EFFECT OF BUCKET SIZE ON LEAD TIME DISCRETIZATION ERROR IN MULTI-LEVEL PRODUCTION PLANNING SYSTEMS

by

Engin Emir

B.S., Mechanical Engineering, ITU, 2011

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Industrial Engineering

Boğaziçi University

2015

ACKNOWLEDGEMENTS

It is a pleasure to have the opportunity to work with Assoc. Prof. Z. Caner Taşkın as my thesis supervisor. I am very thankful for his patience and his perception in my demoralized times and for his endless support in every struggle that I face indeed. Throughout this study, he boosted me to enhance my mentality as well as motivated me in my down times. I could not afford to complete this study without his luminary advices.

I am also indebted to Assoc. Prof. Ali Tamer Ünal for suggestions with his wise and broad knowledge in thesis subject as well as for participating in my thesis jury. I also would like to express my gratitude to Assist. Prof. Ethem Çanakoğlu for taking part in my thesis jury.

I am grateful to my friends İmren, Murat, and İrem for trusting and encouraging me. I let go all the thoughts, wiped out my blurry mind and charged myself up during the times I spend with them.

It is hard to say the correct words and expressions to reflect my indebtedness and appreciation to my family. Their trust, support and endless love lie behind all my success and achievements. I owe them a life, my life.

I also thank TÜBİTAK for their financial support during my graduate study.

ABSTRACT

EFFECT OF BUCKET SIZE ON LEAD TIME DISCRETIZATION ERROR IN MULTI-LEVEL PRODUCTION PLANNING SYSTEMS

Production planning systems are models that hold and analyze planning and controlling of physical transformation and transferral processes in production systems. These systems consist of various planning models such as Material Requirements Planning (MRP) and Job Shop Scheduling (JSS). These models generally use a discretized time frame to determine replenishment times of items or starting times of operations. To discretize time frame, planning horizon is divided into time buckets such as shifts, days, weeks or months. Length of time buckets may not be equal to Greatest Common Divisor (GCD) of the lead times, and they are typically larger than GCD of the lead times. This phenomenon results in errors in rounding of the lead times of the items or operations. In this thesis, we propose a mathematical model that can be applied to problems that have an acyclic Bill of Materials (BOM) structure, or precedence relations with minimization of sum of all cumulative errors objective. Hence, we aim to make optimal rounding choices of the lead times that result in less errors for all items and less error fluctuations through item chains. We also develop a constructive heuristic for the optimal rounding problem. These approaches are applicable to MRP-type problems but thay are not applicable for JSS type models since rounding of processing times does not give sufficient sequence information for JSS problems. There may occur some ties between starting times of the operations. Thus, we develop a second phase model that consist of a mathematical model and a heuristic. We compare the results of second phase model with JSS model in terms of error introduced and Central Processing Unit (CPU) time of the solutions. Our results show that the model can have good discretization scheme with an acceptable rate of errors for the sake of less CPU times. There is negative correlation between bucket size and solution quality in general.

ÖZET

ÇOK SEVİYELİ ÜRETİM PLANLAMA SİSTEMLERİNDE DÖNEM UZUNLUĞUNUN TEDARİK ZAMANI KESİKLİLEŞTİRME HATASI ÜZERİNE ETKİSİ

Üretim planlama sistemleri, fiziksel dönüşüm ve transfer süreçlerinin planlamasının, kontrolünün ve analizinin yapıldığı üretim sistemleridir. Bu sistemler malzeme ihtiyaç planlaması ve atölye çizelgeleme gibi çeşitli planlama modellerinden oluşur. Bu modeller, üretim yenileme ve operasyon başlangıç zamanlarının belirlenmesi için genellikle kesiklileştirilmiş zaman çerçevesi kullanırlar. Planlama süresi, zaman çerçevesinin kesiklileştirilmesi için vardiya, gün hafta veya ay gibi zaman aralıklarına bölünür. Zaman aralıkları tedarik sürelerinin en büyük ortak katına eşit olmayabilir ve genellikle de ondan büyüktür. Bu olgu tedarik sürelerinin yuvarlanmasında hataya neden olur. Bu tezde, döngüsüz ürün ağacı yapısına yada döngüsüz öncüllük-ardıllık ilişkisine sahip, amaç fonksiyonu kümülatif hataların toplamının minimizasyonu olan problemler için matematik bir model önerilmektedir. Böylece, tedarik zamanlarının yuvarlanmasında daha iyi seçimler yapılmış olup, tüm öğe ve operasyonlarda daha az hata yapılması ve öğe zincirlerinde hata dalgalanmasının azalması sağlanmıştır. Ayrıca, en iyilenmiş yuvarlama problemi için bir yapıcı sezgisel geliştirilmiştir. Fakat, bu yaklaşımlar malzeme ihtiyaç planlaması tipindeki problemlere uygulanabilirken, atölye çizelgeleme tipindeki problemlere uygulanamazlar. İşlem sürelerinin yuvarlanması atölye çizelgeleme problemi için yeterli sıralama bilgisi vermez. Bazı operasyonların başlangıç zamanlarında çakışmalar görülebilir. Bu sebeple, matematik model ve sezgisel içerikli ikinci faz bir model geliştirilmiştir. İkinci faz model ile atölye çizelgeleme modeli, oluşturulan hata ve merkezi işlemci birimi süreleri açısından kıyaslanmıştır. Sonuçlar, oluşturulan modelin iyi bir kesiklileştirme şeması oluşturabildiğine ve azaltılan merkezi işlemci birimi süresi uğruna kabul edilebilir hata oranlarına sahip olunabildiğini göstermiştir. Dönem uzunluğu ile çözüm kalitesi arasında genellikle ters orantı bulunmaktadır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

A	Assigned items list
A_i	Auxiliary variable that holds absolute value in objective function
c^*	Chosen operation to be assigned
C_{max}	Makespan
$C_{max_{opt}}$	Optimum makespan
CE_i	Cumulative error of item i
$D(i)$	Set of successor operation of operation i
EI	Set of lowest level items
FT_i	Finish time of operation i
$FT_{j(i)}$	Finish time of job j that includes operation i
$FT_{m(i)}$	Finish time of machine m that operation i is assigned to
I	Set of items
I_l	Set of items at level l∈L
J	Set of jobs
J_i	The job which operation i belongs to
L	Set of levels
lt_i	Rounded down lead time of item i divided by bucket size
LT_i	Rounding variable of item i
ltp_i	Lead time of item i
M	Set of machines
M_i	The machine which operation i is to be processed on
N_j	Total number of operations of the first j jobs
O	Set of operations
p	Bucket size
p_i	Processing time of operation i
$P(i)$	Set of items that preceeds item i
$PR(i)$	Set of operations that are scheduled by optimization model to start earlier than operation i on the same machine

S	Set of operations that are the first operation of a job
ST_i	Start time of operation i
t_i	The start time of the operation i
T	An upper bound of the makespan
U	An upper bound on CE_i
W_{ij}	Variable that CE_j is the minimum amongst predecessor items of i
x_{ij}	An auxiliary binary variable that holds operation precedence
Y	Set of ready operations
Z_i	Minimum cumulative error of predecessor items of item i
Z^+	Set of positive integer numbers
τ	Factor of how well the solution is

