o5r3lr1t6_i2 | 3661 | 7 | 0.078 | 79 | 3661 | 0.093 | 9 |
| j3o5r3lr1t6_i3 | 2377 | 2321 | 36.796 | 9370 | 2395.4 | 36.953 | 2327 |
| j3o5r3lr1.2t5_i1 | 2282 | 41 | 0.265 | 258 | 2282 | 0.312 | 51 |
| j3o5r3lr1.2t6_i2 | 164 | 0 | 0 | 20 | 164 | 0.046 | 5 |
| j3o5r3lr1.2t5_i3 | 988 | 0 | 0 | 1 | 988 | 0.031 | 3 |
| j3o8r5lr0.8t8_i1 | 2235 | 4 | 0.063 | 24 | 2235 | 0.078 | 7 |
| j3o8r5lr0.8t7_i2 | 1889 | 1 | 0.047 | 19 | 1889 | 0.047 | 3 |
| j3o8r5lr0.8t8_i3 | 3034 | 174 | 3.687 | 1891 | 3071.97 | 4.421 | 213 |
| j3o8r5lr1t6_i1 | 266 | 1 | 0.031 | 7 | 266 | 0.031 | 3 |
| j3o8r5lr1t5_i2 | 1175 | 6 | 0.046 | 22 | 1175 | 0.046 | 7 |
| j3o8r5lr1t6_i3 | 1610 | 9 | 0.11 | 79 | 1610 | 0.125 | 11 |
| j3o8r5lr1.2t4_i1 | 649 | 0 | 0 | 1 | 649 | 0.016 | 3 |
| j3o8r5lr1.2t4_i2 | 1102 | 1 | 0.031 | 3 | 1102 | 0.031 | 3 |
| j3o8r5lr1.2t5_i3 | 1806 | 0 | 0.031 | 16 | 1806 | 0.031 | 1 |
| j3o10r5lr0.8t9_i1 | 1620 | 3 | 0.094 | 39 | 1620 | 0.11 | 5 |
| j3o10r5lr0.8t9_i2 | 3938 | 133 | 4.625 | 51436 | 4107.27 | 900.39 | 6239 |
| j3o10r5lr0.8t9_i3 | 2563 | 24 | 0.375 | 1216 | 2582.52 | 3.406 | 259 |
| j3o10r5lr1t8_i1 | 6645 | 427 | 13.469 | 3910 | 6686.4 | 14.422 | 453 |
| j3o10r5lr1t7_i2 | 861 | 0 | 0 | 1 | 861 | 0.062 | 3 |
| j3o10r5lr1t7_i3 | 3171 | 8 | 0.234 | 104 | 3171 | 0.234 | 9 |
| j3o10r5lr1.2t6_i1 | 0 | 0 | 0 | 0 | 0 | 0.031 | 3 |
| j3o10r5lr1.2t5_i2 | 3004 | 1 | 0.031 | 10 | 3004 | 0.047 | 3 |
| j3o10r5lr1.2t6_i3 | 137 | 0 | 0 | 1 | 137 | 0.031 | 3 |
| j5o3r3lr0.8t7_i1 | 1729 | 29 | 0.109 | 103 | 1729 | 0.109 | 31 |
| j5o3r3lr0.8t8_i2 | 2259 | 30 | 0.141 | 98 | 2259 | 0.141 | 31 |
| j5o3r3lr0.8t8_i3 | 2627 | 71 | 0.235 | 155 | 2627 | 0.25 | 73 |
| j5o3r3lr1t6_i1 | 3621 | 15 | 0.094 | 122 | 3621 | 0.094 | 17 |
| j5o3r3lr1t6_i2 | 3031 | 55 | 0.172 | 197 | 3031 | 0.172 | 57 |
| j5o3r3lr1t5_i3 | 3041 | 73 | 0.266 | 396 | 3041 | 0.266 | 75 |
| j5o3r3lr1.2t4_i1 | 1312 | 22 | 0.063 | 97 | 1312 | 0.063 | 23 |
| j5o3r3lr1.2t5_i2 | 1321 | 25 | 0.094 | 105 | 1321 | 0.094 | 29 |
| j5o3r3lr1.2t5_i3 | 2670 | 49 | 0.187 | 1713 | 2674.13 | 1.718 | 325 |
| j5o5r3lr0.8t10_i1 | 2635 | 138 | 1.203 | 50357 | 2675.87 | 900.015 | 14627 |
| j5o5r3lr0.8t14_i2 | 3404 | 87 | 0.953 | 450 | 3404 | 0.985 | 89 |
| j5o5r3lr0.8t12_i3 | 6891 | 107 | 0.797 | 326 | 6891 | 0.813 | 109 |
| j5o5r3lr1t10_i1 | 3862 | 62 | 0.578 | 334 | 3862 | 0.578 | 63 |
| j5o5r3lr1t10_i2 | 3176 | 171 | 2.625 | 61797 | 3221.05 | 900.11 | 8681 |
| j5o5r3lr1t11_i3 | 5992 | 307 | 4.078 | 1633 | 5992 | 4.094 | 309 |
| j5o5r3lr1.2t8_i1 | 1182 | 85 | 0.641 | 409 | 1182 | 0.656 | 87 |

| Instance | Best Integer Solution | Node of Best Integer Solution | Time to Best Integer Solution (seconds) | Columns Generated | Best Bound | Runtime (seconds) | Total Nodes |
|---|---|---|---|---|---|---|---|
| j5o5r3lr1.2t9_i2 | 3114 | 2024 | 74.672 | 18289 | 3136 | 87.172 | 2201 |
| j5o5r3lr1.2t8_i3 | 1700 | 14 | 0.109 | 71 | 1700 | 0.109 | 15 |
| j5o8r5lr0.8t11_i1 | 2725 | 19 | 0.422 | 439 | 2735.96 | 1.11 | 59 |
| j5o8r5lr0.8t13_i2 | 8630.5 | 3562 | 544.39 | 64786 | 8773.76 | 900.343 | 4615 |
| j5o8r5lr0.8t12_i3 | 11059 | 149 | 5.656 | 61079 | 11209.7 | 900.406 | 6341 |
| j5o8r5lr1t9_i1 | 1559 | 6429 | 622.36 | 50076 | 1588.72 | 711.875 | 6915 |
| j5o8r5lr1t9_i2 | 3650 | 77 | 2.156 | 59277 | 3736.14 | 900.516 | 7323 |
| j5o8r5lr1t9_i3 | 2751 | 101 | 2.813 | 3835 | 2802.8 | 13.141 | 499 |
| j5o8r5lr1.2t9_i1 | 3752 | 164 | 4.625 | 1790 | 3764.19 | 5.672 | 199 |
| j5o8r5lr1.2t8_i2 | 2059 | 83 | 1.703 | 4518 | 2120.61 | 15.188 | 627 |
| j5o8r5lr1.2t8_i3 | 4229 | 0 | 0 | 48534 | 7121.13 | 900.531 | 9111 |
| j5o10r5lr0.8t15_i1 | 6774 | 1195 | 109.094 | 56508 | 6918.58 | 900.531 | 4719 |
| j5o10r5lr0.8t14_i2 | 2970 | 442 | 22.031 | 59252 | 3041.38 | 900.14 | 5843 |
| j5o10r5lr0.8t16_i3 | 9826 | 3908 | 813.172 | 55019 | 9967.47 | 900.172 | 4123 |
| j5o10r5lr1t12_i1 | 5693 | 259 | 25.578 | 57217 | 6023.01 | 900.61 | 4753 |
| j5o10r5lr1t12_i2 | 6277.33 | 450 | 31.047 | 52202 | 6334.43 | 900.281 | 6589 |
| j5o10r5lr1t12_i3 | 6011 | 413 | 44.344 | 61202 | 6317.32 | 900.313 | 3545 |
| j5o10r5lr1.2t10_i1 | 3615 | 51 | 2 | 1028 | 3616.29 | 3.5 | 103 |
| j5o10r5lr1.2t10_i2 | 5659 | 313 | 22.906 | 53155 | 5842.08 | 900.343 | 6117 |
| j5o10r5lr1.2t10_i3 | 4361 | 1098 | 69.516 | 56294 | 4497.4 | 900.141 | 5753 |
| j8o3r3lr0.8t13_i1 | 2295 | 231 | 2.188 | 851 | 2295 | 2.188 | 233 |
| j8o3r3lr0.8t13_i2 | 4341 | 109 | 0.812 | 312 | 4341 | 0.828 | 111 |
| j8o3r3lr0.8t11_i3 | 4193 | 57 | 0.344 | 217 | 4193 | 0.36 | 59 |
| j8o3r3lr1t9_i1 | 2889 | 65 | 0.281 | 167 | 2889 | 0.297 | 69 |
| j8o3r3lr1t9_i2 | 2069 | 260 | 1.281 | 547 | 2069 | 1.281 | 261 |
| j8o3r3lr1t10_i3 | 3218 | 50 | 0.25 | 149 | 3218 | 0.25 | 51 |
| j8o3r3lr1.2t8_i1 | 2963 | 44 | 0.281 | 248 | 2963 | 0.281 | 45 |
| j8o3r3lr1.2t8_i2 | 5436 | 193 | 1.156 | 777 | 5436 | 1.156 | 195 |
| j8o3r3lr1.2t8_i3 | 3202 | 241 | 1.406 | 766 | 3202 | 1.406 | 243 |
| j8o5r3lr0.8t20_i1 | 5940 | 155 | 2.469 | 613 | 5940 | 2.5 | 157 |
| j8o5r3lr0.8t20_i2 | 5941 | 566 | 12.203 | 1773 | 5941 | 12.203 | 567 |
| j8o5r3lr0.8t20_i3 | 6942 | 9293 | 679.485 | 53960 | 6954 | 900.172 | 10845 |
| j8o5r3lr1t17_i1 | 6739 | 388 | 14.235 | 64480 | 6915.17 | 900.203 | 5817 |
| j8o5r3lr1t14_i2 | 5724 | 493 | 7.64 | 2123 | 5724 | 7.656 | 495 |
| j8o5r3lr1t16_i3 | 4009 | 128 | 1.313 | 429 | 4009 | 1.313 | 129 |
| j8o5r3lr1.2t14_i1 | 5373 | 583 | 16.094 | 64322 | 5405.5 | 900.016 | 7217 |
| j8o5r3lr1.2t14_i2 | 5759 | 431 | 7.875 | 56914 | 5807 | 900.234 | 9699 |
| j8o5r3lr1.2t12_i3 | 3991 | 464 | 9.688 | 64058 | 4029 | 900.063 | 7377 |
| j8o8r5lr0.8t21_i1 | 7046 | 719 | 155.563 | 49896 | 7252 | 900.641 | 3553 |
| j8o8r5lr0.8t18_i2 | 6748 | 568 | 29.781 | 57211 | 6802 | 900.156 | 5455 |
| j8o8r5lr0.8t19_i3 | 14641 | 428 | 66.672 | 58312 | 15031.8 | 900.641 | 3713 |
| j8o8r5lr1t15_i1 | 15751 | 354 | 49.5 | 57209 | 16230.2 | 900.844 | 3441 |
| j8o8r5lr1t16_i2 | 9331 | 974 | 178.297 | 50851 | 11032.9 | 900.125 | 3841 |
| j8o8r5lr1t16_i3 | 10889 | 1020 | 133.062 | 52694 | 10996 | 900.109 | 4179 |
| j8o8r5lr1.2t13_i1 | 10082 | 2381 | 439.156 | 61897 | 10369.6 | 900.735 | 3883 |
| j8o8r5lr1.2t13_i2 | 7154 | 318 | 33.141 | 60971 | 7347.76 | 900.438 | 3921 |
| j8o8r5lr1.2t13_i3 | 4484 | 1634 | 197.438 | 67009 | 4546.92 | 900.36 | 4003 |
| j8o10r5lr0.8t26_i1 | 17087 | 1408 | 465.718 | 43904 | 17432 | 900.453 | 2327 |
| j8o10r5lr0.8t23_i2 | 11107 | 1334 | 368.563 | 47942 | 11258 | 900.797 | 2373 |
| j8o10r5lr0.8t23_i3 | 18045 | 1465 | 284.531 | 48316 | 18213 | 900.219 | 3083 |
| j8o10r5lr1t19_i1 | 12758 | 388 | 77.516 | 57787 | 13067.4 | 900.125 | 2839 |
| j8o10r5lr1t17_i2 | 9056 | 422 | 59.125 | 59340 | 9354.86 | 900.188 | 3939 |
| j8o10r5lr1t19_i3 | 8721 | 554 | 85.922 | 57969 | 8870.84 | 900.594 | 3351 |
| j8o10r5lr1.2t15_i1 | 8803 | 1977 | 522.125 | 54429 | 9225.48 | 900.235 | 2911 |

| Instance | Best Integer Solution | Node of Best Integer Solution | Time to Best Integer Solution (seconds) | Columns Generated | Best Bound | Runtime (seconds) | Total Nodes |
|---|---|---|---|---|---|---|---|
| j8o10r5lr1.2t16_i2 | 14616 | 482 | 140.094 | 50381 | 15121 | 900.532 | 1851 |
| j8o10r5lr1.2t16_i3 | 10269 | 721 | 151.938 | 52648 | 10665.6 | 900.329 | 3263 |
| j10o3r3lr0.8t15_i1 | 7366 | 205 | 2.281 | 595 | 7366 | 2.297 | 207 |
| j10o3r3lr0.8t15_i2 | 3529 | 317 | 3.812 | 1064 | 3529 | 3.843 | 319 |
| j10o3r3lr0.8t14_i3 | 4676 | 303 | 2.766 | 644 | 4676 | 2.766 | 305 |
| j10o3r3lr1t12_i1 | 3639 | 186 | 1.422 | 388 | 3639 | 1.422 | 187 |
| j10o3r3lr1t12_i2 | 5218 | 198 | 2.438 | 933 | 5218 | 2.438 | 199 |
| j10o3r3lr1t11_i3 | 4658 | 166 | 1.297 | 556 | 4658 | 1.297 | 167 |
| j10o3r3lr1.2t10_i1 | 4683 | 197 | 1.469 | 569 | 4683 | 1.485 | 199 |
| j10o3r3lr1.2t9_i2 | 4373 | 218 | 1.296 | 621 | 4373 | 1.296 | 219 |
| j10o3r3lr1.2t10_i3 | 4306 | 221 | 2.046 | 50546 | 4323.5 | 900.015 | 13921 |
| j10o5r3lr0.8t26_i1 | 9833 | 744 | 29.922 | 2439 | 9833 | 29.922 | 745 |
| j10o5r3lr0.8t28_i2 | 11760 | 1770 | 180.312 | 47298 | 11817 | 900.437 | 5643 |
| j10o5r3lr0.8t25_i3 | 9189 | 499 | 15.812 | 1575 | 9189 | 15.953 | 501 |
| j10o5r3lr1t20_i1 | 4672 | 481 | 10.39 | 1284 | 4672 | 10.422 | 483 |
| j10o5r3lr1t19_i2 | 6145 | 1175 | 45.641 | 56551 | 6156 | 900.172 | 8419 |
| j10o5r3lr1t19_i3 | 7072 | 399 | 12.906 | 2070 | 7072 | 13.015 | 405 |
| j10o5r3lr1.2t16_i1 | 3884 | 747 | 19.141 | 64747 | 3896 | 900.453 | 8487 |
| j10o5r3lr1.2t18_i2 | 6193 | 200 | 5.203 | 55699 | 6221 | 900.203 | 9411 |
| j10o5r3lr1.2t18_i3 | 7350 | 272 | 6.906 | 1379 | 7350 | 6.906 | 273 |
| j10o8r5lr0.8t24_i1 | 13218.5 | 877 | 120.672 | 44386 | 13293 | 900.829 | 3333 |
| j10o8r5lr0.8t24_i2 | 10518 | 1849 | 396.969 | 37037 | 10589.1 | 900.234 | 3921 |
| j10o8r5lr0.8t22_i3 | 11262 | 1556 | 202.953 | 47638 | 11289 | 900.656 | 4641 |
| j10o8r5lr1t17_i1 | 12053 | 2140 | 507.344 | 52869 | 12294 | 900.484 | 3277 |
| j10o8r5lr1t18_i2 | 8552 | 227 | 16.5 | 63420 | 8716.18 | 900.14 | 3879 |
| j10o8r5lr1t19_i3 | 9162 | 983 | 119.672 | 57238 | 9346 | 900.579 | 3847 |
| j10o8r5lr1.2t16_i1 | 13469 | 365 | 37.469 | 52620 | 13790 | 900.579 | 4915 |
| j10o8r5lr1.2t16_i2 | 13040 | 997 | 118.219 | 57951 | 13293 | 900.063 | 4237 |
| j10o8r5lr1.2t16_i3 | 9951 | 3383 | 805.156 | 54541 | 10160.5 | 900.281 | 3649 |
| j10o10r5lr0.8t29_i1 | 18435 | 884 | 146.593 | 30370 | 18491 | 900.828 | 3361 |
| j10o10r5lr0.8t29_i2 | 11507 | 1767 | 780.703 | 27938 | 12091 | 900.297 | 2211 |
| j10o10r5lr0.8t30_i3 | 16690.5 | 1447 | 662.328 | 29949 | 17464 | 902.219 | 1919 |
| j10o10r5lr1t24_i1 | 10682 | 1127 | 547.016 | 37154 | 11023.9 | 900.406 | 1741 |
| j10o10r5lr1t23_i2 | 15578 | 773 | 219.25 | 47741 | 16092.2 | 900.719 | 2983 |
| j10o10r5lr1t25_i3 | 15369 | 754 | 204.968 | 41056 | 15980.6 | 900.562 | 2435 |
| j10o10r5lr1.2t19_i1 | 17632 | 636 | 170.969 | 50688 | 18177.4 | 900.172 | 2417 |
| j10o10r5lr1.2t21_i2 | 12637 | 951 | 361.156 | 49398 | 12867 | 901.719 | 1933 |
| j10o10r5lr1.2t19_i3 | 15726 | 1229 | 299.328 | 43617 | 15882.9 | 900.688 | 3377 |

