ABSTRACT

BAKIR, ILKE. Linear Programming Formulations for Production Planning Problems with Alternative Routings. (Under the direction of Dr. Reha Uzsoy).

Linear programming (LP) is a common tool for solving production planning problems. However, LP formulations tend to get very large, when used for modeling large and complex production systems, such as production systems with alternative routings. In this thesis, we define production planning problems with alternative routings, investigate different modeling approaches to such problems, and propose a solution method.

We first present the "Path Based Formulation", which grows rapidly in terms of number of variables. Then we propose a Column Generation formulation to solve the path-based problem more efficiently, without generating all variables corresponding to the alternative routings. We also present some compact LP formulations, which are used for modeling production planning problems with alternative routings, from existing literature; namely "Direct Product Mix Formulation" of Leachman and Carmon (1992) and "Capacity Partition Approach" of Hung and Cheng (2001).

The results of our computational experiments, performed on randomly generated data, show that our column generation approach has potential in solving very large problems that cannot be solved by a direct LP approach in a reasonable amount of computational time.

Linear Programming Formulations for Production Planning Problems with Alternative Routings

by
İlke Bakır

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Industrial Engineering

Raleigh, North Carolina

2011

APPROVED BY:

_____	_____
Dr. Reha Uzsoy Committee Chair	Dr. Thom J. Hodgson
_____	
Dr. Brian Denton	

DEDICATION

To my family, who made me the person I am today…

BIOGRAPHY

İlke Bakır was born in Ankara, Turkey on September 13th, 1987. In July 2009, she received her Bachelor of Science degree in Industrial Engineering from Boğaziçi University in İstanbul, Turkey. She has been a graduate student at North Carolina State University starting from August 2009 to the present time, and has received teaching and research assistantships during her studies. She is expected to receive her Master of Science degree in Industrial Engineering in 2011. Bakir is currently a student in Doctorate of Philosophy in Industiral Engineering program at Georgia Institute of Technology, in Atlanta, GA.

ACKNOWLEDGMENTS

I would like to sincerely thank my advisor, Dr. Reha Uzsoy, a million times for making this thesis possible by guiding me through my research, constantly supporting me in all the problems I face, and patiently tolerating my reactions to tough situations, especially towards the end. He has been not only an academic advisor, but also a mentor and a counselor to me. I also would like to thank the members of my advisory committee, Dr. Thom J. Hodgson and Dr. Brian Denton for taking the time to serve in my committee, review and critique my thesis.

I am grateful to the faculty and staff of Edward P. Fitts Department of Industrial Engineering for giving me the opportunity to initiate my graduate studies, providing me with an academic "home" for my research activities, and give me the knowledge basis necessary to continue my studies.

I would also like to express my appreciation to my family in Turkey and my friends who supported me during my studies. Without the unconditional love and support of my family, I would have never been able to overcome the constant pressure of graduate studies. Lastly, I thank my friends (especially Müge, who have been my family in Raleigh and supported me as a full-time job) for always being there for me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. Introduction

Linear programming (LP) is widely used for solving a wide range of production planning problems. However, LP formulations can get very large, especially with large and complex production systems such as those encountered in semiconductor manufacturing; hence the challenge has always been finding the most compact model possible while still achieving the desired level of accuracy. In particular, production planning problems where products can be produced using different combinations of flexible resources have proven particularly difficult to model.

There are many different production planning models available in the literature; both for deterministic and stochastic demand structures (Johnson and Montgomery, 1974). This thesis focuses on deterministic production planning problems with alternative machine types. The production planning models considered in this research aim at determining an optimal, capacity-feasible product mix for each planning period over some planning horizon in a production facility with following characteristics:

- Multiple production stages
- Alternative machine types, i.e. machine types that are functionally different but capable of performing the same manufacturing operation, at each stage, perhaps at different quality levels and with different processing times
- Discrete time periods
- Single commodity

The particular production system we will consider within the scope of this thesis has multiple stages and multiple machines at each stage. All the machines at each stage are capable of performing the same operation on the product. It can thus be classified as a flexible flow shop structure. Although the models we develop can be extended to more complex production systems, such as those with reentrant product flows, we focus on the flexible flow shop environment due to the ease of controlling the number of alternative paths through the system, which simplifies the design of our computational experiments.

The common approach to solve the production planning problem for a manufacturing environment with alternative machines types has been path-based LP formulations in which the decision variables specify the quantity of a specific product type produced by a specific process at a specific period (Johnson and Montgomery, 1974). In such a formulation, the number of variables grows rapidly since the number of possible process routes, i.e., admissible

sequences of operations that can produce a product, grows exponentially with increasing number of stages. These formulations provide detailed information on the workload assigned to each machine or the proportion of jobs that use a particular machine type in each planning period. The purpose of the planning process, however, is finding a capacity-feasible job release schedule that specifies the amounts of the product to be released into the production system in each time period while satisfying the demand constraints.

Path-based formulations are mathematical models that keep track of how much work is allocated to each possible sequence of operations. These are called Path-Based formulations, since we are explicitly specifying the amount of each product that will be launched on each possible path through the production system in each period. Such a formulation is depicted in Figure 1.1, which represents a production system with four stages with a number of alternative machines at each stage, represented by the boxes. Path-Based formulations, referred to as Process Selection Models by Johnson and Montgomery (1974), have been known for quite some time, but have the obvious drawback that in a production system of any complexity, the number of possible paths through the system that a product can follow, and hence the number of decision variables in a time period, grows exponentially in the number of alternatives at each stage and the number of stages. It is a fact that Path-Based formulations can be very flexible in the sense that they can be used to model almost any production environment; but their flexibility comes at a cost of significant computational complexity resulting from the necessity to generate all possible paths in advance and assign a variable to each path for each period.

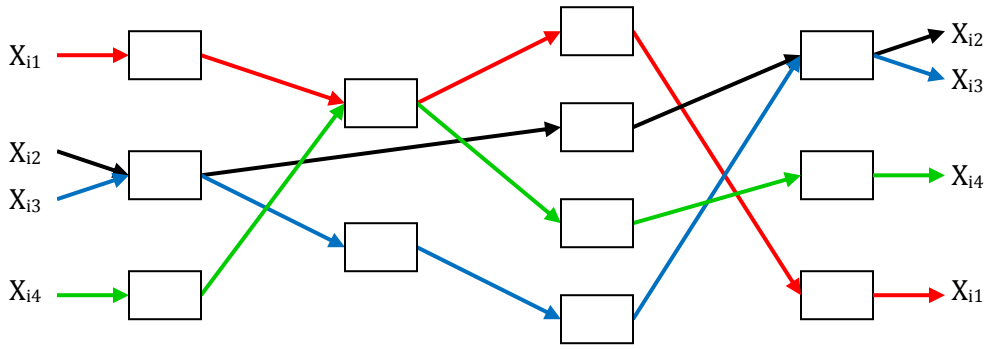

Figure 1.1: Path-Based Formulation of Alternative Resources

The concept of the path-based formulation immediately suggests an approach where paths are generated as they are needed, and do not all have to be generated a priori resulting in an exponentially large LP formulation. Column generation approaches based on Dantzig-Wolfe decomposition (Bertsimas and Tsitsiklis, 1997) and similar approaches have been used to solve

large problems in many areas. One of the early examples of the method is the method used by Ford and Fulkerson (1958) in their paper on maximal flow problems on multi-commodity networks. They state that solving such problems by Simplex Method would not be possible with available machine capacity, so they suggest performing the "pricing" operation of Simplex Method (i.e. the determination of a vector to enter the basis) implicitly. They show that finding the entering variable that would improve objective value the most is equivalent to solving a shortest chain problem. They also show that the shortest chain problem can be reduced to a standard transshipment problem, which may be solved in various simple ways.

Tomlin (1966), in his paper on minimum cost multi-commodity network flows, shows that such problems can be solved by a technique similar to the method proposed by Ford and Fulkerson (1958); i.e. determining the entering variable by solving a subproblem that obtains the column whose introduction to the basis would improve the objective value the most, while satisfying the feasibility constraints. In this case, subproblems have the form of a minimum cost flow problem in an uncapacitated network, and such problems can easily be solved using one of the efficient shortest chain algorithms (Ahuja, 1956 and Bazaara, 2010).

The column generation approach of Ford and Fulkerson (1958) has been extended and used for solving multi-commodity network flow problems by Folie and Tiffin (1976), Farvolden, Powell, and Lustig (1993), Barnhart et al. (1995), Desaulniers et al. (1997), Venkatadri et al. (2006), and Alvelos and Carvalho (2007).

The paper by Alvelos and Carvalho (2007) extends the type of work pioneered by Ford and Fulkerson (1958) by defining a new model and proposing a column generation approach to the minimal cost multi-commodity flow problem. Their extended model incorporates some additional variables, converts some simple constraints into ranged constraints, and requires an additional procedure to give an optimal solution to the original problem. Even though their extended model has more variables and more complex constraints, it has the advantage of reducing the number of iterations it takes for the column generation approach to solve the problem, and their experimentation show that the column generation approach implemented on the extended model improves the solution times greatly.

A different view that can be considered a Path-Based approach is introduced in the paper by Jain and Palekar (2005). They suggest a Configuration-Based formulation for the aggregate production planning problem in continuous processes with alternative machine types, which does not allow intermediate buffers between machines. Configuration-Based formulation incorporates the bottleneck logic into the mathematical model, as it is necessary in

a production environment with continuous processes. Jain and Palekar (2005) indicate that their Configuration-Based model, which considers process and setup times of each machine configuration, i.e. path, separately allows them to overcome the limitations of Resource-Based models, which represent the workload on machines independently from the process path being used. Even though the Configuration-Based model increases accuracy significantly, it also increases the problem size in terms of number of variables and number of constraints. The authors also present some heuristics to limit the number of variables and show that they are effective in solving real-life problems.

Along with Path-Based approaches, there are also alternative formulations for the production planning problem. A number of authors have proposed more compact formulations of the problem with reduced numbers of decision variables and constraints. Leachman and Carmon (1992) propose three alternative production planning formulations that are often, although not always, more compact than conventional path-based models that accommodate alternative machine types. These models aggregate the capacity constraints to eliminate unnecessary details, and consequently reduce the solution time significantly. In particular, these models omit the representation of work flow within the system.

Among the three formulations of Leachman and Carmon (1992), the formulation of particular interest to this thesis is the Direct Product Mix Formulation. The Direct Product Mix Formulation, unlike most production planning models, does not use any allocation variables. It accomplishes this by introducing capacity constraints for specific sets of alternative machine types, identified by an algorithm called the Capacity Set Generation Procedure. The structure of those sets depends on the machine usage patterns appearing in the problem data. The authors prove that this formulation is exact by formulating the problem in a single period as a bipartite network flow problem and examining the optimality conditions.

An important limitation of Direct Product Mix Formulation is that it is only valid under the "uniformity assumption", which requires the processing times for a given product among alternative machine types to be either identical or else proportional across all operations performed by the machines. This means that the value obtained by dividing the time to process an operation on a specific machine by the time to process that operation on a predetermined 'standard' machine is the same for all operations and equal to a constant value. A demonstration of this statement can be observed in the equation below, where a_{ijk} denotes the time to perform operation j of product i on machine k, and machine k_1 denotes the standard machine.

$$\frac{a_{i1k}}{a_{i1k_1}} = \frac{a_{i2k}}{a_{i2k_1}} = \cdots = \frac{a_{iJk}}{a_{iJk_1}} = s_k, \qquad k = 1, \ldots, K$$

Under uniformity assumption, processing times and capacities for the machine types may be scaled in terms of some "standard" machine type to achieve identical processing times and hence eliminate the machine index from variables. The uniformity assumption can be an obstacle for real-life planners, since not all production systems satisfy this assumption. An example of this situation is given by Bermon and Hood (1999), who conclude that an accurate model is worth the high model complexity and solution time because the Direct Product Mix Formulation erroneously calculated that additional capacity, which would cost millions of dollars, was needed to meet demand.

The Capacity Set Generation Procedure and the main logic of the Direct Product Mix Formulation were later extended by Hung and Wang (1997), and applied to the bin allocation planning problem (also called the alternative material allocation planning problem) in semiconductor manufacturing. This application of the set generation procedure is a bit different than the Direct Product Mix Formulation, because it addresses alternative materials instead of alternative equipment. Even though the Acceptable Material Set Generation Procedure of Hung and Wang (1997) reduces the number of constraints by approximately one-third when compared to conventional bin allocation planning models, the main limitation of Direct Product Mix Formulation, i.e. uniformity assumption, remains effective in this formulation.

The restriction caused by uniformity assumption of Direct Product Mix Formulation was later overcome by the modified formulation proposed by Hung and Cheng (2002). They relax the uniformity assumption by dividing the capacity of machines that belong to two or more machine sets into partitions and assigning each partition to its corresponding machine set. They provide an algorithm to generate the appropriate machine set partitions. Their formulation relaxes the uniformity assumption, and hence ensures model accuracy, at the expense of about 15% increase in solution time.

We will discuss the Direct Product Mix Formulation of Leachman and Carmon (1992), and its extension proposed by Hung and Cheng (2002) in detail in Chapter 2, along with our Path-Based formulation and the Column Generation approach built on it. Also, the methodology and implementation tools will be presented in the same section.

In Chapter 3, we will provide the reader with the experimental design parameters; and then we will present the experimentation results in Chapter 4. Finally, in Chapters 5 and 6, we

will discuss the conclusions we have drawn and the future directions of this research, respectively.

2. Problem Definition and Formulations

A. Problem Definition

As stated in the introduction, the main purpose of this thesis is the examination of production planning models for solving production planning problems with alternative routings. The models of particular interest are the Direct Product Mix Formulation of Leachman and Carmon (1992), the Partitioning formulation of Hung and Cheng (1992), and our Column Generation formulation, which is inspired by the approach of Ford and Fulkerson (1958), based on the Path-Based model.

This topic is of interest to us because real-life planning generally requires quick response to unexpected events at the shop floor level, and solving detailed mathematical models may be computationally cumbersome, preventing the planners from responding quickly. Therefore, we will present several mathematical modeling approaches to production planning problem that provide the planners with a certain level of accuracy, while keeping the problem size relatively small and manageable, and allowing solution in modest CPU times. In production planning models, high levels of detail and accuracy come at a cost of computational complexity. The converse statement is also true most of the time: Small formulations that provide quick results come at a cost of loss of generality, which imposes the necessity to meet certain assumptions, rendering them valid for only certain types of production systems.

The analysis will focus mainly on comparing the performances of the aforementioned models based on both experimentation and information provided by the authors who proposed the models in their papers.

The models of Leachman and Carmon (1992) and Hung and Cheng (2002) are presented in detail in the upcoming section. Their sizes in terms of number of variables and constraints will be studied; along with best and worst case solution time performances and their limitations.

We depart from the conjecture that in many production systems with alternative routings, some paths will have clear advantage (low cost, short processing times, and/or high revenue) against other paths; hence the mathematical model will tend to assign nonnegative release quantities to those paths only. In this case, only a small proportion of all possible paths will actually be used in an optimal solution. This implies that most of the columns in the Path-Based formulation are unnecessary, and we may not need to incorporate these unnecessary

columns into the mathematical model. The Column Generation formulation is developed to generate only the paths that will be used as they are needed during the solution process. In the upcoming chapters, we will experiment with production systems of different sizes and characteristics to observe what proportion of the possible columns are actually being used for production. We will also compare the size of the Column Generation formulation to those of the full Path-Based formulation and the models of Leachman and Carmon (1992) and Hung and Cheng (2002).