LIST OF ACRONYMS/ABBREVIATIONS

BOM	Bill of Materials
CLSP	Capacitated Lot Sizing Problem
CPU	Central Processing Unit
GCD	Greatest Common Divisor
JSS	Job Shop Scheduling
MIP	Mixed Integer Programming
MPS	Master Production Schedule
MRP	Materials Requirements Planning
MRP-II	Manufacturing Resource Planning
PLSP	Proportional Lot Sizing and Scheduling Problems
RCPSP	Resource Capacitated Project Scheduling Problem
SPT	Shortest Processing Time

1. INTRODUCTION

Production planning systems are software that are used to plan and control physical transformation processes in production systems. Planning is the coordination of a group of activities that required transforming raw materials into finished products to satisfying customer demand in an efficient way [1]. In practice, most production planning systems apply the well-known Manufacturing Resource Planning (MRP-II) logic. MRP-II essentially is a planning system that starts with a Master Production Schedule (MPS) that is built from the demand of end products. Together with inventory records, lot sizing rules, expected lead times, resource requirements and BOM, MRP-II is used to compute the time-phased material requirements at the lower production levels [2]. MRP-II also considers resource requirements of the capacitated resources. Scheduling of the shop floor, which is more detailed, precise and complicated than MRP or MRP-II, is an another aspect of production planning. Using the result data of higher level production planning systems such as MPS, MRP or MRP-II, job assignments to the machines and sequence of the operations can be done with scheduling tools.

Almost all production planning systems including MRP and JSS systems have some decision parameters. Planning horizon length, bucket size, number of frozen periods are considered as some basic decision parameters in these type of production planning systems. Reizebos [3] explains some important design factors of MRP systems. He reasons that choosing the proper time bucket length, offset times, cycle times, and the structuring of the bill of materials could assist to improve overall system performance.

Lead time is one of the important parameters of the production planning systems. Lead time is defined by Pochet [1] as: "The total time needed to complete a procurement or production order, including preparation, administration, waiting, production, quality control and tests, and delivery, and are measured as an integer number of time periods." Performing procurement, production and transportation activities instantaneously at the same time is not possible since the system has lead times. In

conclusion, lead time is a considerable parameter having difficulties in its estimation and implementation in models and has an important effect on production planning systems .

Lead time consists of various and different components such as purchasing, production, transportation, waiting, delivery, etc. These type of lead times are considered for all components in all levels of the BOM structure to create more realistic production plans with fewer errors. Implementation of the lead time into a production planning model has an important effect on production system and has also difficulties in its estimation, determination and calculation. Lead times are typically implemented to the production planning model using time buckets. Bucket size is defined by Reizebos [4] as: "Planning systems use time buckets for time-phasing the co-ordination of several stages of production, during each of which one or more operations are being performed." Lead times and bucket size of the production planning model are highly related with each other and they are couple of parameters that affect the overall production planning system performance in many ways.

Aggregate production planning problem, scheduling problem, multi-site network planning problem and supply chain network problem have some decision parameters such as bucket size as MRP system does. These parameters are not directly related with the problem data, they are parameters that to be chosen appropriately with the problem structure and problem data in the beginning of the problem. Reizebos [5] cited that replanning interval is restricted by the size of time bucket, which also has an effect on internal work order lead times. Bucket size determination is a difficult problem in few aspects. Brandimarte and Villa [6] explain that bucket size determination is one of the challenging task in aggregate production planning systems and proposes several models from the literature to handle this problem. Maravelias and Grossmann [7] show that bucket size is an important design parameter in their short-term scheduling problem. They propose different approaches to Continuous time State-Task Network problem avoiding preventive number of time buckets when bucket sizes is equal to GCD of the processing times.

Production planning systems usually work in discrete time domain. An important issue is how to discretize lead times in continuous time domain into period based lead times in discrete time domain. Bucket size determination is one of the significant sides of the time domain discretization and it is highly related with the discretization process. Discretization of the lead times is strictly dependent on the bucket size. The selection of the bucket size affects the errors in discretization procedure, hence, solution quality of the overall production planning system. Determining the bucket size is a difficult problem since a trade-off exists between accuracy and computational time. Choosing a relatively small bucket size increases the complexity of the problem and sometimes leads to unsolvable models in large scale scheduling models. Lin and Chen [8] point out that considering small time bucket in their multi-site network planning problem is aiming towards having more accurate plans. As a common sense, days, weeks or months are used for bucket size for aggregate planning while hours, shifts or days are used for detailed scheduling purposes.

The difference between small bucket and big bucket is defined by Stadtler [9] as if an operation begins in a time bucket must be completed by the end of the same time bucket, time bucket is called big time bucket. More than one setup will usually take place in a big time bucket. In contrast, small time buckets can preserve the setup state of a resource. In small bucket size models, there can be at most one setup in the same bucket of a resource. Hence, less setup times and costs will incur to the solution of a model where the model has small time buckets compared to big time buckets [9]. Suerie and Stadler [10] state that to have more accurate and errorless plans, smaller bucket size usage has an increasing trend in practice. This trend particularly practiced in the multilevel case where big bucket size models have a handicap of large planned lead times inherently. On the other side, in small-bucket problems the time horizon must be sliced into more buckets to obtain a solution similar to a big bucket model. The problem complexity is increased by this approach in terms of the number of constraints and variables necessary in a Mixed Integer Programming (MIP) model formulation where many production planning systems such as JSS have MIP model formulations.

Choosing a proper bucket size is a challenging task in which a trade-off exists between an errorless plan and computational effort in terms of number of variables and constraints. Choosing an appropriate bucket size means how to discretize lead times and how much error to introduce in model. Discretization is sufficient for MRP type systems where one can use discrete lead times in the model directly to have a utilizable solution. However, in some cases such as JSS, discrete lead times may not be sufficient and may not be used as an appropriate final result. When solving JSS problem only via discrete lead times, the solution does not necessarily map directly with the continuous time space. The main result that can be gathered from discrete solution of the JSS is the starting times of the operations. This data can be used with continuous lead times and blended to come up with a real time continuous space solution. This secondary process is named as continuization in our study. This continuization process also introduces an error to the model.

According to Pochet [1] lead times are measured as an integer number of time periods. Our main aim in this study is to investigate the effect of bucket size choice in environments where lead times are continuous. To this end, we propose a minimum error discretization and continuization procedure with or without an integer bucket size. Our proposed discretization procedure is designed in a way that it can be applied to models that have an acyclic BOM or sequence structure as in MRP, classical JSS problem or resource capacitated project scheduling problem (RCPSP). On the other hand, continuization procedure is designed in a way that to work only with the JSS problem.

The rest of the thesis is organized as follows: In Chapter 2, we give a detailed definition of the investigated problem and discuss some related problems. In Chapter 3, a literature review of the topic is given. Available literature is investigated and analyzed from the view of studied bucket size determination and lead time implementations. In Chapter 4, we construct a MIP formulation of the discretization procedure and construct a corresponding heuristic for MIP formulation. To deal with continuization issue, we construct a final sequencing heuristic in Section 4.2. In Chapter 5, we present the results of our proposed methodology. Finally, we provide conclusions and

suggestions for further research in Chapter 6.