# APPENDIX G

| Instance | Best Integer Solution | Best Integer @ node | Time to Best Integer Solution (seconds) | Number of Columns | Best Bound | Runtime (seconds) | Total Nodes |
|---|---|---|---|---|---|---|---|
| j3o3r3lr0.8t4_i1 | 794 | 15 | 0.031 | 51 | 794 | 0.031 | 17 |
| j3o3r3lr0.8t5_i2 | 1135 | 3 | 0.016 | 24 | 1135 | 0.031 | 5 |
| j3o3r3lr0.8t5_i3 | 1807 | 1 | 0.016 | 24 | 1807 | 0.031 | 3 |
| j3o3r3lr1t4_i1 | 1120 | 1 | 0.031 | 23 | 1120 | 0.031 | 3 |
| j3o3r3lr1t4_i2 | 1169 | 3 | 0.031 | 26 | 1169 | 0.031 | 5 |
| j3o3r3lr1t3_i3 | 729 | 8 | 0.031 | 40 | 729 | 0.031 | 9 |
| j3o3r3lr1.2t2_i1 | 92 | 0 | 0 | 1 | 92 | 0.016 | 3 |
| j3o3r3lr1.2t3_i2 | 274 | 0 | 0 | 3 | 274 | 0.016 | 3 |
| j3o3r3lr1.2t3_i3 | 1202 | 1 | 0.031 | 24 | 1202 | 0.031 | 3 |
| j3o5r3lr0.8t8_i1 | 2781 | 165 | 2.484 | 2110 | 2782 | 3.343 | 233 |
| j3o5r3lr0.8t6_i2 | 3416 | 11 | 0.078 | 82 | 3416 | 0.078 | 13 |
| j3o5r3lr0.8t7_i3 | 2157 | 19 | 0.203 | 183 | 2157 | 0.203 | 21 |
| j3o5r3lr1t6_i1 | 2134 | 3 | 0.063 | 73 | 2134 | 0.063 | 5 |
| j3o5r3lr1t6_i2 | 3661 | 7 | 0.063 | 79 | 3661 | 0.078 | 9 |
| j3o5r3lr1t6_i3 | 2377 | 565 | 4.875 | 2709 | 2377 | 4.891 | 567 |
| j3o5r3lr1.2t5_i1 | 2282 | 35 | 0.281 | 258 | 2282 | 0.297 | 41 |
| j3o5r3lr1.2t6_i2 | 164 | 0 | 0 | 20 | 164 | 0.047 | 5 |
| j3o5r3lr1.2t5_i3 | 988 | 0 | 0 | 1 | 988 | 0.016 | 3 |
| j3o8r5lr0.8t8_i1 | 2235 | 4 | 0.063 | 24 | 2235 | 0.078 | 7 |
| j3o8r5lr0.8t7_i2 | 1889 | 1 | 0.046 | 19 | 1889 | 0.062 | 3 |
| j3o8r5lr0.8t8_i3 | 3034 | 442 | 13.171 | 4244 | 3034.5 | 13.171 | 443 |
| j3o8r5lr1t6_i1 | 266 | 1 | 0.046 | 7 | 266 | 0.046 | 3 |
| j3o8r5lr1t5_i2 | 1175 | 6 | 0.046 | 22 | 1175 | 0.046 | 7 |
| j3o8r5lr1t6_i3 | 1610 | 9 | 0.11 | 79 | 1610 | 0.125 | 11 |
| j3o8r5lr1.2t4_i1 | 649 | 0 | 0 | 1 | 649 | 0.015 | 3 |
| j3o8r5lr1.2t4_i2 | 1102 | 1 | 0.031 | 3 | 1102 | 0.031 | 3 |
| j3o8r5lr1.2t5_i3 | 1806 | 0 | 0.046 | 16 | 1806 | 0.046 | 1 |
| j3o10r5lr0.8t9_i1 | 1620 | 3 | 0.094 | 39 | 1620 | 0.125 | 5 |
| j3o10r5lr0.8t9_i2 | 2375 | 0 | 0 | 50377 | 4047.08 | 900.016 | 6177 |
| j3o10r5lr0.8t9_i3 | 2563 | 367 | 5.578 | 1959 | 2563.49 | 6.156 | 403 |
| j3o10r5lr1t8_i1 | 6645 | 621 | 19.203 | 8330 | 6645.86 | 41.875 | 1115 |
| j3o10r5lr1t7_i2 | 861 | 0 | 0 | 1 | 861 | 0.031 | 3 |
| j3o10r5lr1t7_i3 | 3171 | 8 | 0.219 | 104 | 3171 | 0.219 | 9 |
| j3o10r5lr1.2t6_i1 | 1E-07 | 0 | 0 | 0 | 1E-07 | 0.031 | 3 |
| j3o10r5lr1.2t5_i2 | 3004 | 1 | 0.031 | 10 | 3004 | 0.031 | 3 |
| j3o10r5lr1.2t6_i3 | 137 | 0 | 0 | 1 | 137 | 0.031 | 3 |
| j5o3r3lr0.8t7_i1 | 1729 | 23 | 0.078 | 75 | 1729 | 0.078 | 25 |
| j5o3r3lr0.8t8_i2 | 2259 | 179 | 0.641 | 343 | 2259 | 0.641 | 181 |
| j5o3r3lr0.8t8_i3 | 2627 | 986 | 4.047 | 1780 | 2627 | 4.047 | 987 |
| j5o3r3lr1t6_i1 | 3621 | 50 | 0.25 | 291 | 3621 | 0.25 | 51 |
| j5o3r3lr1t6_i2 | 3031 | 3650 | 44.235 | 10508 | 3031 | 44.235 | 3651 |
| j5o3r3lr1t5_i3 | 3041 | 1601 | 17.485 | 7889 | 3041 | 17.5 | 1603 |
| j5o3r3lr1.2t4_i1 | 1312 | 23 | 0.062 | 100 | 1312 | 0.078 | 25 |
| j5o3r3lr1.2t5_i2 | 1321 | 41 | 0.172 | 193 | 1321 | 0.188 | 43 |
| j5o3r3lr1.2t5_i3 | 2670 | 1073 | 11.046 | 5603 | 2670 | 11.062 | 1075 |
| j5o5r3lr0.8t10_i1 | 2657 | 10314 | 506.281 | 36648 | 2657 | 506.313 | 10315 |
| j5o5r3lr0.8t14_i2 | 3404 | 2893 | 34.891 | 6135 | 3404 | 34.922 | 2895 |
| j5o5r3lr0.8t12_i3 | 6891 | 16196 | 783.297 | 40542 | 6891 | 783.328 | 16197 |
| j5o5r3lr1t10_i1 | 3411 | 0 | 0 | 50117 | 3862 | 900.141 | 14813 |
| j5o5r3lr1t10_i2 | 2658 | 0 | 0 | 69198 | 3218.78 | 900.218 | 6775 |
| j5o5r3lr1t11_i3 | 5767 | 0 | 0 | 48036 | 5992 | 900.141 | 15391 |

| Instance | Best Integer Solution | Best Integer @ node | Time to Best Integer Solution (seconds) | Number of Columns | Best Bound | Runtime (seconds) | Total Nodes |
|---|---|---|---|---|---|---|---|
| j5o5r3lr1.2t8_i1 | 1182 | 735 | 7.281 | 3054 | 1182 | 7.297 | 737 |
| j5o5r3lr1.2t9_i2 | 3114 | 1678 | 78.359 | 19339 | 3114.88 | 91.125 | 1859 |
| j5o5r3lr1.2t8_i3 | 1700 | 16 | 0.125 | 76 | 1700 | 0.125 | 17 |
| j5o8r5lr0.8t11_i1 | 2725 | 26 | 0.485 | 667 | 2725 | 1.656 | 87 |
| j5o8r5lr0.8t13_i2 | 7905 | 0 | 0 | 58321 | 8748.8 | 900.063 | 5301 |
| j5o8r5lr0.8t12_i3 | 10094 | 0 | 0 | 58946 | 11184.8 | 900.203 | 6247 |
| j5o8r5lr1t9_i1 | 1259 | 213 | 3.531 | 54681 | 1571.88 | 900.485 | 7447 |
| j5o8r5lr1t9_i2 | 2127 | 0 | 0 | 58608 | 3700.03 | 900.297 | 6981 |
| j5o8r5lr1t9_i3 | 2767.8 | 2742 | 186.656 | 32011 | 2768.71 | 328.469 | 3947 |
| j5o8r5lr1.2t9_i1 | 3752 | 214 | 6.766 | 1962 | 3752 | 6.781 | 215 |
| j5o8r5lr1.2t8_i2 | 2059 | 525 | 15.61 | 10551 | 2060 | 48.391 | 1313 |
| j5o8r5lr1.2t8_i3 | 4229 | 0 | 0 | 67428 | 7090.16 | 900.438 | 5813 |
| j5o10r5lr0.8t15_i1 | 5419 | 0 | 0 | 52996 | 6902.17 | 900.328 | 5267 |
| j5o10r5lr0.8t14_i2 | 755 | 0 | 0 | 59933 | 2985.11 | 900.5 | 4389 |
| j5o10r5lr0.8t16_i3 | 7241 | 0 | 0 | 61672 | 9955.26 | 900.844 | 3495 |
| j5o10r5lr1t12_i1 | 4123 | 0 | 0 | 66217 | 6001.42 | 900.219 | 2713 |
| j5o10r5lr1t12_i2 | 2828 | 0 | 0 | 52691 | 6307.99 | 900.156 | 5863 |
| j5o10r5lr1t12_i3 | 4177 | 0 | 0 | 66364 | 6309.76 | 900.75 | 3267 |
| j5o10r5lr1.2t10_i1 | 3615 | 236 | 8.047 | 2087 | 3615 | 8.063 | 237 |
| j5o10r5lr1.2t10_i2 | 5019 | 0 | 0 | 61266 | 5772.75 | 901.063 | 3559 |
| j5o10r5lr1.2t10_i3 | 2185 | 0 | 0 | 56904 | 4468.03 | 900.766 | 5445 |
| j8o3r3lr0.8t13_i1 | 2263 | 0 | 0 | 53845 | 2295 | 900.203 | 15907 |
| j8o3r3lr0.8t13_i2 | 4341 | 43 | 0.36 | 250 | 4341 | 0.375 | 45 |
| j8o3r3lr0.8t11_i3 | 4057 | 0 | 0 | 45159 | 4193 | 900.031 | 19415 |
| j8o3r3lr1t9_i1 | 2889 | 2475 | 17.906 | 4528 | 2889 | 17.906 | 2477 |
| j8o3r3lr1t9_i2 | 1616 | 0 | 0 | 43764 | 2069 | 900.11 | 20159 |
| j8o3r3lr1t10_i3 | 3218 | 1376 | 9.141 | 2797 | 3218 | 9.141 | 1377 |
| j8o3r3lr1.2t8_i1 | 2963 | 50 | 0.312 | 306 | 2963 | 0.312 | 51 |
| j8o3r3lr1.2t8_i2 | 5436 | 10210 | 361.781 | 33668 | 5436 | 361.796 | 10211 |
| j8o3r3lr1.2t8_i3 | 2641 | 0 | 0 | 51307 | 3202 | 900.063 | 17485 |
| j8o5r3lr0.8t20_i1 | 3993 | 0 | 0 | 41417 | 5940 | 900.016 | 16305 |
| j8o5r3lr0.8t20_i2 | 5681 | 0 | 0 | 42904 | 5941 | 900.141 | 14067 |
| j8o5r3lr0.8t20_i3 | 6630 | 0 | 0 | 41107 | 6954 | 900.062 | 14283 |
| j8o5r3lr1t17_i1 | 5833 | 0 | 0 | 40237 | 6915.17 | 900.015 | 13483 |
| j8o5r3lr1t14_i2 | 4989 | 0 | 0 | 42307 | 5724 | 900.031 | 17739 |
| j8o5r3lr1t16_i3 | 1698 | 0 | 0 | 43244 | 4009 | 900.125 | 16919 |
| j8o5r3lr1.2t14_i1 | 4338 | 0 | 0 | 53583 | 5405.5 | 900.172 | 13445 |
| j8o5r3lr1.2t14_i2 | 4040 | 0 | 0 | 53823 | 5807 | 900.125 | 12943 |
| j8o5r3lr1.2t12_i3 | 3224 | 0 | 0 | 4834 | 4029 | 358.985 | 30001 |
| j8o8r5lr0.8t21_i1 | 6360 | 0 | 0 | 56119 | 7252 | 900.156 | 4865 |
| j8o8r5lr0.8t18_i2 | 5514 | 0 | 0 | 47939 | 6802 | 900.094 | 10717 |
| j8o8r5lr0.8t19_i3 | 13046 | 0 | 0 | 54221 | 15023.6 | 900.078 | 3171 |
| j8o8r5lr1t15_i1 | 14347 | 0 | 0 | 63849 | 16213.5 | 900.328 | 3297 |
| j8o8r5lr1t16_i2 | 8145 | 0 | 0 | 54627 | 11032.9 | 900.187 | 5791 |
| j8o8r5lr1t16_i3 | 9819 | 0 | 0 | 59293 | 10996 | 900.437 | 6825 |
| j8o8r5lr1.2t13_i1 | 6443 | 0 | 0 | 60277 | 10350.5 | 900.453 | 3763 |
| j8o8r5lr1.2t13_i2 | 5481 | 0 | 0 | 68924 | 7344.64 | 900.984 | 2729 |
| j8o8r5lr1.2t13_i3 | 2824 | 0 | 0 | 61133 | 4535.7 | 900.453 | 3663 |
| j8o10r5lr0.8t26_i1 | 15038 | 0 | 0 | 48761 | 17432 | 900.437 | 5053 |
| j8o10r5lr0.8t23_i2 | 8724 | 0 | 0 | 49889 | 11258 | 900.484 | 5537 |
| j8o10r5lr0.8t23_i3 | 16541 | 0 | 0 | 50786 | 18213 | 900.188 | 6993 |
| j8o10r5lr1t19_i1 | 7217 | 0 | 0 | 49206 | 13065.4 | 900.109 | 2717 |
| j8o10r5lr1t17_i2 | 7382 | 0 | 0 | 55177 | 9343.65 | 900.703 | 3721 |

