Elsevier Editorial System(tm) for European Journal of Operational Research
Manuscript Draft

Manuscript Number:

Title: A Branch and Price Solution Approach for Order Acceptance and Capacity Planning in Make-to-Order Operations

Article Type: Theory and Methodology Paper

Section/Category: Manufacturing

Keywords: Order Acceptance, Branch-and-Price, Capacity Planning, Make-to-Order Operations, and Large-Scale Optimization.

Corresponding Author: Prof. Purushothaman Damodaran, PhD

Corresponding Author's Institution: Northern Illinois University

First Author: Siddharth Mestry, M.S.

Order of Authors: Siddharth Mestry, M.S.; Purushothaman Damodaran, PhD; Chin-Sheng Chen, PhD

Abstract: Make-to-order (MTO) operations have to effectively manage their capacity to make long-term sustainable profits. This objective can be met by selectively accepting available customer orders and simultaneously planning for capacity. We model a MTO operation of a job-shop with multiple resources having regular and non-regular capacity. The MTO firm has a set of customer orders at time zero with fixed due-dates. The process route, processing times, and sales price for each order are all given. Since orders compete for limited resources, the firm can only accept some orders. In this paper we formulate a Mixed-Integer Linear Program (MILP) to aid an operational manager to decide which orders to accept and how to allocate resources such that the overall profit is maximized. A branch-and-price algorithm is devised to solve the MILP effectively. The MILP is first decomposed into a master problem and several sub-problems using Dantzig-Wolfe decomposition. Each sub-problem is represented as a network flow problem and an exact procedure is proposed to solve the sub-problems efficiently. We also propose an approximate branch-and-price scheme, Lagrangian bounds, and approximations to fathom nodes in the branch-and-bound tree. Computational analysis shows that the proposed branch-and-price algorithm can solve large problem instances with relatively short time.

Cover Letter

Click here to view linked References

NORTHERN ILLINOIS UNIVERSITY

(815) 753-6748
pdamodaran@niu.edu

November 2, 2009

Dear Editor,

We are submitting our manuscript entitled "A Branch and Price Solution Approach for Order Acceptance and Capacity Planning in Make-to-Order Operations" for review and publication in European Journal of Operational Research.

The problem under study can be observed at several manufacturing facilities. The co-authors have personally interacted with several make-to-order firms. The model and the solution proposed in this paper can benefit industry and stimulate academic research to consider several extensions of this problem.

We hope you will find this paper and its contribution worthy to publish in your journal. If you need additional information with regards to the manuscript, please write to me. I look forward to hearing your comments and the reviewers' comments/suggestions on our manuscript.

Sincerely,

Purushothaman Damodaran

Department of Industrial and Systems Engineering
Northern Illinois University
De Kalb, IL 60115

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

Siddharth Mestry[1], Purushothaman Damodaran[2*], Chin-Sheng Chen[1]

[1] Department of Industrial and Systems Engineering,
Florida International University, Miami, FL.

[2] Department of Industrial and Systems Engineering,
Northern Illinois University, DeKalb, IL.

[*] Corresponding author: pdamodaran@niu.edu; (815) 753-6748.

ABSTRACT

Make-to-order (MTO) operations have to effectively manage their capacity to make long-term sustainable profits. This objective can be met by selectively accepting available customer orders and simultaneously planning for capacity. We model a MTO operation of a job-shop with multiple resources having regular and non-regular capacity. The MTO firm has a set of customer orders at time zero with fixed due-dates. The process route, processing times, and sales price for each order are all given. Since orders compete for limited resources, the firm can only accept some orders. In this paper we formulate a Mixed-Integer Linear Program (MILP) to aid an operational manager to decide which orders to accept and how to allocate resources such that the overall profit is maximized. A branch-and-price algorithm is devised to solve the MILP effectively. The MILP is first decomposed into a master problem and several sub-problems using Dantzig-Wolfe decomposition. Each sub-problem is represented as a network flow problem and an exact procedure is proposed to solve the sub-problems efficiently. We also propose an approximate branch-and-price scheme, Lagrangian bounds, and approximations to fathom nodes in the branch-and-bound tree. Computational analysis shows that the proposed branch-and-price algorithm can solve large problem instances with relatively short time.

Keywords: Order Acceptance, Branch-and-Price, Capacity Planning, Make-to-Order Operations, and Large-Scale Optimization.

1. INTRODUCTION

1.1. Background

The focus on innovation and customer satisfaction has led to shortened product development life cycles and mass customization, compelling the manufacturers to remain agile and flexible. These factors have contributed to an increase in the popularity of make-to-order (MTO) operational philosophy (Jalora, 2006). MTO firms are process-focused, as the products manufactured share the same kind of operations but differ in the design details making them efficient not only for unique product manufacturing but also for producing greater product variety at lower cost (Gallien et al., 2004). This policy allows a high degree of operational flexibility and the products manufactured are one of a kind or in small batches. It is advantageous when the end product is customer specific with high component-level customization unique to each customer. A MTO firm starts working on an order only after it has been placed by the customer. Typical examples of MTO operations are found in engineering tooling, industrial boilers, chemical equipment, construction, and general engineering/contracting industries (Chen, 2006).

MTO is characterized by back orders with zero inventories as each customer order is unique and cannot be manufactured in advance. The only way to make sustainable profits is by managing the customer demands which is achieved by effectively and efficiently using available capacity. Because the main driver in MTO operations is customer orders, it is vital to coordinate operations and sales functions for effective use of available resources by managing the demand placed on the system (Mehmet and Sridharan, 2005). 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 thus will tend to accept all orders, regardless of the available capacity, because its goal is to maximize the sales revenue. Manufacturing on the other hand, is concerned with limited capacity and tries to maximize resources utilization, while minimizing the number of tardy deliveries. Without effective coordination and look-forward mechanisms, order acceptance decisions are often made without involving production department or with incomplete information on the available capacity (Slotnick and Morton, 2007). Accepting each available order is the tendency of the sales department, which often leads to an over-loaded production system, making it difficult to meet deadlines and other delivery commitments. To deal with this short-term capacity problem, management usually relies on additional non-regular capacity like overtime and outsourcing, thereby increasing its costs. This may lead to lower profit margins or even negative profits. Tardy deliveries may lead to penalty costs and possibly loss of customer goodwill (Ebben et al., 2005, Slotnick and Morton, 2007).

While negotiating contracts in a MTO environment, the company can either adjust the price or lead time for an order. If the order has non-negotiable tight due-dates, the MTO firm can charge a premium for accepting that order as it might have to be expedited with the use of non-regular capacity. Recent experience of firms such as Amazon.com, however, indicates that customers may be unwilling to accept dynamic pricing as fair (Streitfeld, 2000). 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. With multiple orders, each providing a different profit contribution, the capacity allocation problem becomes an order acceptance or refusal problem (Harris and Pinder, 1995; Barut and Sridharan, 2004).