B. Path-Based Formulation

The Path-Based approach is the most detailed and accurate formulation among all the approaches that will be presented here, and requires fewer assumptions. The only assumptions that are required to hold is that total cost of a path must be the sum of costs at each stage, and costs at each stage must be independent from the previous stages.

The accuracy of this formulation comes at a cost of computational complexity. This formulation requires the planner to generate all possible paths for each product through the production system and assign a decision variable to each path in each time period, specifying the number of products to be released on that path in that period. All the costs and revenues associated with that particular path must also be computed a priori. Naturally, this increases the problem size and computational complexity of this formulation, resulting in the size of the formulation growing exponentially with the number of alternative machines.

The main decision variables of the Path-Based formulation represent the number of job releases onto each path in each time period. There are also variables representing the backorder quantities in each time period. The path-based formulation consists of capacity and inventory balance constraints, and an objective function that aims at maximizing the net profit. The notation and the model formulation are presented below.

Indices:

i : Product type $i = 1, \dots, I$

j : Path (process route) $j = 1, \dots, J$

k : Machine $k = 1, \dots, K$

t : Time periods $t = 1, \dots, T$

Parameters:

L_{ij}: Overall lead time (from start to finish) of product i when it is produced using path j

L_{ijk}: Lead time up to initiation of the operation at machine k when product i is produced using path j

$a^i_{jk\tau t}$: Time t capacity consumption at machine k by one unit of product i released at time τ to path in period t

C_{kt} : Available capacity (machine hours) at machine k in time period t

D_{it} : Demand for product i in period t

I_{i0} : Initial inventory of product i at the beginning of the planning horizon

p_{it} : The revenue obtained by producing and selling one unit of product i in time period t

c^i_{jt} : The production cost of one unit of product i released to path j in period t

h_{it} : The cost of holding one unit of product i in time period t

b_{it} : The backorder cost of product i in time period t

Decision variables:

X^i_{jt} : Amount of product i to be produced in period t using path j

I_{it} : The inventory of product i at the end of period t

B_{it} : The backorder quantity of product i at the end of period t

The LP formulation can now be stated as follows:

$$max \sum_{i=1}^{I}\sum_{t=1}^{T}\left[\sum_{j=1}^{J}(p_{it}-c^i_{jt})X^i_{jt} - h_{it}I_{it} - b_{it}B_{it}\right]$$

$s.t.$

$$\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{\tau=1}^{t-L_{ijk}} a^i_{jk\tau t}X^i_{jt} \le C_{kt} \qquad k=1,\dots,K \qquad t=1,\dots,T$$

$$I_{i0} + \sum_{j=1}^{J}\sum_{\tau=1}^{t-L_{ij}} X^i_{j\tau} - B_{i,t-1} + B_{it} - \sum_{\tau=1}^{t} D_{i\tau} \ge 0 \qquad i=1,\dots,I \qquad t=1,\dots T$$

$$X^i_{jt} \ge 0 \qquad i=1,\dots,I \qquad j=1,\dots,J \qquad t=1,\dots,T$$

$$I_{it}, B_{it} \ge 0 \qquad i=1,\dots,I \qquad t=1,\dots,T$$

The objective function is to minimize net profit, minus inventory holding and backorder costs, which is written as

$$z = \sum_{i=1}^{I} \sum_{t=1}^{T} \left[\sum_{j=1}^{J} (p_{it} - c_{jt}^{i}) X_{jt}^{i} - h_{it} I_{it} - b_{it} B_{it} \right]$$

Since $I_{it} = I_{i0} + \sum_{j=1}^{J} \sum_{\tau=1}^{t-L_{ij}} X_{j\tau}^{i} - \sum_{\tau=1}^{t} D_{i\tau}$ for all time periods t, upon substitution the objective function becomes

$$z = \sum_{i=1}^{I} \sum_{t=1}^{T} \left[\sum_{j=1}^{J} (p_{it} - c_{jt}^{i}) X_{jt}^{i} - h_{it} \left(I_{i0} + \sum_{j=1}^{J} \sum_{\tau=1}^{t-L_{ij}} X_{j\tau}^{i} - \sum_{\tau=1}^{t} D_{i\tau} \right) - b_{it} B_{it} \right]$$

In the Path-Based formulation, the objective is to maximize profit, and the constraint set consists of capacity, inventory balance, and non-negativity constraints. The first constraint is the capacity constraint, stating that the total processing time of a machine in a period cannot be greater than the available capacity of that machine. The second constraint is the inventory balance constraint, which states that the total input into the system must equal the total outcome of the system. The third and fourth constraints are the non-negativity constraints related to the production and backorder variables. Please note in this formulation that because all machines have fixed lead times, it is possible to off-line compute the time at which a particular job order will consume capacity.

The number of production variables in this formulation is JT, the number of paths times the number of time periods. The number of paths, J, is determined by number of production stages, and number of alternative machines in each stage. Let S be number of stages, and N_s be the number of machines at stage s. Then total number of paths can be calculated as $J = \prod_{s=1}^{S} N_s$. Here, we assume that all machines in a stage can perform the same operation, and it does not matter which machine performs the operation. The calculation of number of paths is valid only when this assumption is satisfied.

The number of capacity constraints is KT, the number of machines times the number of time periods, and the number of inventory balance constraints is T. Thus the total number of constraints is $(K + 1)T$.

The path-based formulation is the largest of all the formulations we present here. The column generation that will be presented in the next section is derived based on this path-based

formulation. The comparison between these two formulations in terms of solution time will be performed by designed experiments in Chapter 3.

C. Column Generation Formulation for Path-Based Model

The Path-Based formulation presented in the previous section has the advantage of being the most accurate formulation compared to the other formulations considered in this thesis. However, for large problems the Path-Based formulation may not be solved in a short time, since the number of paths increases exponentially as number of stages increases. Therefore, we seek an approach that has the accuracy level of the Path-Based Formulation, while still having a manageable problem size. Since we hypothesize that only a small fraction of production paths are actually assigned nonzero production in the optimal solution of the Path-Based formulation, introducing the most profitable production paths into the mathematical model one-by-one can improve solution time. Therefore, we propose here a column generation approach to the Path-Based Formulation.

The purpose of our Column Generation approach is to generate only the path variables corresponding to the most preferable paths, and ignoring the rest. By not incorporating the path variables for unfavorable paths into the formulation, this model prevents the number of variables from increasing exponentially. This means keeping the advantages of the Path-Based Formulation, while eliminating its main disadvantage of exponentially increasing problem size.

Column Generation is defined as an algorithm that generates the columns when simplex method needs them, instead of tabulating all the columns at the beginning (Lasdon, 1970). It does so by iteratively solving a set of independent subproblems and a restricted master problem. The subproblems receive simplex multipliers (dual variables) from the restricted master problem, and send their solutions back to the master problem, which integrates the input from the subproblems to the original restricted master problem and computes new simplex multipliers. This process continues until the optimality test is passed.

In this formulation, the term "column" is used to depict a column of the LP coefficient matrix corresponding to the production variable denoting a specific path in a specific time period. It means that each X_{jt}^{i} variable, which denotes the amount of product i to be produced in period t using path j, in the Path-Based Formulation correspond to a column in this model. From this point on, the Column Generation formulation will be presented as a single commodity

problem, even though it can easily be generalized for multiple commodity production environments. Hence, for the sake of simplicity, all the modeling aspects of our Column Generation approach will be developed for single commodity environments from this point on, and we will consider each column to represent a X_{jt} variable, instead of a X_{jt}^{i} variable. This means that each column corresponds to a (path, period) pair.

In light of this basic information, the restricted master model of our Column Generation formulation is presented below. Also, the model parameters and decision variables, which are defined differently from the Path-Based model, are presented.

Indices and Sets:

s : Column index

S : The set of all columns that were introduced into the restricted master problem

Parameters:

$\hat{p}_s$: Net profit associated with column s, calculated using all revenues and costs. (The calculation of this parameter will be discussed later in detail)

b_t : The backorder cost for one unit of product in time period t

j_s : The path associated with column s

t_s : The period associated with column s

L_s : Overall lead time (from start to finish) of a product when it is produced using path associated with column s

L_{sk} : Lead time up to initiation of the operation at machine k, when product is released from the path associated with column s in time period associated with column s

a_{kt}^{s} : The amount of machine k capacity consumed in period t by one unit of product released on the path associated with column s in time period associated with column s

Decision Variables:

X_s : Release quantity assigned to the path associated with column s in time period associated with column s

B_t : The backorder quantity at the end of period t

$$max \sum_{s \in S} \hat{p}_s X_s - \sum_{t=1}^{T} b_t B_t$$

$$s.t.$$

$$\sum_{s:\, t_s < t - L_{sk}} a_{kt}^{s} X_s \leq C_{kt} \qquad k = 1, \dots, K \qquad t = 1, \dots, T$$

$$I_0 + \sum_{s:\, t_s < t - L_s} X_s - B_{t-1} + B_t - \sum_{\tau=1}^{t} D_\tau \geq 0 \qquad t = 1, \dots T$$

$$X_s \geq 0 \qquad s \in S$$

$$B_t \geq 0 \qquad t = 1, \dots, T$$

In order to select an entering column in a column generation approach, one needs to find a column associated with a nonbasic variable that would improve the objective value, if made basic. Since the objective is to maximize net profit in our Path-Based formulation, the entering variable needs to have positive reduced cost, so that the objective value increases when that variable enters the basis. In column generation formulations, the convention is to select the column that would improve the objective function the most. Therefore, we will look for the column that has the maximum positive reduced cost.

The calculation of reduced cost, denoted as $\boldsymbol{r}$, is shown for a linear programming problem in standard form is as follows:

$$\boldsymbol{r = c - A^T \alpha}$$

The following formulation defines the standard form of LP problems:

$$min \quad \boldsymbol{c^T x}$$

$$s.t.$$

$$\boldsymbol{Ax \leq b} \qquad : \boldsymbol{\alpha \geq 0}$$

$$\boldsymbol{x \geq 0}$$

$\boldsymbol{A}$ defines the coefficient matrix, $\boldsymbol{b}$ defines the right-hand side vector, $\boldsymbol{c}$ defines the objective function cost vector, $\boldsymbol{x}$ defines the vector of decision variables, and $\boldsymbol{\alpha}$ defines the vector of dual variables corresponding to this specific LP model.

Our Path-Based formulation is in the following form:

$$
\begin{aligned}
& max \quad \boldsymbol{c}^T \boldsymbol{x} \\
& s.t. \\
& \quad \boldsymbol{Ax} \leq \boldsymbol{b} && : \boldsymbol{\alpha} \geq \boldsymbol{0} \\
& \quad \boldsymbol{Cx} \geq \boldsymbol{d} && : \boldsymbol{\gamma} \leq \boldsymbol{0} \\
& \quad \boldsymbol{x} \geq \boldsymbol{0}
\end{aligned}
$$

Therefore, the reduced cost vector can be calculated as:

$$\boldsymbol{r} = \boldsymbol{c} - \boldsymbol{A^T \alpha} - \boldsymbol{C^T \gamma}$$

To determine the reduced cost of a column, one needs the objective function coefficient, and the corresponding column of the constraint matrix. As it can be seen in the previous section, the objective function in the Path-Based Formulation is not organized to have a specific coefficient corresponding to each X_{jt} variable. Therefore, it is necessary to reorganize the terms in the objective function to bring into the following form:

$$\sum_{t=1}^{T} \sum_{j=1}^{J} \hat{p}_{jt} \, X_{jt} - \sum_{t=1}^{T} b_t B_t$$

For that purpose, the objective function of the Path-Based model is transformed as demonstrated in the following example. The example consists of one product type and three production paths and four time periods. Let the lead times of paths be $L_1 = 1, L_2 = 2, L_3 = 3$. The objective function of the Path-Based Formulation for this example can be written as:

$$z = \sum_{t=1}^{4} \left[\sum_{j=1}^{3} (p_t - c_{jt}) X_{jt} - h_t \left(I_0 + \sum_{j=1}^{3} \sum_{\tau=1}^{t-L_j} X_{j\tau} - \sum_{\tau=1}^{t} D_\tau \right) - b_t B_t \right]$$

$$= [(p_1 - c_{11})X_{11} + (p_1 - c_{21})X_{21} + (p_1 - c_{31})X_{31} - h_1(I_0 - D_1) - b_1B_1]$$
$$+ [(p_2 - c_{12})X_{12} + (p_2 - c_{22})X_{22} + (p_2 - c_{32})X_{32} - h_2(I_0 + X_{11} - D_1 - D_2)$$
$$- b_2B_2]$$
$$+ [(p_3 - c_{13})X_{13} + (p_3 - c_{23})X_{23} + (p_3 - c_{33})X_{33}$$
$$- h_3(I_0 + X_{11} + X_{12} + X_{21} - D_1 - D_2 - D_3) - b_3B_3]$$
$$+ [(p_4 - c_{14})X_{14} + (p_4 - c_{24})X_{24} + (p_4 - c_{34})X_{34}$$
$$- h_4(I_0 + X_{11} + X_{12} + X_{13} + X_{21} + X_{22} + X_{31} - D_1 - D_2 - D_3 - D_4) - b_4B_3]$$

$$= (p_1 - c_{11} - h_2 - h_3 - h_4)X_{11} + (p_2 - c_{12} - h_3 - h_4)X_{12} + (p_3 - c_{13} - h_4)X_{13} + (p_4 - c_{14})X_{14}$$
$$+ (p_1 - c_{21} - h_3 - h_4)X_{21} + (p_2 - c_{22} - h_4)X_{22} + (p_3 - c_{23})X_{23}$$
$$+ (p_4 - c_{24})X_{24} + (p_1 - c_{31} - h_4)X_{31} + (p_2 - c_{32})X_{32} + (p_3 - c_{33})X_{33}$$
$$+ (p_4 - c_{34})X_{34} + (-h_1 - h_2 - h_3 - h_4)I_0 + (h_1 + h_2 + h_3 + h_4)D_1$$
$$+ (h_2 + h_3 + h_4)D_2 + (h_3 + h_4)D_3 + (h_4)D_4 - b_1B_1 - b_2B_2 - b_3B_3 - b_4B_3$$

$$= (p_1 - c_{11} - h_2 - h_3 - h_4)X_{11} + (p_2 - c_{12} - h_3 - h_4)X_{12} + (p_3 - c_{13} - h_4)X_{13} + (p_3 - c_{13})X_{14}$$
$$+ (p_1 - c_{21} - h_3 - h_4)X_{21} + (p_2 - c_{22} - h_4)X_{22} + (p_2 - c_{23})X_{23}$$
$$+ (p_2 - c_{24})X_{24} + (p_1 - c_{31} - h_4)X_{31} + (p_2 - c_{32})X_{32} + (p_3 - c_{33})X_{33}$$
$$+ (p_4 - c_{34})X_{34} - b_1B_1 - b_2B_2 - b_3B_3 - b_4B_3 + C$$

$$z = \sum_{t=1}^{4}\sum_{j=1}^{3}\left(p_t - c_{jt} - \sum_{\tau=t+L_j}^{4} h_\tau\right)X_{jt} - \sum_{t=1}^{4} b_tB_t$$

When this condensed version of the objective function is generalized for one product type, it becomes

$$z = \sum_{t=1}^{T}\sum_{j=1}^{J}\left(p_t - c_{jt} - \sum_{\tau=t+L_j}^{T} h_\tau\right)X_{jt} - \sum_{t=1}^{T} b_tB_t$$