| Instance | Best Integer Solution | Best Integer @ node | Time to Best Integer Solution (seconds) | Number of Columns | Best Bound | Runtime (seconds) | Total Nodes |
|---|---|---|---|---|---|---|---|
| j8o10r5lr1t19_i3 | 5850 | 0 | 0 | 55114 | 8867.88 | 900.25 | 4031 |
| j8o10r5lr1.2t15_i1 | 5440 | 0 | 0 | 56454 | 9210.53 | 900.125 | 2759 |
| j8o10r5lr1.2t16_i2 | 13092 | 0 | 0 | 49455 | 15121 | 900.125 | 3771 |
| j8o10r5lr1.2t16_i3 | 8131 | 0 | 0 | 58395 | 10652.4 | 900.625 | 2725 |
| j10o3r3lr0.8t15_i1 | 6757 | 0 | 0 | 50850 | 7366 | 900.125 | 16497 |
| j10o3r3lr0.8t15_i2 | 3529 | 14375 | 640.75 | 37731 | 3529 | 640.906 | 14377 |
| j10o3r3lr0.8t14_i3 | 4532 | 0 | 0 | 48726 | 4676 | 900.016 | 17423 |
| j10o3r3lr1t12_i1 | 2813 | 0 | 0 | 39861 | 3639 | 900.047 | 20659 |
| j10o3r3lr1t12_i2 | 4781 | 0 | 0 | 50621 | 5218 | 900.125 | 15111 |
| j10o3r3lr1t11_i3 | 3663 | 0 | 0 | 51481 | 4658 | 900.031 | 16873 |
| j10o3r3lr1.2t10_i1 | 3542 | 0 | 0 | 1701 | 4683 | 194.64 | 30001 |
| j10o3r3lr1.2t9_i2 | 3125 | 0 | 0 | 47009 | 4373 | 900.015 | 19059 |
| j10o3r3lr1.2t10_i3 | 2989 | 0 | 0 | 50292 | 4323.5 | 900.188 | 17097 |
| j10o5r3lr0.8t26_i1 | 9405 | 0 | 0 | 5649 | 9833 | 827.657 | 30001 |
| j10o5r3lr0.8t28_i2 | 10474 | 0 | 0 | 49126 | 11817 | 900 | 9115 |
| j10o5r3lr0.8t25_i3 | 8585 | 0 | 0 | 43950 | 9189 | 900.125 | 12283 |
| j10o5r3lr1t20_i1 | 4215 | 0 | 0 | 36826 | 4672 | 900.094 | 17207 |
| j10o5r3lr1t19_i2 | 5903 | 0 | 0 | 57187 | 6156 | 900.094 | 10189 |
| j10o5r3lr1t19_i3 | 6083 | 0 | 0 | 53297 | 7072 | 900.25 | 12093 |
| j10o5r3lr1.2t16_i1 | 3062 | 0 | 0 | 48248 | 3896 | 900.047 | 13917 |
| j10o5r3lr1.2t18_i2 | 4564 | 0 | 0 | 45725 | 6221 | 900.094 | 14467 |
| j10o5r3lr1.2t18_i3 | 4050 | 0 | 0 | 43211 | 7350 | 900 | 14153 |
| j10o8r5lr0.8t24_i1 | 12568 | 0 | 0 | 39774 | 13293 | 900.172 | 6353 |
| j10o8r5lr0.8t24_i2 | 9930 | 0 | 0 | 48398 | 10589.1 | 900.109 | 6271 |
| j10o8r5lr0.8t22_i3 | 10337 | 0 | 0 | 54217 | 11289 | 900.188 | 6987 |
| j10o8r5lr1t17_i1 | 10408 | 0 | 0 | 55672 | 12294 | 900.141 | 7205 |
| j10o8r5lr1t18_i2 | 6189 | 0 | 0 | 61227 | 8713.3 | 900.297 | 5383 |
| j10o8r5lr1t19_i3 | 6268 | 0 | 0 | 46532 | 9346 | 900.157 | 7299 |
| j10o8r5lr1.2t16_i1 | 7276 | 0 | 0 | 61194 | 13790 | 900.594 | 7185 |
| j10o8r5lr1.2t16_i2 | 10204 | 0 | 0 | 59153 | 13293 | 900.047 | 7371 |
| j10o8r5lr1.2t16_i3 | 8061 | 0 | 0 | 53898 | 10160.5 | 900.047 | 4593 |
| j10o10r5lr0.8t29_i1 | 15839 | 0 | 0 | 30209 | 18491 | 900.359 | 6155 |
| j10o10r5lr0.8t29_i2 | 10919 | 0 | 0 | 29526 | 12091 | 900.015 | 4593 |
| j10o10r5lr0.8t30_i3 | 15273 | 0 | 0 | 32658 | 17464 | 900.234 | 6347 |
| j10o10r5lr1t24_i1 | 9028 | 0 | 0 | 51289 | 11023.9 | 901.453 | 2175 |
| j10o10r5lr1t23_i2 | 7497 | 0 | 0 | 47829 | 16092.2 | 900.344 | 3811 |
| j10o10r5lr1t25_i3 | 11700 | 0 | 0 | 27377 | 15980.6 | 900.015 | 7251 |
| j10o10r5lr1.2t19_i1 | 13871 | 0 | 0 | 43390 | 18177.4 | 900.469 | 5379 |
| j10o10r5lr1.2t21_i2 | 6483 | 0 | 0 | 49759 | 12864.4 | 901.047 | 1957 |
| j10o10r5lr1.2t19_i3 | 10791 | 0 | 0 | 47793 | 15882.9 | 900.016 | 5003 |

# APPENDIX H

| Instance | Best Integer Solution | Best Integer @ node | Time to Best Integer Solution (seconds) | Number of Columns | Best Bound | Runtime (seconds) | Total Nodes |
|---|---|---|---|---|---|---|---|
| j3o3r3lr0.8t4_i1 | 794 | 14 | 0.063 | 41 | 794 | 0.063 | 15 |
| j3o3r3lr0.8t5_i2 | 1135 | 3 | 0.031 | 24 | 1135 | 0.031 | 5 |
| j3o3r3lr0.8t5_i3 | 1807 | 1 | 0.031 | 24 | 1807 | 0.031 | 3 |
| j3o3r3lr1t4_i1 | 1120 | 1 | 0.016 | 23 | 1120 | 0.031 | 3 |
| j3o3r3lr1t4_i2 | 1169 | 3 | 0.016 | 26 | 1169 | 0.031 | 5 |
| j3o3r3lr1t3_i3 | 729 | 8 | 0.031 | 40 | 729 | 0.031 | 9 |
| j3o3r3lr1.2t2_i1 | 92 | 0 | 0 | 1 | 92 | 0.016 | 3 |
| j3o3r3lr1.2t3_i2 | 274 | 0 | 0 | 3 | 274 | 0.016 | 3 |
| j3o3r3lr1.2t3_i3 | 1202 | 1 | 0.031 | 24 | 1202 | 0.031 | 3 |
| j3o5r3lr0.8t8_i1 | 2781 | 23 | 0.312 | 711 | 2825.36 | 0.89 | 71 |
| j3o5r3lr0.8t6_i2 | 3416 | 15 | 0.093 | 92 | 3416 | 0.093 | 17 |
| j3o5r3lr0.8t7_i3 | 2157 | 7 | 0.11 | 109 | 2157 | 0.125 | 9 |
| j3o5r3lr1t6_i1 | 2134 | 3 | 0.063 | 73 | 2134 | 0.063 | 5 |
| j3o5r3lr1t6_i2 | 3661 | 7 | 0.078 | 79 | 3661 | 0.078 | 9 |
| j3o5r3lr1t6_i3 | 2367 | 649 | 5.687 | 2972 | 2413.7 | 6.234 | 691 |
| j3o5r3lr1.2t5_i1 | 2282 | 41 | 0.281 | 228 | 2282 | 0.281 | 43 |
| j3o5r3lr1.2t6_i2 | 164 | 0 | 0 | 20 | 164 | 0.062 | 5 |
| j3o5r3lr1.2t5_i3 | 988 | 0 | 0 | 1 | 988 | 0.031 | 3 |
| j3o8r5lr0.8t8_i1 | 2235 | 4 | 0.063 | 20 | 2235 | 0.063 | 5 |
| j3o8r5lr0.8t7_i2 | 1889 | 1 | 0.047 | 19 | 1889 | 0.047 | 3 |
| j3o8r5lr0.8t8_i3 | 3023 | 101 | 2.359 | 1172 | 3077.92 | 2.562 | 111 |
| j3o8r5lr1t6_i1 | 266 | 1 | 0.047 | 7 | 266 | 0.047 | 3 |
| j3o8r5lr1t5_i2 | 1175 | 6 | 0.047 | 22 | 1175 | 0.047 | 7 |
| j3o8r5lr1t6_i3 | 1610 | 9 | 0.14 | 79 | 1610 | 0.156 | 11 |
| j3o8r5lr1.2t4_i1 | 649 | 0 | 0 | 1 | 649 | 0.031 | 3 |
| j3o8r5lr1.2t4_i2 | 1102 | 1 | 0.063 | 3 | 1102 | 0.078 | 3 |
| j3o8r5lr1.2t5_i3 | 1806 | 0 | 0.062 | 16 | 1806 | 0.062 | 1 |
| j3o10r5lr0.8t9_i1 | 1620 | 3 | 0.11 | 39 | 1620 | 0.125 | 5 |
| j3o10r5lr0.8t9_i2 | 3951.5 | 374 | 15.219 | 53432 | 4117.38 | 900.11 | 6097 |
| j3o10r5lr0.8t9_i3 | 2563 | 24 | 0.375 | 197 | 2632.48 | 0.5 | 31 |
| j3o10r5lr1t8_i1 | 6589 | 83 | 2.484 | 1651 | 6691.83 | 4.984 | 197 |
| j3o10r5lr1t7_i2 | 861 | 0 | 0 | 1 | 861 | 0.031 | 3 |
| j3o10r5lr1t7_i3 | 3171 | 8 | 0.219 | 104 | 3171 | 0.219 | 9 |
| j3o10r5lr1.2t6_i1 | 1E-07 | 0 | 0 | 0 | 1E-07 | 0.031 | 3 |
| j3o10r5lr1.2t5_i2 | 3004 | 1 | 0.046 | 10 | 3004 | 0.046 | 3 |
| j3o10r5lr1.2t6_i3 | 137 | 0 | 0 | 1 | 137 | 0.031 | 3 |
| j5o3r3lr0.8t7_i1 | 1729 | 29 | 0.109 | 103 | 1729 | 0.109 | 31 |
| j5o3r3lr0.8t8_i2 | 2259 | 30 | 0.141 | 98 | 2259 | 0.141 | 31 |
| j5o3r3lr0.8t8_i3 | 2627 | 71 | 0.25 | 155 | 2627 | 0.25 | 73 |
| j5o3r3lr1t6_i1 | 3621 | 15 | 0.094 | 122 | 3621 | 0.094 | 17 |
| j5o3r3lr1t6_i2 | 3031 | 55 | 0.171 | 197 | 3031 | 0.187 | 57 |
| j5o3r3lr1t5_i3 | 3041 | 73 | 0.281 | 396 | 3041 | 0.297 | 75 |
| j5o3r3lr1.2t4_i1 | 1312 | 22 | 0.062 | 97 | 1312 | 0.062 | 23 |
| j5o3r3lr1.2t5_i2 | 1321 | 25 | 0.093 | 105 | 1321 | 0.109 | 29 |
| j5o3r3lr1.2t5_i3 | 2670 | 49 | 0.171 | 223 | 2670 | 0.187 | 51 |
| j5o5r3lr0.8t10_i1 | 2635 | 138 | 1.203 | 3847 | 2684.62 | 8.985 | 765 |
| j5o5r3lr0.8t14_i2 | 3404 | 87 | 0.953 | 450 | 3404 | 0.969 | 89 |
| j5o5r3lr0.8t12_i3 | 6891 | 107 | 0.797 | 326 | 6891 | 0.797 | 109 |
| j5o5r3lr1t10_i1 | 3862 | 62 | 0.578 | 334 | 3862 | 0.578 | 63 |
| j5o5r3lr1t10_i2 | 3167 | 103 | 1.703 | 68180 | 3221.25 | 900 | 6541 |
| j5o5r3lr1t11_i3 | 5992 | 307 | 4.078 | 1633 | 5992 | 4.078 | 309 |

| Instance | Best Integer Solution | Best Integer @ node | Time to Best Integer Solution (seconds) | Number of Columns | Best Bound | Runtime (seconds) | Total Nodes |
|---|---|---|---|---|---|---|---|
| j5o5r3lr1.2t8_i1 | 1171 | 58 | 0.453 | 288 | 1171 | 0.453 | 59 |
| j5o5r3lr1.2t9_i2 | 3092 | 59 | 0.875 | 2393 | 3136 | 5.016 | 281 |
| j5o5r3lr1.2t8_i3 | 1700 | 14 | 0.11 | 71 | 1700 | 0.11 | 15 |
| j5o8r5lr0.8t11_i1 | 2725 | 19 | 0.421 | 181 | 2725 | 0.437 | 21 |
| j5o8r5lr0.8t13_i2 | 8600 | 149 | 7.969 | 66269 | 8775.25 | 900.281 | 3943 |
| j5o8r5lr0.8t12_i3 | 11059 | 149 | 6.079 | 57856 | 11227.4 | 900.172 | 5463 |
| j5o8r5lr1t9_i1 | 1559 | 45 | 0.843 | 2972 | 1599.36 | 8.234 | 429 |
| j5o8r5lr1t9_i2 | 3650 | 77 | 2.203 | 58081 | 3749.83 | 900.203 | 6323 |
| j5o8r5lr1t9_i3 | 2751 | 101 | 2.812 | 2534 | 2803.45 | 8.078 | 305 |
| j5o8r5lr1.2t9_i1 | 3752 | 164 | 4.672 | 1441 | 3752 | 4.688 | 165 |
| j5o8r5lr1.2t8_i2 | 2059 | 78 | 1.75 | 2320 | 2122.91 | 6.313 | 259 |
| j5o8r5lr1.2t8_i3 | 4229 | 0 | 0 | 48234 | 7121.13 | 900.234 | 9057 |
| j5o10r5lr0.8t15_i1 | 6759 | 114 | 9.141 | 50345 | 6920.54 | 900.156 | 5245 |
| j5o10r5lr0.8t14_i2 | 2967 | 118 | 6.391 | 30280 | 3026.17 | 286.313 | 2289 |
| j5o10r5lr0.8t16_i3 | 9810 | 551 | 65.094 | 55001 | 9967.47 | 900.812 | 2923 |
| j5o10r5lr1t12_i1 | 5693 | 259 | 25.687 | 59400 | 6023.01 | 901.062 | 4157 |
| j5o10r5lr1t12_i2 | 6269 | 63 | 4.219 | 2158 | 6348.84 | 10.437 | 155 |
| j5o10r5lr1t12_i3 | 5993 | 205 | 21.141 | 63623 | 6325.47 | 900.531 | 3215 |
| j5o10r5lr1.2t10_i1 | 3593 | 40 | 1.672 | 530 | 3593 | 1.672 | 41 |
| j5o10r5lr1.2t10_i2 | 5628 | 122 | 8.36 | 55920 | 5842.08 | 900.735 | 4729 |
| j5o10r5lr1.2t10_i3 | 4348 | 3420 | 453.422 | 54911 | 4499.46 | 900.344 | 5113 |
| j8o3r3lr0.8t13_i1 | 2295 | 231 | 2.171 | 851 | 2295 | 2.187 | 233 |
| j8o3r3lr0.8t13_i2 | 4341 | 109 | 0.828 | 312 | 4341 | 0.828 | 111 |
| j8o3r3lr0.8t11_i3 | 4193 | 57 | 0.343 | 217 | 4193 | 0.359 | 59 |
| j8o3r3lr1t9_i1 | 2889 | 65 | 0.281 | 167 | 2889 | 0.297 | 67 |
| j8o3r3lr1t9_i2 | 2069 | 260 | 1.281 | 547 | 2069 | 1.281 | 261 |
| j8o3r3lr1t10_i3 | 3218 | 50 | 0.25 | 149 | 3218 | 0.25 | 51 |
| j8o3r3lr1.2t8_i1 | 2963 | 44 | 0.281 | 248 | 2963 | 0.281 | 45 |
| j8o3r3lr1.2t8_i2 | 5417 | 75 | 0.438 | 329 | 5417 | 0.438 | 77 |
| j8o3r3lr1.2t8_i3 | 3202 | 241 | 1.406 | 766 | 3202 | 1.422 | 243 |
| j8o5r3lr0.8t20_i1 | 5940 | 155 | 2.5 | 613 | 5940 | 2.516 | 157 |
| j8o5r3lr0.8t20_i2 | 5941 | 566 | 12.281 | 1773 | 5941 | 12.281 | 567 |
| j8o5r3lr0.8t20_i3 | 6942 | 243 | 4.438 | 821 | 6942 | 4.453 | 245 |
| j8o5r3lr1t17_i1 | 6739 | 388 | 14.313 | 62893 | 6915.17 | 900.344 | 6647 |
| j8o5r3lr1t14_i2 | 5724 | 493 | 7.672 | 2123 | 5724 | 7.688 | 495 |
| j8o5r3lr1t16_i3 | 4009 | 128 | 1.328 | 429 | 4009 | 1.328 | 129 |
| j8o5r3lr1.2t14_i1 | 5369 | 238 | 5.188 | 1754 | 5369 | 5.188 | 239 |
| j8o5r3lr1.2t14_i2 | 5719 | 259 | 4.485 | 46766 | 5807 | 900.016 | 11225 |
| j8o5r3lr1.2t12_i3 | 3991 | 464 | 9.687 | 3144 | 3991 | 9.687 | 465 |
| j8o8r5lr0.8t21_i1 | 7046 | 719 | 156.359 | 52623 | 7252 | 900.593 | 2819 |
| j8o8r5lr0.8t18_i2 | 6748 | 277 | 13.813 | 2036 | 6748 | 13.891 | 279 |
| j8o8r5lr0.8t19_i3 | 14641 | 428 | 66.625 | 56358 | 15031.8 | 900.187 | 3165 |
| j8o8r5lr1t15_i1 | 15751 | 354 | 49.531 | 56903 | 16230.2 | 900.984 | 2975 |
| j8o8r5lr1t16_i2 | 9331 | 974 | 178.281 | 52344 | 11032.9 | 900.312 | 3391 |
| j8o8r5lr1t16_i3 | 10858 | 645 | 84.359 | 54052 | 10996 | 900.859 | 4179 |
| j8o8r5lr1.2t13_i1 | 10067 | 1966 | 355.062 | 61010 | 10373.8 | 900.187 | 3371 |
| j8o8r5lr1.2t13_i2 | 7154 | 318 | 33.156 | 56363 | 7347.76 | 900.984 | 3933 |
| j8o8r5lr1.2t13_i3 | 4484 | 120 | 8.578 | 65031 | 4556.04 | 901.047 | 3991 |
| j8o10r5lr0.8t26_i1 | 17003 | 793 | 236.328 | 42222 | 17432 | 900.672 | 2469 |
| j8o10r5lr0.8t23_i2 | 11073 | 686 | 164.891 | 45519 | 11258 | 901.438 | 2281 |
| j8o10r5lr0.8t23_i3 | 18021 | 753 | 127.172 | 45769 | 18213 | 901.156 | 2723 |
| j8o10r5lr1t19_i1 | 12758 | 388 | 77.625 | 51867 | 13067.4 | 900 | 3081 |
| j8o10r5lr1t17_i2 | 9056 | 422 | 59.188 | 62094 | 9354.86 | 900.407 | 2915 |