2. PROBLEM DEFINITION

Lead time always becomes one of the important parameters of any production planning system, and it is commonly modified to the discretized model world. Time space discretization is a difficult issue when transforming real world continuous times to the discretized model world. When discretization exist in such production planning systems mentioned in Chapter 1, time bucket size determination is an essential and helpful instrument. If a production planning model that introduces no discretization error is desired to be built, one can choose a bucket size that is equal to GCD of the lead times. However, this may lead to an excessive number variables and constraints in the underlying optimization model and the model may become unsolvable. On the other hand, choosing a bucket size that is larger than the GCD of the lead times results in a more compact optimization model at the expense of introducing discretization error. It is known that we cannot introduce such a small bucket size because of the statements mentioned above. There can be a way of achieving this as opposed to Pochet [1] where discrete lead time errors can cancel each other which can lead to errorless discretization process where lead times are bigger than the GCD of lead times.

Table 2.1. Data of discretization example.

		Discretized Lead Time [day]		
Tasks	Lead Time [day]	Rounded up to a week	Rounded to a week	Rounded down to a week
1	7	7	7	7
2	6	7	7	0
3	3	7	0	0
4	8	14	7	7
Sum	24	35	21	14
Error	0	+11	-3	-10

To illustrate the lead time discretization error problem, the following example may become useful. Consider a single-chain project scheduling problem that only consist of four successor tasks. Suppose that the bucket size is chosen to be seven days (a week) and the tasks have related lead times in days that are shown in Table 2.1. The scheduling is illustrated on a GANTT Chart. The GANTT Chart of the sample project

that uses continuous lead times is shown in Figure 2.1 where there is no discretization error since one day bucket size is chosen, which is less than or equal to GCD of the lead times, and the resulting project completion time is 24 days. When there is a bucket size larger than GCD of lead times of the items, let us consider three simple methods which are rounding, rounding up and rounding down. We will investigate more complex methods in the following chapters, but these three are sufficient to illustrate lead time discretization problem and error occurrence

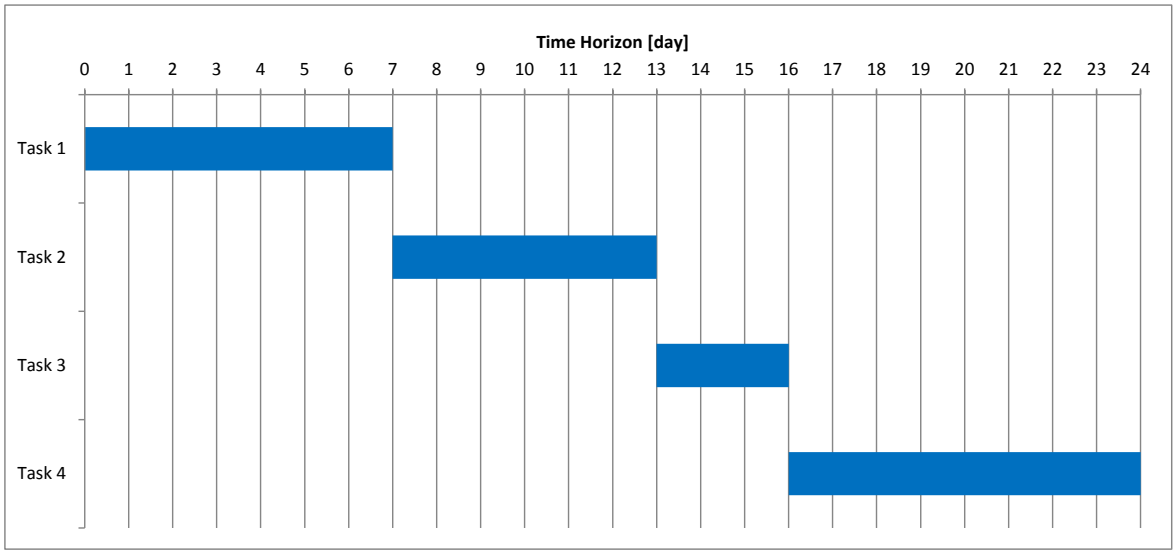

Figure 2.1. GANTT Chart of tasks when bucket size is a day.

In Figure 2.2, bucket size is seven days and we rounded up the lead times to the bucket size of seven or its integer multiples. First task remains as it is, lead time of Task 2 is rounded up to seven days, Task 3 is rounded up to seven days and finally Task 4 is rounded up to 14 days. The resulting completion time is 35 days, and 11 days error compared to the one day bucket size. This error is positive that means rounding up procedure is over-estimating.

Rounding can be thought as second alternative which is shown in Figure 2.3. First task remains as it is, Task 2 is rounded to seven days, Task 3 is rounded to zero and Task 4 is rounded to seven days. It has 21 days of completion time and three days error compared to the one day bucket size which is negative that means rounding is under-estimating.

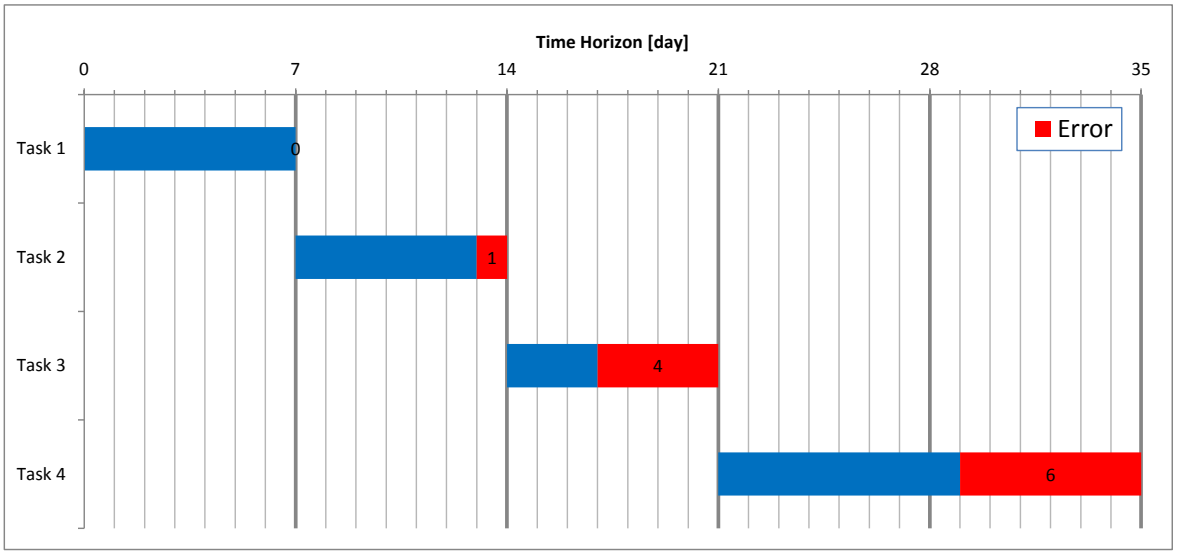

Figure 2.2. GANTT Chart of tasks when bucket size is seven days and lead times are rounded up.