As mentioned earlier, the purpose of the subproblem is to find the most profitable entering column. For that purpose, the subproblem needs to obtain the column with maximum reduced cost. Below are the calculations for obtaining the reduced cost value for a column,

given the cost data and dual variables from the master problem. We define the following additional notation:

$\alpha_{k\tau}$: Value of the dual variable associated with the capacity constraint belonging to machine k in period τ

γ_τ : Value of the dual variable corresponding to the inventory balance constraint of period τ

c_k : Unit production cost incurred when an operation is performed on machine k

a_k : Capacity consumption when an operation is performed on machine k

M^s : The set of machines that perform operations on the path associated with column s

$$\boldsymbol{r} = \boldsymbol{c} - \boldsymbol{A^T\alpha} - \boldsymbol{C^T\gamma}$$

$$r_s = c_s - \boldsymbol{A_s}^{\boldsymbol{T}}\boldsymbol{\alpha} - \boldsymbol{C_s}^{\boldsymbol{T}}\boldsymbol{\gamma}$$

$$= \hat{p}_s - \sum_{k=1}^{K}\sum_{\tau}^{T} a_{k\tau}^{s}\alpha_{k\tau} - \sum_{\tau=t+L_j}^{T}\gamma_\tau$$

$$= p_{t_s} - c_{j_s,t_s} - \sum_{\tau=t_s+L_s}^{T} h_\tau - \sum_{k=1}^{K}\sum_{\tau}^{T} a_{k\tau}^{s}\alpha_{k\tau} - \sum_{\tau=t_s+L_s}^{T}\gamma_\tau$$

$$= p_{t_s} - \sum_{k\epsilon M^s} c_k - \sum_{\tau=t_s+L_s}^{T} h_\tau - \sum_{k\epsilon M^s} a_k\alpha_{k,t_s+L_{sk}} - \sum_{\tau=t_s+L_s}^{T}\gamma_\tau$$

$$= p_{t_s} - \sum_{k\epsilon M^s} c_k - \left(\sum_{\tau=t_s}^{T} h_\tau - \sum_{\tau=t_s}^{t_s+L_s-1} h_\tau\right) - \sum_{k\epsilon M^s} a_k\alpha_{k,t_s+L_{sk}} - \left(\sum_{\tau=t_s}^{T}\gamma_\tau - \sum_{\tau=t_s}^{t_s+L_s-1}\gamma_\tau\right)$$

$$= p_{t_s} - \sum_{k\epsilon M^s} c_k - \sum_{\tau=t_s}^{T} h_\tau + \sum_{\tau=t_s}^{t_s+L_s-1} h_\tau - \sum_{k\epsilon M^s} a_k\alpha_{k,t_s+L_{sk}} - \sum_{\tau=t_s}^{T}\gamma_\tau + \sum_{\tau=t_s}^{t_s+L_s-1}\gamma_\tau$$

$$= \left(p_{t_s} - \sum_{\tau=t_s}^{T} h_\tau - \sum_{\tau=t_s}^{T}\gamma_\tau\right) - \sum_{k\epsilon M^s} c_k + \sum_{\tau=t_s}^{t_s+L_s-1} h_\tau - \sum_{k\epsilon M^s} a_k\alpha_{k,t_s+L_{sk}} + \sum_{\tau=t_s}^{t_s+L_s-1}\gamma_\tau$$

The first three components of reduced cost are constants, so they do not affect the selection of the entering column. Therefore, these terms can be excluded from the subproblem formulation and added back later, when computing the actual reduced cost values. The last four terms, constitute the components of the subproblem cost function. One should note that these cost components are divisible with respect to machines and time periods.

Since the production stages consist of non-overlapping machine sets, and all the machines in a particular stage perform the same operation with same quality, the problem of selecting the column with largest reduced cost can be modeled as a Longest Path Problem with no cycles. This way, the four cost components of the objective function can be distributed between the arcs that define the production paths.

After solving the restricted master problem and obtaining the values of dual variables, we solve a series of subproblems. At each iteration of the restricted master problem, a subproblem is solved for each time period, in order to generate multiple columns at once and speed up the solution process. (If we were interested in the multi-commodity problem, one subproblem for each product in each time period would be solved at each iteration.)

The network structure of the Column Generation subproblem is shown in Figure 2.2. The network consists of nodes that correspond to specific machines in specific time periods, and arcs that correspond to transfer of products from one machine to another while passing from one production stage to another. According to this structure, an arc exists from node (k_1, t_1) to node (k_2, t_2) only if there is a production path suggesting that the product should be processed at machine k_1 in period t_1, and at machine k_2 in period t_2. Other arcs in the network are artificial, and they have a zero arc weight, so that they wouldn't have any effect in the selection of the longest path.

The arc costs for the longest path network should be determined so that the existing production paths have total arc weight $-\sum_{k\epsilon M^s} c_k + \sum_{\tau=t_s}^{t_s+L_s-1} h_\tau - \sum_{k\epsilon M^s} a_k \alpha_{k,t_s+L_{sk}} + \sum_{\tau=t_s}^{t_s+L_s-1} \gamma_\tau$, as this expression constitutes the objective value of the subproblem that needs to be maximized. For this purpose, the first and the third components are partitioned into machines; while the second and the fourth components are partitioned into time periods. When this logic is generalized, the arc costs between two nodes are determined as follows:

Figure 2.1: Representation of Two Nodes and an Arc in the Subproblem Network

$$w_{ij} = \begin{cases} -c_{k_j} - a_{k_j}\,\alpha_{k_j,t_j}, & if\ \ (i,j)\epsilon P\ and\ i = s; \\ -c_{k_j} + \sum_{\tau=t_i}^{t_j-1} h_\tau - a_{k_j}\,\alpha_{k_j,t_j} + \sum_{\tau=t_i}^{t_j-1} \gamma_\tau\,, & if\ \ (i,j)\epsilon P\ and\ k_j < K+1; \\ \sum_{\tau=t_i}^{t_j-1} h_\tau + \sum_{\tau=t_i}^{t_j-1} \gamma_\tau\,, & if\ \ (i,j)\epsilon P\ and\ k_j = K+1; \\ 0, & if\ \ (i,j)\epsilon P\ and\ i = t; \\ +\infty, & otherwise \end{cases}$$

where $(i,j)\epsilon P$ denotes the set of node pairs (i,j) where a product can be processed at machine k_i in period t_i, and at machin k_j in period t_j.

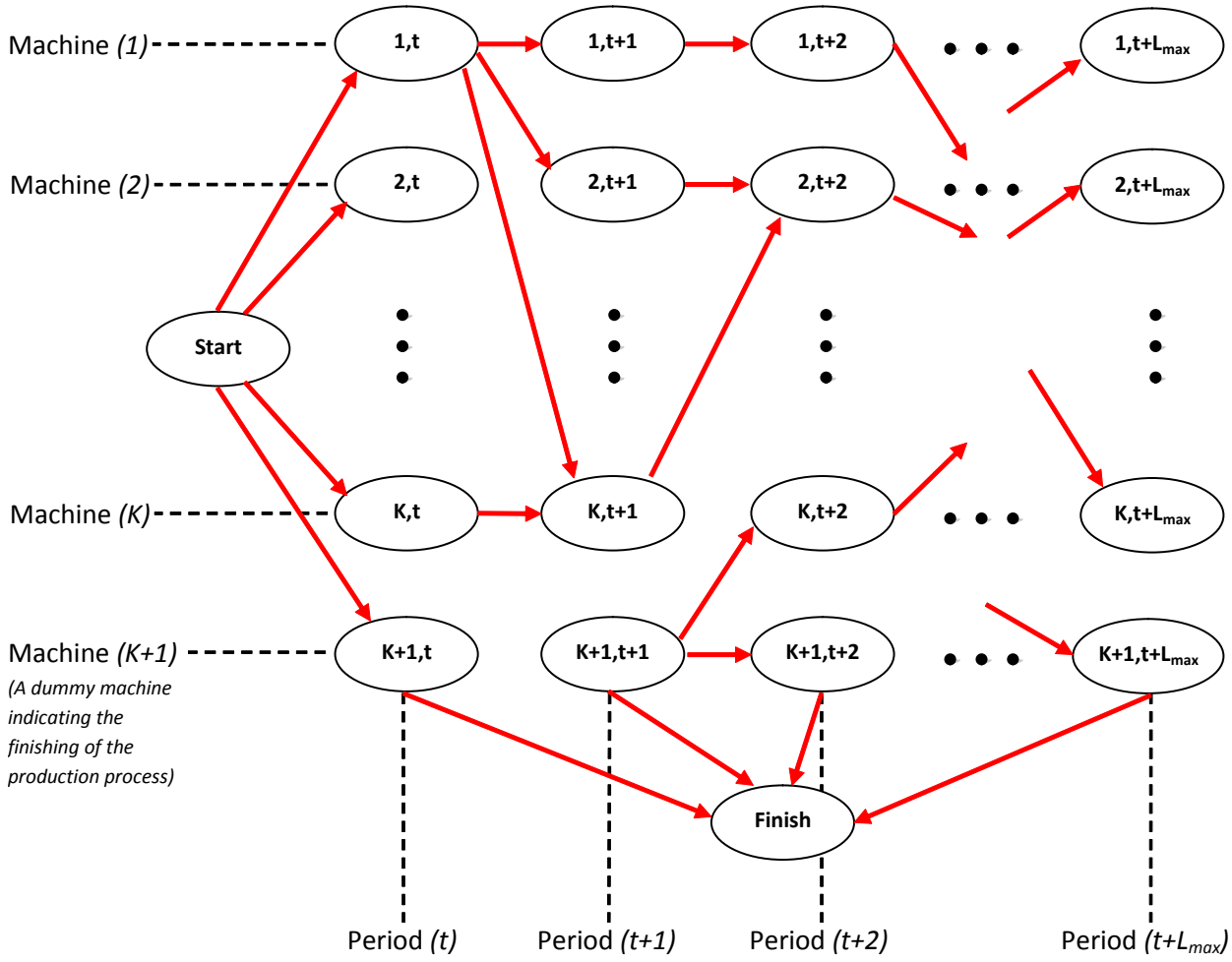

Figure 2.2: The Directed Graph G=(N,A) Defining the Column Generation Subproblem

The Longest Path Problem for a graph $G = (N, A)$ with source node s and sink node t can be formulated as follows:

Indices:

$i \epsilon N, j \epsilon N$ denote the nodes in the network

$ij \epsilon A$ denotes the arcs in the network

Parameters:

w_{ij} : Weight of arc ij, determined by the method explained above, so that the optimal solution of the subproblem corresponds to the column with maximum reduced cost in the master problem

Decision Variables:

$$Y_{ij} = \begin{cases} 1, & if\ arc\ ij\ is\ a\ member\ of\ the\ shortest\ path \\ 0, & otherwise \end{cases}$$

$$max \sum_{ij \epsilon A} w_{ij} Y_{ij}$$

$$s.t.$$

$$\sum_{j \epsilon N} Y_{ij} - \sum_{j \epsilon N} Y_{ji} = \begin{cases} 1, & if\ i = s; \\ -1, & if\ j = t; \\ 0, & otherwise \end{cases} \qquad \forall i \epsilon N$$

$$Y_{ij} \geq 0 \qquad \forall ij \epsilon A$$

The master problem and subproblems of our Column Generation approach are built and solved using IBM Optimization Programming Language (OPL) (IBM, 2011). It is also noticeable that there are relatively efficient algorithms that solve the Longest Path Problem; but we solve it using linear programming, which is probably less efficient in terms of solution time.

D. Direct Product Mix Formulation

The Direct Product Mix Formulation is the most compact formulation suggested by Leachman and Carmon (1992) in their paper. It is an exact formulation that does not use any variables that denote the workload in specific process-steps, or the workload in specific

machines. It does not use any allocation variables and its capacity constraints, which are defined for very specific sets of alternative machine types, only contain variables indicating the release quantities of each product type in each period.

Leachman and Carmon (1992) present a systematic Capacity Set Generation Procedure (CSGP) for generating alternative machine sets, and formulate capacity constraints for each of these generated sets. The design of such capacity constraints requires defining two sets: a set of machine types whose capacities are summed to form the right hand side of the constraint, and a set of operations (product-steps) whose capacity usages are summed to form the left-hand side of the constraint.

The CSGP is defined with the following three steps:

Step 1. List all alternative machine types capable of performing one or more operations (product-steps). The set of operations should include all operations that load the identified set of machine types or any proper subset.

Step 2. For all machine sets identified in Step 1 that have elements in common, form unions of these machine sets and unions of the corresponding operations. Here, larger sets are formed.

Step 3. Continue forming unions of intersecting machine sets so as to combine sets created in Step 2 with each other of with sets identified in Step 1. Terminate when no new machine sets can be generated.

The worst-case complexity of CSGP is shown to be $O(PN + P2^K)$, where P is the maximal cardinality of the alternative machine sets, K the number of machine nodes of the maximal connected component of the bipartite graph of product-steps and machine types, and N the number of product-steps in the problem data. Although the procedure is exponential, it is said to be practically useful since K, the maximal number of machine types in a connected component of the bipartite graph, is typically relatively small, especially in semiconductor manufacturing settings.

In order to demonstrate the alternative machine set concept and the outcomes of CSGP, we present an example here. Let there be a production system in which operations can be performed by the machines as shown in Table 2.1.

Table 2.1: Data for CSGP Example

Operation	Machines
A	1, 2
B	1, 3
C	1
D	2
E	1, 2, 3

Table 2.2: Outcome of CSGP Example

Set	Operations	Machines
1	A, C, D	1, 2
2	B, C	1, 3
3	C	1
4	D	2
5	A, B, C, D, E	1, 2, 3

The CSGP works with the following logic for this example:

- Operation A can be performed by machines 1 and 2, operation C can be performed by machine 1, and operation D can be performed by machine 2. Since the machine sets {1,2}, {1}, and {2} have elements in common, we can form a union to get operation set {A, C, D} and machine set {1,2}.
- Operation B can be performed by machines 1 and 3. The machines that can perform operation C (namely machine 1) is a subset of machine set {1, 3}. Therefore we can form a union set containing operations {B, C} and machines {1, 3}.
- The set containing operation {C} and machine {1}, and the set containing operation {D} and machine {2} must also considered as sets by themselves (along with the union sets containing them).
- When we add the first two union sets, namely {A, C, D, 1, 2} and {B, C, 1, 3}, together; we must also add operation E, since it can be performed by all the machines in machine set {1, 2, 3}. Therefore, the set generated by the union of the union sets becomes {A, B, C, D, E, 1, 2, 3}.

All the sets generated by CSGP for this example are given in Table 2.2.

After the alternative machine sets for which capacity constraints are necessary are generated, the parameters defining the processing times and the available machine capacities need to be scaled and expressed in terms of standard machine hours. The uniformity assumption, which requires the processing times for a given product among alternative machine types to be either identical or else proportional across all operations performed by the machines, makes this scaling procedure possible, and also reduces the problem size drastically. To convert processing times and machine capacities into standard machine hours, a machine is randomly determined to be the “standard” machine, and all the machine capacities are corrected so that the processing times are the same for all machines. This way, the processing time parameter does not need to have a machine index, hence reducing the problem size. An example of capacity and processing time correction is as follows:

Let machine 1 with available capacity of 100 minutes perform operation A in 2 minutes, and operation B in 4 minutes; and machine 2 with available capacity of 900 minutes perform operation A in 6 minutes, and operation B in 12 minutes. If we set machine 1 to be the "standard machine", then the corrected process times of machine 2 would be 2 minutes for operation A, and 4 minutes for operation B; and the corrected capacity of machine 2 would be $\frac{2}{6} \times 900 = 300$ minutes. (Here, it must be noted that the uniformity assumption holds, since $\frac{2}{6} = \frac{4}{12}$.) Since machine 1 is the standard machine, its process time and available capacity values are not corrected and they remain the same.