| Instance | Best Integer Solution | Best Integer @ node | Time to Best Integer Solution (seconds) | Number of Columns | Best Bound | Runtime (seconds) | Total Nodes |
|---|---|---|---|---|---|---|---|
| j8o10r5lr1t19_i3 | 8645 | 295 | 41.656 | 60511 | 8870.84 | 901.406 | 2769 |
| j8o10r5lr1.2t15_i1 | 7899 | 441 | 76.829 | 54134 | 9229.78 | 900.25 | 2717 |
| j8o10r5lr1.2t16_i2 | 14616 | 482 | 140.532 | 56460 | 15121 | 901.813 | 1809 |
| j8o10r5lr1.2t16_i3 | 10269 | 721 | 151.984 | 53186 | 10666.8 | 901.016 | 2563 |
| j10o3r3lr0.8t15_i1 | 7366 | 205 | 2.281 | 595 | 7366 | 2.312 | 207 |
| j10o3r3lr0.8t15_i2 | 3529 | 317 | 3.812 | 1064 | 3529 | 3.843 | 319 |
| j10o3r3lr0.8t14_i3 | 4676 | 303 | 2.765 | 644 | 4676 | 2.781 | 305 |
| j10o3r3lr1t12_i1 | 3639 | 186 | 1.422 | 388 | 3639 | 1.422 | 187 |
| j10o3r3lr1t12_i2 | 5218 | 198 | 2.438 | 933 | 5218 | 2.438 | 199 |
| j10o3r3lr1t11_i3 | 4658 | 166 | 1.296 | 556 | 4658 | 1.296 | 167 |
| j10o3r3lr1.2t10_i1 | 4683 | 197 | 1.468 | 569 | 4683 | 1.484 | 199 |
| j10o3r3lr1.2t9_i2 | 4373 | 218 | 1.297 | 621 | 4373 | 1.297 | 219 |
| j10o3r3lr1.2t10_i3 | 4278 | 112 | 1.031 | 49085 | 4323.5 | 900.078 | 13969 |
| j10o5r3lr0.8t26_i1 | 9833 | 744 | 29.907 | 2439 | 9833 | 29.907 | 745 |
| j10o5r3lr0.8t28_i2 | 11760 | 1770 | 180.437 | 15366 | 11760 | 180.469 | 1771 |
| j10o5r3lr0.8t25_i3 | 9189 | 499 | 15.875 | 1575 | 9189 | 16.015 | 501 |
| j10o5r3lr1t20_i1 | 4672 | 481 | 10.469 | 1284 | 4672 | 10.5 | 483 |
| j10o5r3lr1t19_i2 | 6104 | 649 | 23.688 | 3980 | 6104 | 23.797 | 651 |
| j10o5r3lr1t19_i3 | 7072 | 399 | 12.906 | 2061 | 7072 | 12.922 | 401 |
| j10o5r3lr1.2t16_i1 | 3884 | 747 | 19.172 | 3156 | 3884 | 19.219 | 749 |
| j10o5r3lr1.2t18_i2 | 6193 | 200 | 5.188 | 1006 | 6193 | 5.188 | 201 |
| j10o5r3lr1.2t18_i3 | 7350 | 272 | 6.906 | 1379 | 7350 | 6.906 | 273 |
| j10o8r5lr0.8t24_i1 | 13218.5 | 877 | 120.391 | 7107 | 13218.5 | 120.454 | 879 |
| j10o8r5lr0.8t24_i2 | 10424 | 1043 | 207.625 | 42640 | 10589.1 | 901.188 | 3111 |
| j10o8r5lr0.8t22_i3 | 11262 | 1556 | 203.125 | 13643 | 11262 | 203.156 | 1557 |
| j10o8r5lr1t17_i1 | 11909 | 785 | 129.454 | 46238 | 12294 | 900.5 | 3465 |
| j10o8r5lr1t18_i2 | 8552 | 227 | 16.516 | 4519 | 8713.3 | 26.703 | 319 |
| j10o8r5lr1t19_i3 | 9162 | 983 | 119.765 | 55750 | 9346 | 900.75 | 3303 |
| j10o8r5lr1.2t16_i1 | 13469 | 365 | 37.484 | 51499 | 13790 | 900.375 | 3303 |
| j10o8r5lr1.2t16_i2 | 12961.5 | 491 | 51.297 | 50673 | 13293 | 900.031 | 5163 |
| j10o8r5lr1.2t16_i3 | 9903 | 628 | 97.672 | 54366 | 10160.5 | 900.359 | 3699 |
| j10o10r5lr0.8t29_i1 | 18435 | 884 | 146.531 | 5435 | 18435 | 146.547 | 885 |
| j10o10r5lr0.8t29_i2 | 11507 | 1767 | 781.234 | 27917 | 12091 | 900.016 | 2209 |
| j10o10r5lr0.8t30_i3 | 16690.5 | 1447 | 662.407 | 29892 | 17464 | 900.328 | 1917 |
| j10o10r5lr1t24_i1 | 10682 | 1127 | 547.515 | 37143 | 11023.9 | 900.422 | 1739 |
| j10o10r5lr1t23_i2 | 15578 | 773 | 218.984 | 43074 | 16092.2 | 900.703 | 2779 |
| j10o10r5lr1t25_i3 | 15369 | 754 | 205.031 | 42278 | 15980.6 | 900.672 | 2263 |
| j10o10r5lr1.2t19_i1 | 17632 | 636 | 171.25 | 52075 | 18177.4 | 900.031 | 2233 |
| j10o10r5lr1.2t21_i2 | 12593 | 480 | 166.984 | 52139 | 12867 | 900 | 1645 |
| j10o10r5lr1.2t19_i3 | 15726 | 1229 | 299.328 | 22335 | 15726 | 299.969 | 1231 |

# APPENDIX I

| Instance | Best Integer Solution | Best Integer @ node | Time to Best Integer Solution (seconds) | Number of Columns | Best Bound | Runtime (seconds) | Total Nodes |
|---|---|---|---|---|---|---|---|
| j3o3r3lr0.8t4_i1 | 794 | 14 | 0.046 | 41 | 794 | 0.046 | 15 |
| j3o3r3lr0.8t5_i2 | 1135 | 3 | 0.031 | 24 | 1135 | 0.031 | 5 |
| j3o3r3lr0.8t5_i3 | 1807 | 1 | 0.016 | 24 | 1807 | 0.031 | 3 |
| j3o3r3lr1t4_i1 | 1120 | 1 | 0.016 | 23 | 1120 | 0.016 | 3 |
| j3o3r3lr1t4_i2 | 1169 | 3 | 0.031 | 26 | 1169 | 0.031 | 5 |
| j3o3r3lr1t3_i3 | 729 | 8 | 0.031 | 40 | 729 | 0.031 | 9 |
| j3o3r3lr1.2t2_i1 | 92 | 0 | 0 | 1 | 92 | 0.016 | 3 |
| j3o3r3lr1.2t3_i2 | 274 | 0 | 0 | 3 | 274 | 0.031 | 3 |
| j3o3r3lr1.2t3_i3 | 1202 | 1 | 0.016 | 24 | 1202 | 0.031 | 3 |
| j3o5r3lr0.8t8_i1 | 2781 | 23 | 0.328 | 257 | 2781 | 0.343 | 25 |
| j3o5r3lr0.8t6_i2 | 3366 | 0 | 0 | 38 | 3366 | 0.047 | 3 |
| j3o5r3lr0.8t7_i3 | 2157 | 7 | 0.109 | 109 | 2157 | 0.109 | 9 |
| j3o5r3lr1t6_i1 | 2134 | 3 | 0.062 | 73 | 2134 | 0.078 | 5 |
| j3o5r3lr1t6_i2 | 3661 | 7 | 0.078 | 79 | 3661 | 0.093 | 9 |
| j3o5r3lr1t6_i3 | 2339 | 84 | 0.453 | 388 | 2339 | 0.453 | 85 |
| j3o5r3lr1.2t5_i1 | 2186 | 26 | 0.172 | 223 | 2296.4 | 0.25 | 41 |
| j3o5r3lr1.2t6_i2 | 164 | 0 | 0 | 19 | 164 | 0.047 | 3 |
| j3o5r3lr1.2t5_i3 | 988 | 0 | 0 | 1 | 988 | 0.031 | 3 |
| j3o8r5lr0.8t8_i1 | 2195 | 0 | 0 | 17 | 2195 | 0.063 | 3 |
| j3o8r5lr0.8t7_i2 | 1889 | 1 | 0.046 | 19 | 1889 | 0.062 | 3 |
| j3o8r5lr0.8t8_i3 | 3023 | 101 | 2.297 | 1083 | 3023 | 2.313 | 103 |
| j3o8r5lr1t6_i1 | 266 | 1 | 0.031 | 7 | 266 | 0.031 | 3 |
| j3o8r5lr1t5_i2 | 1156 | 3 | 0.031 | 18 | 1156 | 0.031 | 5 |
| j3o8r5lr1t6_i3 | 1550 | 0 | 0 | 30 | 1550 | 0.063 | 3 |
| j3o8r5lr1.2t4_i1 | 649 | 0 | 0 | 1 | 649 | 0.016 | 3 |
| j3o8r5lr1.2t4_i2 | 1102 | 1 | 0.031 | 3 | 1102 | 0.031 | 3 |
| j3o8r5lr1.2t5_i3 | 1806 | 0 | 0.047 | 16 | 1806 | 0.047 | 1 |
| j3o10r5lr0.8t9_i1 | 1620 | 3 | 0.109 | 39 | 1620 | 0.125 | 5 |
| j3o10r5lr0.8t9_i2 | 3938 | 133 | 4.625 | 1560 | 3938 | 4.672 | 135 |
| j3o10r5lr0.8t9_i3 | 2563 | 24 | 0.375 | 142 | 2563 | 0.375 | 25 |
| j3o10r5lr1t8_i1 | 6589 | 83 | 2.5 | 943 | 6589 | 2.531 | 85 |
| j3o10r5lr1t7_i2 | 861 | 0 | 0 | 1 | 861 | 0.031 | 3 |
| j3o10r5lr1t7_i3 | 3171 | 8 | 0.234 | 104 | 3171 | 0.234 | 9 |
| j3o10r5lr1.2t6_i1 | 1E-07 | 0 | 0 | 0 | 1E-07 | 0.031 | 3 |
| j3o10r5lr1.2t5_i2 | 3004 | 1 | 0.046 | 10 | 3004 | 0.046 | 3 |
| j3o10r5lr1.2t6_i3 | 137 | 0 | 0 | 1 | 137 | 0.031 | 3 |
| j5o3r3lr0.8t7_i1 | 1729 | 29 | 0.109 | 103 | 1729 | 0.125 | 31 |
| j5o3r3lr0.8t8_i2 | 2259 | 30 | 0.141 | 98 | 2259 | 0.141 | 31 |
| j5o3r3lr0.8t8_i3 | 2627 | 71 | 0.235 | 155 | 2627 | 0.235 | 73 |
| j5o3r3lr1t6_i1 | 3621 | 15 | 0.109 | 122 | 3621 | 0.109 | 17 |
| j5o3r3lr1t6_i2 | 3031 | 55 | 0.172 | 197 | 3031 | 0.172 | 57 |
| j5o3r3lr1t5_i3 | 2975 | 94 | 0.328 | 348 | 2975 | 0.328 | 95 |
| j5o3r3lr1.2t4_i1 | 1312 | 22 | 0.078 | 97 | 1312 | 0.078 | 23 |
| j5o3r3lr1.2t5_i2 | 1321 | 25 | 0.094 | 99 | 1321 | 0.094 | 27 |
| j5o3r3lr1.2t5_i3 | 2670 | 49 | 0.172 | 223 | 2670 | 0.188 | 51 |
| j5o5r3lr0.8t10_i1 | 2577 | 66 | 0.656 | 397 | 2577 | 0.656 | 67 |
| j5o5r3lr0.8t14_i2 | 3404 | 87 | 1.016 | 450 | 3404 | 1.032 | 89 |
| j5o5r3lr0.8t12_i3 | 6891 | 107 | 0.797 | 326 | 6891 | 0.797 | 109 |
| j5o5r3lr1t10_i1 | 3862 | 62 | 0.687 | 334 | 3862 | 0.687 | 63 |
| j5o5r3lr1t10_i2 | 3100 | 78 | 1.312 | 808 | 3100 | 1.312 | 79 |
| j5o5r3lr1t11_i3 | 5767 | 0 | 0 | 60 | 5767 | 0.109 | 3 |

| Instance | Best Integer Solution | Best Integer @ node | Time to Best Integer Solution (seconds) | Number of Columns | Best Bound | Runtime (seconds) | Total Nodes |
|---|---|---|---|---|---|---|---|
| j5o5r3lr1.2t8_i1 | 1171 | 58 | 0.5 | 288 | 1171 | 0.5 | 59 |
| j5o5r3lr1.2t9_i2 | 3092 | 59 | 1 | 489 | 3092 | 1.032 | 61 |
| j5o5r3lr1.2t8_i3 | 1700 | 14 | 0.141 | 71 | 1700 | 0.141 | 15 |
| j5o8r5lr0.8t11_i1 | 2725 | 19 | 0.547 | 181 | 2725 | 0.578 | 21 |
| j5o8r5lr0.8t13_i2 | 8600 | 149 | 8.266 | 1987 | 8600 | 8.282 | 151 |
| j5o8r5lr0.8t12_i3 | 11059 | 149 | 5.735 | 1634 | 11059 | 5.797 | 151 |
| j5o8r5lr1t9_i1 | 1559 | 45 | 0.843 | 427 | 1559 | 0.875 | 47 |
| j5o8r5lr1t9_i2 | 3650 | 77 | 2.156 | 867 | 3650 | 2.187 | 79 |
| j5o8r5lr1t9_i3 | 2751 | 101 | 2.782 | 1011 | 2751 | 2.829 | 103 |
| j5o8r5lr1.2t9_i1 | 3752 | 164 | 4.75 | 1441 | 3752 | 4.75 | 165 |
| j5o8r5lr1.2t8_i2 | 2059 | 39 | 0.828 | 375 | 2059 | 0.844 | 41 |
| j5o8r5lr1.2t8_i3 | 4229 | 0 | 0 | 48435 | 7121.13 | 900.33 | 9105 |
| j5o10r5lr0.8t15_i1 | 6759 | 114 | 9.064 | 1793 | 6759 | 9.064 | 115 |
| j5o10r5lr0.8t14_i2 | 2967 | 118 | 6.298 | 1857 | 2967 | 6.313 | 119 |
| j5o10r5lr0.8t16_i3 | 9715 | 201 | 19.612 | 3412 | 9715 | 19.752 | 203 |
| j5o10r5lr1t12_i1 | 5693 | 259 | 25.55 | 53865 | 6023.01 | 900.418 | 4283 |
| j5o10r5lr1t12_i2 | 6269 | 63 | 4.188 | 998 | 6269 | 4.219 | 65 |
| j5o10r5lr1t12_i3 | 5993 | 205 | 21.408 | 65702 | 6325.47 | 900.356 | 3111 |
| j5o10r5lr1.2t10_i1 | 3593 | 40 | 1.656 | 530 | 3593 | 1.656 | 41 |
| j5o10r5lr1.2t10_i2 | 5628 | 122 | 7.954 | 1869 | 5628 | 7.954 | 123 |
| j5o10r5lr1.2t10_i3 | 4325 | 90 | 3.922 | 1189 | 4325 | 3.938 | 91 |
| j8o3r3lr0.8t13_i1 | 2263 | 0 | 0 | 56 | 2263 | 0.078 | 3 |
| j8o3r3lr0.8t13_i2 | 4181 | 0 | 0 | 53 | 4181 | 0.063 | 3 |
| j8o3r3lr0.8t11_i3 | 4057 | 0 | 0 | 56 | 4057 | 0.062 | 3 |
| j8o3r3lr1t9_i1 | 2889 | 65 | 0.281 | 167 | 2889 | 0.296 | 67 |
| j8o3r3lr1t9_i2 | 2012 | 159 | 0.75 | 376 | 2012 | 0.75 | 161 |
| j8o3r3lr1t10_i3 | 3218 | 50 | 0.265 | 149 | 3218 | 0.265 | 51 |
| j8o3r3lr1.2t8_i1 | 2963 | 44 | 0.281 | 248 | 2963 | 0.281 | 45 |
| j8o3r3lr1.2t8_i2 | 5417 | 75 | 0.437 | 329 | 5417 | 0.453 | 77 |
| j8o3r3lr1.2t8_i3 | 3081 | 80 | 0.516 | 334 | 3081 | 0.516 | 81 |
| j8o5r3lr0.8t20_i1 | 5940 | 155 | 2.485 | 613 | 5940 | 2.5 | 157 |
| j8o5r3lr0.8t20_i2 | 5681 | 0 | 0 | 74 | 5681 | 0.172 | 5 |
| j8o5r3lr0.8t20_i3 | 6630 | 0 | 0 | 88 | 6630 | 0.172 | 3 |
| j8o5r3lr1t17_i1 | 6739 | 388 | 14.204 | 2904 | 6739 | 14.204 | 389 |
| j8o5r3lr1t14_i2 | 5652 | 322 | 5.188 | 1494 | 5652 | 5.188 | 323 |
| j8o5r3lr1t16_i3 | 4009 | 128 | 1.313 | 429 | 4009 | 1.313 | 129 |
| j8o5r3lr1.2t14_i1 | 5369 | 238 | 5.141 | 1754 | 5369 | 5.141 | 239 |
| j8o5r3lr1.2t14_i2 | 5719 | 259 | 4.438 | 1234 | 5719 | 4.454 | 261 |
| j8o5r3lr1.2t12_i3 | 3875 | 275 | 5.687 | 1965 | 3875 | 5.719 | 277 |
| j8o8r5lr0.8t21_i1 | 7046 | 719 | 155.714 | 13467 | 7046 | 155.948 | 721 |
| j8o8r5lr0.8t18_i2 | 6748 | 277 | 13.844 | 2036 | 6748 | 13.923 | 279 |
| j8o8r5lr0.8t19_i3 | 14641 | 428 | 66.63 | 7435 | 14641 | 66.645 | 429 |
| j8o8r5lr1t15_i1 | 15751 | 354 | 49.457 | 6850 | 15751 | 49.472 | 355 |
| j8o8r5lr1t16_i2 | 9146 | 448 | 78.505 | 51689 | 11032.9 | 900.199 | 2797 |
| j8o8r5lr1t16_i3 | 10732 | 362 | 42.565 | 5494 | 10732 | 42.565 | 363 |
| j8o8r5lr1.2t13_i1 | 9986 | 338 | 34.783 | 5918 | 9986 | 34.799 | 339 |
| j8o8r5lr1.2t13_i2 | 7154 | 318 | 33.111 | 5847 | 7154 | 33.111 | 319 |
| j8o8r5lr1.2t13_i3 | 4484 | 120 | 8.547 | 2261 | 4484 | 8.547 | 121 |
| j8o10r5lr0.8t26_i1 | 17003 | 793 | 236.342 | 14141 | 17003 | 236.467 | 795 |
| j8o10r5lr0.8t23_i2 | 11073 | 686 | 164.697 | 12599 | 11073 | 164.728 | 687 |
| j8o10r5lr0.8t23_i3 | 18021 | 753 | 127.117 | 10120 | 18021 | 127.492 | 755 |
| j8o10r5lr1t19_i1 | 12758 | 388 | 77.441 | 8003 | 12758 | 77.457 | 389 |
| j8o10r5lr1t17_i2 | 9056 | 422 | 59.096 | 7415 | 9056 | 59.112 | 423 |