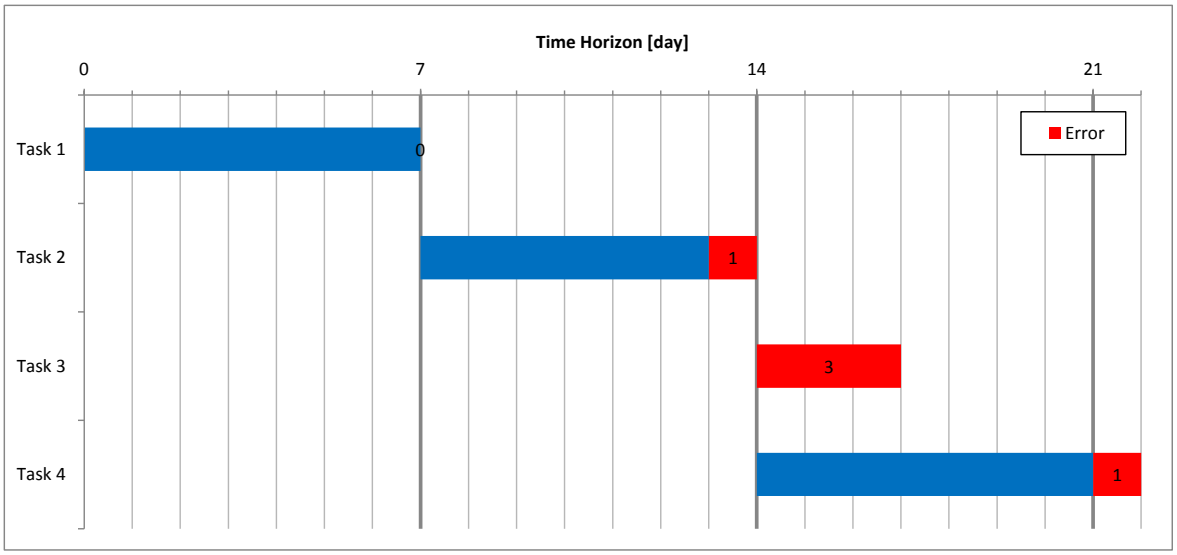

Figure 2.3. GANTT Chart of tasks when bucket size is seven days and lead times are rounded.

Rounding down can be thought as a third alternative where related GANNT chart is shown in Figure 2.4. First task remains as it is as seven days, Task 2 is rounded down to zero days, Task 3 is rounded down to zero and Task 4 is rounded down to seven days. It has 14 days of completion time and 10 days error compared to the one day bucket size which is negative that means rounding is under-estimating.

In the light of these examples, lead time discretization is a problem that a proper discretization procedure need to be implemented in besides choosing a appropriate bucket size. Over-estimating and under-estimating are the key decision parameter of the discretization process. Besides over-estimating or under-estimating the whole process, estimating the individual errors of the tasks is also an important factor changing from implementation to implementation. For instance, total of the absolute errors of the individual items is 11 for rounding up, five for rounding and 10 for rounding down. So, fluctuations of errors of individual items are important as well as makespan of the instance.

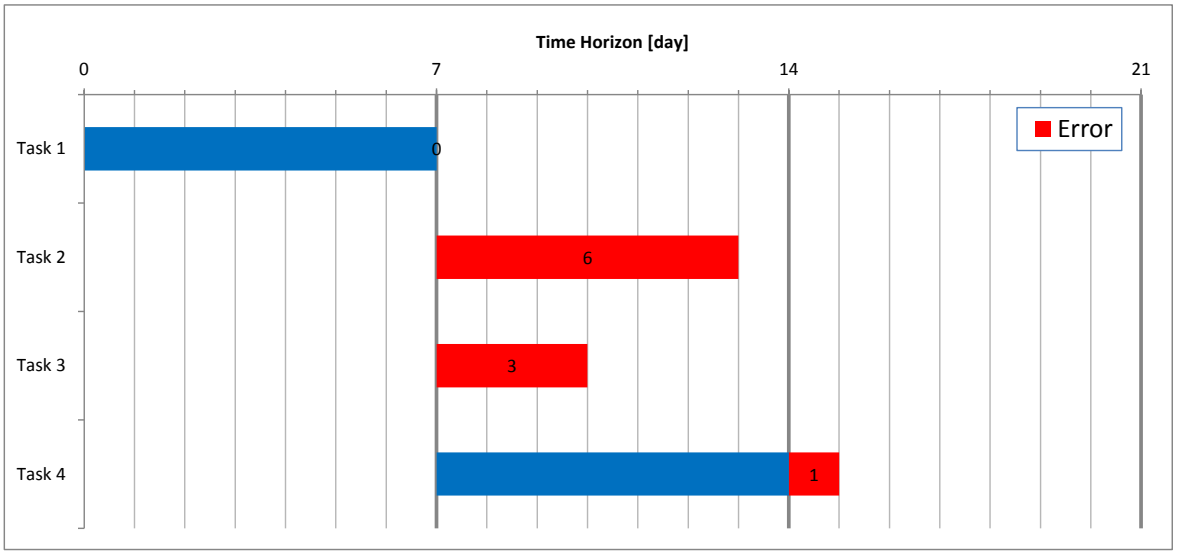

Figure 2.4. GANTT Chart of tasks when bucket size is seven days and lead times are rounded down.

Another important issue beside the discretization process is converting discrete lead times to utilizable continuous lead times for some of the implementations such as JSS problem. In MRP type problems, discrete lead times can be used directly since the sequence of the items is pre-defined, the only thing is to determine the lead times of the items. However, this is not the case where JSS takes place. In JSS problem, sequences of the operations of different jobs are not pre-defined. The makespan of the problem may not be evaluated with the discrete lead times realistically. The only data that can be gathered from discretization process is discrete lead times and these are not sufficient since they do not imply sequence of the operations. To determine the sequence of the operations, a JSS model can be used with the discrete lead times. The

results of the JSS model give us the starting time of the operations. The sequence can be organized with continuous lead times to achieve a real continuous and applicable to the real world schedule and a makespan.

3. LITERATURE REVIEW

Determination of bucket size is a difficult problem as explained in the Chapter 2. Several researchers tend to develop different types of approaches to this problem. Every approach has its own drawbacks and powerful aspects. The below three models incorporates properties of the small bucket size and big bucket size. Lin and Chen [8] proposed a monolithic multi-site supply network planning model that uses variable bucket sizes. The model uses narrower bucket size for closer periods that are more detailed and wide bucket size for further periods. It divides the first month to days and other months are bucketed as months. They show that their model is superior to the hierarchical model and combines the properties of big-bucket and small-bucket models with less computational effort. The included time scheme can be seen in Figure 3.1.

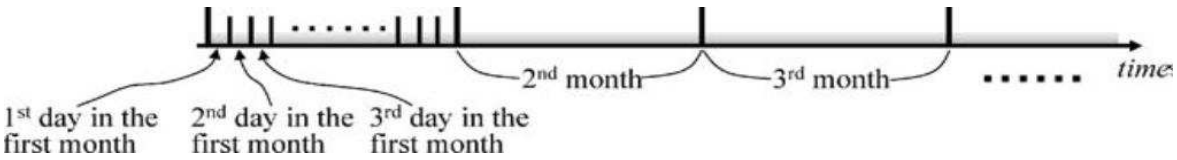

Figure 3.1. Two different time bucket lengths in a single optimization model [8].