1.2. Problem Description

A make-to-order operation in a job shop environment is considered in this research. The MTO firm has a set of bids or customer orders to consider. A customer order is referred to as jobs in the context of this research. The decision to be made is which customer order to accept and how to schedule it in order to maximize the profit and to fulfill the accepted orders by the due date. Both decisions should be made simultaneously, otherwise an order may be accepted but the available residual capacity may not permit on-time 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 resource type has one or more machines. Furthermore, job recirculation is allowed, which means that the jobs can visit the same resource more than once. The cost of using a resource depends on its source. The objective considered is to maximize the operational profit over a planning horizon considering only the sales price and the manufacturing costs by accepting a subset of customer orders. The planning horizon is discretized into time buckets of equal length know as time periods. Without loss of generality we assume that each time period is one day. Furthermore each day is divided into two capacity sources viz. regular time and overtime. Overtime is usually considered more expensive. The decision of accepting or rejecting the orders is done at the beginning of the day. Figure 1 shows a schematic representation of a typical order acceptance problem in a job shop environment functioning under a MTO operation mode.

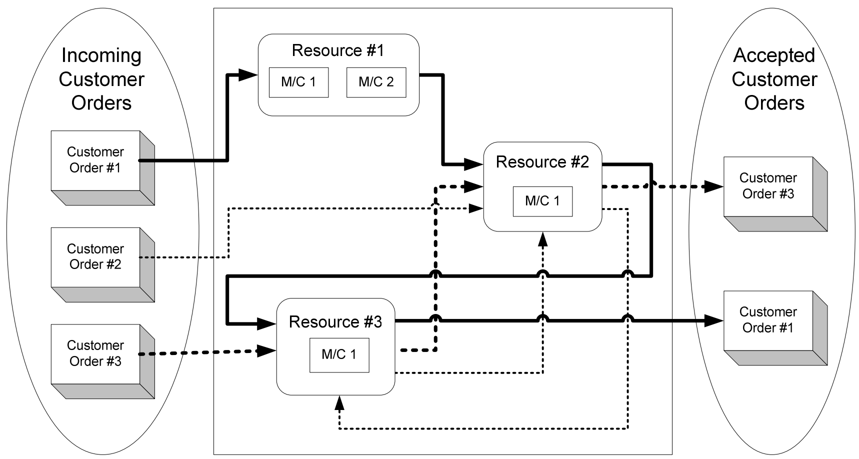


Figure 1. Customer order processing in a job shop-MTO operation

The job shop used for illustration purpose has three resources. Resource 1 has two machines of the same type, while resources 2 and 3 each have a single machine. 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 order acceptance and capacity planning process is to decide which order to accept at the current decision time, and the number of hours for which each resource has to be assigned in each time period and source to each of the accepted jobs, while considering all the constraints stated in the problem description.

The primary objective of this research is to formulate the MTO problem under study and develop solution approaches which can solve large problem instances effectively and efficiently. The problem under study is modeled as a Mixed-Integer Linear Program (MILP). However, the proposed MILP takes prohibitively long runtimes for solving problems with more than 5 jobs as illustrated in Section 3.3. In order to use the proposed model in practice it is important to device efficient solution methodologies. Since the proposed MILP inherits the block diagonal structure, Dantzig-Wolfe decomposition procedure is applied to decompose the MILP into a master problem and several sub-problems (one sub-problem for each customer order). Later a branch-and-price (B&P) algorithm is proposed for solving the proposed MILP.

The rest of this paper is organized as follows. Section 2 reviews relevant literature. Section 3 presents a formal definition of the problem and a mathematical model along with its assumptions and limitations. Section 4 proposes an exact and approximate B&P algorithm to solve the problem under consideration and various approximation schemes for exploring the branch and bound tree. The experimentation and computational results are presented in Section 5, with concluding remarks and future extensions given in Section 6.

2. LITERATURE REVIEW

Order acceptance in manufacturing is closely related to the principles of revenue management (RM) which is commonly used in the service industry for order acceptance and refusal process, with differential pricing, capacity reallocation and overbooking (Harris and Pinder, 1995). There has been an emerging interest in applying RM to the manufacturing industry for both MTO and make-to-stock (MTS) operations. 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. Order acceptance in MTO can be broadly classified by static and dynamic arrivals of customer orders. The problem under study falls in the category of static arrival of customer orders. Section 2.1 focuses on the static arrivals. Section 2.2 focuses on the applications of column generation technique, especially in the area of scheduling.

2.1. 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 selecting and ordering job elements from a given set so as to optimize an objective function was considered by Bahram et al. (2001). They present a generalization of the best-in rule that in many cases can solve the problem while the best-in rule does not.

Slotnick and Morton (1996) 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 (1997) proves that the Slotnick and Morton (1996) version of the job selection problem is NP-Hard. He also proposed two dynamic programs.

In an extension to Slotnick and Morton (1996), Herbert and Slotnick (2002) 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 number of time periods (stages) within a given time horizon. The firm has several 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. it never sends another job to be processed within the planning horizon).

Slotnick and Morton (2007) model a manufacturing facility that considers a pool of orders, and chooses for processing a subset that results in the highest profit. In addition to the problem characteristics in Slotnick and Morton (1996) 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 (2009) also propose a genetic algorithm (GA) to solve the order acceptance problem with tardiness penalties.

2.2. Applications of column generation in scheduling

Column generation has been successfully used in job scheduling for common due date (Van den Akker et al., 1997), parallel machines (Van den Akker et al., 1999a), and single machines (Van den Akker et al., 1999b, 2000). For a detailed taxonomy of the column generation literature we refer to Wilhelm (2001). Hans (2001) developed a B&P loading method that is an exact approach for solving the pre-emptive resource loading problem. The objective is to generate a schedule for each order, such that the total costs of the required non-regular capacity and the tardiness penalties are minimized.

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. (2005); but they do not consider non-regular capacity (overtime) and the customer arrivals are dynamic. A MILP formulation is proposed for the problem under study, its structure is studied and later exploited to develop a B&P algorithm to solve large problem instances of practical interest. To the best of our knowledge the B&P approach has never been used for order acceptance; although Hans (2001) has developed a B&P resource loading (BPRL) approach for scheduling orders which have already been selected. Ebben et al. (2005) use the BPRL technique in their simulation for scheduling the already accepted orders.

Table 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 1. Summary of relevant literature and contribution of proposed research

Research	Objective	Order Acceptance	Multiple Resources	Non-regular Capacity	Fixed Due-dates	Solution Approach
Hans, 2001	Minimize non-regular capacity costs and tardiness penalties	No	Yes	Yes	No	Branch and Price
Slotnick and Morton, 1996	Maximize Profit	Yes	No	No	No	Heuristic
Slotnick and Morton, 2007	Maximize Profit	Yes	No	No	No	Branch and Bound
Lewis and Slotnick, 2002	Maximize Profit	Yes	No	No	No	DP, Heuristic
Proposed Research	Maximize Profit	Yes	Yes	Yes	Yes	Branch and Price

3. PROBLEM DEFINITION

3.1. Mathematical formulation

The MTO operation 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 assumed to be 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.