| Instance | Best Integer Solution | Best Integer @ node | Time to Best Integer Solution (seconds) | Number of Columns | Best Bound | Runtime (seconds) | Total Nodes |
|---|---|---|---|---|---|---|---|
| j8o10r5lr1t19_i3 | 8645 | 295 | 41.58 | 5412 | 8645 | 41.721 | 297 |
| j8o10r5lr1.2t15_i1 | 7899 | 441 | 76.723 | 49190 | 9229.78 | 900.103 | 3047 |
| j8o10r5lr1.2t16_i2 | 14616 | 482 | 140.021 | 12795 | 14616 | 140.052 | 483 |
| j8o10r5lr1.2t16_i3 | 10107 | 380 | 71.472 | 54651 | 10666.9 | 900.081 | 2577 |
| j10o3r3lr0.8t15_i1 | 7366 | 205 | 2.265 | 595 | 7366 | 2.281 | 207 |
| j10o3r3lr0.8t15_i2 | 3529 | 317 | 3.813 | 1064 | 3529 | 3.844 | 319 |
| j10o3r3lr0.8t14_i3 | 4532 | 0 | 0 | 70 | 4532 | 0.078 | 3 |
| j10o3r3lr1t12_i1 | 3639 | 186 | 1.406 | 388 | 3639 | 1.406 | 187 |
| j10o3r3lr1t12_i2 | 5218 | 198 | 2.438 | 933 | 5218 | 2.438 | 199 |
| j10o3r3lr1t11_i3 | 4658 | 166 | 1.296 | 556 | 4658 | 1.296 | 167 |
| j10o3r3lr1.2t10_i1 | 4683 | 197 | 1.469 | 569 | 4683 | 1.485 | 199 |
| j10o3r3lr1.2t9_i2 | 4263 | 119 | 0.719 | 380 | 4263 | 0.735 | 121 |
| j10o3r3lr1.2t10_i3 | 4278 | 112 | 1.046 | 603 | 4278 | 1.046 | 113 |
| j10o5r3lr0.8t26_i1 | 9405 | 0 | 0 | 120 | 9405 | 0.312 | 3 |
| j10o5r3lr0.8t28_i2 | 11643 | 847 | 85.503 | 7588 | 11643 | 85.566 | 849 |
| j10o5r3lr0.8t25_i3 | 9189 | 499 | 15.782 | 1575 | 9189 | 15.907 | 501 |
| j10o5r3lr1t20_i1 | 4672 | 481 | 10.406 | 1284 | 4672 | 10.438 | 483 |
| j10o5r3lr1t19_i2 | 5903 | 0 | 0 | 118 | 5903 | 0.204 | 3 |
| j10o5r3lr1t19_i3 | 7072 | 399 | 12.876 | 2061 | 7072 | 12.907 | 401 |
| j10o5r3lr1.2t16_i1 | 3815 | 457 | 12.266 | 1954 | 3815 | 12.297 | 459 |
| j10o5r3lr1.2t18_i2 | 6193 | 200 | 5.188 | 1006 | 6193 | 5.188 | 201 |
| j10o5r3lr1.2t18_i3 | 7350 | 272 | 6.891 | 1379 | 7350 | 6.891 | 273 |
| j10o8r5lr0.8t24_i1 | 13218.5 | 877 | 120.348 | 7107 | 13218.5 | 120.426 | 879 |
| j10o8r5lr0.8t24_i2 | 9930 | 0 | 0 | 44805 | 10589.1 | 900.496 | 2555 |
| j10o8r5lr0.8t22_i3 | 11262 | 1556 | 202.974 | 13643 | 11262 | 203.005 | 1557 |
| j10o8r5lr1t17_i1 | 11909 | 785 | 129.41 | 13296 | 11909 | 129.613 | 787 |
| j10o8r5lr1t18_i2 | 8552 | 227 | 16.626 | 2679 | 8552 | 16.782 | 229 |
| j10o8r5lr1t19_i3 | 8942 | 629 | 69.455 | 6937 | 8942 | 69.549 | 631 |
| j10o8r5lr1.2t16_i1 | 13469 | 365 | 37.611 | 4767 | 13469 | 37.72 | 367 |
| j10o8r5lr1.2t16_i2 | 12961.5 | 491 | 50.736 | 6559 | 12961.5 | 50.97 | 493 |
| j10o8r5lr1.2t16_i3 | 9903 | 628 | 97.659 | 9811 | 9903 | 97.675 | 629 |
| j10o10r5lr0.8t29_i1 | 18435 | 884 | 146.409 | 5435 | 18435 | 146.425 | 885 |
| j10o10r5lr0.8t29_i2 | 11507 | 1767 | 780.421 | 27938 | 12091 | 900.126 | 2211 |
| j10o10r5lr0.8t30_i3 | 16690.5 | 1447 | 661.887 | 22628 | 16690.5 | 662.184 | 1449 |
| j10o10r5lr1t24_i1 | 10682 | 1127 | 546.806 | 23433 | 10682 | 547.322 | 1129 |
| j10o10r5lr1t23_i2 | 15578 | 773 | 218.502 | 15248 | 15578 | 218.737 | 775 |
| j10o10r5lr1t25_i3 | 15369 | 754 | 204.924 | 13562 | 15369 | 204.955 | 755 |
| j10o10r5lr1.2t19_i1 | 17632 | 636 | 171.049 | 14011 | 17632 | 171.08 | 637 |
| j10o10r5lr1.2t21_i2 | 12593 | 480 | 166.846 | 12687 | 12593 | 166.877 | 481 |
| j10o10r5lr1.2t19_i3 | 15336 | 679 | 155.736 | 12888 | 15336 | 156.018 | 681 |

# APPENDIX J

| Instance | Best Integer Solution | Best Integer @ node | Time to Best Integer Solution (seconds) | Columns Added | Best Bound | Runtime (seconds) | Total Nodes |
|---|---|---|---|---|---|---|---|
| j3o3r3lr0.8t4_i1 | 794 | 7 | 0.031 | 28 | 794 | 0.031 | 9 |
| j3o3r3lr0.8t5_i2 | 1135 | 3 | 0.031 | 30 | 1135 | 0.031 | 5 |
| j3o3r3lr0.8t5_i3 | 1807 | 1 | 0.031 | 26 | 1807 | 0.031 | 3 |
| j3o3r3lr1t4_i1 | 1120 | 2 | 0.016 | 20 | 1120 | 0.016 | 3 |
| j3o3r3lr1t4_i2 | 1169 | 3 | 0.031 | 25 | 1169 | 0.031 | 5 |
| j3o3r3lr1t3_i3 | 721 | 17 | 0.046 | 81 | 721 | 0.046 | 19 |
| j3o3r3lr1.2t2_i1 | 92 | 0 | 0 | 1 | 92 | 0.016 | 3 |
| j3o3r3lr1.2t3_i2 | 274 | 0 | 0 | 3 | 274 | 0.015 | 3 |
| j3o3r3lr1.2t3_i3 | 1202 | 3 | 0.016 | 30 | 1202 | 0.016 | 5 |
| j3o5r3lr0.8t8_i1 | 2723 | 46 | 0.531 | 437 | 2737.6 | 0.547 | 49 |
| j3o5r3lr0.8t6_i2 | 3416 | 10 | 0.093 | 107 | 3416 | 0.093 | 11 |
| j3o5r3lr0.8t7_i3 | 2157 | 10 | 0.125 | 130 | 2157 | 0.14 | 13 |
| j3o5r3lr1t6_i1 | 2134 | 3 | 0.062 | 77 | 2134 | 0.078 | 5 |
| j3o5r3lr1t6_i2 | 3661 | 8 | 0.078 | 80 | 3661 | 0.078 | 9 |
| j3o5r3lr1t6_i3 | 2367 | 12 | 0.094 | 208 | 2377 | 0.172 | 27 |
| j3o5r3lr1.2t5_i1 | 2282 | 18 | 0.125 | 123 | 2282 | 0.125 | 19 |
| j3o5r3lr1.2t6_i2 | 164 | 0 | 0 | 20 | 164 | 0.047 | 5 |
| j3o5r3lr1.2t5_i3 | 988 | 0 | 0 | 1 | 988 | 0.031 | 3 |
| j3o8r5lr0.8t8_i1 | 2235 | 3 | 0.047 | 22 | 2235 | 0.063 | 7 |
| j3o8r5lr0.8t7_i2 | 1889 | 1 | 0.046 | 19 | 1889 | 0.062 | 3 |
| j3o8r5lr0.8t8_i3 | 2998 | 22 | 0.531 | 642 | 3086.86 | 1.234 | 67 |
| j3o8r5lr1t6_i1 | 266 | 1 | 0.031 | 7 | 266 | 0.031 | 3 |
| j3o8r5lr1t5_i2 | 1175 | 5 | 0.046 | 21 | 1175 | 0.046 | 7 |
| j3o8r5lr1t6_i3 | 1610 | 9 | 0.125 | 86 | 1610 | 0.14 | 11 |
| j3o8r5lr1.2t4_i1 | 649 | 0 | 0 | 1 | 649 | 0.016 | 3 |
| j3o8r5lr1.2t4_i2 | 1102 | 1 | 0.016 | 3 | 1102 | 0.016 | 3 |
| j3o8r5lr1.2t5_i3 | 1806 | 0 | 0.031 | 16 | 1806 | 0.031 | 1 |
| j3o10r5lr0.8t9_i1 | 1620 | 3 | 0.156 | 57 | 1620 | 0.171 | 5 |
| j3o10r5lr0.8t9_i2 | 3923 | 110 | 4.531 | 2321 | 3996.5 | 6.343 | 161 |
| j3o10r5lr0.8t9_i3 | 2563 | 18 | 0.36 | 163 | 2563 | 0.36 | 19 |
| j3o10r5lr1t8_i1 | 6645 | 38 | 0.844 | 476 | 6659.33 | 1.235 | 59 |
| j3o10r5lr1t7_i2 | 861 | 0 | 0 | 1 | 861 | 0.031 | 3 |
| j3o10r5lr1t7_i3 | 3171 | 6 | 0.218 | 105 | 3171 | 0.218 | 7 |
| j3o10r5lr1.2t6_i1 | 1E-07 | 0 | 0 | 0 | 1E-07 | 0.031 | 3 |
| j3o10r5lr1.2t5_i2 | 3004 | 1 | 0.046 | 10 | 3004 | 0.046 | 3 |
| j3o10r5lr1.2t6_i3 | 137 | 0 | 0 | 1 | 137 | 0.031 | 3 |
| j5o3r3lr0.8t7_i1 | 1729 | 13 | 0.078 | 107 | 1729 | 0.078 | 15 |
| j5o3r3lr0.8t8_i2 | 2259 | 11 | 0.078 | 66 | 2259 | 0.093 | 13 |
| j5o3r3lr0.8t8_i3 | 2627 | 13 | 0.063 | 68 | 2627 | 0.078 | 15 |
| j5o3r3lr1t6_i1 | 3621 | 9 | 0.063 | 82 | 3621 | 0.078 | 11 |
| j5o3r3lr1t6_i2 | 3031 | 27 | 0.11 | 179 | 3031 | 0.125 | 29 |
| j5o3r3lr1t5_i3 | 3041 | 11 | 0.063 | 114 | 3041 | 0.063 | 13 |
| j5o3r3lr1.2t4_i1 | 1312 | 8 | 0.046 | 58 | 1312 | 0.046 | 9 |
| j5o3r3lr1.2t5_i2 | 1321 | 28 | 0.125 | 130 | 1321 | 0.125 | 29 |
| j5o3r3lr1.2t5_i3 | 2652 | 32 | 0.171 | 316 | 2670 | 0.218 | 43 |
| j5o5r3lr0.8t10_i1 | 2657 | 21 | 0.266 | 253 | 2657 | 0.281 | 23 |
| j5o5r3lr0.8t14_i2 | 3404 | 23 | 0.421 | 224 | 3404 | 0.421 | 25 |
| j5o5r3lr0.8t12_i3 | 6891 | 26 | 0.359 | 237 | 6891 | 0.359 | 27 |
| j5o5r3lr1t10_i1 | 3862 | 22 | 0.375 | 246 | 3862 | 0.375 | 23 |
| j5o5r3lr1t10_i2 | 3182 | 72 | 1.265 | 1197 | 3194 | 1.625 | 89 |
| j5o5r3lr1t11_i3 | 5992 | 27 | 0.515 | 352 | 5992 | 0.531 | 29 |