A similar approached used in research by Timpe and Kallrath [11]. Their model includes small and big bucket sizes for different types of activities. Big bucket size is used for marketing aspects that is called commercial time scale where marketing is less exact in terms of forecasting and small bucket sizes is used for production related activities. Hence, there are two period indices for decision variables. The model divides the planning horizon into commercial time scales and each commercial time scale is again divided into production time scales where finer production time scales is used at the beginning of the planning horizon for sales and distribution purposes. The introduced time scale is shown in Figure 3.2.

Another approach is presented by Suerie and Stadler [10] that incorporates properties of the small and big bucket sized models. Big bucket sized models allow more

than one setup state but do not carry the setup state to the next period which are similar to capacitated lot sizing problem (CLSP). In contrast, small bucket size allows at most one setup state per period and allow carrying the setup state to the next periods which are actually proportional lot sizing and scheduling problems (PLSP). The proposed model in [10] includes a characteristic of small bucket sized models that is the carrying setup states to the next period while protecting its big bucket sized properties. They conclude that the model yields tight lower bound as well as good primal solutions with respect to some test instances. This type of approaches reduce the number of variables compared to the small bucket sized models, but they introduce new coupling constraints to tie different time scales. Chou and Hong [12] conclude that

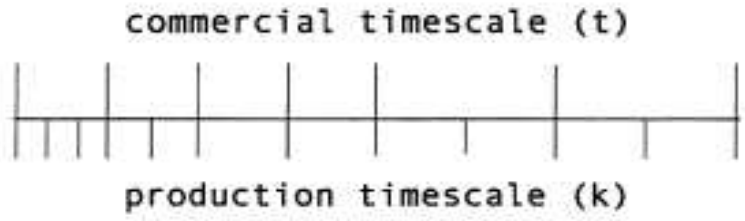

Figure 3.2. Two time scales for different purposes for the same horizon [11].

bucket size determination is important and it affects the machine workload variation in product mix planning in semiconductor foundry manufacturing. They argue that work release frequency should be at the same scale as the model bucket size.

The re-planning frequency and internal work order lead times are affected and constrained by the length of the time bucket size. Riezebos [13] considers effect of bucket size as a design parameter on the performance of MRP systems with lot splitting approach. The research is focused on the effect of bucket size for different number of sub-batches. He proposes a nonlinear model that optimizes the total cost by solving the bucket size problem and lot splitting decisions simultaneously. The conclusion in this research is that bucket length for one sub-batch increases the total cost, but this does not hold for lot splitting models. In these models, the cost decreases down to some point and increases respectively with the increasing bucket size. This last phenomenon is not taken into account since we do not consider lot splitting in our model.

An alternative approach proposed by Maravelias and Grossmann [7] is used in state-task network formulation. They give a general continuous time formulation for short term scheduling. They argue that the discrete time representations can only be used when the bucket size is equal to greatest common factor of the processing times to have a schedule without errors. To meet all processing times, bucket size may become so small that makes the model unrealistic and unsolvable. They propose a continuous time representation where the time horizon is split into time intervals of unequal and unknown sizes. Stadtler [9] argued that this model will conclude to more computational burden, although this is the most accurate and precise model can be built.

While the problem of dividing the time horizon into time buckets have been investigated from different approaches, to the best of our knowledge there is no study on the literature that focused on the lead time discretization, bucket size selection or error estimation when bucket size is bigger than GCD of the lead times.

4. PROPOSED METHODOLOGY

The aim of this study is to investigate problems mentioned in Chapters 2 and 3, namely, determining a proper bucket size to obtain reasonable solution times for the scheduling problems and to utilize more usable lead times for MRP type problems at the expense of introduced error. Our proposed approach can be used in many problem types that contain multiple levels such as MRP, JSS, RCPSP. The classical JSS problem, which is NP-hard, is chosen for the test of discretization process.

The proposed methodology is consist of three main processes which are discretization, solving and continuization. These three main sections of the proposed methodology can be seen in Figure 4.1. In the following sections these will be explained in detail.

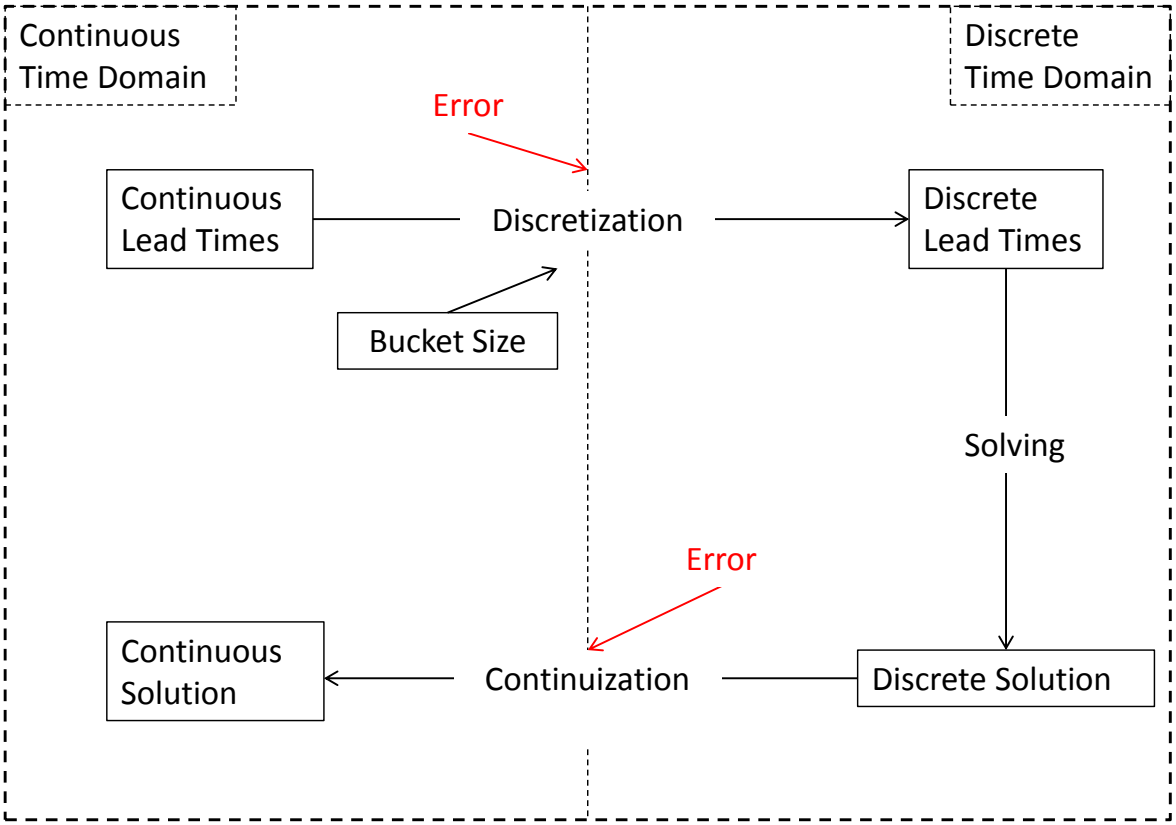

Figure 4.1. Scheme of proposed methodology.