The decision variables used in the model are:

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

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

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

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

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

subject to

$$\sum_{j\in J}\sum_{o\in O_j} X_{jorts} \leq b_{rts} \qquad \forall r\in R, t\in T, s\in S \quad \textbf{(2)}$$

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

$$\sum_{o\in O_j}\sum_{r\in R} X_{jorts} \leq l_{ts} \qquad \forall j\in J, t\in T, s\in S \quad \textbf{(4)}$$

$$X_{jorts} \geq \tau Y_{jorts} \qquad \forall j\in J, o\in O_j, r\in R, t\in T, s\in S \quad \textbf{(5)}$$

$$X_{jorts} \leq p_{jor} Y_{jorts} \qquad \forall j\in J, o\in O_j, r\in R, t\in T, s\in S \quad \textbf{(6)}$$

$$\sum_{r\in R} t Y_{j|O_j|rts} \leq d_j U_j \qquad \forall j\in J, t\in T, s\in S \quad \textbf{(7)}$$

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

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

$$X_{jorts} \geq 0 \qquad \forall j\in J, o\in O_j, r\in R, t\in T, s\in S \quad \textbf{(10)}$$

$$Y_{jorts} \in \{0,1\} \qquad \forall j\in J, o\in O_j, r\in R, t\in T, s\in S \quad \textbf{(11)}$$

$$U_j \in \{0,1\} \qquad \forall j\in J \quad \textbf{(12)}$$

Objective (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 labor or manufacturing cost. Constraint set (2) ensures that the capacity of resource r of source s in time period t is not violated. Constraint set (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 (=) in constraint (3) can be replaced with an inequality (≥). The second term in the objective will prevent allocating more resources than what is required.

Constraint set (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 (5) and (6) set the Y_{jorts} decision variables to either 1 or 0. 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 Y_{jorts} variables are used to ensure the precedence relationship. The parameter τ in constraint (5) indicates that whenever an operation is processed on a resource it should be processed for at least τ units of time. The constraint set (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. Constraint set (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 (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 (8) represents the number of hours allocated to process operation (o-1) in time period t during regular hours. The constraint set (9) ensures that operation o of job j can be processed in period t during overtime only after completing operation (o-1). Constraint sets (10) – (12) impose the non-negativity restrictions on the decision variables. In particular, the constraint sets (11) and (12) impose the binary restrictions on the decision variables Y and U.

3.2. Decision support using the proposed model

The model proposed in the previous section can help the operations manager/decision maker to determine which subset of incoming customer orders should be selected to maximize profits. It can be integrated into a decision support system which can be used to make decisions on day-to-day basis for selecting customer orders and planning for their capacity 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(s) have to be selected for strategic reasons, a corresponding subset of order(s) that will maximize the profits can also be determined. The model is also useful to reschedule the already accepted orders when new orders have to be accepted. We present an example to illustrate how the user can utilize this model.

Consider a job shop with 3 resources having a pool of three customer orders namely jobs 1, 2 and 3 at the start of time period 1. Table 2 shows the characteristics of these three customer orders. The cost of using each resource in different sources namely, regular time (RT) and overtime (OT) are given in Table 3. It is assumed that regular production time and overtime is 8 hours each. 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. The MILP model for the example problem is solved using the commercial MILP solver CPLEX to determine the optimum solution. The optimum profit is $570 when customer orders 2 and 3 are accepted and the corresponding capacity plan is shown in Figure 2(a). Now consider that at the start of time period 2 two more orders (for jobs 4 and 5) are received, which have to be delivered by time period 4. Table 4 shows the characteristics of these two new orders.

Table 2. Customer orders available at time zero

Customer Order (Job j)	Sales Price (q_j)	Due Date (d_j)	Operation (o_j)	Resource (r)	Processing Time (p_{jor})
1	\$800	2	1	1	10
			2	3	8
2	\$950	3	1	1	8
			2	2	6
			3	2	12
3	\$1560	3	1	1	10
			2	2	8
			3	3	12

Table 3. Resource cost ($/hr) for each source

Resource # (r)	Regular time (RT) cost ($/hr)	Overtime (OT) cost ($/hr)
1	40	60
2	20	30
3	30	45

Table 4. Customer order available at time one

Customer Order (Job j)	Sales Price (q_j)	Due Date (d_j)	Operation (o_j)	Resource (r)	Processing Time (p_{jor})
4	$400	4	1	2	8
			2	3	6
5	$800	4	1	1	8
			2	3	8
			3	2	6

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 at the beginning of the first time period. When the mathematical model was solved with the new information, job 5 was chosen. The capacity planned for jobs 2 and 3 in period 1 cannot be changed, however the capacity allocated can be altered for the subsequent time periods. The model revises the capacity for job 2 and job 3 in periods 2 and 3 so as to optimally process job 5. The new capacity plan is shown in Figure 2(b). When jobs 2 and 3 were initially accepted the model prescribed a profit of $200 and $370, respectively. After job 5 was accepted, the profit of job 2 was reduced to $150, but by accepting job 5 the overall profit was increased to $770.

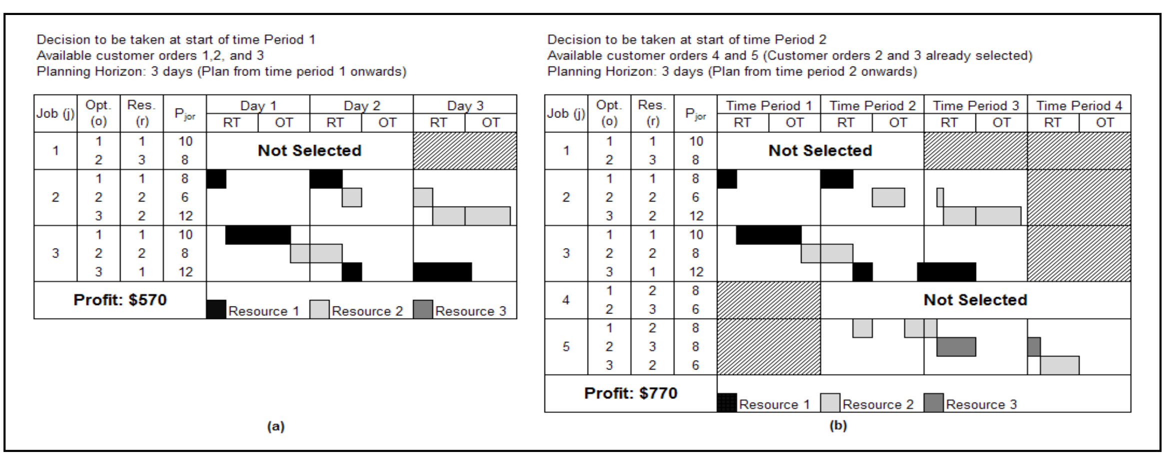


Figure 2. Capacity plan for accepted orders at start of period 1 and 2

3.3. Computational runtime analysis

The commercial solver CPLEX was used to experiment with the model proposed. 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 A) 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 (13), over the planning horizon with $|T|$ time periods. If the total demand and the available resources are known, problem instances can be generated by computing the number of time periods required for a fixed DC ratio using equation (14).