Once the alternative machine sets are generated and the aforementioned parameters are appropriately scales, the LP model presented below can be solved using the capacity constraints corresponding to the identified capacity sets.

Indices:

i : Product type $i = 1, \dots, I$

t : Time period $t = 1, \dots, T$

(i, j) : Process step j of product type i, where $i = 1, \dots, I$ and $j = 1, \dots, J$

Parameters:

L_i : Average flow time of product i from start to finish of its entire production process ("lead time" for the process)

L_{ij} : Average flow time for product i from the start of its production process until initiation of step j ("lead time" up to step j)

p_{it} : Estimated net discounted cash flow from producing and selling one unit of product i in period t

h_{it} : Estimated cost of holding one unit of inventory of product i at time t

a_{ij} : Time required to process one unit of product i in step j (This parameter is corrected for different machine types, with regard to the "uniformity assumption")

C_{kt} : The available capacity (in time units) of machine type k in period t (This parameter is also corrected with regard to the "uniformity assumption")

D_{it} : The maximum cumulative quantity of product i that can be sold by the end of period t

d_{it} : The minimum cumulative quantity of product i that can be sold by the end of period t

Decision Variables:

X_{it} : Production orders released to the system in period t for product i

I_{it} : Inventory of product i at the end of period t

$$max \sum_{t=1}^{T} \left(p_{it} X_{i,t-L_i} - h_{it} I_{it} \right)$$

$$s.t.$$

$$\sum_{(i,j)\epsilon S} a_{ij} X_{i,t-L_{ij}} \leq \sum_{(k)\epsilon S} C_{kt} \qquad for\ all\ generated\ sets\ S;\ t = 1, \dots, T$$

$$\sum_{\tau=1}^{t-L_i} X_{i\tau} - I_{it} + B_{it} = D_{it} \qquad i = 1, \dots, I \qquad t = 1, \dots, T-1$$

$$\sum_{\tau=1}^{T-L_i} X_{i\tau} + B_{iT} = D_{iT} \qquad i = 1, \dots, I$$

$$B_{it} \leq D_{it} - d_{it} \qquad i = 1, \dots, I \qquad t = 1, \dots, T$$

$$X_{it}, I_{it}, B_{it} \geq 0 \qquad i = 1, \dots, I \qquad t = 1, \dots, T$$

The number of variables appearing in capacity constraints each time period is equal to I, the number of product types; so the total number of variables is IT, number of product types times number of time periods. The number of capacity constraints per period is the number of generated sets $\boldsymbol{S}$ if machine types, which depends on the usage patterns appearing in the problem data. The cardinality of set $\boldsymbol{S}$ can be as small as K in the best case or as large as $2^K - 1$ in the worst case. But Leachman and Carmon (1992) discuss that it is close the number of machine types K for most practical cases, and the Direct Product Mix Formulation is usually only slightly larger than the standard production planning formulation that does not admit alternative machine types.

When they describe the alternative machine sets and CSGP, Leachman and Carmon (1992) that these machine sets correspond to dominant cutsets of the bipartite network comprised of a set of nodes corresponding to all the machines in the system, and another set of nodes corresponding to all the process-steps that can be performed in the production system. The bipartite network can be observed in Figure 2.3. A dominant cutset has the property that if the capacity constraints corresponding to that set of machines and the corresponding

operations are satisfied, all capacity constraints for subsets of that machine set and its operation set are also satisfied. In order for a set to be considered a dominant cutset, it needs to satisfy the following three characteristics:

(i) If $(i-j) \in \boldsymbol{S}$, then $(k) \in \boldsymbol{S}$ $\forall\ (k, i-j) \in \boldsymbol{A}$

(ii) If there exists an $(i-j*)$ such that whenever $(k, i-j*) \in \boldsymbol{A}$ we have $(k) \in \boldsymbol{S}$, then $(i-j*) \in \boldsymbol{S}$

(iii) $\boldsymbol{S}$ is connected.

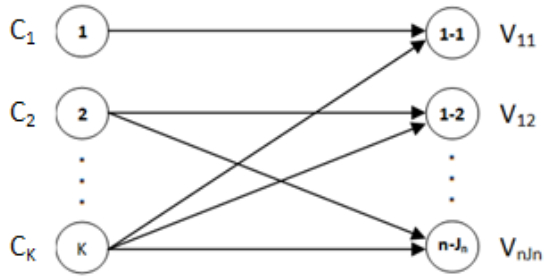

Figure 2.3: Network Representation for Capacity Constraints of Direct Product Mix Formulation

Leachman and Carmon (1992) show that the dominant cutsets are the cutsets that correspond to the required capacity constraints in Direct Product Mix Formulation. They use Gale's Flow Feasibility Theorem for Transshipment Networks (Gale, 1960) to show that cutsets for which one or more of the conditions above are violated are dominated by the cutsets that satisfy all the conditions above. Hence, the outcomes of CSGP (and equivalently, the dominant cutsets) generate the minimum number of capacity constraints that ensure capacity feasibility.

Even though the Direct Product Mix Formulation seems to be a good way of representing a multi-stage production system with alternative resources, it has a number of limitations that originate from the following assumptions:

(i) The set of machine types suitable for performing a particular processing step is independent of the machine types selected to perform other steps on the same product.

(ii) For each product there are exactly J process-steps (operations) loading one or more of the K machine types.

(iii) Production cost is independent of the product's processing route.

(iv) Formulation is valid under the "uniformity assumption". This assumption requires the processing times among alternative machine types to be either identical or else proportional across all operations performed by the machines. Under this

assumption, within a particular machine set, the allocated machine hours of each machine type are scaled into "standard" machine hours according to a "standard" machine type.

The most limiting assumption of this formulation is the uniformity assumption, which may not always be satisfied in some production systems. The other three planning models presented in this chapter are not limited by this assumption; while the first two assumptions have to hold in all the formulations presented within the scope of this thesis. Path-Based model and the Column Generation approach require only assumptions (i) and (ii), while the Capacity Partition Approach also requires assumption (iii).

E. Capacity Partition Approach

Hung and Cheng (2002) discuss two possible shortcomings of the Direct Product Mix Formulation, and suggested a solution approach to overcome them, called "Partition Approach". The first and most important of these shortcomings is the uniformity assumption, and the second one is the union operation of two machine sets used in CSGP.

As mentioned earlier, uniformity assumption can cause serious inaccuracies in real life production planning problems. The method Hung and Cheng (2002) propose to relax the uniformity assumption is the Partition Approach, which divides the capacity of machines that can perform multiple jobs into partitions, and allocating each partition to a machine set in a manner similar to the Workload Allocation formulation of Leachman and Carmon (1992). This approach requires additional variables and constraints: Partition variables to represent each capacity portion for all machines, and equality constraints to enforce the sum of partition variables of a machine type to be equal to its capacity.

Hung and Cheng (2002) introduce the Capacity Partition Generation Procedure (CPGP) for their Partition Approach. CPGP is summarized below:

Step 1. List each operation and the machine set that performs it.

Step 2. For each operation, find a partition that satisfies the uniformity assumption. If such a partition can be found, add this operation and its corresponding machine set to the partition. If it cannot be found, create a new partition and add this operation along with its corresponding machine set to the new partition.

Step 3. Within each partition, perform a union on two machine sets with common machine types, and add the operations that can be performed by these machine sets.

Step 4. For each machine set in each partition, perform union operations until there is no more possible union within a partition.

Step 5. Designate a standard machine for each partition, and scale the available capacities and processing times according to the standard machine.

Step 6. Form the capacity constraint for each machines set of each partition and the capacity conservation constraint for each machine type.

CSGP, which is the set generation procedure that needs to be performed before solving the mathematical model in the Direct Product Mix Formulation, requires generating some machine sets initially and then forming unions of these machine sets to obtain new machine sets. As the number of machine sets increases, the number of capacity constraints increases, which can cause the size of the formulation to grow considerably. Hung and Cheng (2002) argue that the number of rows in an LP matrix usually affects solution time more than the number of columns, so a large number of capacity constraints can cause the Direct Product Mix Formulation to perform badly in terms of solution time. They show that the Partition Approach eliminates the union operation, and sometimes results in fewer capacity constraints. However, they conclude, after a series of experiments, that the LP models based on the union operation of CSGP do not require significantly more CPU time than the Partition Approach. Therefore, the Partition Approach ensures accuracy when uniformity assumption is not satisfied, but it does not provide an advantage in terms of solution time by eliminating the union operation.

An example for the CPGP is presented below.

Table 2.3 shows the operation and processing time data for a production system with three machines: Machine A, Machine B, and Machine C. The distribution of operations on the machines is shown in Figure 2.4.

CPGP requires generating the partitions within which the uniformity assumption holds. We see that operation sets {5, 6}, {7, 8}, and {9, 10} satisfies the uniformity assumptions, so these sets of operations and their corresponding machines form partitions. The remaining operations and their corresponding machines form partitions by themselves. After determining the partitions, CPGP suggest that we perform union operations within each partition. For the partitions with a single operation, forming unions is not possible. For the partitions with more than one operation, we form unions with the operations sharing the same resources. In this

particular example, operation sets {5, 6}, {7, 8}, and {9, 10} can be performed on same machines; so the unions are formed. The resulting partitions can be observed in Table 2.4. The next step of the CPGP is building the capacity constraints corresponding to these partitions. In this example, there will be nine capacity constraints for each partition, and three constraints (one for each machine) to ensure that the sum of partition capacities equals available machine capacity.

Table 2.3: Data for CPGP Example

Operation	Processing Time in Machine A	Processing Time in Machine B	Processing Time in Machine C
1	50	90	100
2	40	60	90
3	80	40	-
4	35	70	-
5	20	-	30
6	60	-	45
7	-	30	50
8	-	60	75
9	45	-	-
10	50	-	-
11	-	55	-
12	-	-	15

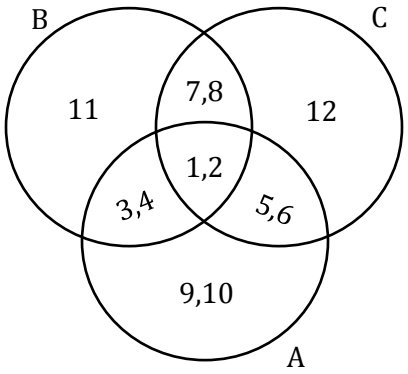

Figure 2.4: Capacity Partitions for CPGP Example

Table 2.4: Outcome of CPGP Example

Partition Index	Partition
1	{A, B, C, 1}
2	{A, B, C, 2}
3	{A, B, 3}
4	{A, B, 4}
5	{A, C, 5, 6}
6	{B, C, 7, 8}
7	{A, 9, 10}
8	{B, 11}
9	{C, 12}

The mathematical model based on the Partition Approach is as follows:

Indices:

i : Product type $i = 1, \dots, I$

k : Machine type $k = 1, \dots, K$

j : Operation $j = 1, \dots, J$

r : Partition $r = 1, \dots, R$

n : Machine set contained in a partition $n = 1, \dots, N^r$

Parameters:

$\hat{a}_{i,j}^r$: Processing time of operation j on product i by standard machine $\hat{k}$ of partition r

$\hat{s}_k^r$: The factor used in converting original machine hours of machine k to standard machine hours in partition r

N^r : Number of machine sets in partition r

G_n^r : The n^{th} machine set contained in partition r

Q_n^r : The set of operations that can be performed by n^{th} machine set of partition r

P_k : The set of partitions that contain machine k

L_{ij} : The flow time of product i from the start of its production until operation j

L_i : The flow time of product i for its entire production route

C_{kt} : Available capacity (machine hours) of machine k in time period t

D_{it} : The maximum cumulative quantity of product i that can be sold by the end of period t

d_{it} : The minimum cumulative quantity of product i that can be sold by the end of period t

p_{it} : The net profit from producing and selling one unit of product i in time period t

b_{it} : The backorder cost of product i in time period t

h_{it} : The cost of holding one unit of product i in time period t

Decision Variables:

M_{kt}^r : Machine hours of machine type k allocated to partition r in period t

X_{it} : The quantity of product i that will be released in period t

I_{it} : The quantity of product i that is in the inventory at the end of period t

B_{it} : The backorder quantity of product i at the end of period t

$$max \sum_{i=1}^{I} \sum_{t=1}^{T} \left(p_{it} X_{it-L_i} - h_{it} I_{it} - b_{it} B_{it} \right)$$

$s.t.$

$$\sum_{(i,j) \in Q_n^r} \hat{a}_{i,j}^r X_{it-L_{ij}} \leq \sum_{(k) \in G_n^r} \hat{s}_k^r M_{kt}^r \qquad n = 1, \dots, N^r \qquad r = 1, \dots, R \qquad t = 1, \dots, T$$

$$\sum_{r \in P_k} M_{kt}^r \leq C_{kt} \qquad k = 1, \dots, K \qquad t = 1, \dots, T$$

$$\sum_{\tau=1}^{t} X_{i,\tau-L_i} - I_{it} + I_{i,t-1} + B_{it} - B_{i,t-1} = D_{it} \qquad i = 1, \dots, I \qquad t = 1, \dots, T$$

$$\sum_{\tau=1}^{T} X_{i,\tau-L_i} + B_{iT} - B_{i,T-1} = D_{iT} \qquad i = 1, \dots, I$$

$$B_{it} \leq D_{it} - d_{it} \qquad i = 1, \dots, I \qquad t = 1, \dots, T$$

$$X_{it} \geq 0, \quad I_{it} \geq 0, \quad B_{it} \geq 0 \qquad i = 1, \dots, I \qquad t = 1, \dots, T$$

$$M_{kt}^r \geq 0 \qquad r \in P_k \qquad k = 1, \dots, K$$

The number of production variables is the same as Direct Product Mix Formulation, which is IT, number of product types times number of time periods. However, there are also partition variables in this formulation, different than the previous one; and the number of partition variables is KRT, where K is the number of machine types, R the number of partitions, and T the number of time periods.

There is one capacity constraint for each machine set in each partition, so the number of capacity constraints is equal to the total number of machine sets. In addition to capacity constraints, there are also constraints that ensure that sum of partition capacities is equal to the available machine capacity. The number of such constraints is KT.

As mentioned earlier, the number of partition variables and capacity constraints depends on the machine usage pattern appearing in the problem data. As Hung and Cheng (2002) point out, there will be K partitions in the best case; hence there will be a total of $2K$ constraints (K capacity constraints, and K constraints that ensure the sum of partition capacities is equal to available machine capacities) regarding capacity. In the worst case, the

Partition Approach will behave the same as Direct Product Mix Formulation; hence total number of constraints regarding capacity will be $2^K + K - 1$.

The Partition Approach generates an LP model with larger number of variables compared to the Direct Product Mix Formulation. But we cannot be certain regarding the comparison of number of constraints in both formulations, because the Partition Approach has fewer constraints in the best case (since Partition Approach eliminates the union operations of CSGP), but more constraints in the worst case. When Hung and Cheng (2002) compared the two formulations with experiments, they concluded that the Partition Approach takes approximately 15% more time than the Direct Product Mix Formulation in generating the machine sets and solving LP models.