| Instance | Best Integer Solution | Best Integer @ node | Time to Best Integer Solution (seconds) | Columns Added | Best Bound | Runtime (seconds) | Total Nodes |
|---|---|---|---|---|---|---|---|
| j5o5r3lr1.2t8_i1 | 1182 | 31 | 0.375 | 339 | 1182 | 0.375 | 33 |
| j5o5r3lr1.2t9_i2 | 3070 | 38 | 0.641 | 606 | 3116.63 | 0.922 | 51 |
| j5o5r3lr1.2t8_i3 | 1700 | 9 | 0.109 | 85 | 1700 | 0.125 | 11 |
| j5o8r5lr0.8t11_i1 | 2725 | 13 | 0.344 | 191 | 2725 | 0.422 | 19 |
| j5o8r5lr0.8t13_i2 | 8527 | 69 | 4.969 | 16900 | 8666.37 | 83.188 | 809 |
| j5o8r5lr0.8t12_i3 | 11024 | 328 | 20.953 | 8617 | 11158.6 | 30.312 | 437 |
| j5o8r5lr1t9_i1 | 1502 | 72 | 1.531 | 826 | 1502 | 1.531 | 73 |
| j5o8r5lr1t9_i2 | 3667 | 39 | 1.281 | 577 | 3667 | 1.313 | 41 |
| j5o8r5lr1t9_i3 | 2751 | 36 | 1.187 | 564 | 2788.88 | 1.359 | 43 |
| j5o8r5lr1.2t9_i1 | 3733 | 43 | 1.719 | 670 | 3733 | 1.719 | 45 |
| j5o8r5lr1.2t8_i2 | 2059 | 26 | 0.64 | 665 | 2119.44 | 1.468 | 77 |
| j5o8r5lr1.2t8_i3 | 6796.5 | 120 | 3.985 | 2255 | 6928.87 | 4.828 | 141 |
| j5o10r5lr0.8t15_i1 | 6769 | 213 | 22.782 | 61553 | 6915.16 | 901.1 | 3557 |
| j5o10r5lr0.8t14_i2 | 2784 | 89 | 6.86 | 2417 | 2899.8 | 8.11 | 103 |
| j5o10r5lr0.8t16_i3 | 9682 | 182 | 30.078 | 46728 | 9935.32 | 482.269 | 1777 |
| j5o10r5lr1t12_i1 | 5534 | 2935 | 765.895 | 67678 | 5954.96 | 901.037 | 3247 |
| j5o10r5lr1t12_i2 | 6175 | 34 | 3.063 | 1733 | 6237.16 | 6.485 | 67 |
| j5o10r5lr1t12_i3 | 5879 | 2717 | 599.598 | 66885 | 6286.69 | 900.74 | 3447 |
| j5o10r5lr1.2t10_i1 | 3589 | 67 | 2.828 | 925 | 3589 | 2.891 | 71 |
| j5o10r5lr1.2t10_i2 | 5471 | 249 | 21.032 | 6453 | 5692.76 | 26.078 | 291 |
| j5o10r5lr1.2t10_i3 | 4383 | 202 | 10.625 | 3795 | 4441.17 | 12.641 | 243 |
| j8o3r3lr0.8t13_i1 | 2295 | 37 | 0.437 | 311 | 2295 | 0.468 | 39 |
| j8o3r3lr0.8t13_i2 | 4341 | 65 | 0.766 | 498 | 4341 | 0.797 | 67 |
| j8o3r3lr0.8t11_i3 | 4193 | 19 | 0.172 | 154 | 4193 | 0.188 | 21 |
| j8o3r3lr1t9_i1 | 2889 | 27 | 0.203 | 168 | 2889 | 0.219 | 31 |
| j8o3r3lr1t9_i2 | 2069 | 43 | 0.297 | 282 | 2069 | 0.297 | 45 |
| j8o3r3lr1t10_i3 | 3218 | 25 | 0.218 | 174 | 3218 | 0.234 | 27 |
| j8o3r3lr1.2t8_i1 | 2963 | 15 | 0.14 | 160 | 2963 | 0.14 | 17 |
| j8o3r3lr1.2t8_i2 | 5436 | 17 | 0.234 | 253 | 5436 | 0.25 | 19 |
| j8o3r3lr1.2t8_i3 | 3202 | 56 | 0.469 | 447 | 3202 | 0.469 | 57 |
| j8o5r3lr0.8t20_i1 | 5940 | 60 | 1.765 | 715 | 5940 | 1.765 | 61 |
| j8o5r3lr0.8t20_i2 | 5941 | 74 | 3 | 924 | 5941 | 3 | 75 |
| j8o5r3lr0.8t20_i3 | 6942 | 1503 | 60.969 | 33282 | 6954 | 270.595 | 4161 |
| j8o5r3lr1t17_i1 | 6787 | 2192 | 223.064 | 61562 | 6852.04 | 515.956 | 3613 |
| j8o5r3lr1t14_i2 | 5724 | 90 | 2 | 945 | 5724 | 2 | 91 |
| j8o5r3lr1t16_i3 | 4009 | 30 | 0.594 | 289 | 4009 | 0.594 | 31 |
| j8o5r3lr1.2t14_i1 | 5385 | 418 | 17.313 | 37494 | 5405.5 | 226.595 | 2293 |
| j8o5r3lr1.2t14_i2 | 5807 | 39 | 1.36 | 742 | 5807 | 1.406 | 41 |
| j8o5r3lr1.2t12_i3 | 4029 | 1566 | 83.173 | 22585 | 4029 | 83.204 | 1567 |
| j8o8r5lr0.8t21_i1 | 6976 | 189 | 47.719 | 63373 | 7252 | 900.771 | 2261 |
| j8o8r5lr0.8t18_i2 | 6766 | 3705 | 598.217 | 63682 | 6802 | 900.316 | 4725 |
| j8o8r5lr0.8t19_i3 | 14456 | 1145 | 278.127 | 68958 | 14855.9 | 900.412 | 2821 |
| j8o8r5lr1t15_i1 | 15748 | 1969 | 627.973 | 69665 | 16200.2 | 900.537 | 2465 |
| j8o8r5lr1t16_i2 | 10686 | 1589 | 512.816 | 69088 | 10965.7 | 901.6 | 2345 |
| j8o8r5lr1t16_i3 | 10780 | 86 | 12.609 | 65103 | 10996 | 900.252 | 2879 |
| j8o8r5lr1.2t13_i1 | 9914 | 541 | 75.876 | 74524 | 10187.3 | 900.287 | 2859 |
| j8o8r5lr1.2t13_i2 | 7185 | 2666 | 745.692 | 69986 | 7325.49 | 901.381 | 2997 |
| j8o8r5lr1.2t13_i3 | 4397 | 352 | 44.516 | 11631 | 4428.51 | 60.735 | 455 |
| j8o10r5lr0.8t26_i1 | 16940 | 446 | 257.424 | 50640 | 17432 | 902.459 | 1557 |
| j8o10r5lr0.8t23_i2 | 10866 | 104 | 32.016 | 54375 | 11258 | 901.091 | 1611 |
| j8o10r5lr0.8t23_i3 | 17926 | 90 | 21.578 | 62888 | 18213 | 902.005 | 2031 |
| j8o10r5lr1t19_i1 | 12560 | 131 | 31.282 | 66367 | 12863.2 | 900.162 | 2347 |
| j8o10r5lr1t17_i2 | 8914 | 1099 | 221.767 | 60610 | 9314.21 | 900.069 | 3055 |

| Instance | Best Integer Solution | Best Integer @ node | Time to Best Integer Solution (seconds) | Columns Added | Best Bound | Runtime (seconds) | Total Nodes |
|---|---|---|---|---|---|---|---|
| j8o10r5lr1t19_i3 | 8709 | 610 | 148.142 | 65446 | 8801.83 | 900.124 | 2283 |
| j8o10r5lr1.2t15_i1 | 8661 | 307 | 89.032 | 64537 | 9183.11 | 901.162 | 2287 |
| j8o10r5lr1.2t16_i2 | 14370 | 1409 | 643.458 | 59671 | 15103.6 | 900.475 | 1733 |
| j8o10r5lr1.2t16_i3 | 10243 | 1199 | 404.55 | 62480 | 10590.5 | 900.069 | 2155 |
| j10o3r3lr0.8t15_i1 | 7366 | 33 | 0.547 | 290 | 7366 | 0.563 | 35 |
| j10o3r3lr0.8t15_i2 | 3529 | 35 | 0.578 | 376 | 3529 | 0.594 | 37 |
| j10o3r3lr0.8t14_i3 | 4676 | 37 | 0.438 | 282 | 4676 | 0.453 | 39 |
| j10o3r3lr1t12_i1 | 3639 | 30 | 0.438 | 323 | 3639 | 0.438 | 31 |
| j10o3r3lr1t12_i2 | 5218 | 55 | 0.985 | 649 | 5218 | 1.016 | 57 |
| j10o3r3lr1t11_i3 | 4658 | 99 | 1.11 | 821 | 4658 | 1.11 | 101 |
| j10o3r3lr1.2t10_i1 | 4683 | 40 | 0.515 | 355 | 4683 | 0.515 | 41 |
| j10o3r3lr1.2t9_i2 | 4373 | 24 | 0.203 | 166 | 4373 | 0.203 | 25 |
| j10o3r3lr1.2t10_i3 | 4306 | 54 | 0.703 | 2631 | 4323.5 | 3.531 | 253 |
| j10o5r3lr0.8t26_i1 | 9833 | 76 | 4.922 | 928 | 9833 | 4.922 | 77 |
| j10o5r3lr0.8t28_i2 | 11817 | 958 | 135.594 | 20085 | 11817 | 135.641 | 959 |
| j10o5r3lr0.8t25_i3 | 9189 | 58 | 2.953 | 691 | 9189 | 2.953 | 59 |
| j10o5r3lr1t20_i1 | 4672 | 144 | 6.781 | 1740 | 4672 | 6.781 | 145 |
| j10o5r3lr1t19_i2 | 6156 | 54 | 2.75 | 972 | 6156 | 2.75 | 55 |
| j10o5r3lr1t19_i3 | 7072 | 61 | 3.484 | 1110 | 7072 | 3.64 | 67 |
| j10o5r3lr1.2t16_i1 | 3896 | 93 | 3.547 | 1376 | 3896 | 3.578 | 95 |
| j10o5r3lr1.2t18_i2 | 6200 | 447 | 21.766 | 7530 | 6221 | 32.75 | 617 |
| j10o5r3lr1.2t18_i3 | 7350 | 39 | 2.438 | 951 | 7350 | 2.516 | 41 |
| j10o8r5lr0.8t24_i1 | 13264 | 2324 | 875.532 | 64685 | 13293 | 901.11 | 2361 |
| j10o8r5lr0.8t24_i2 | 10475 | 590 | 138.579 | 58375 | 10589.1 | 900.005 | 2243 |
| j10o8r5lr0.8t22_i3 | 11251 | 2599 | 846.943 | 69720 | 11289 | 900.334 | 2709 |
| j10o8r5lr1t17_i1 | 11999 | 352 | 94.11 | 68135 | 12294 | 900.225 | 2129 |
| j10o8r5lr1t18_i2 | 8518 | 58 | 6.375 | 72622 | 8716.19 | 900.093 | 2759 |
| j10o8r5lr1t19_i3 | 9089 | 85 | 12.906 | 73850 | 9346 | 900.865 | 2469 |
| j10o8r5lr1.2t16_i1 | 13436 | 1224 | 309.862 | 70151 | 13790 | 900.084 | 2645 |
| j10o8r5lr1.2t16_i2 | 12954 | 359 | 62.954 | 67727 | 13293 | 900.178 | 2997 |
| j10o8r5lr1.2t16_i3 | 9986 | 170 | 29.422 | 68425 | 10160.4 | 900.132 | 2591 |
| j10o10r5lr0.8t29_i1 | 18323 | 128 | 29.36 | 50941 | 18491 | 900.6 | 2151 |
| j10o10r5lr0.8t29_i2 | 11760 | 143 | 48.219 | 47956 | 12091 | 901.537 | 1545 |
| j10o10r5lr0.8t30_i3 | 17129 | 391 | 247.126 | 42764 | 17464 | 900.365 | 1113 |
| j10o10r5lr1t24_i1 | 10490 | 621 | 536.831 | 50884 | 11023.9 | 901.146 | 1067 |
| j10o10r5lr1t23_i2 | 15561 | 190 | 68.141 | 57537 | 16083.5 | 900.162 | 2069 |
| j10o10r5lr1t25_i3 | 15390 | 218 | 88.829 | 56452 | 15980.6 | 900.74 | 1683 |
| j10o10r5lr1.2t19_i1 | 17323 | 289 | 131.438 | 57506 | 18176.6 | 900.568 | 1981 |
| j10o10r5lr1.2t21_i2 | 12694 | 177 | 64.25 | 60206 | 12818.8 | 901.43 | 1687 |
| j10o10r5lr1.2t19_i3 | 15639 | 82 | 24.672 | 61918 | 15882.9 | 900.443 | 1699 |