4.1. Discretization Procedure

Finding an errorless discretization scheme is a difficult problem unless GCD of the lead times is chosen as bucket size. This selection increases the number of variables and constraints and makes the model complicated and computationally difficult in terms of CPU time. Furthermore, such an errorless and detailed model may not be needed in practice.

In this part of our study, we aim to discretize continuous lead times or processing times with the minimum desired error function. Some restricting constraints are crucial not to introduce peak or highly fluctuating errors for the individual BOM items as well as minimizing overall error with the given objective function. So, a mathematical model that minimizes a given error function will be constructed in Section 4.1.1 and this will be explanatory for the discretization process and a following discretization heuristic in Section 4.1.2 will be explained in detail.

4.1.1. Discretization Model

The basic function of the discretization model is to round up or down the lead times of the related items to the integer multiples of the bucket size. To do this, first, all lead times are pre-processed such that they are rounded down to the nearest integer multiple of the bucket size. These pre-processed lead times are lt_i's. then, the model decides whether to round up or holds rounded down with binary variables LT_i. Besides these, there are some constraints that restrict the cumulative errors of the items.

Cumulative error term is the backbone of our model as well as the heuristic. Cumulative error is an error term that related with the error of item itself and the error of predecessor items. To tie predecessor-successor relationship in error term, cumulative error term is a crucial term that fits well for the problems that include BOM structure or predecessor-successor relationship that do not circulate such as JSS, MRP, RCPSP. Cumulative error term is also beneficial to control the error fluctuations or peak errors of the whole item chains by adding some restrictive constraints to the

model for cumulative error. In the light of these, we developed a MIP model (4.1)-(4.10) as follows:

Sets:

- I: Set of items
- EI: Set of lowest level items
- $P(i)$: Set of items that precedes item i

Variables:

- LT_i : $\begin{cases} 1 & \text{if the lead time of item } i \text{ is rounded up} \\ 0 & \text{o/w} \end{cases}$
- W_{ij} : $\begin{cases} 1 & \text{if the predecessor item } j \text{ of item } i \text{ has the minimum cumulative error among all other predecessor items of item } i \\ 0 & o/w \end{cases}$
- CE_i : Cumulative error of an item i
- Z_i : Minimum of the cumulative errors of the preceeding items i

Parameters:

- ltp_i: Lead time of item i
- p: Bucket size
- $lt_i : \lfloor \frac{ltp_i}{p} \rfloor$
- U: An upper bound on CE_i

Model formulation is:

$$\min \quad \sum_{i \in I} \frac{|ltp_i - (lt_i + LT_i).p|}{ltp_i.|I|} \tag{4.1}$$

$$\text{s.t.} \quad CE_i = (lt_i + LT_i).p - ltp_i + Z_i \qquad \forall i \in I \tag{4.2}$$

$$CE_i \leq p \qquad \forall i \in I \tag{4.3}$$

$$CE_i \geq -p \qquad \forall i \in I \tag{4.4}$$

$$CE_j \geq Z_i \qquad \forall i \in I \qquad \forall j \in P(i) \tag{4.5}$$

$$Z_i \geq CE_j - (1 - W_{ij}).U \qquad \forall i \in I \qquad \forall j \in P(i) \tag{4.6}$$

$$\sum_{j \in P(i)} W_{ij} \geq 1 \qquad \forall i \in I \tag{4.7}$$

$$Z_i = 0 \qquad \forall i \in EI \tag{4.8}$$

$$LT_i \in \{0, 1\} \qquad \forall i \in I \tag{4.9}$$

$$W_{ij} \in \{0, 1\} \qquad \forall i \in I \qquad \forall j \in P(i) \tag{4.10}$$

(4.1) is the objective function that minimizes average absolute lead times errors. (4.2) represents the definition of cumulative error. The term $(lt_i + LT_i).p$ is discrete lead time, ltp_i is continuous lead time and Z_i is the minimum cumulative error of lower level items. (4.3) and (4.4) limit the cumulative error not to exceed predefined bucket size. (4.5) and (4.6) ensure choosing minimum cumulative error of the predecessor items of item i, Z_i is the minimum of cumulative errors of the predecessor items. (4.5) satisfies that minimum cumulative error of the predecessor items is less than every cumulative error of the predecessor items. (4.6) satisfies that if W_{ij} is one, then, the specific predecessor item j has the minimum cumulative error amongst other predecessors, 0 o/w and becomes redundant. (4.7) ensures that there must be at least one item that has minimum cumulative error amongst predecessor items of item i. (4.8) satisfies that lowest level items has no predecessor items hence its Z_i must be equal to zero. The big-M type constant U in (4.6) is two times of the bucket size p since CE_i has bounds (4.3) and (4.4). Cumulative error of an item is the sum of the error of that item and the minimum cumulative error of predecessor items of that item. Cumulative errors of lowest level items equal to only the error of that item.

The model is not linear with this structure of the objective function. The absolute value term in the objective function needs to be linearized. To do this, we can introduce a continuous variable that represents the absolute value and two auxiliary constraints. The term $|ltp_i - (lt_i + LT_i).p|$ is represented by A_i and following two additional constraints are introduced:

$$A_i \geq ltp_i - (LT_i + lt_i).p \qquad \forall i \in I$$
$$A_i \geq -ltp_i + (LT_i + lt_i).p \qquad \forall i \in I$$

The model becomes:

$$\min \quad \sum_{i \in I} \frac{A_i}{ltp_i.|I|} \tag{4.11}$$

$$\text{s.t.} \quad CE_i = (lt_i + LT_i).p - ltp_i + Z_i \qquad \forall i \in I \tag{4.12}$$

$$CE_i \leq p \qquad \forall i \in I \tag{4.13}$$

$$CE_i \geq -p \qquad \forall i \in I \tag{4.14}$$

$$CE_j \geq Z_i \qquad \forall i \in I \qquad \forall j \in P(i) \tag{4.15}$$

$$Z_i \geq CE_j - (1 - W_{ij}).M \qquad \forall i \in I \qquad \forall j \in P(i) \tag{4.16}$$

$$\sum_{j \in P(i)} W_{ij} \geq 1 \qquad \forall i \in I \tag{4.17}$$

$$Z_i = 0 \qquad \forall i \in EI \tag{4.18}$$

$$A_i \geq ltp_i - (LT_i + lt_i).p \qquad \forall i \in I \tag{4.19}$$

$$A_i \geq -ltp_i + (LT_i + lt_i).p \qquad \forall i \in I \tag{4.20}$$

$$LT_i \in \{0, 1\} \qquad \forall i \in I \tag{4.21}$$

$$W_{ij} \in \{0, 1\} \qquad \forall i \in I \qquad \forall j \in P(i) \tag{4.22}$$

4.1.2. Discretization Heuristic

The mathematical discretization model is a MIP model so solution times may become larger for large-scale problems. To handle large solution times and to find a

quite simple way of discretization, it is needed to develop a constructive heuristic. In this part of the study, a heuristic that can handle the properties of the model (4.11)–(4.22) is developed and a similar objective function that can handle the properties of (4.1) is sought since minimization of sum of lead time errors may not be applicable in a heuristic properly. So, an objective function that consist of minimization of cumulative errors of all items is tried to be incorporated. Minimization of errors of highest level items especially end items is another choice but it is not beneficial since it causes fluctuating cumulative errors for lower level items, which is undesired. The following Figure 4.2 is the pseudo-code of our discretization heuristic.