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

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

Table 5 presents the data used for Experiment A. Number of jobs and numbers of operations for each job are the two factors which are varied. 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 model to optimality was reported. Figure 3 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 operations, CPLEX was not able to find an optimal solution even after running for more than 16 hours; hence for those instances the optimality gap is reported in Figure 3.

Table 5. Data for Experiment A (computational runtime analysis)

Factors	Levels
Number of jobs	3, and 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

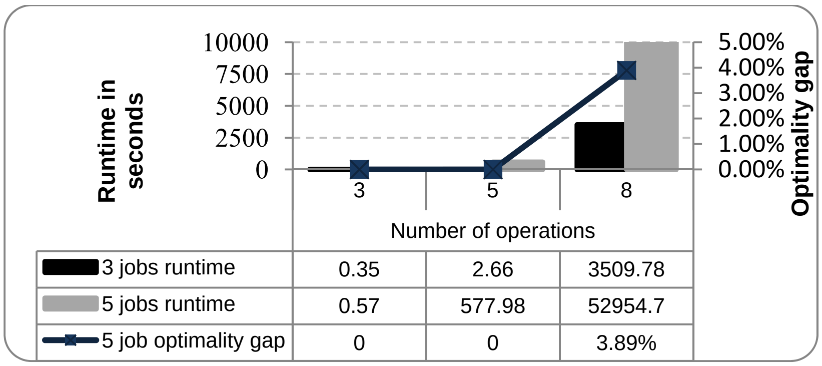


Figure 3. Computation runtime to solve MTO mathematical model to optimality

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. While negotiating with the customers (during quotation process), sales department may have to consider different scenarios before accepting an order. Many customers are sensitive to time and may require the manufacturer to respond in a timely fashion. On the other hand the manufacturer should assess the current workload and available capacity to make judicious decisions. 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. In the next section we present a B&P algorithm for solving the MILP proposed in Section 3.1.

4. BRANCH AND PRICE ALGORITHM

4.1. Model decomposition

The proposed MTO model inherits a block diagonal or angular structure as shown in Figure 4. This special structure is well suited for applying the Dantzig-Wolfe decomposition principle. In Dantzig-Wolfe decomposition, the original formulation is decomposed into a master problem and one or more sub-problems. Instead of enumerating all the variables (columns) in the master problem, columns which improve the objective are generated as needed by solving the sub problem(s). The sub problem (or pricing problem) and the master problem (or restricted master problem) is solved iteratively until no columns can be generated. When the master problem is solved the integer restrictions on the variables are typically relaxed. Consequently, when no improving column can be generated a branch and bound search procedure is implemented to fix non-integer decision variables. At each node of the branch and bound search tree the column generation procedure is applied. This entire process is referred to as branch-and-price in the literature. For a detailed discussion on B&P, we refer the reader to Wilhelm (2001).

The capacity constraint (2) is the binding or complicating constraint in our formulation. 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 restricted master problem (RMP).

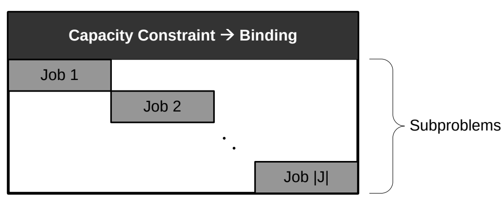


Figure 4. Decomposition of the MTO model (Block-Diagonal Structure)

The MTO restricted master problem is formulated as follows:

$$\text{Maximize } Z_{RMP}^{LP} = \sum_{j \in J} q_j U_j - \sum_{j \in J} \sum_{o \in O_j} \sum_{r \in R} \sum_{t \in T} \sum_{s \in S} \sum_{k \in K_j} (c_{rs} x_{jorts}^{k}) \lambda_j^k \qquad \textbf{(15)}$$

Subject to

$$\sum_{j \in J} \sum_{o \in O_j} \sum_{k \in K_j} x_{jorts}^{k} \lambda_j^k \le b_{rts} \qquad \forall r \in R, t \in T, s \in S \qquad \textbf{(16)}$$

$$\sum_{k \in K_j} \lambda_j^k = U_j \qquad \forall j \in J \qquad \textbf{(17)}$$

$$\lambda_j^k \ge 0 \text{ binary} \qquad \forall j \in J, k \in K_j \qquad \textbf{(18)}$$

$$U_j \ge 0 \text{ binary} \qquad \forall j \in J \qquad \textbf{(19)}$$

Where, K_j is the set of columns generated from solving the sub-problem for job j. A column is a feasible schedule for the corresponding job. An initial feasible solution to the RMP is determined by a greedy heuristic presented in Section 4.3.

A feasible schedule for job j should satisfy the processing time constraint (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 (7) and the precedence constraints (8) and (9). The corresponding formulation for the sub-problem or pricing problem of job j will consist of the constraint set (3) to (11) with an objective of minimizing the total manufacturing cost. The objective function for the pricing problem is formulated as,

$$\text{Minimize } Z_{sp}^{j} = \sum_{o \in O_j} \sum_{r \in R} \sum_{t \in T} \sum_{s \in S} (c_{rs} + w_{rts}) x_{orts} + \alpha_j \quad \textbf{(20)}$$

Where, w_{rts} and α is the dual variables of constraints (16) and (17), respectively. Ideally, the solution approach for solving the sub-problem should be fast as it has to be solved many times during the B&P procedure. In B&P the sub-problems need not be solved to optimality, a heuristic can be used to generate improving columns. Upon further analyzing the structure of each sub-problem, a network flow representation is identified and exploited to solve the sub-problems efficiently. The construction of the network and the solution approach to solve the network flow problem to obtain feasible schedule for each job is presented in the next section.

4.2. Exact procedure for solving the sub-problem

The sub-problem for job j is represented 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 $\mathrm{SA}_{e,h}\{o\in O_j, i\in I_o^j, r\in R, t\in T, s\in S\}$. The set of arcs consists of two distinct types, set of idle arcs $\{\mathrm{IArcs}\subseteq A^j\}$ and set of processing arcs $\{\mathrm{PArcs}\subseteq A^j\}$. 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 5 shows a general DAG representation of the sub-problem.

It is apparent from Figure 5 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. 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. Suppose we discretize time in units of one hour, the corresponding graph for the sub-problem is shown in Figure 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. This logic can be extended to all the split operations and time instances $\left\{e \in E^j,\ h \in H^j\right\}$ to eliminate the unwanted nodes.