# APPENDIX K

| Instance | Best Integer Solution | Best Integer @ node | Time to Best Integer Solution (seconds) | Columns Added | Best Bound | Runtime (seconds) | Total Nodes |
|---|---|---|---|---|---|---|---|
| j3o3r3lr0.8t4_i1 | 794 | 7 | 0.078 | 28 | 794 | 0.078 | 9 |
| j3o3r3lr0.8t5_i2 | 1135 | 3 | 0.031 | 30 | 1135 | 0.031 | 5 |
| j3o3r3lr0.8t5_i3 | 1807 | 1 | 0.031 | 26 | 1807 | 0.031 | 3 |
| j3o3r3lr1t4_i1 | 1120 | 2 | 0.031 | 20 | 1120 | 0.031 | 3 |
| j3o3r3lr1t4_i2 | 1169 | 3 | 0.016 | 25 | 1169 | 0.016 | 5 |
| j3o3r3lr1t3_i3 | 721 | 17 | 0.046 | 81 | 721 | 0.046 | 19 |
| j3o3r3lr1.2t2_i1 | 92 | 0 | 0 | 1 | 92 | 0.015 | 3 |
| j3o3r3lr1.2t3_i2 | 274 | 0 | 0 | 3 | 274 | 0.016 | 3 |
| j3o3r3lr1.2t3_i3 | 1202 | 3 | 0.031 | 30 | 1202 | 0.031 | 5 |
| j3o5r3lr0.8t8_i1 | 2703 | 16 | 0.219 | 379 | 2749.54 | 0.453 | 37 |
| j3o5r3lr0.8t6_i2 | 3416 | 10 | 0.093 | 107 | 3416 | 0.093 | 11 |
| j3o5r3lr0.8t7_i3 | 2157 | 10 | 0.125 | 121 | 2157 | 0.125 | 11 |
| j3o5r3lr1t6_i1 | 2134 | 3 | 0.063 | 77 | 2134 | 0.063 | 5 |
| j3o5r3lr1t6_i2 | 3661 | 8 | 0.078 | 80 | 3661 | 0.078 | 9 |
| j3o5r3lr1t6_i3 | 2367 | 12 | 0.094 | 119 | 2367 | 0.11 | 17 |
| j3o5r3lr1.2t5_i1 | 2282 | 18 | 0.125 | 123 | 2282 | 0.14 | 19 |
| j3o5r3lr1.2t6_i2 | 164 | 0 | 0 | 20 | 164 | 0.062 | 5 |
| j3o5r3lr1.2t5_i3 | 988 | 0 | 0 | 1 | 988 | 0.016 | 3 |
| j3o8r5lr0.8t8_i1 | 2235 | 3 | 0.062 | 20 | 2235 | 0.062 | 5 |
| j3o8r5lr0.8t7_i2 | 1889 | 1 | 0.031 | 19 | 1889 | 0.047 | 3 |
| j3o8r5lr0.8t8_i3 | 2998 | 22 | 0.531 | 447 | 3086.86 | 0.844 | 43 |
| j3o8r5lr1t6_i1 | 266 | 1 | 0.031 | 7 | 266 | 0.031 | 3 |
| j3o8r5lr1t5_i2 | 1175 | 5 | 0.046 | 21 | 1175 | 0.046 | 7 |
| j3o8r5lr1t6_i3 | 1610 | 9 | 0.125 | 86 | 1610 | 0.141 | 11 |
| j3o8r5lr1.2t4_i1 | 649 | 0 | 0 | 1 | 649 | 0.016 | 3 |
| j3o8r5lr1.2t4_i2 | 1102 | 1 | 0.016 | 3 | 1102 | 0.016 | 3 |
| j3o8r5lr1.2t5_i3 | 1806 | 0 | 0.031 | 16 | 1806 | 0.031 | 1 |
| j3o10r5lr0.8t9_i1 | 1620 | 3 | 0.156 | 57 | 1620 | 0.171 | 5 |
| j3o10r5lr0.8t9_i2 | 3911 | 98 | 4.016 | 1874 | 4013.17 | 5 | 127 |
| j3o10r5lr0.8t9_i3 | 2563 | 18 | 0.375 | 163 | 2563 | 0.375 | 19 |
| j3o10r5lr1t8_i1 | 6634 | 20 | 0.531 | 231 | 6634 | 0.531 | 21 |
| j3o10r5lr1t7_i2 | 861 | 0 | 0 | 1 | 861 | 0.031 | 3 |
| j3o10r5lr1t7_i3 | 3171 | 6 | 0.203 | 105 | 3171 | 0.203 | 7 |
| j3o10r5lr1.2t6_i1 | 1E-07 | 0 | 0 | 0 | 1E-07 | 0.031 | 3 |
| j3o10r5lr1.2t5_i2 | 3004 | 1 | 0.047 | 10 | 3004 | 0.047 | 3 |
| j3o10r5lr1.2t6_i3 | 137 | 0 | 0 | 1 | 137 | 0.031 | 3 |
| j5o3r3lr0.8t7_i1 | 1729 | 13 | 0.079 | 107 | 1729 | 0.079 | 15 |
| j5o3r3lr0.8t8_i2 | 2259 | 11 | 0.078 | 66 | 2259 | 0.078 | 13 |
| j5o3r3lr0.8t8_i3 | 2627 | 13 | 0.078 | 68 | 2627 | 0.078 | 15 |
| j5o3r3lr1t6_i1 | 3621 | 9 | 0.078 | 82 | 3621 | 0.078 | 11 |
| j5o3r3lr1t6_i2 | 3031 | 27 | 0.125 | 179 | 3031 | 0.14 | 29 |
| j5o3r3lr1t5_i3 | 3041 | 11 | 0.063 | 114 | 3041 | 0.063 | 13 |
| j5o3r3lr1.2t4_i1 | 1312 | 8 | 0.047 | 58 | 1312 | 0.047 | 9 |
| j5o3r3lr1.2t5_i2 | 1311 | 17 | 0.093 | 94 | 1311 | 0.093 | 21 |
| j5o3r3lr1.2t5_i3 | 2652 | 32 | 0.171 | 231 | 2652 | 0.171 | 33 |
| j5o5r3lr0.8t10_i1 | 2657 | 21 | 0.281 | 253 | 2657 | 0.281 | 23 |
| j5o5r3lr0.8t14_i2 | 3404 | 23 | 0.422 | 224 | 3404 | 0.422 | 25 |
| j5o5r3lr0.8t12_i3 | 6891 | 26 | 0.36 | 237 | 6891 | 0.36 | 27 |
| j5o5r3lr1t10_i1 | 3862 | 22 | 0.39 | 246 | 3862 | 0.39 | 23 |
| j5o5r3lr1t10_i2 | 3176 | 23 | 0.485 | 453 | 3217.71 | 0.563 | 27 |
| j5o5r3lr1t11_i3 | 5992 | 27 | 0.516 | 352 | 5992 | 0.547 | 29 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| j5o5r3lr1.2t8_i1 | 1171 | 15 | 0.203 | 184 | 1171 | 0.219 | 17 |
| j5o5r3lr1.2t9_i2 | 3048 | 17 | 0.375 | 500 | 3136 | 0.781 | 43 |
| j5o5r3lr1.2t8_i3 | 1700 | 9 | 0.11 | 85 | 1700 | 0.11 | 11 |
| j5o8r5lr0.8t11_i1 | 2725 | 13 | 0.359 | 166 | 2725 | 0.375 | 15 |
| j5o8r5lr0.8t13_i2 | 8512 | 46 | 3.406 | 7873 | 8666.37 | 31.031 | 353 |
| j5o8r5lr0.8t12_i3 | 10999 | 46 | 2.672 | 3320 | 11166.4 | 9.563 | 147 |
| j5o8r5lr1t9_i1 | 1502 | 72 | 1.531 | 826 | 1502 | 1.531 | 73 |
| j5o8r5lr1t9_i2 | 3667 | 39 | 1.266 | 577 | 3667 | 1.297 | 41 |
| j5o8r5lr1t9_i3 | 2751 | 36 | 1.171 | 564 | 2788.88 | 1.359 | 43 |
| j5o8r5lr1.2t9_i1 | 3733 | 43 | 1.688 | 670 | 3733 | 1.703 | 45 |
| j5o8r5lr1.2t8_i2 | 2059 | 26 | 0.64 | 545 | 2120.61 | 1.171 | 53 |
| j5o8r5lr1.2t8_i3 | 6796.5 | 116 | 3.922 | 2052 | 6928.87 | 4.422 | 129 |
| j5o10r5lr0.8t15_i1 | 6713 | 65 | 6.844 | 61873 | 6916.71 | 900.312 | 2983 |
| j5o10r5lr0.8t14_i2 | 2784 | 98 | 7.782 | 2648 | 2897.65 | 8.719 | 109 |
| j5o10r5lr0.8t16_i3 | 9682 | 54 | 6.703 | 8381 | 9944.86 | 41.922 | 203 |
| j5o10r5lr1t12_i1 | 5526 | 48 | 6.656 | 68219 | 5956.73 | 901.047 | 2675 |
| j5o10r5lr1t12_i2 | 6175 | 34 | 3.047 | 1116 | 6237.16 | 4.219 | 43 |
| j5o10r5lr1t12_i3 | 5937 | 556 | 61.063 | 66812 | 6283.92 | 900.781 | 3065 |
| j5o10r5lr1.2t10_i1 | 3559 | 23 | 1.359 | 614 | 3615.09 | 1.859 | 39 |
| j5o10r5lr1.2t10_i2 | 5461 | 39 | 3.766 | 3222 | 5707.14 | 11.797 | 119 |
| j5o10r5lr1.2t10_i3 | 4370 | 111 | 6.125 | 2875 | 4463.52 | 9.219 | 177 |
| j8o3r3lr0.8t13_i1 | 2295 | 35 | 0.422 | 311 | 2295 | 0.453 | 37 |
| j8o3r3lr0.8t13_i2 | 4313 | 26 | 0.313 | 184 | 4313 | 0.313 | 27 |
| j8o3r3lr0.8t11_i3 | 4193 | 19 | 0.187 | 154 | 4193 | 0.203 | 21 |
| j8o3r3lr1t9_i1 | 2889 | 27 | 0.188 | 168 | 2889 | 0.203 | 29 |
| j8o3r3lr1t9_i2 | 2069 | 43 | 0.297 | 282 | 2069 | 0.297 | 45 |
| j8o3r3lr1t10_i3 | 3218 | 25 | 0.218 | 174 | 3218 | 0.234 | 27 |
| j8o3r3lr1.2t8_i1 | 2963 | 15 | 0.14 | 160 | 2963 | 0.156 | 17 |
| j8o3r3lr1.2t8_i2 | 5436 | 17 | 0.234 | 253 | 5436 | 0.25 | 19 |
| j8o3r3lr1.2t8_i3 | 3202 | 56 | 0.469 | 447 | 3202 | 0.469 | 57 |
| j8o5r3lr0.8t20_i1 | 5940 | 60 | 1.766 | 715 | 5940 | 1.766 | 61 |
| j8o5r3lr0.8t20_i2 | 5941 | 74 | 3 | 924 | 5941 | 3 | 75 |
| j8o5r3lr0.8t20_i3 | 6924 | 37 | 1.125 | 423 | 6924 | 1.172 | 39 |
| j8o5r3lr1t17_i1 | 6739 | 133 | 6.375 | 83342 | 6915.17 | 900.094 | 4699 |
| j8o5r3lr1t14_i2 | 5724 | 90 | 2 | 945 | 5724 | 2.016 | 91 |
| j8o5r3lr1t16_i3 | 4009 | 30 | 0.578 | 289 | 4009 | 0.578 | 31 |
| j8o5r3lr1.2t14_i1 | 5371 | 193 | 7.265 | 3178 | 5371 | 7.296 | 195 |
| j8o5r3lr1.2t14_i2 | 5807 | 39 | 1.344 | 742 | 5807 | 1.406 | 41 |
| j8o5r3lr1.2t12_i3 | 4004 | 128 | 3.063 | 1770 | 4004 | 3.063 | 129 |
| j8o8r5lr0.8t21_i1 | 6942 | 177 | 45.094 | 60409 | 7252 | 900.515 | 2077 |
| j8o8r5lr0.8t18_i2 | 6784 | 697 | 59.5 | 9563 | 6784 | 59.64 | 699 |
| j8o8r5lr0.8t19_i3 | 14415 | 55 | 12.156 | 69131 | 14865.7 | 900.14 | 2861 |
| j8o8r5lr1t15_i1 | 15741 | 131 | 30.672 | 77517 | 16218.5 | 902.672 | 1645 |
| j8o8r5lr1t16_i2 | 10644 | 771 | 198.703 | 68139 | 10965.7 | 900.386 | 2285 |
| j8o8r5lr1t16_i3 | 10780 | 86 | 12.453 | 68735 | 10996 | 900.906 | 2297 |
| j8o8r5lr1.2t13_i1 | 9776 | 941 | 151.141 | 72692 | 10187.3 | 900.125 | 2763 |
| j8o8r5lr1.2t13_i2 | 7055 | 197 | 31.39 | 69320 | 7342.7 | 900.719 | 2709 |
| j8o8r5lr1.2t13_i3 | 4382 | 245 | 36.062 | 8255 | 4441.61 | 39.687 | 283 |
| j8o10r5lr0.8t26_i1 | 16650 | 97 | 38.484 | 52009 | 17432 | 901.083 | 1199 |
| j8o10r5lr0.8t23_i2 | 10866 | 104 | 32 | 55694 | 11258 | 900.266 | 1259 |
| j8o10r5lr0.8t23_i3 | 18116 | 1147 | 531.765 | 43534 | 18116 | 531.984 | 1149 |
| j8o10r5lr1t19_i1 | 12560 | 131 | 31.375 | 65360 | 12863.2 | 901.594 | 1621 |
| j8o10r5lr1t17_i2 | 8884 | 94 | 19.297 | 64462 | 9314.21 | 900.706 | 2865 |
| j8o10r5lr1t19_i3 | 8704 | 360 | 89.953 | 13987 | 8809.95 | 101.297 | 395 |
| j8o10r5lr1.2t15_i1 | 8599 | 411 | 92.079 | 64714 | 9200.57 | 902.188 | 2083 |
| j8o10r5lr1.2t16_i2 | 14060 | 101 | 44.782 | 59747 | 15110.5 | 901.485 | 1909 |
| j8o10r5lr1.2t16_i3 | 10233 | 287 | 84.672 | 61304 | 10591.5 | 901.281 | 1905 |
| j10o3r3lr0.8t15_i1 | 7366 | 33 | 0.532 | 290 | 7366 | 0.547 | 35 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| j10o3r3lr0.8t15_i2 | 3529 | 35 | 0.563 | 376 | 3529 | 0.578 | 37 |
| j10o3r3lr0.8t14_i3 | 4676 | 37 | 0.437 | 282 | 4676 | 0.453 | 39 |
| j10o3r3lr1t12_i1 | 3639 | 30 | 0.422 | 323 | 3639 | 0.422 | 31 |
| j10o3r3lr1t12_i2 | 5195 | 32 | 0.594 | 400 | 5195 | 0.61 | 33 |
| j10o3r3lr1t11_i3 | 4658 | 139 | 1.813 | 1350 | 4658 | 1.828 | 141 |
| j10o3r3lr1.2t10_i1 | 4683 | 40 | 0.531 | 355 | 4683 | 0.531 | 41 |
| j10o3r3lr1.2t9_i2 | 4373 | 24 | 0.188 | 166 | 4373 | 0.188 | 25 |
| j10o3r3lr1.2t10_i3 | 4306 | 54 | 0.703 | 555 | 4306 | 0.703 | 55 |
| j10o5r3lr0.8t26_i1 | 9833 | 76 | 4.906 | 928 | 9833 | 4.906 | 77 |
| j10o5r3lr0.8t28_i2 | 11817 | 319 | 37.579 | 6646 | 11817 | 37.844 | 321 |
| j10o5r3lr0.8t25_i3 | 9189 | 58 | 2.937 | 691 | 9189 | 2.937 | 59 |
| j10o5r3lr1t20_i1 | 4672 | 144 | 6.781 | 1740 | 4672 | 6.781 | 145 |
| j10o5r3lr1t19_i2 | 6156 | 54 | 2.766 | 972 | 6156 | 2.766 | 55 |
| j10o5r3lr1t19_i3 | 7072 | 61 | 3.485 | 1098 | 7072 | 3.563 | 63 |
| j10o5r3lr1.2t16_i1 | 3896 | 93 | 3.562 | 1376 | 3896 | 3.578 | 95 |
| j10o5r3lr1.2t18_i2 | 6193 | 37 | 1.735 | 569 | 6193 | 1.766 | 39 |
| j10o5r3lr1.2t18_i3 | 7350 | 39 | 2.437 | 951 | 7350 | 2.5 | 41 |
| j10o8r5lr0.8t24_i1 | 13191 | 224 | 43.579 | 5239 | 13191 | 43.594 | 225 |
| j10o8r5lr0.8t24_i2 | 10399 | 109 | 22.204 | 58867 | 10589.1 | 900.69 | 2105 |
| j10o8r5lr0.8t22_i3 | 11163 | 101 | 11.406 | 71622 | 11289 | 900.281 | 2365 |
| j10o8r5lr1t17_i1 | 12070 | 1404 | 574.188 | 68225 | 12294 | 901 | 1835 |
| j10o8r5lr1t18_i2 | 8518 | 58 | 6.375 | 76483 | 8716.19 | 901.047 | 2241 |
| j10o8r5lr1t19_i3 | 9181 | 2847 | 878.093 | 63098 | 9346 | 901.656 | 2903 |
| j10o8r5lr1.2t16_i1 | 13412 | 214 | 42.516 | 69163 | 13790 | 900.672 | 2577 |
| j10o8r5lr1.2t16_i2 | 13027 | 457 | 95.063 | 72173 | 13293 | 900.094 | 2271 |
| j10o8r5lr1.2t16_i3 | 9986 | 170 | 29.391 | 69556 | 10160.4 | 900.454 | 2249 |
| j10o10r5lr0.8t29_i1 | 18323 | 128 | 29.282 | 2307 | 18323 | 29.282 | 129 |
| j10o10r5lr0.8t29_i2 | 11760 | 143 | 48 | 48338 | 12091 | 901.847 | 1489 |
| j10o10r5lr0.8t30_i3 | 17129 | 391 | 246.844 | 47893 | 17464 | 901 | 1031 |
| j10o10r5lr1t24_i1 | 10422 | 241 | 188.75 | 52179 | 11023.9 | 901.282 | 933 |
| j10o10r5lr1t23_i2 | 15561 | 190 | 68.312 | 62622 | 16083.5 | 900.234 | 1585 |
| j10o10r5lr1t25_i3 | 15390 | 218 | 88.813 | 56203 | 15980.6 | 900.505 | 1289 |
| j10o10r5lr1.2t19_i1 | 17337 | 336 | 140.75 | 57634 | 18177.1 | 900.547 | 1349 |
| j10o10r5lr1.2t21_i2 | 12694 | 177 | 64.172 | 8026 | 12694 | 69.578 | 187 |
| j10o10r5lr1.2t19_i3 | 15639 | 82 | 24.625 | 65165 | 15882.9 | 900.063 | 1649 |

# APPENDIX L

| Instance | Best Integer Solution | Best Integer @ node | Time to Best Integer Solution (seconds) | Columns Added | Best Bound | Runtime (seconds) | Total Nodes |
|---|---|---|---|---|---|---|---|
| j3o3r3lr0.8t4_i1 | 794 | 7 | 0.031 | 28 | 794 | 0.031 | 9 |
| j3o3r3lr0.8t5_i2 | 1135 | 3 | 0.031 | 30 | 1135 | 0.031 | 5 |
| j3o3r3lr0.8t5_i3 | 1807 | 1 | 0.031 | 26 | 1807 | 0.031 | 3 |
| j3o3r3lr1t4_i1 | 1120 | 2 | 0.031 | 20 | 1120 | 0.031 | 3 |
| j3o3r3lr1t4_i2 | 1169 | 3 | 0.016 | 25 | 1169 | 0.016 | 5 |
| j3o3r3lr1t3_i3 | 721 | 17 | 0.031 | 81 | 721 | 0.047 | 19 |
| j3o3r3lr1.2t2_i1 | 92 | 0 | 0 | 1 | 92 | 0.031 | 3 |
| j3o3r3lr1.2t3_i2 | 274 | 0 | 0 | 3 | 274 | 0.016 | 3 |
| j3o3r3lr1.2t3_i3 | 1202 | 3 | 0.031 | 30 | 1202 | 0.031 | 5 |
| j3o5r3lr0.8t8_i1 | 2703 | 16 | 0.234 | 191 | 2703 | 0.234 | 17 |
| j3o5r3lr0.8t6_i2 | 3366 | 0 | 0 | 49 | 3366 | 0.047 | 3 |
| j3o5r3lr0.8t7_i3 | 2157 | 10 | 0.125 | 121 | 2157 | 0.125 | 11 |
| j3o5r3lr1t6_i1 | 2134 | 3 | 0.062 | 77 | 2134 | 0.078 | 5 |
| j3o5r3lr1t6_i2 | 3661 | 8 | 0.078 | 80 | 3661 | 0.078 | 9 |
| j3o5r3lr1t6_i3 | 2367 | 12 | 0.093 | 107 | 2367 | 0.093 | 13 |
| j3o5r3lr1.2t5_i1 | 2282 | 18 | 0.14 | 123 | 2282 | 0.14 | 19 |
| j3o5r3lr1.2t6_i2 | 164 | 0 | 0 | 19 | 164 | 0.046 | 3 |
| j3o5r3lr1.2t5_i3 | 988 | 0 | 0 | 1 | 988 | 0.015 | 3 |
| j3o8r5lr0.8t8_i1 | 2195 | 0 | 0 | 17 | 2195 | 0.047 | 3 |
| j3o8r5lr0.8t7_i2 | 1889 | 1 | 0.031 | 19 | 1889 | 0.047 | 3 |
| j3o8r5lr0.8t8_i3 | 2998 | 22 | 0.531 | 287 | 2998 | 0.531 | 23 |
| j3o8r5lr1t6_i1 | 266 | 1 | 0.046 | 7 | 266 | 0.046 | 3 |
| j3o8r5lr1t5_i2 | 1156 | 3 | 0.031 | 18 | 1156 | 0.031 | 5 |
| j3o8r5lr1t6_i3 | 1550 | 0 | 0 | 29 | 1550 | 0.047 | 3 |
| j3o8r5lr1.2t4_i1 | 649 | 0 | 0 | 1 | 649 | 0.031 | 3 |
| j3o8r5lr1.2t4_i2 | 1102 | 1 | 0.016 | 3 | 1102 | 0.016 | 3 |
| j3o8r5lr1.2t5_i3 | 1806 | 0 | 0.031 | 16 | 1806 | 0.031 | 1 |
| j3o10r5lr0.8t9_i1 | 1620 | 3 | 0.156 | 57 | 1620 | 0.171 | 5 |
| j3o10r5lr0.8t9_i2 | 3911 | 98 | 4.016 | 1536 | 3911 | 4.031 | 99 |
| j3o10r5lr0.8t9_i3 | 2541 | 14 | 0.328 | 144 | 2541 | 0.328 | 15 |
| j3o10r5lr1t8_i1 | 6634 | 20 | 0.516 | 231 | 6634 | 0.516 | 21 |
| j3o10r5lr1t7_i2 | 861 | 0 | 0 | 1 | 861 | 0.031 | 3 |
| j3o10r5lr1t7_i3 | 3171 | 6 | 0.218 | 105 | 3171 | 0.218 | 7 |
| j3o10r5lr1.2t6_i1 | 1E-07 | 0 | 0 | 0 | 1E-07 | 0.031 | 3 |
| j3o10r5lr1.2t5_i2 | 3004 | 1 | 0.032 | 10 | 3004 | 0.032 | 3 |
| j3o10r5lr1.2t6_i3 | 137 | 0 | 0 | 1 | 137 | 0.031 | 3 |
| j5o3r3lr0.8t7_i1 | 1729 | 13 | 0.078 | 107 | 1729 | 0.078 | 15 |
| j5o3r3lr0.8t8_i2 | 2259 | 11 | 0.078 | 66 | 2259 | 0.078 | 13 |
| j5o3r3lr0.8t8_i3 | 2627 | 13 | 0.078 | 68 | 2627 | 0.078 | 15 |
| j5o3r3lr1t6_i1 | 3621 | 9 | 0.078 | 82 | 3621 | 0.078 | 11 |
| j5o3r3lr1t6_i2 | 2961 | 18 | 0.078 | 123 | 2961 | 0.078 | 19 |
| j5o3r3lr1t5_i3 | 3041 | 11 | 0.063 | 114 | 3041 | 0.063 | 13 |
| j5o3r3lr1.2t4_i1 | 1312 | 8 | 0.031 | 58 | 1312 | 0.031 | 9 |
| j5o3r3lr1.2t5_i2 | 1311 | 17 | 0.078 | 93 | 1311 | 0.078 | 19 |
| j5o3r3lr1.2t5_i3 | 2670 | 33 | 0.187 | 268 | 2670 | 0.187 | 35 |
| j5o5r3lr0.8t10_i1 | 2657 | 21 | 0.281 | 253 | 2657 | 0.281 | 23 |
| j5o5r3lr0.8t14_i2 | 3404 | 23 | 0.453 | 224 | 3404 | 0.453 | 25 |
| j5o5r3lr0.8t12_i3 | 6891 | 26 | 0.36 | 237 | 6891 | 0.36 | 27 |
| j5o5r3lr1t10_i1 | 3862 | 22 | 0.375 | 246 | 3862 | 0.375 | 23 |
| j5o5r3lr1t10_i2 | 3176 | 23 | 0.485 | 412 | 3176 | 0.516 | 25 |
| j5o5r3lr1t11_i3 | 5767 | 0 | 0 | 75 | 5767 | 0.094 | 3 |