The heuristic works likes in following way and explained step by step. In steps 2-5, initial values of all variables are set to zero. Step 6 ensures that each and every item is processed and if all are processed then the algorithm stops. L is the set of levels. I_l is the set of items at level l. In steps 9-11, if an item has no predecessor, its Z_i variable stay as zero and then cumulative error of the item is calculated with set values of Z_i and LT_i. If cumulative error of the item is less than half of the chosen period length, the item is placed in assigned items list A in steps 12-14, else its LT_i is set to one, item is placed in list A and its cumulative error is calculated again since its LT_i is set to one from zero in steps 15-18. In steps 19-22 if an item has at least one predecessor , its Z_i is set to minimum cumulative error of predecessor items of that item. Its LT_i is set to zero and its cumulative error is calculated. In steps 23-24, If cumulative error of that item is less than half of the period length, it is placed in list A. If its cumulative error is greater than half of the period length, its LT_i is set to one and its cumulative error is calculated again with new LT_i value and it is placed in list A in step 25-28.

In conclusion, disceritization heuristic works in a way that cumulative errors of the individual items starting from the lowest level items are minimized. This heuristic is down to up type but its reverse can also be tried that starts from the end items. Starting from the lowest level items, their cumulative error is calculated. If pre-calculated cumulative error of one item is lower than half of the bucket size, its rounding variable is set to zero, or to one otherwise. This continues for the next higher level items. All

other calculations are similar to mathematical model from evaluation of the cumulative error to assignment of the Z_i's.

4.2. Continuization

The second part of the proposed methodology is continuization process as shown in Figure 4.1. The aim in this section is to convert discrete lead times or processing times into utilizable continuous lead times. This section consist of two procedures: Scheduling and Final sequencing. The processing times or lead times taken from the discretization process are not directly applicable in the sense of real world continuous time analysis. So, to be able to use the discretized processing times, these processing times are used in a scheduling model that gives a crude sequencing of the operations that actually give start time of the jobs but this crude information is also not adequate since it uses discrete processing times. There occurs some tie between starting times of the operations. Suppose that, both of successor operations in a specific job is started at the same time due to under estimating of one of the processing time of an operation as zero. In this case, a tie occurs, which is solved with heuristic explained in Section 4.2.2.

4.2.1. Scheduling Model

In this part of the methodology, incorporating the processing times that are discretized in Section 4.1 is explained. A classical JSS model will be used to get finishing times of the operations to be used in Section 4.2.2 for final sequencing. According to Seda [14], the classical JSS problem can be defined as follows: Each job has its own sequence of operations that each operation has to be processed on different machine with given processing times with no preemption. Machines are allowed only one operation at the same time. A schedule is the sequence of operations on the machines related with the above rules. The mathematical model (4.23)-(4.31) of the classical JSS is described below as defined in [14]:

```
L is the set of levels
I_l is the set of items at level l ∈ L
1. A ← ∅ {assigned items list}
2.      foreach(i in I)
3.          Z_i ← 0
4.          LT_i ← 0
5.          CE_i ← 0
6. while(A ≠ I)
7.      foreach(l in L)
8.          foreach(i in I_l)
9.           if   P(i)= ∅
10.                  Z_i ← 0
11.                  CE_i = lt_i.p − ltp_i
12.                 if  |CE_i|<p/2
13.                      LT_i ← 0
14.                      A ← A ∪ {i}
15.                 else
16.                      LT_i ← 1
17.                      CE_i = (lt_i + 1).p − ltp_i
18.                      A ← A ∪ {i}
19.             else
20.                  Z_i ← min_{k∈P(i)} CE_k
21.                  CE_i = lt_i.p − ltp_i + Z_i
22.                  if  |CE_i|<p/2
23.                      LT_i ← 0
24.                      A ← A ∪ {i}
25.                  else
26.                       LT_i ← 1
27.                       CE_i = (lt_i + 1).p − ltp_i + Z_i
28.                       A ← A ∪ {i}
```

Figure 4.2. Pseudo code of discretization algorithm.

Sets:

- J: Set of jobs
- O: Set of operations
- M: Set of machines
- J_i: The job which operation i belongs to
- M_i: The machine which operation i is to be processed on
- N_j: Total number of operations of the first j jobs

Variables:

- C_{max}: Makespan
- t_i: The start time for the operation i
- x_{ij}: $\begin{cases} 1, & t_j \geq t_i + p_i, \text{ if operation } i \text{ precedes operation } j \\ 0, & t_i \geq t_j + p_j \end{cases}$

Parameters:

- p_i: Processing time of operation i
- T: An upper bound of the makespan

Model formulation:

$$\min \quad C_{max} \tag{4.23}$$

$$\text{s.t.} \quad t_i \geq 0 \qquad \forall i \in O \tag{4.24}$$

$$t_{i+1} \geq t_i + p_i \qquad \forall j \in J \qquad N_{j-1} + 1 \leq i \leq N_j - 1 \tag{4.25}$$

$$C_{max} \geq t_{N_j} + p_{N_j} \qquad \forall j \in J \tag{4.26}$$

$$t_j \geq t_i + p_i x_{ij} - T(1 - x_{ij}) \qquad \forall i, j \in O \qquad i \neq j, \quad M_i = M_j \tag{4.27}$$

$$t_i \geq t_j + p_j(1 - x_{ij}) - T x_{ij} \qquad \forall i, j \in O \qquad i \neq j, \quad M_i = M_j \tag{4.28}$$

$$C_{max} \in Z^+ \tag{4.29}$$

$$t_i \in Z^+ \qquad \forall i \tag{4.30}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in O \qquad i \neq j, \quad M_i = M_j \tag{4.31}$$

(4.23) is well-known makespan objective, which is defined as "Maximum of the completion times of the last operations in the jobs"[14]. (4.24) ensures every start time of the operations is greater than or equal to 0. The condition that an operation that belongs to a job must not start before the predecessor operation finishes is ensured by (4.25). (4.26) is the definition of the makespan. The makespan is defined as the maximum of the completion times of the last operations of each jobs. There is no need for an upper bound constraint for the makespan since objective function only includes minimization of the makespan. (4.27)-(4.28) constraints ensure machine capacity restrictions that at most one operation can be processed on each machine at a time [14]. (4.29)-(4.30) constraints are integrality constraints for time variables. Finally, (4.31) is the binary constraint for auxilary variable x_{ij}, which provides either-or type constraint for machine capacity constraints set.

Note that p_i may refer to continuous lead times or rounded lead times. In case of rounding, the only data gathered from this classical JSS model is the starting times of the operations. If some processing times are rounded down to zero, we may not be able to obtain an exact sequence or predecessor-successor relationships from this model since starting times ties can occur. To solve these ties, starting times of the operations will be used in final sequencing part for continuzation of the schedule.

4.2.2. Final Sequencing

Up to now, processing times are discretized with discretization model or discretization heuristic. These discrete times are used in the scheduling model to get starting times of the operations. The results of the scheduling model is created with discrete processing times. This is not so meaningful since there is no connection with the real continuous processing times. The useful data in scheduling results is not the sequence of items since it can start immediate predecessor items at the same time because of one of the processing time of the items may become discretized to zero and it is not appropriate with real life situation to start both immediate predecessor items at the same time.