3. Design of Experiments

The purpose of our computational experimentation is to observe how our Column Generation formulation performs, and whether or not our claims about the approach were valid. Our hypothesis regarding the CG approach is that it would work efficiently in large problems, because the LP solution would assign nonzero production variables to a very small fraction of the possible process paths in a production environment with alternative process paths. This premise motivated us towards experimenting on different production environments to see what proportion of paths are being assigned nonzero production variables. We also wish to compare the computational performance of the CG approach to the performance of PB formulation, in terms of solution time. For this purpose, some time measures related to these two approaches are compared. These time measures will be discussed in detail in Chapter 4.

We hypothesized that the two main factors that would affect the performance of our Column Generation approach would be problem size, as expressed by the number of alternative production paths, and level of distinction between different paths. If some paths are markedly better than others in terms of cost and lead time, we would expect an optimal solution to use these paths predominantly. If, on the other hand, the cost differences between different paths tend to be very small, we would expect a great many slightly different paths to appear in an optimal solution. For observing the effects of these factors, the data sets are generated to examine the effect of different values of five key parameters: number of stages, number of machines at each stage, process time variability, lead time variability and machine utilization.

Number of Stages: This parameter denotes the number of process stages in the production system, which is one of the two determinants of problem size. It takes three values: 2, 4, and 6.

Number of Machines at Each Stage: This parameter denotes the number of alternative machines at each stage, which is the other factor determining the problem size. It takes three values: 2, 4, and 6.

Process Time Variability: This parameter is defined as the difference between process times of machines that perform the same operation. In other words, it determines how much the machines at a particular stage differ from each other. It takes two values: ±10%, and ±25%.

In our data generation procedure, process times for the machines in a particular stage are randomly generated from the distribution $Uniform(LB_s, UB_s)$, where s denotes the stage index. Lower bound and upper bound values are determined by the equations $LB_s = (1 - p)\mu_s$

and $UB_s = (1 + p)\mu_s$, where p denotes the process time variability and $\mu_s$the mean processing time for that stage.

Lead Time Variability: This parameter shows whether or not lead time differs from one machine to another within a given stage. It takes two values: 0 or U[1,3]; which means either there is no lead time variability and all lead time values are set to 1, or there is lead time variability and the lead time values are randomly generated to be an integer between 1 and 3.

Machine Utilization: This parameter is defined as the target machine utilization, and it determines what proportion of the available machine capacity will be used on the average. It takes two values: 0.7, which demonstrates an average utilization level, and 0.9, which demonstrates a high utilization level.

The Column Generation approach is tested with 360 different problems, with 72 different settings and five replications in each setting. The different settings are generated by combining all possible values of the five parameters described above. As it can be seen on Table 3.1, there are nine possible problem sizes, and eight possible combinations of other production environment parameter levels.

Data sets are generated and managed using VBA in Microsoft Excel. Detailed information on data generation, including the parameter values that are the same for all production settings and how each mathematical model input is calculated, can be found in Appendix A.

Table 3.1: Design of Experiments with Different Values of Parameters

Number of Stages	Number of Machines at Each Stage
2	2
2	4
2	6
4	2
4	4
4	6
6	2
6	4
6	6

Process Time Variability	Lead Time Variability	Machine Utilization
±10%	0	0.7
±10%	0	0.9
±10%	U[1,3]	0.7
±10%	U[1,3]	0.9
±25%	0	0.7
±25%	0	0.9
±25%	U[1,3]	0.7
±25%	U[1,3]	0.9

4. Computational Results

A. Proportion of Generated and Used Columns

While suggesting the solution approach and designing the experiments, we predicted that only a small proportion of possible columns would actually be generated by the Column Generation (CG) approach and assigned a non-zero production quantity, even though the number of all possible paths is very large. To verify this hypothesis, we consider the following two performance measures in our computational experiments:

Proportion of Generated Columns: This measure is given by the number of generated columns divided by the number of all possible columns, which is given by the number of possible production paths times the number of time periods. Recall that in a production system with n stages and m alternative machines at each stage, the total number of possible columns, i.e., the number of decision variables in the path-based model described in Chapter 2, is given by Tn^m.

Proportion of Used Columns: This measure is defined as the number of columns whose associated decision variable is positive, representing the number of paths actually used in production in some period, divided by the number of all possible columns.

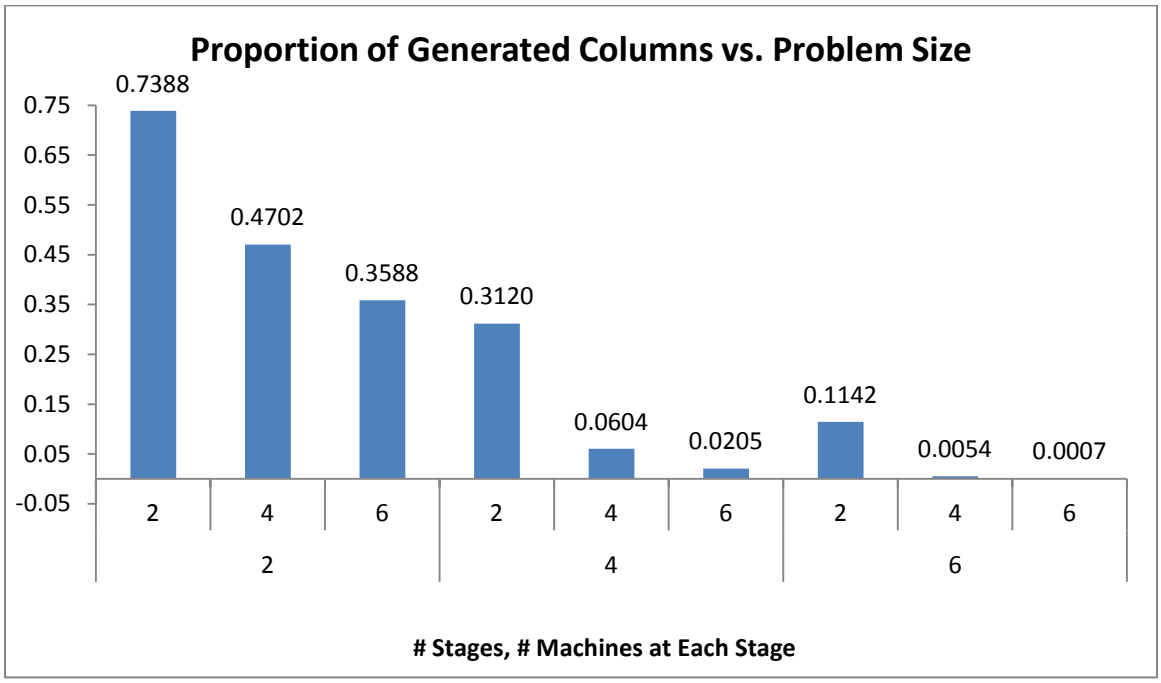

Figure 4.1: Proportion of Generated Columns with Changing Problem Size

Our results show that the Column Generation approach gets more and more efficient as problem size increases. Figure 4.1 shows the proportion of generated columns as a function of the problem size, defined by the number of production stages and the number of machines in each stage. The proportion of generated columns decreases as problem size increases, and gets as small as 0.0007 on the average.

Table 4.1: Proportion of Generated Columns vs. Number of Possible Paths

Number of Stages	Number of Machines at Each Stage	Number of Possible Production Paths	Proportion of Generated Columns
2	2	4	0.739
2	4	16	0.47
4	2	16	0.312
2	6	36	0.359
6	2	64	0.114
4	4	256	0.06
4	6	1296	0.021
6	4	4096	0.005
6	6	46656	0.001

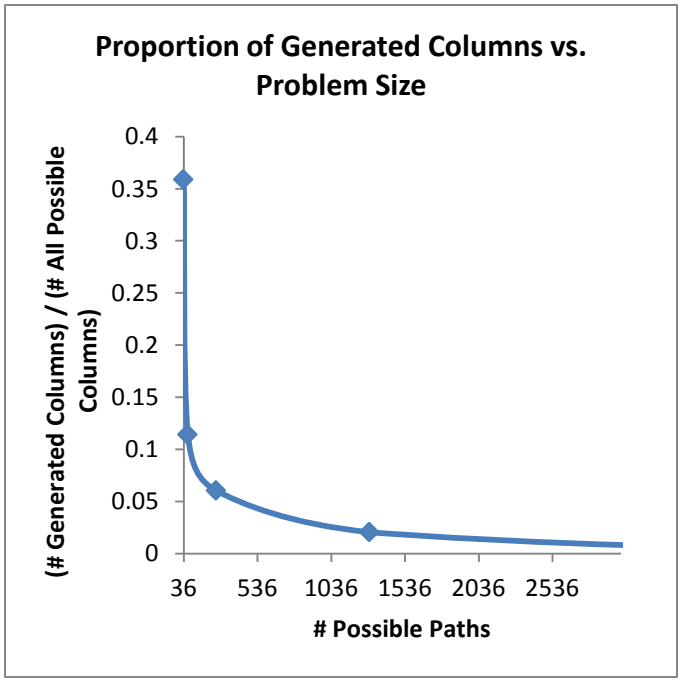

Figure 4.2: Proportion of Generated Columns with Changing Number of Possible Paths

Table 4.1 shows the proportion of generated columns for different values of problem size, expressed as the number of possible production paths. Figure 4.2 shows how the proportion of generated columns changes with the number of possible paths. The proportion of generated columns decreases exponentially with increasing number of paths.

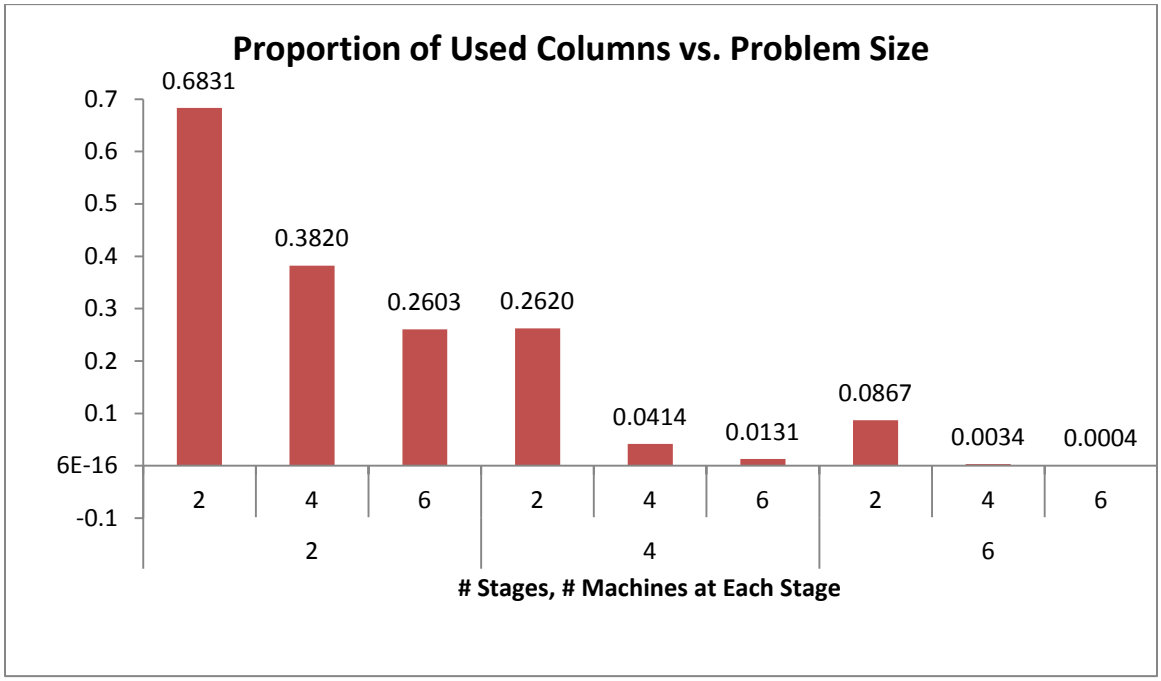

Figure 4.3: Proportion of Used Columns with Changing Problem Size

Table 4.2: Proportion of Used Columns vs. Number of Possible Paths

Number of Stages	Number of Machines at Each Stage	Number of Possible Production Paths	Proportion of Used Columns
2	2	4	0.683
2	4	16	0.382
4	2	16	0.262
2	6	36	0.260
6	2	64	0.087
4	4	256	0.041
4	6	1296	0.013
6	4	4096	0.003
6	6	46656	0.0004

The proportion of used columns changes with changing problem size in a similar way to the proportion of generated columns. As seen in Figure 4.3, the proportion of used columns

decreases with increasing problem size. Table 4.2 and Figure 4.4 suggest that the decrease in the proportion of used columns seems to be exponential.

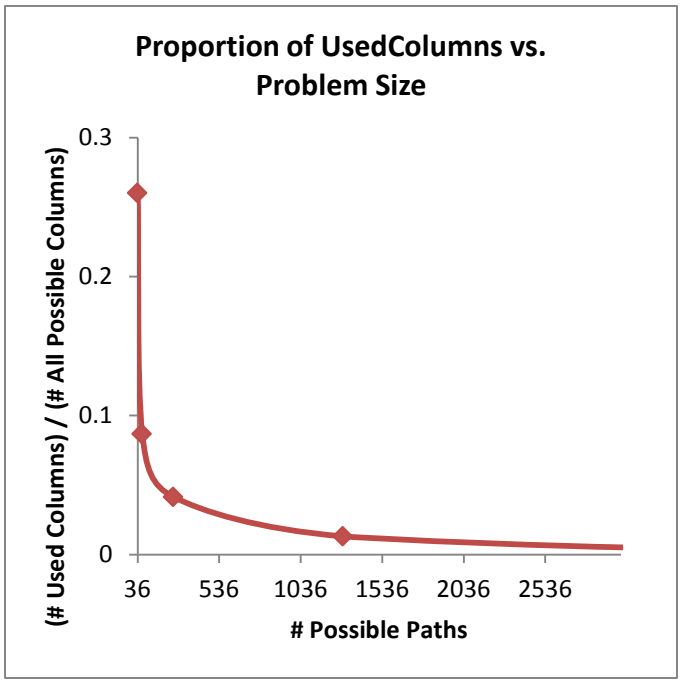

Figure 4.4: Proportion of Used Columns with Changing Number of Possible Paths

B. Solution Times of CG, and Comparison with PB

In this section we shall examine the computation times of the Path-Based and CG formulations. It is important to note that our CG implementation cannot be expected to compete with the Path-Based formulation in terms of solution time for several reasons. First of all, the CG method involves considerable overhead in terms of reading data files and constructing subproblem instances as well as the control logic for the CG approach that are not needed in the Path-Based model. Secondly, we have made no attempt to engineer our implementation of the CG method for efficiency. Several steps that could be taken in that direction include efficient implementation of the shortest path algorithm used to solve the subproblems, instead of solving as a linear program; restarting the master problem at each iteration from the final basis of the previous iteration, as opposed to restarting the solution of the master problem from scratch; and improving the efficiency of reading and outputting data through a more sophisticated OPL implementation. All these issues can be addressed directly if the CG method is implemented in a programming language like C++. However, our intention in this thesis is to explore the potential

of the CG method for large problems with many alternative routings, by examining the number of columns actually generated and used.

We define the following performance measures:

Solve Time: The time, expressed in milliseconds, it takes to actually solve the mathematical models, both in the Column Generation approach and the Path-Based approach. This measure of time excludes all types of initialization, data reading, data modification, model building, or result interpretation processes. It only measures the time it takes for the LP solving algorithm to find the optimal solutions to the LP problems. For Column Generation (CG) approach, this time measure is the total time it takes to solve all the master problems and subproblems, since this approach iteratively solves these problems multiple times. For Path-Based (PB) approach, this time measure only denotes the time to solve the Path-Based model once.