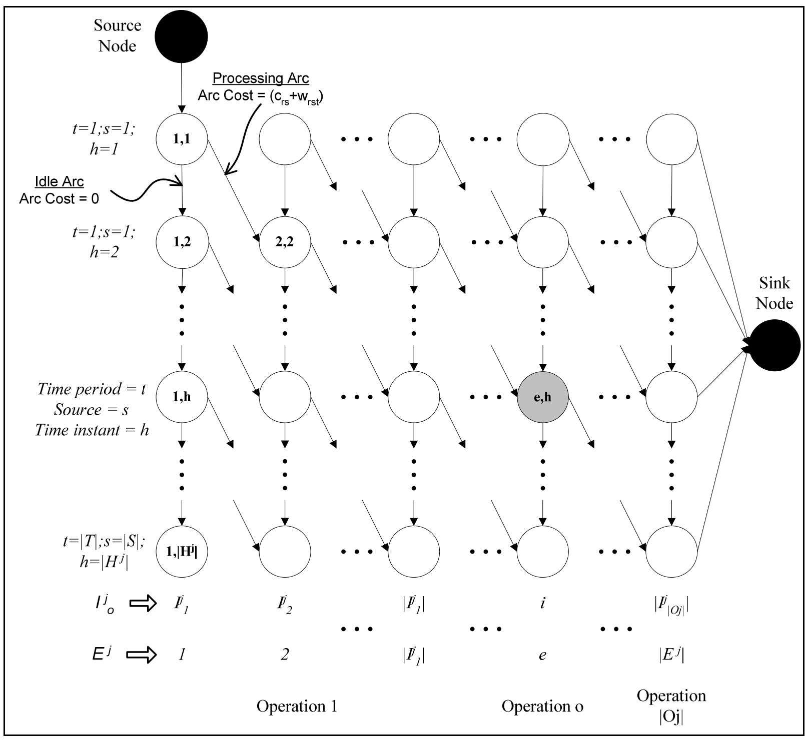


Figure 5. General Directed Acyclic Graph (DAG) representation of the sub-problem

Figure 6 shows a feasible path from the source node to the sink node. The nodes visited in the 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, operation 1 is processed for four hours. This information is used to determine the X variables (i.e. X_{j1r11} = 4) for each column.

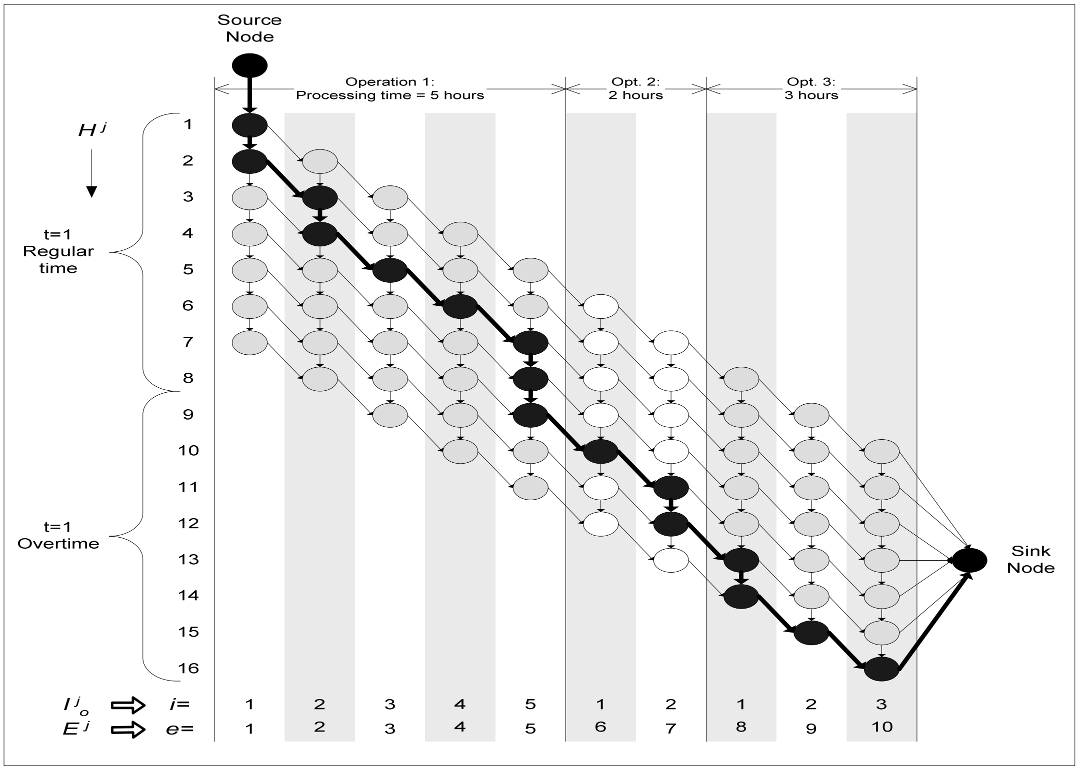


Figure 6. Illustration of DAG for sub-problem representation

The pseudo code for finding the shortest path for acyclic digraph (Rardin, 1998) and extracting the schedule from the shortest path solution are given below:

Algorithm for finding the shortest path:

BEGIN
$v[N\{1,1\}] \leftarrow 0$
$optPathTo[N\{1,1\}] \leftarrow source\ node$
$For\ N\{e',h'\} \in OperationTimeNodes\ \{e \in E^j, h \in H^j\} \backslash N\{1,1\}$
  $If\ N\{e',h'\}\ exists\ then$
    $For\ N\{e,h\} \in \{N\{e',h'\text{-}1\}, N\{e'\text{-}1,h'1\text{-}\}\}$
      $v[N\{e',h'\}] \leftarrow min\ \{v[N\{e,h\}] + C_{eh}^{e'h'} : (A_{eh}^{e'h'}\ exists\ )\}$
      $optPathTo[N\{e',h'\}] \leftarrow node\ N\{e,h\}\ achieving\ the\ minimum\ cost$
    $End\ for$
  $End\ if$
$End\ for$
$Let\ v[sink\ node] \leftarrow min\ \{v[N\{|E^j|,h\}] + C_{eh}^{sink} : (A_{|E^j|h}^{sink} exists, \forall h \in H^j\ )\}$
$Let\ optPathTo[sink\ node] \leftarrow node\ N\{e,h\}\ achieving\ the\ minimum\ cost$
END

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

BEGIN

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

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} ≠ source node)

    Let N{e',h'}←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}←N{e',h'}

End while

END

4.3. Greedy heuristic for initial solution to RMP

A greedy heuristic is proposed to obtain the initial basic feasible solution to the RMP. 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. To determine the schedule, the sub-problem solution approach described in Section 4.2 is followed. If the schedule determined improves the objective function value (i.e. the total profit) then the job is accepted and the residual capacities for the resources are updated, else the job is rejected. 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.

4.4. Branching

4.4.1. Definition of an integer feasible solution to RMP

We formally define a feasible integer solution to the RMP.

Definition 1: Consider a set of columns $k \in K_j$ for job *j* represented by the basic variables λ_j^k in the RMP, such that $\sum_{k\in K_j}\lambda_j^k=1$, which implies $U_j=1$ (from constraint (17)). The convex combination $\theta_j = \sum_{k \in K_j} x^k_{jorts} \lambda^k_j$ is a feasible integer solution to the RMP for job j if for any pair of operations $(o_i, o_{i+1})\{\forall i=1,\ldots,|O_j|-1\}$ in θ_j, there is no precedence violation.