| Instance | Best Integer Solution | Best Integer @ node | Time to Best Integer Solution (seconds) | Columns Added | Best Bound | Runtime (seconds) | Total Nodes |
|---|---|---|---|---|---|---|---|
| j5o5r3lr1.2t8_i1 | 1171 | 15 | 0.203 | 184 | 1171 | 0.219 | 17 |
| j5o5r3lr1.2t9_i2 | 3048 | 17 | 0.375 | 245 | 3048 | 0.375 | 19 |
| j5o5r3lr1.2t8_i3 | 1700 | 9 | 0.109 | 85 | 1700 | 0.125 | 11 |
| j5o8r5lr0.8t11_i1 | 2725 | 13 | 0.344 | 166 | 2725 | 0.375 | 15 |
| j5o8r5lr0.8t13_i2 | 8512 | 46 | 3.406 | 1120 | 8512 | 3.406 | 47 |
| j5o8r5lr0.8t12_i3 | 10839 | 23 | 1.453 | 612 | 10839 | 1.531 | 25 |
| j5o8r5lr1t9_i1 | 1502 | 78 | 1.796 | 976 | 1502 | 1.796 | 79 |
| j5o8r5lr1t9_i2 | 3530 | 34 | 1.187 | 515 | 3530 | 1.187 | 35 |
| j5o8r5lr1t9_i3 | 2751 | 36 | 1.156 | 481 | 2751 | 1.156 | 37 |
| j5o8r5lr1.2t9_i1 | 3733 | 43 | 1.688 | 670 | 3733 | 1.703 | 45 |
| j5o8r5lr1.2t8_i2 | 2059 | 26 | 0.625 | 303 | 2059 | 0.625 | 27 |
| j5o8r5lr1.2t8_i3 | 6796.5 | 96 | 3.453 | 1614 | 6796.5 | 3.453 | 97 |
| j5o10r5lr0.8t15_i1 | 6628 | 40 | 4.515 | 1048 | 6628 | 4.531 | 41 |
| j5o10r5lr0.8t14_i2 | 2719 | 35 | 2.937 | 1267 | 2925.25 | 3.953 | 47 |
| j5o10r5lr0.8t16_i3 | 9682 | 54 | 6.672 | 1631 | 9682 | 6.672 | 55 |
| j5o10r5lr1t12_i1 | 5526 | 48 | 6.609 | 70360 | 5961.02 | 900.484 | 2027 |
| j5o10r5lr1t12_i2 | 6175 | 34 | 3.032 | 780 | 6175 | 3.032 | 35 |
| j5o10r5lr1t12_i3 | 5985 | 433 | 58.938 | 11937 | 6287.74 | 64.156 | 465 |
| j5o10r5lr1.2t10_i1 | 3501 | 16 | 1.047 | 368 | 3501 | 1.047 | 17 |
| j5o10r5lr1.2t10_i2 | 5461 | 39 | 3.719 | 1048 | 5461 | 3.75 | 41 |
| j5o10r5lr1.2t10_i3 | 4346 | 72 | 4 | 1337 | 4346 | 4 | 73 |
| j8o3r3lr0.8t13_i1 | 2263 | 0 | 0 | 56 | 2263 | 0.063 | 3 |
| j8o3r3lr0.8t13_i2 | 4181 | 0 | 0 | 64 | 4181 | 0.078 | 3 |
| j8o3r3lr0.8t11_i3 | 4057 | 0 | 0 | 56 | 4057 | 0.063 | 3 |
| j8o3r3lr1t9_i1 | 2889 | 27 | 0.188 | 168 | 2889 | 0.203 | 29 |
| j8o3r3lr1t9_i2 | 2012 | 24 | 0.171 | 147 | 2012 | 0.171 | 25 |
| j8o3r3lr1t10_i3 | 3218 | 25 | 0.219 | 174 | 3218 | 0.219 | 27 |
| j8o3r3lr1.2t8_i1 | 2963 | 15 | 0.125 | 160 | 2963 | 0.14 | 17 |
| j8o3r3lr1.2t8_i2 | 5436 | 17 | 0.235 | 253 | 5436 | 0.25 | 19 |
| j8o3r3lr1.2t8_i3 | 3081 | 29 | 0.25 | 236 | 3081 | 0.265 | 31 |
| j8o5r3lr0.8t20_i1 | 5859 | 34 | 0.985 | 381 | 5859 | 0.985 | 35 |
| j8o5r3lr0.8t20_i2 | 5681 | 0 | 0 | 78 | 5681 | 0.172 | 5 |
| j8o5r3lr0.8t20_i3 | 6630 | 0 | 0 | 88 | 6630 | 0.172 | 3 |
| j8o5r3lr1t17_i1 | 6820 | 142 | 7.109 | 2423 | 6820 | 7.109 | 143 |
| j8o5r3lr1t14_i2 | 5724 | 90 | 1.968 | 945 | 5724 | 1.968 | 91 |
| j8o5r3lr1t16_i3 | 4009 | 30 | 0.61 | 289 | 4009 | 0.61 | 31 |
| j8o5r3lr1.2t14_i1 | 5199 | 65 | 2.25 | 1022 | 5199 | 2.281 | 67 |
| j8o5r3lr1.2t14_i2 | 5807 | 39 | 1.344 | 742 | 5807 | 1.391 | 41 |
| j8o5r3lr1.2t12_i3 | 3951 | 41 | 1 | 688 | 3951 | 1 | 43 |
| j8o8r5lr0.8t21_i1 | 6930 | 87 | 24.297 | 3360 | 6930 | 24.437 | 89 |
| j8o8r5lr0.8t18_i2 | 6565 | 80 | 5.359 | 1176 | 6565 | 5.359 | 81 |
| j8o8r5lr0.8t19_i3 | 14415 | 55 | 12.016 | 2267 | 14415 | 12.25 | 57 |
| j8o8r5lr1t15_i1 | 15617 | 83 | 18.609 | 3936 | 15617 | 18.687 | 85 |
| j8o8r5lr1t16_i2 | 10303 | 89 | 18.578 | 71755 | 10965.7 | 901.234 | 1479 |
| j8o8r5lr1t16_i3 | 10780 | 86 | 12.282 | 2430 | 10780 | 12.297 | 87 |
| j8o8r5lr1.2t13_i1 | 9643 | 1480 | 385.281 | 44536 | 10211.6 | 408.828 | 1541 |
| j8o8r5lr1.2t13_i2 | 6984 | 60 | 10.031 | 3349 | 6984 | 14.203 | 83 |
| j8o8r5lr1.2t13_i3 | 4329 | 32 | 3.64 | 1173 | 4329 | 3.64 | 33 |
| j8o10r5lr0.8t26_i1 | 16650 | 97 | 38.25 | 4117 | 16650 | 38.516 | 99 |
| j8o10r5lr0.8t23_i2 | 10866 | 104 | 31.844 | 3891 | 10866 | 31.86 | 105 |
| j8o10r5lr0.8t23_i3 | 17926 | 90 | 21.344 | 2854 | 17926 | 21.36 | 91 |
| j8o10r5lr1t19_i1 | 12339 | 74 | 16.032 | 2624 | 12339 | 16.047 | 75 |
| j8o10r5lr1t17_i2 | 8884 | 94 | 18.953 | 3434 | 8884 | 18.953 | 95 |

| Instance | Best Integer Solution | Best Integer @ node | Time to Best Integer Solution (seconds) | Columns Added | Best Bound | Runtime (seconds) | Total Nodes |
|---|---|---|---|---|---|---|---|
| j8o10r5lr1t19_i3 | 8530 | 63 | 15.672 | 2501 | 8530 | 15.735 | 65 |
| j8o10r5lr1.2t15_i1 | 8466 | 79 | 24.219 | 67664 | 9203.25 | 900.373 | 1665 |
| j8o10r5lr1.2t16_i2 | 14060 | 101 | 44.422 | 66222 | 15117 | 900.688 | 1047 |
| j8o10r5lr1.2t16_i3 | 10269 | 419 | 130.406 | 15683 | 10269 | 130.765 | 421 |
| j10o3r3lr0.8t15_i1 | 7366 | 33 | 0.516 | 290 | 7366 | 0.516 | 35 |
| j10o3r3lr0.8t15_i2 | 3529 | 35 | 0.563 | 376 | 3529 | 0.578 | 37 |
| j10o3r3lr0.8t14_i3 | 4532 | 0 | 0 | 70 | 4532 | 0.078 | 3 |
| j10o3r3lr1t12_i1 | 3639 | 30 | 0.422 | 323 | 3639 | 0.422 | 31 |
| j10o3r3lr1t12_i2 | 5195 | 32 | 0.609 | 400 | 5195 | 0.609 | 33 |
| j10o3r3lr1t11_i3 | 4574 | 26 | 0.328 | 285 | 4574 | 0.328 | 27 |
| j10o3r3lr1.2t10_i1 | 4683 | 40 | 0.531 | 355 | 4683 | 0.531 | 41 |
| j10o3r3lr1.2t9_i2 | 4373 | 24 | 0.187 | 166 | 4373 | 0.187 | 25 |
| j10o3r3lr1.2t10_i3 | 4240 | 31 | 0.438 | 358 | 4240 | 0.453 | 33 |
| j10o5r3lr0.8t26_i1 | 9405 | 0 | 0 | 121 | 9405 | 0.344 | 3 |
| j10o5r3lr0.8t28_i2 | 11709 | 208 | 21.969 | 3851 | 11709 | 21.969 | 209 |
| j10o5r3lr0.8t25_i3 | 9189 | 58 | 2.937 | 691 | 9189 | 2.937 | 59 |
| j10o5r3lr1t20_i1 | 4511 | 61 | 3.016 | 820 | 4511 | 3.063 | 63 |
| j10o5r3lr1t19_i2 | 5903 | 0 | 0 | 138 | 5903 | 0.235 | 3 |
| j10o5r3lr1t19_i3 | 7072 | 61 | 3.469 | 1098 | 7072 | 3.547 | 63 |
| j10o5r3lr1.2t16_i1 | 3809 | 56 | 2.172 | 923 | 3809 | 2.172 | 57 |
| j10o5r3lr1.2t18_i2 | 6193 | 37 | 1.719 | 569 | 6193 | 1.765 | 39 |
| j10o5r3lr1.2t18_i3 | 7350 | 39 | 2.438 | 951 | 7350 | 2.5 | 41 |
| j10o8r5lr0.8t24_i1 | 13197 | 1251 | 566.344 | 47446 | 13197 | 567.391 | 1253 |
| j10o8r5lr0.8t24_i2 | 10458 | 203 | 41.047 | 4370 | 10458 | 41.25 | 205 |
| j10o8r5lr0.8t22_i3 | 11163 | 101 | 11.359 | 1619 | 11163 | 11.641 | 103 |
| j10o8r5lr1t17_i1 | 11929 | 193 | 48.735 | 8196 | 11929 | 48.969 | 195 |
| j10o8r5lr1t18_i2 | 8518 | 58 | 6.359 | 1523 | 8518 | 6.359 | 59 |
| j10o8r5lr1t19_i3 | 9089 | 85 | 13.016 | 2243 | 9089 | 13.141 | 87 |
| j10o8r5lr1.2t16_i1 | 13274 | 68 | 14.688 | 3096 | 13274 | 14.704 | 69 |
| j10o8r5lr1.2t16_i2 | 12875 | 81 | 12.844 | 3024 | 12875 | 13 | 83 |
| j10o8r5lr1.2t16_i3 | 9873 | 77 | 13.657 | 2681 | 9873 | 13.829 | 79 |
| j10o10r5lr0.8t29_i1 | 18323 | 128 | 29.547 | 2307 | 18323 | 29.562 | 129 |
| j10o10r5lr0.8t29_i2 | 11760 | 143 | 48 | 3875 | 11760 | 49.031 | 145 |
| j10o10r5lr0.8t30_i3 | 16664 | 139 | 82.781 | 5932 | 16664 | 83.156 | 141 |
| j10o10r5lr1t24_i1 | 10200 | 125 | 93.766 | 51936 | 11023.9 | 900.857 | 839 |
| j10o10r5lr1t23_i2 | 15561 | 190 | 68.124 | 7955 | 15561 | 68.14 | 191 |
| j10o10r5lr1t25_i3 | 14936 | 104 | 41.078 | 55796 | 15980.6 | 900.438 | 1157 |
| j10o10r5lr1.2t19_i1 | 17161 | 99 | 38.297 | 63451 | 18177.1 | 900.86 | 1061 |
| j10o10r5lr1.2t21_i2 | 12370.5 | 90 | 33.859 | 4285 | 12370.5 | 33.859 | 91 |
| j10o10r5lr1.2t19_i3 | 15639 | 82 | 24.61 | 3424 | 15639 | 24.625 | 83 |

# VITA

## SIDDHARTH D. MESTRY

| | |
|---|---|
| November 9, 1978 | Born, Bombay (Mumbai), India |
| 2001 | B.E. Mechanical Engineering<br>University of Mumbai<br>India |
| 2005 | M.S. Industrial & Systems Engineering<br>Rochester Institute of Technology<br>Rochester, NY<br>USA |

## PUBLICATIONS & PRESENTATIONS

Chen, C.S., Mestry, S., Damodaran, P., Wang, C., "The Capacity Planning Problem in Make-to-order Enterprises," Computer & Mathematical Modeling, Vol. 50, November 2009.

Damodaran, P., Mestry, S., M Krishnamurthi., "Genetic Algorithm to Minimize Makespan of a Capacitated Batch Processing Machine," Journal of Management and Engineering Integration (accepted).

Damodaran, P., Mestry S.D., Zuniga M., Perez J., "A Mathematical Model for Scheduling Shipboard Crew in Cruise Lines", submitted to Proceedings of the 19th Annual Industrial Engineering Research Conference, Cancun, 2010.

Mestry, S., Damodaran, P., Chen, C., Wang, C., "A Capacity Planning Model for Make-to-Order Enterprise," Proceedings of the 13th Annual International Conference of Industry, Engineering and Management Systems, Cocoa Beach, FL, 2007.

Mestry, S., Damodaran, P., Krishnamurthi, M., "Scheduling a Capacitated Batch Processing Machine using Genetic Algorithms," Proceedings of the 16th Annual Industrial Engineering Research Conference, Nashville, TN, 2007.

Mestry S.D., Damodaran, P., and Rao, A.G., "Particle Swarm Optimization for Scheduling Batch Processing Machines in Permutation Flowshops," Proceedings of the 18th Annual Industrial Engineering Research Conference, Miami, 2009.

Mestry, S.D., Damodaran, P., Chen, C.S., "Branch & Price Solution Approach for Solving Order Acceptance and Capacity Planning Problem in Make-to-Order Operations," (submitted to European Journal of Operational Research).

Perez, J., Mestry, S., Damodaran, P., “Heuristics for Minimizing Makepsan of Batch Processing Machines in a Flowshop,” Proceedings of the 17th Annual Industrial Engineering Research Conference, Vancouver, 2008.

Stiebitz, P.H., Carrano, A.L., Taylor, J.B., Plaz, C.R., Mestry, S., “Swarm of Microsystem Particles for Multi-Axial Morphogenic Rapid Prototyping,” Proceedings of the 14th Annual Industrial Engineering Research Conference, Atlanta, GA, 2005.