Data Time: The time, expressed in milliseconds, it takes to manage the data. This includes the time required to read initial data from Excel files, organize and modify the data into the format that the mathematical model requires, make necessary calculations to build the mathematical models, and transform the model outcomes into the necessary format. Most of these actions are required for CG approach, but still, data time is measured for PB approach as well.

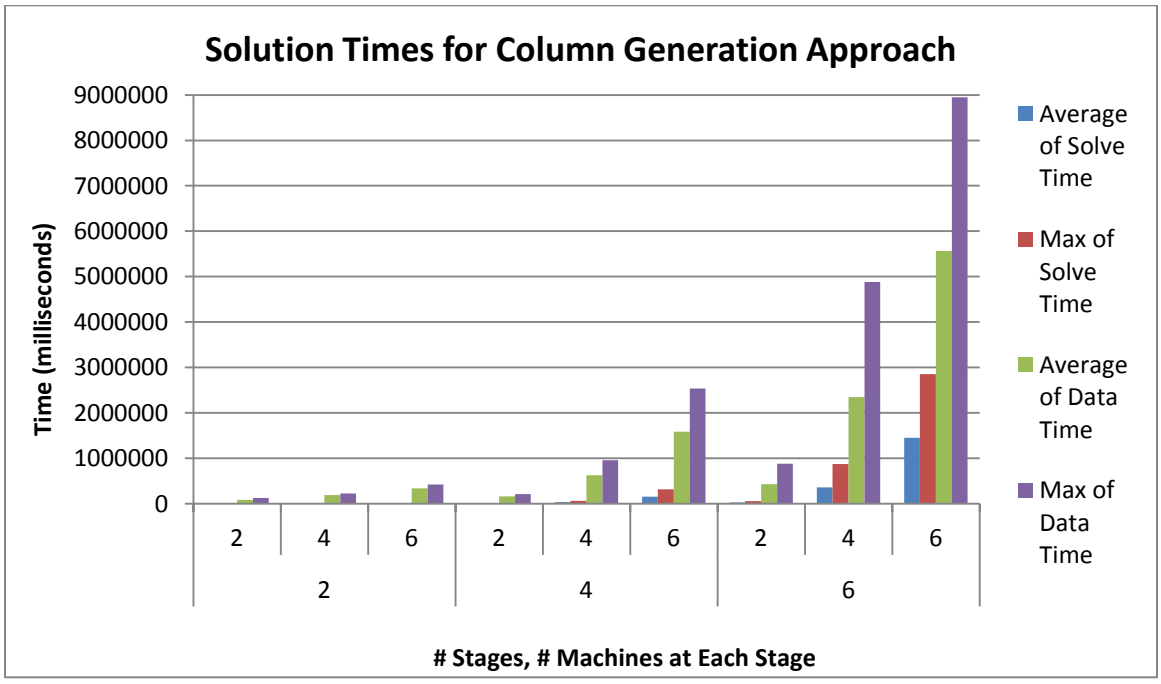

Figure 4.5: Solution Time Outcomes for CG Approach

The total time it takes to solve a problem with CG approach ranges from a minute to three and a half hours, with the data sets we used within the scope of this thesis. On average, around 5% of the total time is solve time, around 92% of it is data time, and the remainder representing the time spent for activities such as building and loading the mathematical models, printing major outcomes to screen, etc. The fraction of total time spent for model solving is around 2% for problems with 2 stages, around 4% for problems with 4 stages, and around 9% for problems with 6 stages. Thus the solution time for the CG approach is dominated by the data time.

The average and maximum values of solve time and data time for different problem sizes are shown in Figure 4.5. Looking at the chart, one can see that the solve time and data time increase exponentially with increasing problem size. In addition, the fraction of solve time in total time increases as problem size increases. This means that the fraction of data time decreases.

As mentioned earlier, the motivation for developing a column generation approach for the production planning problem with alternative machine types was that only a small proportion of all the possible columns would actually be assigned non-zero production. Since the Path-Based model generates all the possible columns in the beginning and uses the variables for all possible (path, period) pairs in the mathematical model, it might be more time-efficient to use the Column Generation approach instead of solving the Path-Based model. Experimentation is performed to see how solve times differ for CG and PB approaches, and the results can be observed in Figure 4.6 and Figure 4.7. As it can be seen, PB approach performs better than CG in the problems solved within the scope of this research for comparison purposes. CG approach results in solve times much higher than the PB approach. Here, we must take into consideration the following facts:

1. The experimentation is performed for relatively small problem sizes, due to software and time limitations. If the problems could be designed and solved for larger production systems, the Column Generation approach would probably perform better than PB model at some point, because CG approach tends to be more time-efficient with increasing problem size and the number of variables in PB model increases exponentially with increasing number of production stages.
2. The subproblems are in form of the shortest path problem, which can be solved efficiently with algorithms such as Dijkstra's algorithm. But, we use an LP solver to solve the subproblems in our solution mechanism. Therefore, the total solve time in

CG approach would be smaller, if more efficient ways were used to solve the subproblems.

3. OPL does not have a very efficient model building mechanism when there are multiple models in a control loop. Using a different programming language would reduce the time lost because of model building/solving inefficiencies, hence reducing the total solution time.

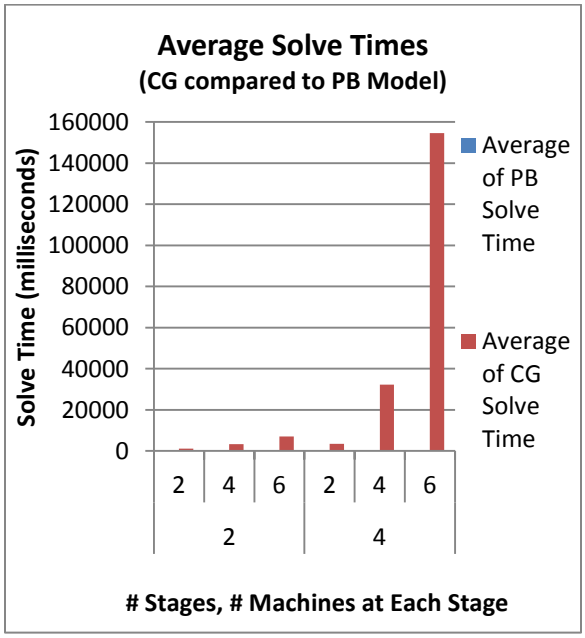

Figure 4.6: Average Solve Time Comparison for CG and PB

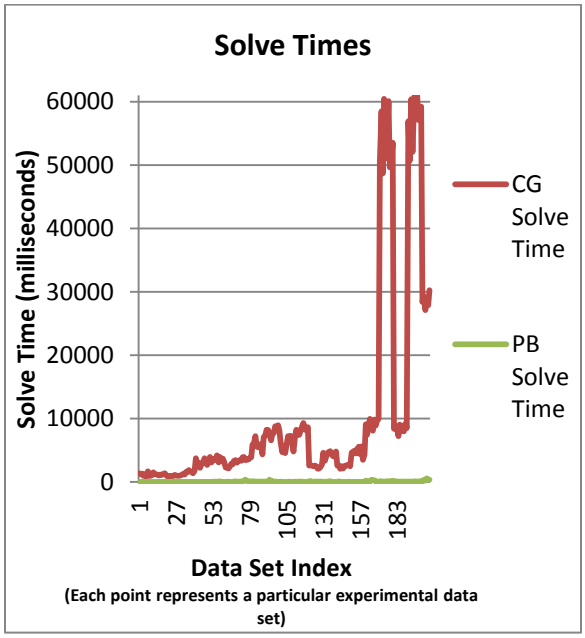

Figure 4.7: Solve Time Values for CG and PB Approaches

Even though the solve time of CG approach is much higher than the solve time of PB model, CG approach is still promising in terms of solving very large problems. We previously claimed that the solve time of PB model would increase exponentially with increasing problem size, while CG formulation would still remain manageable even for very large problem instances. To support our claim, we performed a regression analysis for solve times and problem sizes.

As seen on Figure 4.8 and Figure 4.9, PB solve time increases with the equation $y=37.927e^{0.0019x}$, while CG solve time increases with the equation $y=10643e^{1E-04x}$. Since the exponent of the PB equation is almost 20 times the exponent of the CG equation, we can conclude with confidence that the solution time of PB model increases with a much higher rate than the CG formulation. This result shows that our Column Generation approach has potential

for solving very large problem instances, and we expect it to work even when the PB model fails to generate an optimal solution with reasonable amount of time.

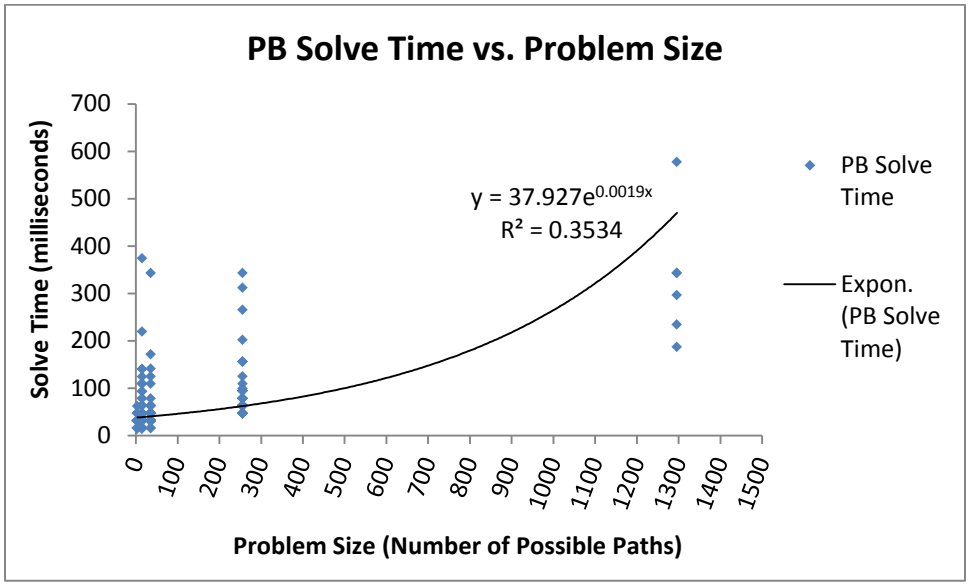

Figure 4.8: Regression for PB Solve Time with Changing Values of Problem Size

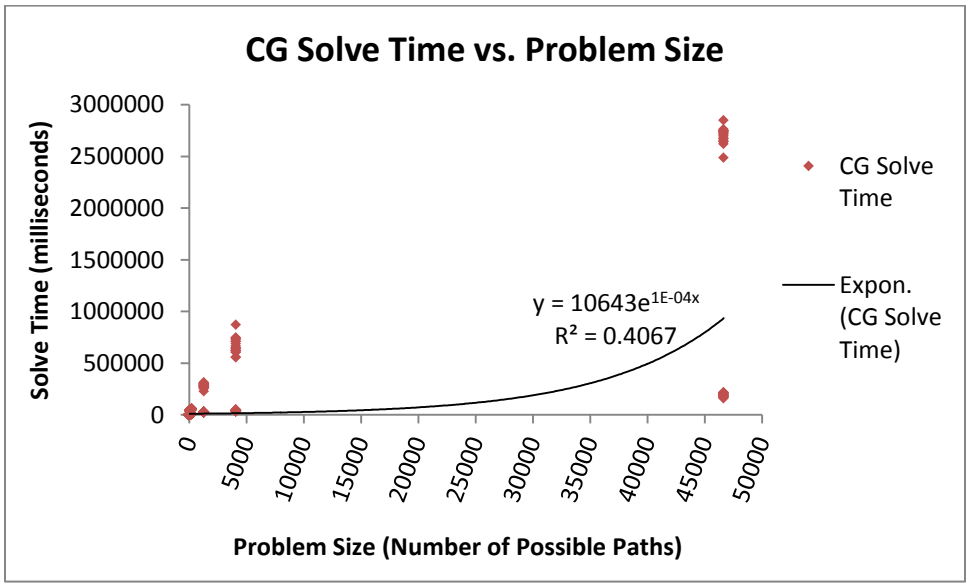

Figure 4.9: Regression for CG Solve Time with Changing Values of Problem Size

We mentioned earlier that it is possible to improve the solve time of subproblems. Since this is the case, experimentation is performed in order to see how solve time of only master problems in CG approach compares to the solve time of PB model. In general, solve time of master problems constitutes around 3.6% of total solve time (master problem solve time plus

subproblem solve time), and around 0.09% of total time (time it takes from the start to the end of whole solution procedure). To be more precise, master problem solve time constitutes around 6% of total solve time in problems with 2 stages; where this percentage decreases to 3.5% in problems with 4 stages, and to 1.5% in problems with 6 stages.

Figure 4.10 shows how the average master problem solve time of the CG approach compares to the average solve time of PB approach. Even when subproblem solve time is excluded from the time measure, Path-Based model with all possible path variables is solved faster, because the average PB solve time is smaller than the average master problem solve time of CG approach in all problem sizes experimented here. As mentioned earlier, CG was expected to perform well in terms of time measures in very large problem settings; and problem sizes we could experiment on are relatively small because of software and time limitations. When Figure 4.10 is examined in detail, it can be observed that in larger problems CG master solve time seems to increase with a decreasing rate, while PB solve time increases with an increasing rate. This finding supports the claim that CG might perform better than PB in larger problems.

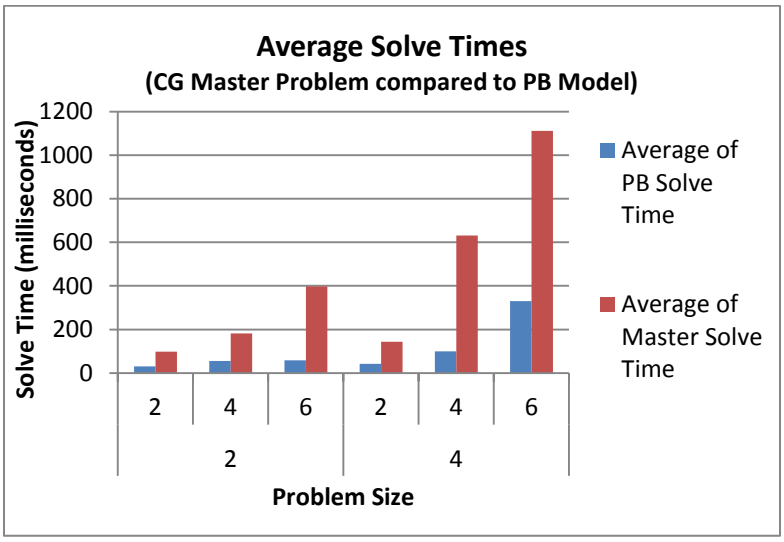

Figure 4.10: Average Solve Time Comparison for Master Models of CG Approach and the PB Model

C. The Effect of Process Time Variability, Lead Time Variability, and Machine Utilization on Proportion of Used Columns

The first parameter that was expected to affect the outcome is the processing time variability within a stage; in other words, how much the machines in the same process stage differ from each other. The predicted result was a decrease in average ratio of used columns

when the process time variability within the stage increases. As the variability within the stage increases, the hypothesis was that some machines in the stage would become more favorable, making paths containing those machines more favorable. When there are noticeably favorable paths, the mathematical model should select those paths as entering variables, and assign as much production as possible to those paths. In contrast, in production systems where there is not much process time variability within the stages, the solution approach would tend to distribute the production within the paths that are more advantageous than others, but not as significantly different from each other.

Despite the expected results explained above, the actual results are as shown in Figure 4.11. The ratio of used columns tends to decrease as process time variability within the stage increases; but the difference is not as visible as expected.