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, λ^1_j=0.45 and λ^2_j=0.55. For an intuitive representation of a schedule a matrix notation is followed, where the rows denote the time period t and source s while the columns denote the operations. Suppose the schedules corresponding to the basic variables are as shown below. Then the convex combination θ_j is as shown below.

$$\lambda^1_j \Rightarrow x^1_{jorts} \Rightarrow \begin{array}{cl} \begin{bmatrix} 4 & 0 & 0 \\ 0 & 0 & 0 \\ 2 & 6 & 0 \\ 0 & 0 & 0 \\ 0 & 4 & 4 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \begin{array}{l} t=1,s=1 \\ t=1,s=2 \\ t=2,\ s=1 \\ t=2,s=2 \\ \\ \vdots \\ \\ t=4,\ s=2 \end{array} \end{array} \qquad \lambda^2_j \Rightarrow x^2_{jorts} \Rightarrow \begin{bmatrix} 0 & 0 & 0 \\ 6 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 2 & 2 \\ 0 & 0 & 2 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \theta_j = \sum_{k=1}^{2} x^k_{jokrt} \lambda^k_j \Rightarrow \begin{bmatrix} 1.8 & 0.0 & 0.0 \\ 3.3 & 0.0 & 0.0 \\ 0.9 & 2.7 & 0.0 \\ 0.0 & 4.4 & 0.0 \\ 0.0 & 2.9 & 2.9 \\ 0.0 & 0.0 & 1.1 \\ 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \end{bmatrix}$$

In θ_j, none of the adjacent operation pairs have a precedence violation; the processing time constraint (3), physical constraint (4) and due-date constraint (7) are satisfied and hence θ_j is an integer feasible solution to RMP for job j. Now consider another basic column λ^3_j with a corresponding schedule given by x^3_{jorts} and the new solution to RMP is λ^1_j=0.15, λ^2_j=0.35, and λ^3_j=0.5. Then the convex combination θ'_j is given as,

$$\lambda^3_j \Rightarrow x^3_{jorts} \Rightarrow \begin{bmatrix} 6 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 4 & 4 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \theta'_j = \sum_{k=1}^{3} x^k_{jokrt} \lambda^k_j \Rightarrow \begin{bmatrix} 3.6 & 0.0 & 0.0 \\ 2.1 & 3.0 & 0.0 \\ 0.3 & 2.9 & 2.0 \\ 0.0 & 2.8 & 0.0 \\ 0.0 & 1.3 & 1.3 \\ 0.0 & 0.0 & 0.7 \\ 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \end{bmatrix}$$

In θ'_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 θ'_j, is an infeasible integer solution to the RMP for job j.

4.4.2. Branch-and-Price Strategy 1 (BPS1)

In the original formulation there are 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 λ in the RMP. From 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 precedence constraints are satisfied. Hence at any node in the branch and bound tree if U_j is 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 all U_j's are either 0 or 1, and their corresponding θ_j is integer feasible as per Definition 1, then an integer feasible solution for the original formulation can be reported.

In the case that U_j is binary, but θ_j is integer infeasible, then the original variable Y has to be fixed. Fixing Y variables is not as straight forward as fixing U, since they are not used in the RMP. In the original formulation Y is an indicator variable used for ensuring that the linear precedence amongst operations is maintained. Whenever θ_j is fractional all the constraints except the precedence constraint of a job are satisfied. In such a case a branching strategy, which attempts to balance the solution space on either nodes, is proposed to restore the precedence constraints.

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 ε_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 (21) computes the precedence error between the violating adjacent pair of operations.

$$\varepsilon_p^o = 2\left(et_o - st_{o+1}\right) + \left(es_o - ss_{o+1}\right) \quad \textbf{(21)}$$

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 θ_j' be an integer infeasible solution at node *n* in the branch and bound tree and all U_j be binary. We try to restore the precedence amongst the violating operation pair with maximum precedence error first. Let this pair be denoted by (*o*,*o*+1). Consider the illustration in Section 4.4.1, with schedule θ_j' being an integer infeasible solution at some node *n* in the branch and bound tree and all U_j are binary. Figure 7 shows the schedule generated from this convex combination. Operation pair (2,3) has the maximum precedence error (ε_p^2) of 2 units. Hence we select this pair to restore precedence feasibility.

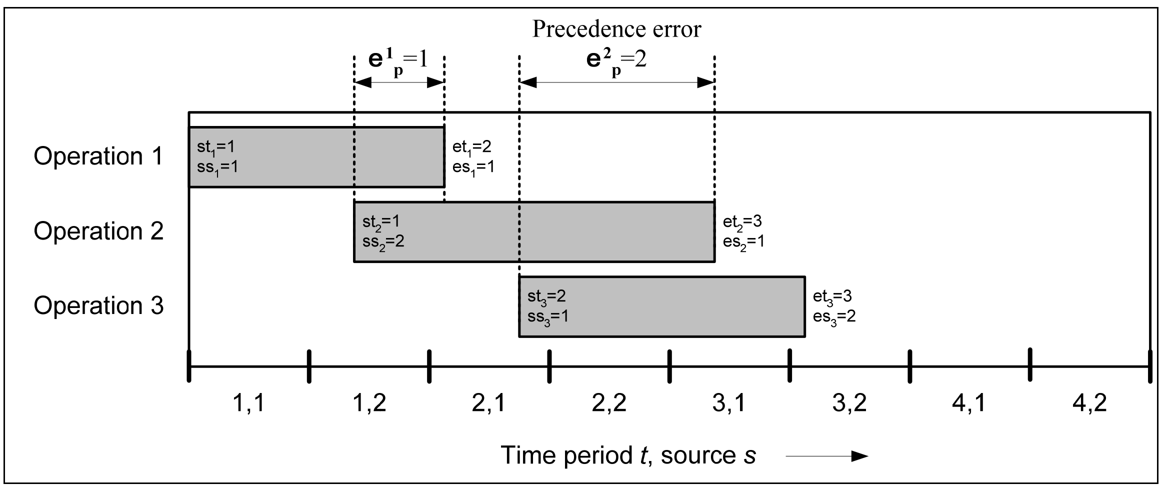


Figure 7. Schedule obtained from the convex combination

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 θ_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 θ_j'. There are no other restrictions on scheduling either these operations or other operations.

For the example under consideration, 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 operation 3 is not to be scheduled during the regular time (s=1) of the second time period. Figure 8 shows these restrictions on each node using a black colored box.

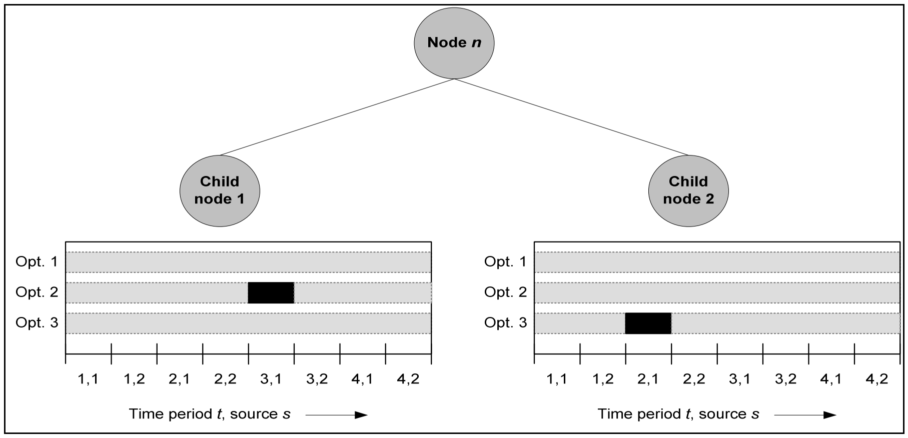


Figure 8. Branching strategy 1 (BPS1)

4.4.3. Branch and Price Strategy 2 (BPS2)

BPS1 guarantees an optimal solution. However, it takes a long time to prove optimality (see section 5.0). Since one variable is fixed at a time in BPS1, the branch and bound tree can grow exponentially. To overcome this problem a B&P heuristic (BPS2) is presented in this section.

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 θ_j', the precedence violation can be viewed 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 θ_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. Hans (2001) has used a similar approach to repair precedence violations of the fractional solution by creating three child nodes. Figure 9 shows the branching strategy using BPS2. We disallow 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 a large value.

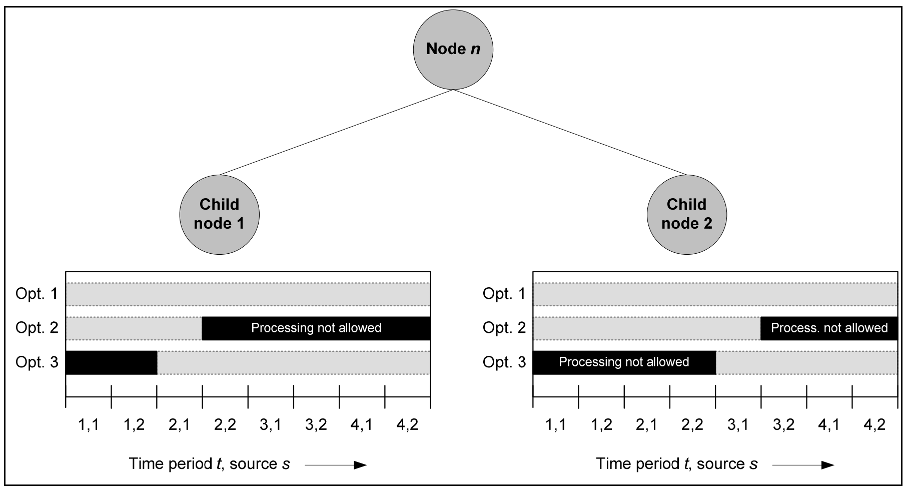


Figure 9. Branching in BPS2

4.4.4. Lagrangian bounds

Column generation process carries out many iterations with very small improvements in 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 at a node in the branch and bound tree is less than the best known integer solution value then the column generation procedure can be terminated and the node can be fathomed without the risk of missing the optimum.

Lasdon (1970) 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 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:

$$\boldsymbol{qU - cx\lambda - wb = qU - cx\lambda - wx\lambda - \alpha(\lambda - U)} \quad \textbf{(22)}$$

$$\boldsymbol{qU - cx\lambda - wb = qU - \lambda[(c + w)x + \alpha\ ] + \alpha U} \quad \textbf{(23)}$$

From Equation (20) 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 $min\left(\sum_j\left(min\ Z_{SP}^j\right), 0\right)$.

$$\boldsymbol{qU - cx\lambda -\ wb \leq qU -}\ \sum_{j\in J}\sum_{k\in K_j}\lambda_j^k\left[min\left(\sum_j\left(min\ Z_{SP}^j\right),0\right)\right] + \boldsymbol{\alpha U} \quad \textbf{(24)}$$

Rearranging the terms in Equation (24), we get,

$$\boldsymbol{qU - cx\lambda \leq qU + wb + \alpha U} - \sum_{j\in J}\sum_{k\in K_j}\lambda_j^k\left[min\left(\sum_j\left(min\ Z_{SP}^j\right),0\right)\right]. \quad \textbf{(25)}$$

The dual objective function value of the RMP is given by $\boldsymbol{qU + wb + \alpha U}$, which is equal to the objective function value of the primal RMP at optimality. We can re-write equation (25) as

$$\boldsymbol{qU - cx\lambda} \leq Z_{RMP}^{LP} \text{-}\ \sum_{j\in J}\sum_{k\in K_j}\lambda_j^k\left[min\left(\sum_j\left(min\ Z_{SP}^j\right),0\right)\right]. \quad \textbf{(26)}$$

4.4.5. Node Selection

Three strategies are implemented for node selection during the branch and bound search process, namely, Depth first Search (DFS), Best First Search (BeFS) and a combination of depth first and best first strategy (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 again apply DFS. Preliminary experimentation showed that

DFS+BeFS strategy performs the best, and hence we use this node selection strategy for further experimentation.

4.4.6. Approximation Algorithms for Branch & Price

The bounds from the decomposition algorithms are generally tighter when compared to the linear programming relaxations of the original formulation and typically feasible solutions are determined early on in the process. Truncated tree search algorithms may provide very good approximations - in truncated tree search algorithms the number of nodes evaluated in the solution process is reduced according to some pre-specified scheme (Savelsbergh, 1997). In the approximation algorithm, which we propose, we introduce a optimality tolerance γ, 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.

5. EXPERIMENTATION AND RESULTS

The branch and price algorithm was implemented in C++ using IBM/ILOG Concert technology (CPLEX 10.1) for solving the RMP. The experiment was conducted on a Intel Core 2 CPU 6330 @ 1.86 having 0.97 GB of RAM running Microsoft Windows XP professional system. The results of B&P were compared to benchmark problems solved using CPLEX 10.1. For many large instances – especially with 8 and 10 jobs, even after running for several hours did not converge to optimum. Consequently, the run time of CPLEX was restricted to 1800 sec and the best integer solution reported was used for comparison purposes.

The complete experimental setup to assess the solution quality of the B&P algorithm is presented in Table 6. A full factorial experiment was conducted. For each factor and level combination three instances were generated. 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-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 (27).

Table 6. Factors and Levels for Experiment

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]

$$q_j = \sum_{o \in O_J} p_{jor} c_{r,s=1} * (1 + (0.7 * U[0,1])) \quad \textbf{(27)}$$

We implemented the approximation algorithms for both BPS1 and BPS2 with γ value of 0.01 and 0.05. For γ 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 brackets. For example, BPS2(0.01) represents branching strategy BPS2 with an optimality tolerance $\gamma = 0.01$. This convention is followed throughout this section.

Table 7 presents the percentage improvement in solution obtained by various B&P strategies over CPLEX. CPLEX performs marginally better than the B&P strategies for instances with 3 jobs. The solution quality of B&P increases as the number of jobs increase. For 8 job and 10 job instances the improvements were at least 35% and 196% respectively.

Table 7. Improvement over CPLEX

		% improvement over CPLEX					
Jobs	Opts.	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

Table 8 presents the computation runtime to solve the various problem instances by the different B&P strategies. BPS1(0.0) takes the most time to solve the problems. When the number of operations are less (i.e. 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 approximate B&P strategies. The results are shown in Table 9. 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.