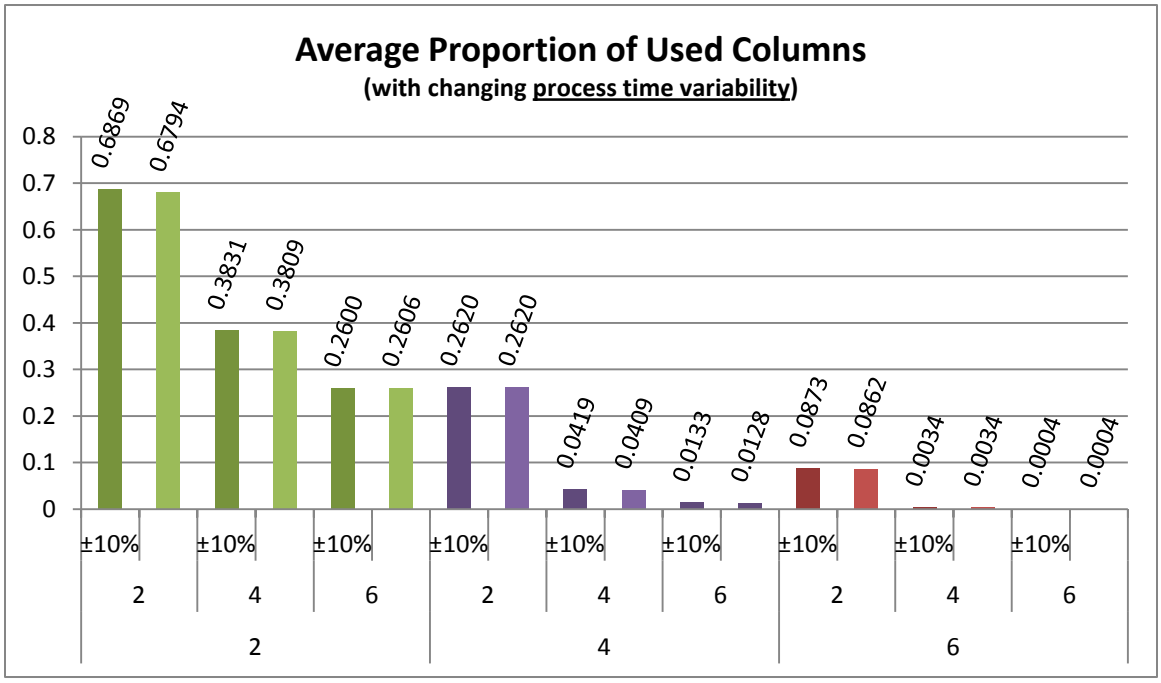

Figure 4.11: Average Proportion of Used Columns vs. Process Time Variability

In order to see whether there is a statistically significant relationship between process time variability and proportion of used columns, t-tests for the following hypotheses are performed for each problem size with 0.05 level of significance.

$$\boldsymbol{H_0}: \mu_1 = \mu_2$$
$$\boldsymbol{H_1}: \mu_1 > \mu_2$$

As seen in Table 4.3, the t-tests did not provide any statistical evidence that the process time variability has any effect on proportion of used columns.

Table 4.3: Hypothesis Testing for the Difference of Means (Low Process Time Variability vs. High Process Time Variability)

Number of Stages	Number of Machines in Each Stage	Process Time Variability	$\bar{x}_2 - \bar{x}_1$	n	Standard Error	t-statistic	df	df (rounded down)	$t_{0.05}$	Result
2	2	±10%	0.008	20	0.022	0.34	37.92	37	1.687	Means are not different
		±25%		20						
	4	±10%	0.002	20	0.01	0.2114	37.81	37	1.6871	Means are not different
		±25%		20						
	6	±10%	-0.001	20	0.007	-0.083	37.37	37	1.6871	Means are not different
		±25%		20						
4	2	±10%	0	20	0.011	0	37.09	37	1.6871	Means are not different
		±10%		20						
	4	±25%	0.001	20	0.001	0.894	37.84	37	1.6871	Means are not different
		±10%		20						
	6	±25%	0	20	0	1.48	38	37	1.6871	Means are not different
		±10%		20						
6	2	±25%	0.001	20	0.003	0.395	37	36	1.6883	Means are not different
		±10%		20						
	4	±10%	0	20	0	0.476	37.88	37	1.6871	Means are not different
		±25%		20						
	6	±10%	0	20	0	-0.047	37.6	37	1.6871	Means are not different
		±25%		20						

The second parameter that was expected to have an effect on the proportion of used columns in Column Generation approach was lead time variability. The data sets were divided into two groups: one group with constant lead time among all machines, and one group where machine lead times are generated randomly with a discrete uniform distribution. Similar to the process time variability parameter, the aim was to make the distinction between different machines more obvious, so that some paths become significantly more desirable than others.

When some paths are more desirable than others in terms of improving the objective value, the model would assign as much production as possible to those models, and try to avoid the other paths unless there is absolutely no other choice. Consequently, a little number of paths would be actually utilized in such production settings. On the contrary, in production settings where there are no paths with distinct advantage, the model would assign production to a larger number of paths with similar characteristics; hence resulting in a higher proportion of used columns.

The experimentation resulted as expected, and the chart below shows that the Column Generation approach utilizes a larger number of columns when solving the problem for data sets with no lead time variability, when compared to the data sets with lead time variability. This outcome provides support to the aforementioned insight stating that a smaller proportion of the possible columns would be utilized when there is clear distinction between different production paths.

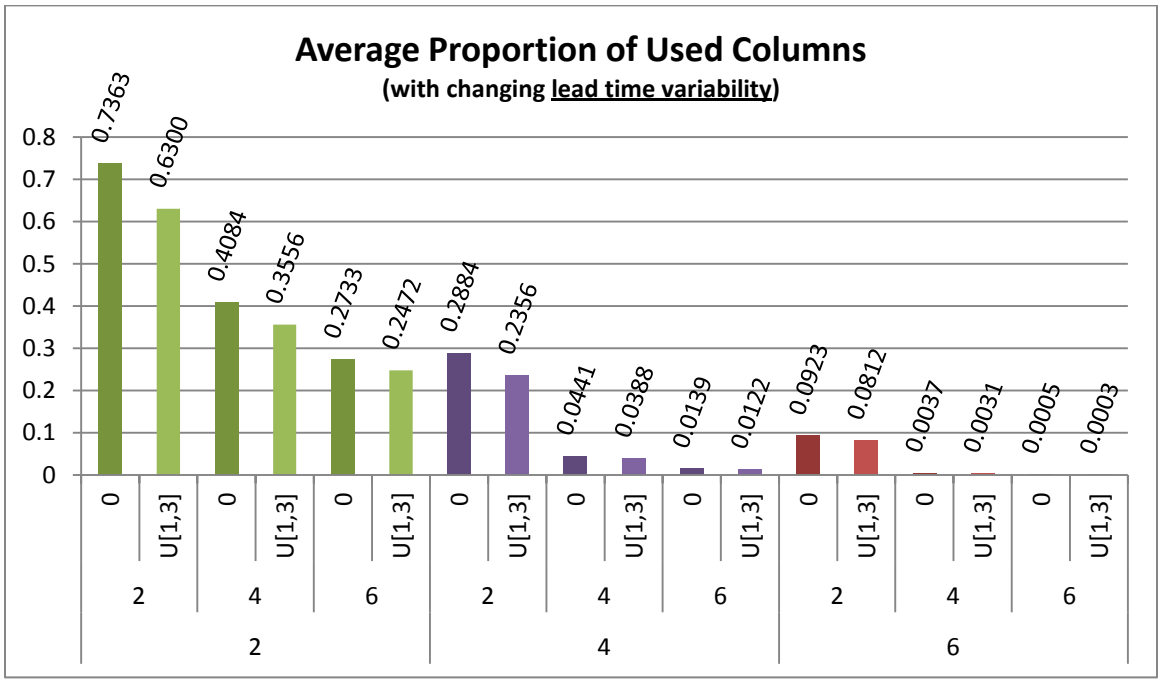

Figure 4.12: Average Proportion of Used Columns vs. Lead Time Variability

Visual inspection of the above chart shows that the problems with lead time variability utilizes a smaller number of possible columns, and this finding can be made more concrete by testing whether the difference of used column proportions between the cases with lead time variability and the cases without lead time variability are statistically significant. For that

purpose, t-tests for the following hypotheses are performed for each problem size with 0.05 level of significance.

$$\boldsymbol{H_0}: \mu_1 = \mu_2$$
$$\boldsymbol{H_1}: \mu_1 > \mu_2$$

The results are shown in Table 4.4. As can be seen, the mean proportion of used columns is significantly lower for the problems where lead time is variable among different machines, when compared to the problems where lead time is constant for all machines.

Table 4.4: Hypothesis Testing for the Difference of Means (Lead Time Variability vs. No Lead Time Variability)

Number of Stages	Number of Machines in Each Stage	Lead Time Variability	$\bar{x}_1 - \bar{x}_2$	n	Standard Error	t-statistic	df	df (rounded down)	$t_{0.05}$	Result
2	2	0 U[1,3]	0.106	20 20	0.014	7.687	19.31	19	1.729	Means are different
	4	0 U[1,3]	0.053	20 20	0.006	9.083	37.2	37	1.687	Means are different
	6	0 U[1,3]	0.026	20 20	0.005	5.03	26.06	26	1.706	Means are different
4	2	0 U[1,3]	0.053	20 20	0.006	8.52	29.5	29	1.699	Means are different
	4	0 U[1,3]	0.005	20 20	0.001	7.54	36.74	36	1.688	Means are different
	6	0 U[1,3]	0.002	20 20	0	8.471	37.79	37	1.687	Means are different
6	2	0 U[1,3]	0.011	20 20	0.002	5.561	35.63	35	1.69	Means are different
	4	0 U[1,3]	0.001	20 20	0	9.317	34.12	34	1.691	Means are different
	6	0 U[1,3]	0	20 20	0	44.234	34.71	34	1.691	Means are different

The third parameter that was expected to change the main outcome, namely the proportion of used columns, was the average target utilization of the machines. The data sets were designed so that there is a distinction between low-utilization and high-utilization settings. Column generation approach is experimented on production settings where 70% of

the machine capacities are utilized on the average, as well as settings where 90% of the machine capacities are utilized. The basic insight was that more paths would be assigned non-zero production when the target utilization is high, and less number of paths would be assigned non-zero production when the target utilization is low. Therefore we expect the problem instances with 70% utilization to have higher proportions of used columns, when compared to the problem instances with 90% utilization.

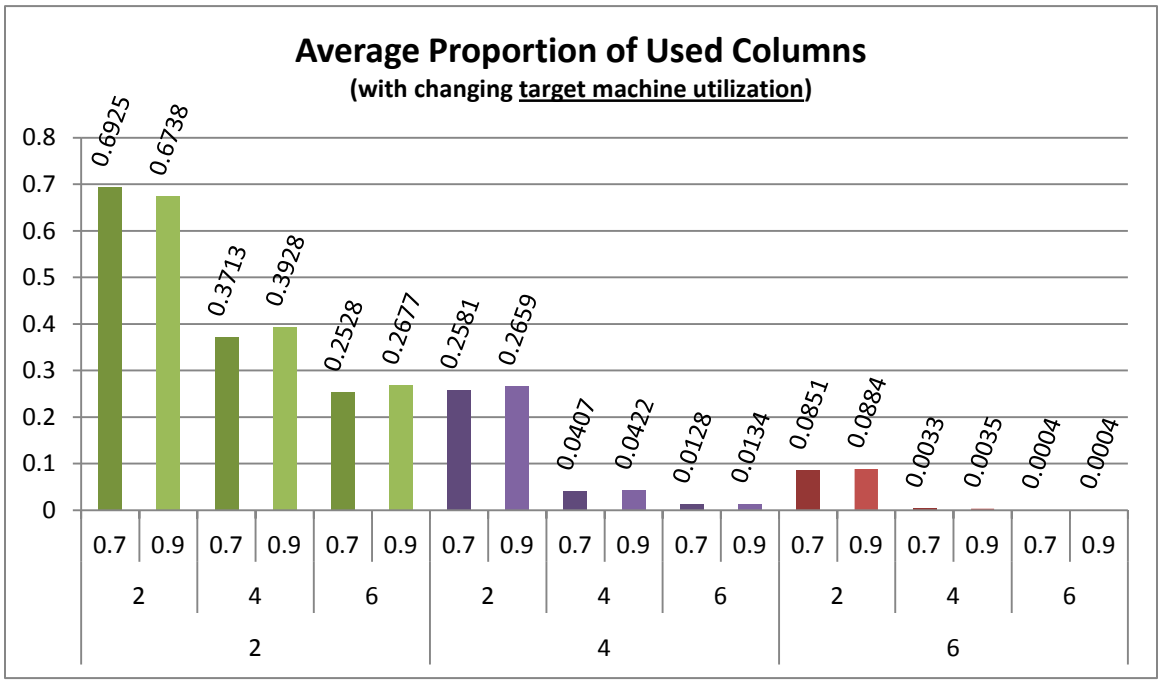

Figure 4.13: Average Proportion of Used Columns vs. Target Machine Utilization

The results are shown in Figure 4.13, and there seems to be no apparent trend in proportion of used columns that can be observed with changing values of utilization. Even though the proportion of used columns show a slight increase with increasing utilization values, the numbers are very close and it is necessary to perform hypothesis testing to see whether utilization has an effect on proportion of used columns.

For the purpose mentioned above, the following hypotheses are tested with 0.05 level of significance:

$$\boldsymbol{H_0}: \mu_1 = \mu_2$$
$$\boldsymbol{H_1}: \mu_2 > \mu_1$$

The results are shown in Table 4.5. As it can be seen, the difference in mean proportions of used columns is not significant for target machine utilization values 70% and 90%. This means that our experiments could not provide enough evidence for the claim that increased machine utilization decreases the proportion of used columns in our Column Generation approach.

Table 4.5: Hypothesis Testing for the Difference of Means (Low Machine Utilization vs. High Machine Utilization)

Number of Stages	Number of Machines in Each Stage	Utilization	$\bar{x}_2 - \bar{x}_1$	n	Standard Error	t-statistic	df	df (rounded down)	$t_{0.05}$	Result
2	2	0.7 0.9	-0.019	20 20	0.022	-0.857	32.81	32	1.694	Means are not different
	4	0.7 0.9	0.022	20 20	0.01	2.213	30.37	30	1.697	Means are different
	6	0.7 0.9	0.015	20 20	0.006	2.377	30.33	30	1.697	Means are different
4	2	0.7 0.9	0.008	20 20	0.011	0.744	37.91	37	1.687	Means are not different
	4	0.7 0.9	0.001	20 20	0.001	1.354	36.83	36	1.688	Means are not different
	6	0.7 0.9	0.001	20 20	0	1.666	35.9	35	1.69	Means are not different
6	2	0.7 0.9	0.003	20 20	0.003	1.235	37.67	37	1.687	Means are not different
	4	0.7 0.9	0	20 20	0	1.013	35.78	35	1.69	Means are not different
	6	0.7 0.9	0	20 20	0	0.243	37.88	37	1.687	Means are not different

In this chapter, we examined our Column Generation approach in terms of solution time performance and potential. First, we demonstrated that the CG approach has potential in solving very large problems by showing that only a very small proportion of possible columns is actually generated and used. This finding provides support for our hypothesis in which we

claimed that problem size of the CG models are likely to stay manageable even though the number of possible columns get very large.