Table 8. 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

Finally we present the maximum number of columns generated (refer to Table 10), and maximum branch and bound nodes generated (refer to Table 11) 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. We do not see a clear pattern as regards to the columns generated and the problem size. But for high number of jobs and operations per job, the number of columns generated by BPS2 is more than BPS1. The maximum number of columns generated were for a 5 job 8 operation problem by BPS2(0.01).

Table 9. Reduction in runtime

		%age reduction in runtime				
Jobs	Opts.	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

The number of nodes in the branch and bound tree are lesser in BPS2 as compared to BPS1, which makes intuitive sense because in BPS2, a set of original variables are fixed to zero based on the concept of time windows, as compared to BPS1, where only one original variable is fixed at a time. 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 and bound is more and hence the increase in computational time.

Table 10. 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

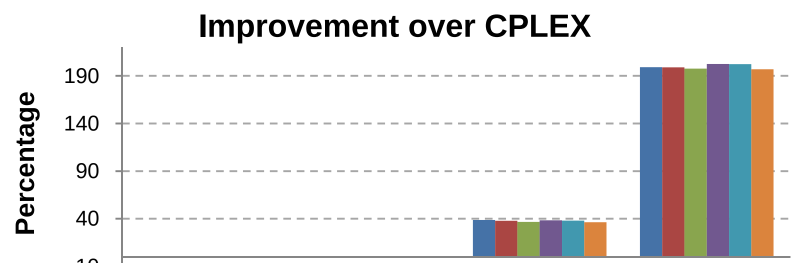
Table 11. Size of branch and bound tree in B&P

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

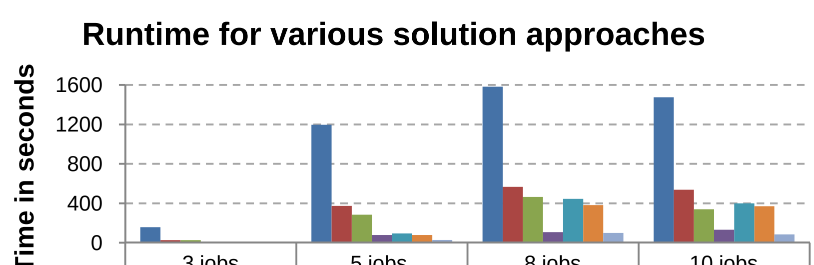
Integrating order acceptance and capacity planning provides tremendous opportunities to maximize the operational profits of make-to-order operations. This is done by selectively accepting jobs from the available pool of customer orders and simultaneously planning for their capacity. This integrated problem 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 it can be extended by considering 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 paper we propose a Mixed-Integer Linear Program (MILP) to model MTO 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. Using the model we illustrated that integrating the two decisions of order acceptance and capacity planning can achieve our goal to maximize the operational profits. Typically order acceptance problems are solved on a 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 frequently 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 and prices while better aligning with the firm's long-term business strategy. To efficiently solve this model we propose an exact branch and price algorithm (BPS1). We present Lagrangian bounds for fathoming the nodes in the branch and bound tree. We further improve the runtime of the solution approach by developing an approximate branching scheme (BPS2). We combine BPS1 and BPS2 with various approximation algorithms for truncating the branch and bound tree.

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 are typically found in real-life applications. Figures 10 and 11 graphically summarize the improvements made by B&P algorithms and the computational runtime of various solution approaches discussed in this paper. 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. BPS2(0.05) can solve, on an average, 10 jobs problems in 85 seconds and making 196% improvements over CPLEX. Thus B&P algorithms are faster and solve problems in reasonable time, and they can be utilized in a decision support system 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 10. 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 11. Runtime for various solution approaches

The problem under consideration is a basic problem found in many industrial settings. The model and solution proposed lay the foundation for other more complex problems of practical interest having variations to this basic one. For example, we consider non-regular capacity as overtime, but some make-to-order operations consider outsourcing options extensively. Integrating outsourcing is another important variation to the problem we have considered. Instead of having orders with a simple deliverable, many MTO operations handle orders that consist of product assemblies made of smaller sub-assemblies. Hence, their precedence relations are non-linear though each sub-assembly may still have linear precedence amongst its own sub-operations or tasks.

REFERENCES

Barut M. and Sridharan V. Design and Evaluation of a Dynamic Capacity Apportionment Procedure. European Journal of Operations Research 2004;155(1); 112-133.

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

Ebben M. J. R., Hans E. W. and Weghuis O. F. M. Workload Based Order Acceptance in Job Shop Environments. OR Spectrum 2005;27; 107-122.

Gallien J., Tallec Y. L. and Schoenmeyr T. A Model for Make-To-Order Revenue Management. MIT Sloan School of Management, Working Paper, November 2004.

Ghosh J. B. Job Selection in a Heavily Loaded Shop. Computers Ops. Res. 1997; 24(2); 141-145.

Hans E. Resource Loading by Branch-and-Price Techniques. University of Twente, Ph.D. Thesis, 2001;

Harris F. H., deB. and Pinder J. P. A Revenue Management Approach to Demand Management and Order Booking in Assemble-to-Order Manufacturing. Journal of Operations Management 1995; 13; 299-309.

Jalora A. Order Acceptance and Scheduling at a Make-To-Order System using Revenue Management. Texas A&M University, Dissertation, August 2006;

Lasdon L. S. Optimization Theory for Large Systems. Macmillan Publishing Co., Inc.: New York;

Mehmet B. and Sridharan V. Revenue Management in Order-Driven Production Systems. Decision Sciences 2005; 36(2); 287-316.

Rom W. O. and Slotnick S. A. Order Acceptance using Genetic Algorithms. Computers & Operations Research 2009; 36(6); 1758-1767.

Slotnick S. A. and Morton T. E. Order Acceptance with Weighted Tardiness. Computers & Operations Research 2007; 34; 3029-3042.

Slotnick S. A. and Morton T. E. Selecting Jobs for Heavily Loaded Shop with Lateness Penalties. Computers & Operations Research 1996; 23(2); 131-140.

Streitfeld D. Amazon pays a price for marketing test. Los Angeles Times; 2000; C1

Van den Akker J. M., Hoogeveen H. and van de Velde S. Parallel Machine Scheduling by Column Generation. Operations Research 1999; 47(6); 862-872.

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

Van den Akker J. M., Hurkens C. A. J. and Savelsbergh M. W. P. A Time-Indexed Formulation for Single-Machine Scheduling Problems: Column Generation. INFORMS Journal of computing 2000;12(2); 111-124.

Van den Akker J. M., Van Hoesel C. P. M. and Savelsbergh M. W. P. A Polyhedral Approach to Single-Machine Scheduling Problems. Mathematical Programming 1999; 85(3); 541-572.

Wilhelm W. E. A Technical Review of Column Generation in Integer Programming. Optimization and Engineering 2001; 2; 159-200.

Zijm W. H. M. Towards Intelligent Manufacturing. OR Spektrum 2000; 22; 313-345.