Then, we illustrated how our CG approach performs when compared to the Path-Based formulation. As discussed in detail earlier in this chapter, we did not expect our CG approach to have a solution time performance comparable to that of the Path-Based model. As expected, Path-Based model performed a lot better when compared to the CG formulation.

Lastly, we experimented on some production environment parameters to determine in what kinds of settings the CG approach has the most potential. In general, we claimed that the CG approach performs the best when the production paths in the system are very different than each other and there are some paths with clear advantage against others. We presented some results that do not provide any strong evidence to support this claim, along with some results that supports it, such as the result showing that increased lead time variability reduces the proportion of used columns in our CG approach.

5. Conclusion

In this thesis, the production planning problem with alternative machine types is described, and existing linear programming formulations and solution methods for that problem are presented. Also, the ideas from existing literature, which motivate our column generation approach are discussed.

Path-Based model, which defines the number of jobs released onto each path in each time period as the decision variable, is the traditional production planning model with high levels of accuracy and flexibility. It is possible to model almost any production environment using this model, but the accuracy and flexibility comes at a cost of exponentially increasing computational complexity. Our computational experiments have shown that only a small fraction of paths are assigned non-zero production by the mathematical model. The facts that the number of variables increase exponentially with increasing number of stages in a Path-Based approach, and the mathematical model tends to assign non-zero production to only a small fraction of paths suggests the idea of a column generation approach to the Path-Based model.

Our Column Generation approach introduces the most profitable columns, i.e. decision variables denoting a specific path in a specific period, into the master model at each iteration until there remains no more profitable columns. It determines the most profitable columns by solving pricing problems, which are in the form of a shortest path problem.

Along with the Path-Based model and the Column Generation formulation, two other modeling approaches, the Direct Product Mix Formulation of Leachman and Carmon (1992) and the Capacity Partition approach of Hung and Cheng (2002), are also presented in this thesis. The main idea of these formulations is eliminating path variables and obtaining a smaller formulation by generating capacity constraints for particular machine sets, instead of single machines. The authors show that the Direct Product Mix Formulation may perform as well as the standard LP planning formulation with no alternative machine types in the best case, but it can also perform as bad as a Path-Based model, in terms of problem size and solution time. The Capacity Partition approach may perform better or worse than the Direct Product Mix Formulation in terms of solution time. But on the average, it is shown to perform 15% worse than the Direct Product Mix Formulation.

The most important drawback of the Direct Product Mix Formulation is the uniformity assumption. Under this assumption, the processing times for a given product among alternative

machine types must be proportional across all operations performed by the machines. For the Capacity Set Generation Procedure (CSGP) and the capacity constraints for alternative machine sets to function properly, the uniformity assumption must hold. As stated by Leachman and Carmon (1992), it holds in many semiconductor manufacturing facilities, but it is very limiting when other types of manufacturing facilities are considered. Of course, it is necessary to experiment on how this formulation performs for the problems where uniformity assumption is violated. But due to time constraints, it remains as future work.

The Capacity Partition Approach is created with the intention of taking advantage of the small problem sizes associated with the Direct Product Mix Formulation, while also not being restricted by the uniformity assumption. The main idea of this approach is to partition the machines capacities and distribute them among different machine sets, and generate capacity constraints accordingly. In reality, we don't know how this approach performs against the other formulations presented in this thesis, because the Capacity Partition Generation Procedure (CPGP) has increased complexity when compared to CSGP, and the mathematical model can be smaller of larger than the other models. Therefore, it remains as future work to perform computational experiments on how efficiently Capacity Partition Approach performs in terms of solution time.

After defining the different solution approaches, computational experiments are performed on some of the formulations for observing the following three outcomes: what proportion of possible columns are generated and used in the Column Generation approach, how the Column Generation formulation performs in terms of solution time when compared to the Path-Based model, and how the proportion of used columns in the Column Generation approach is affected with different factors.

First, we tested our Column Generation approach to see whether our hypothesis that only a small fraction of columns will actually be assigned production, is supported. The results show that the proportion of used columns decreases exponentially as problem size increases. In the largest problem we solved in our computational experiments, the fraction of used columns turned out to be as small as 0.0004; and this number is expected to decrease as problem size increases. This result supports our hypothesis that the Column Generation approach tends to get more and more efficient as the problem gets larger, and shows us that the Column Generation approach has potential. As the number of possible paths (and therefore possible columns) increases exponentially when the number of stages or number of machines at each stage increase, the Path-Based model will eventually fail to solve the problem to optimality in a

reasonable amount of time. But due to its property explained above, the Column Generation approach would solve the problem in reasonable amount of time even for very large problems. Another advantage of the Column Generation approach to the Path-Based model is that by changing the stopping criteria, the Column Generation formulation can be designed to return suboptimal solutions that are very close to optimal, in a short amount of time.

Second, we tested our models for solution time and discussed the outcomes. As expected, the total solution times of the Column Generation approach was not even comparable to those of the Path-Based model due to several reasons. First of all, we did not design our solution approach to be computationally efficient, and the solution time can be improved greatly by modifying it in that sense. Also, it is known that the model building capabilities of OPL in control loops with multiple models is not very efficient. Another reason why the Column Generation approach could not perform as well as the Path-Based model is that the problems for which we tested the two formulations were very small when compared to the size of real-life production planning problems; the Column Generation approach could have performed better if we had the opportunity to solve larger problems.

The computational experiments supported in many ways our hypothesis stating that the Column Generation approach would be more and more efficient as problem size increases. Another finding, revealed as a result of a regression analysis, was that the solution time of the Path-Based model increases exponentially at a high rate, while the solution time of the Column Generation formulation increases at a much slower rate. This is another fact, which shows that our Column Generation formulation has potential in solving production planning problems with alternative machine types.

The third aspect for which we tested our Column Generation approach was the parameters that affect the performance in terms of proportion of used columns. We first asserted that proportion of used columns decrease, hence increasing the potential of the Column Generation approach, as the paths get more different than each other. When paths significantly differ from each other, the Column Generation approach finds the most profitable columns right away and assigns production to those columns. On the other hand, when the paths are similar to each other in terms of profitability and capacity consumption, the mathematical models cannot distinguish between the columns as easily and perform more iterations to find the best columns for assigning production.

The parameters for which we tested the Column Generation approach were process time variability, lead time variability, and machine utilization. The results of the computational

experiments showed us that process time variability and machine utilization have no significant effect on the proportion of used columns, while lead time variability has a significant effect. Increased lead time variability decreases the proportion of used columns, hence showing that the Column Generation approach in fact performs more efficiently when the paths significantly differ from each other.

6. Future Directions

The Direct Product Mix Formulation and the Capacity Partition Approach should be tested to see how they perform against Path-Based model and our Column Generation approach. It is anticipated that these two formulations perform better than the Path-Based model in mathematical model solving time; however, one must also note that both these approaches have a set generation procedure that must be performed before the model solving phase. So experimentation on the total time (set generation procedure plus model solving) is necessary to comment on their efficiencies compared to the Path-Based model and the Column Generation approach.

It is also necessary to test the Direct Product Mix Formulation in environments where the uniformity assumption is violated. Leachman and Carmon (1992) propose that the capacity correction can be performed using average values when the uniformity assumption is violated, but they do not provide any computational experiments. Therefore, it may be interesting to observe how much the solution to the Direct Product Mix Formulation deviates from the real optimal solution in production environments where the uniformity assumption is violated.

As mentioned earlier, the Column Generation approach was not implemented to be computationally efficient. To see how it really performs in comparison to the other approaches, it should be improved in terms of computational efficiency. For that purpose, data input-output systems can be made more efficient, subproblems can be solved using shortest path solution algorithms instead of linear programming, and the control loop can be coded using a more efficient language than OPL. Also, the solution time comparison between different solution approaches must be carried out using larger problem instances in order to see clearly how the Column Generation approach performs, and how its solution time changes with increasing problem size.

Lastly, all the formulations presented here should be tested in multi-commodity environments. All four formulations are capable of solving problems involving multiple product types; so it is possible to test them in such environments.

References

Ahuja, R. K., Magnanti, T. L., Orlin, J. B. (1993). *Network Flows : Theory, Algorithms, and Applications.* Englewood Cliffs, N.J.; Prentice Hall.

Alvelos, F., Carvalho, J.M.V. (2007). An Extended Model and a Column Generation Algorithm for the Planar Multicommodity Flow Problem, *Networks, 50*(1), 3-16.

Barnhart, C., Hane, C.A., Johnson, E.L., Sigismondi, G. (1995). A Column Generation and Partitioning Approach for Multi-Commodity Flow Problems, *Telecommunication Systems, 3*(3), 239-258.

Bazaraa, M. S., Jarvis, J. J., Sherali, H. D. (2010). *Linear Programming and Network Flows.* Hoboken, N.J.; John Wiley & Sons.

Bermon, S., Hood, S.J. (1999). Capacity Optimization Planning System (CAPS), *Interfaces, 29*(5), 31-50.

Bertsimas, D., Tsitsiklis, J. N. (1997). *Introduction to Linear Optimization.* Belmont, Mass.; Athena Scientific.

Desaulniers, G., Desrosiers, J., Dumas, Y., Solomon, M.M., Soumis, F. (1997). Daily Aircraft Routing and Scheduling, *Management Science, 43*(6), 841-855.

Farvolden, J.M., Powell, W.B., Lustig, I.J. (1993). A Primal Partitioning Solution for the Arc-Chain Formulation of a Multicommodity Network Flow Problem, *Operations Research, 41*(4), 669-693.

Folie, M., Tiffin, J. (1976). Solution of a Multi-Product Manufacturing and Distribution Problem, *Management Science, 23*(3), 286-296.

Ford Jr., L.R., Fulkerson D.R. (1958). A Suggested Computation for Maximal Multi-Commodity Network Flows, *Management Science, 5*(1), 97-101.

Gale, D. (1960). *The Theory of Linear Economic Models.* New York, McGraw-Hill, 1960.

Hung, Y.F., Cheng, G.J. (2002). Hybrid Capacity Modeling for Alternative Machine Types in Linear Programming Production Planning, *IIE Transactions, 34*(2), 157-165.

Hung, Y. F., Wang Q.Z. (1997). A New Formulation Technique for Alternative Material Planning -- an Approach for Semiconductor Bin Allocation Planning, *Computers and Industrial Engineering, 32*(2), 281-297.

IBM. (2011). *IBM - Solving Optimization Problems - IBM ILOG CPLEX Optimization Studio - Software.* Retrieved 8/8, 2011, from <http://www-01.ibm.com/software/integration/optimization/cplex-optimization-studio/>

Jain, A., Palekar, U.S. (2005). Aggregate Production Planning for a Continuous Reconfigurable Manufacturing Process, *Computers & Operations Research, 32*(5), 1213-1236.

Johnson, L. A., Montgomery, D. C. (1974). *Operations Research in Production Planning, Scheduling, and Inventory Control.* New York, Wiley [1974].

Lasdon, L. S. (1970). *Optimization Theory for Large Systems.* [New York] Macmillan.

Leachman, R.C., Carmon, T.F. (1992). On Capacity Modeling for Production Planning with Alternative Machine Types, *IIE Transactions, 24*(4), 62-72.

Liberopoulos, G. (2002). Production Capacity Modeling of Alternative, Nonidentical, Flexible Machines, *The International Journal of Flexible Manufacturing Systems, 14*(4), 345-359.

Tomlin, J. A. (1966). Minimum-Cost Multicommodity Network Flows, *Operations Research, 14*(1), 45-51.

Venkatadri, U., Srinivasan, A., Montreuil, B., Saraswat, A. (2006). Optimization-Based Decision Support for Order Promising in Supply Chain Networks, *International Journal of Production Economics, 103*(1), 117-130.

Voss, S., Woodruff, D. L. (2003). *Introduction to Computational Optimization Models for Production Planning in a Supply Chain.* Berlin ; New York; Springer.

Appendices

Appendix A

The problem instances to test the proposed methods are generated using Microsoft Excel and VBA. Some parameters are directly entered to the data generation module, and other parameters are calculated within the data generation module, using the entered parameters. The entered parameters are as follows:

1. **Number of Stages**
2. **Number of Resources at each Stage**
3. **Number of Periods**
4. **Capacity:** Capacity is determined to be 1 for each machine at each time period, because the machine capacities are assumed to be one time period.
5. **Demand Change Percentage:** This parameter determines the range (upper bound and lower bound) that is used for generating random demand. The lower bound and upper bound for demand are computed using the formula presented below. Then, the demand values at each period are generated by a uniform random variable with lower and upper bounds $LB(D_t)$ and $UB(D_t)$, respectively.

$$LB(D_t) = (1 - \delta)D$$
$$UB(D_t) = (1 + \delta)D$$

$LB(D_t)$: ***Lower bound for generating random demand***

$UB(D_t)$: ***Upper bound for generating random demand***

δ : Demand change percentage (between zero and one)

D : ***Expected demand***, calculated as cK (a constant times the number of machine types)

6. **Revenue:** This parameter is a constant, and it denotes the revenue when a unit of product is produced and sold.
7. **Holding Cost:** Denotes the holding cost per unit per time period.
8. **Backorder Cost:** Denotes the backorder cost per unit per time period.
9. **Production Cost Multiple:** A constant used for obtaining production costs proportional to the unit revenue and processing times. The basic idea here is that unit production cost for a machine type must be a percentage of average revenue gained at each production stage multiplied by the processing time at that machine. The formula for calculating actual production costs are depicted below.

$$c_k = \frac{\gamma p}{S} a_k$$

c_k : ***Unit production cost*** incurred when an operation is performed on machine k

γ: Production cost multiple

p: Average revenue obtained by selling one unit of product

S: Number of production stages

a_k : ***Capacity consumption (processing time)*** when an operation is performed on machine k

10. Average Utilization: This parameter is defined for each stage and denotes the average target utilization of the machines at each specific stage. The average utilization values are used for computing the average processing times at each stage by the following formula:

$$a_s = N_s C_s u_s$$

a_s : Average processing time of the machines at stage s

N_s : Number of resources at stage s (This parameter is actually N in our experiments, since we held the number of machines constant at each stage.)

C_s : Average available capacity of the machines at stage s, computed by the equation $C_s = \frac{1}{N_s}\sum_{k\epsilon M(s)} C_k$

u_s : Average target utilization of the machines at stage s

11. Process Time Change Percentage: This parameter, which is a number between zero and one, defines how much the processing times fluctuate within each stage. After the mean processing time at stage s is calculated by the formula $a_s = N_s C_s u_s$, ***processing time lower bound*** $LB(a_s)$ and ***upper bound*** $UB(a_s)$ are calculated using this parameter. Then, processing times for each machine at that stage are generated from a uniform random variable with lower bound $LB(a_s)$ and upper bound $UB(a_s)$.

$$LB(a_s) = (1-\alpha)a_s$$
$$UB(a_s) = (1+\alpha)a_s$$

12. Lead Time LB: This parameter is used in generating the lead time for each machine.

13. Lead Time UB: This parameter is used in generating the lead time for each machine from a discrete uniform distribution that can take any integer value between lead time LB and lead time UB with equal probability.

Above, all the entered parameters are listed (in bold font) and all the computed parameters are described (in bold italic font). In addition to those parameters, there is also one other computed parameter:

Initial Inventory: This parameter determines how much inventory there exists in the production system at the beginning of the planning period. The value of this parameter is determined to be expected demand multiplied by the maximum possible total lead time, i.e. the maximum amount of time necessary to produce one product. Initial inventory is calculated by the method described,with the aim of buffering for the starting effect (there is not going to be throughput for a number of periods in the beginning of planning period).