O *is the set of operations*

A *is the set of assigned operations*

Y *is the set of ready operations to be processed*

S *is the set of operations that are the first operation of a job*

PR(i) *is the set of operations that are scheduled by optimization model to start earlier than operation i on the same machine*

P(i) *is the set of predecessor operation of operation i*

D(i) *is the set of successor operation of operation i*

$FT_{m(i)}$ *is the finish time of machine m that operation i is assigned to*

$FT_{j(i)}$ *is the finish time of job j that includes operation i*

ST_i *is the start time of operation i*

FT_i *is the finish time of operation i*

c^* *is the chosen operation to be assigned*

1. $\boldsymbol{A} \leftarrow \emptyset$
2. $\boldsymbol{Y} \leftarrow \emptyset$
3. $\forall i,\ FT_{m(i)} \leftarrow 0$
4. $\forall i,\ FT_{j(i)} \leftarrow 0$
5. $\forall i,\ FT_i \leftarrow 0$
6. $\forall i,\ ST_i \leftarrow 0$
7. $\boldsymbol{Y} \leftarrow \{i:\ i \in \boldsymbol{S}\ \vee (i:\ i \in \boldsymbol{O} \wedge \boldsymbol{PR(i)}{=}\emptyset)\}$
8. **while(** $\boldsymbol{Y} \neq \emptyset$**)**
9. **if** $\exists c : c \in \boldsymbol{Y},\ \boldsymbol{PR(c)} = \emptyset\ \wedge\ c \in \boldsymbol{S}$
10. $c^* \leftarrow \underset{c \in \boldsymbol{Y},\ \boldsymbol{PR(c)}=\emptyset\ \wedge\ c \in \boldsymbol{S}}{\operatorname{argmin}} p_c$
11. **else**
12. $c^* \leftarrow \underset{c \in \boldsymbol{Y},\ \forall \boldsymbol{PR(c)} \in \boldsymbol{A}}{\operatorname{argmin}} p_c$
13. $\boldsymbol{A} \leftarrow \boldsymbol{A} \cup \{c^*\}$
14. $ST_{c^*} \leftarrow max\{FT_{m(c^*)}, FT_{j(c^*)}\}$
15. $FT_{c^*} \leftarrow ST_{c^*} + p_{c^*}$
16. $FT_{m(c^*)} \leftarrow FT_{c^*}$
17. $FT_{j(c^*)} \leftarrow FT_{c^*}$
18. $\boldsymbol{Y} \leftarrow \boldsymbol{Y} \setminus \{c^*\}$
19. $\boldsymbol{Y} \leftarrow \boldsymbol{Y} \cup \{\boldsymbol{D}(c^*)\}$

Figure 4.3. Pseudo code of final sequencing algorithm.

To overcome this problem, a heuristic that uses scheduling model results, which are starting times of the each item or operation, blended with decision makings where a tie occurs with the discretized data is developed. The sequencing heuristic will become a balance between exact scheduling model and sequencing heuristic that consists of some choosing criteria. The heuristic is close the exact scheduling model solution when bucket size is considerably small with respect to processing times. The heuristic is exactly same with the scheduling model when bucket size is equal to GCD of the processing times. On the other hand, if bucket size is set to the makespan, actually makespan is not known at that time, final sequencing only uses the heuristic and uses no starting time of operations coming from the scheduling data with discretized processing times. The pseudo-code of the heuristic is shown in Figure 4.3.

The final sequencing heuristic works in following way and explained step by step. In steps 1-2, assigned operations set A and ready operations set Y are initialized and empty. In steps 3-4, finish time of each machine and job are set to zero.Finish time and start time of each operation are set to zero in steps 5-6. In step 7, each operation is controlled to be placed in ready operations Set Y. If an operation is a starting operation of a job or has no earlier operation on its machine, it is placed in Set Y. Step 8 ensures that each and every operation is processed and if there is no ready operations to be processed then the algorithm stops. A Top priority operation is chosen from Set Y between steps 9-10. If there exists an operation c in Set Y such that it has no earlier operation on its machine, it is a starting operation then an operation c^* is chosen from Y set with minimum processing time. Else if, an operation c^* is chosen from Set Y among operations, whose all earlier operations on its machine is assigned, with minimum processing time in steps 11-12. This chosen operation c^* is placed in assigned operation Set A in step 13. In steps 15-19, we assign the start and finish time of operations, jobs and machines. Start time of operation c^* is equal to maximum of finish time of machine that operates on c^* and finish time of job that includes operation c^*. Finish time of the operation $c^*$is equal to its start time plus its processing time. Finish time of machine that operates on machine c^* and finish time of job that includes operation c^* is equal to finish time of operation c^*. Finally, in steps 18-19, assigned operation c^* is discard from Set Y and its successor operation is added to Set Y.

When a tie occurs, start time of two or more job that uses same machine, there must be some dispatching rules that breaks the tie. We use a simple single level dispatching rule that is shortest processing time (SPT). There is no well known, good dispatching rule for JSS, so intuitively SPT that is used for parallel machine scheduling problem is used.

4.3. Overall Algorithm

To gather all steps in an algorithm, we have an overall algorithm definition. The overall algorithm is consist of discretization procedure as explained in Section 4.1.1, scheduling model as in Section 4.2.1 and final sequencing heuristic as in Section 4.2.2. The overall algorithm is explained in Figure 4.4.

1. Run the discretization model (4.11)-(4.22) and get discrete processing times $(lt_i + LT_i).p$.
2. Run the scheduling model (4.11)-(4.22) using discrete processing times and get starting times of the operations t_i's from scheduling model (4.11)-(4.22).
3. Run the final sequencing algorithm Figure 4.3 using starting times.

Figure 4.4. Overall algorithm.

5. RESULTS

5.1. Comparison of Discretization Heuristic and Model

The following explanatory example reveals the difference between our proposed heuristic in Section 4.1.2 and the model described in Section 4.1.1.

Table 5.1. Data of example MRP BOM.

Item ID	Predecessor Items	Lead Time	Error in Heuristic	Cumulative Error in Heuristic	Error in Model	Cumulative Error in Model
1	2,3	5	+2	-1	+2	+0
2	4,5	5	-3	-3	+2	-2
3	6	4	-4	-1	-4	-1
4	-	5	+2	+2	+2	+2
5	-	4	+3	+3	-4	-4
6	7,8	2	+5	+3	+5	+3
7	9,10	3	+4	+2	+4	+2
8	-	2	-2	-2	-2	-2
9	-	6	+1	+1	+1	+1
10	-	2	-2	-2	-2	-2
Total				20		19

The longest chain of items which is 1-3-6-7-9 is 20 days as can be seen from Table 5.1 and Figure 5.1. If bucket size is chosen as seven, one may expect 21 days to complete all items or operations by some rounding. However, this is not the case since objective functions of our model and heuristic are based on minimizing sum of all cumulative errors; not the minimize longest item chain. In principle, designing a model that has an objective function that minimizes the longest item chain length is possible. However, in this case, rounding of other items except the longest chain items could yield unexpected results.

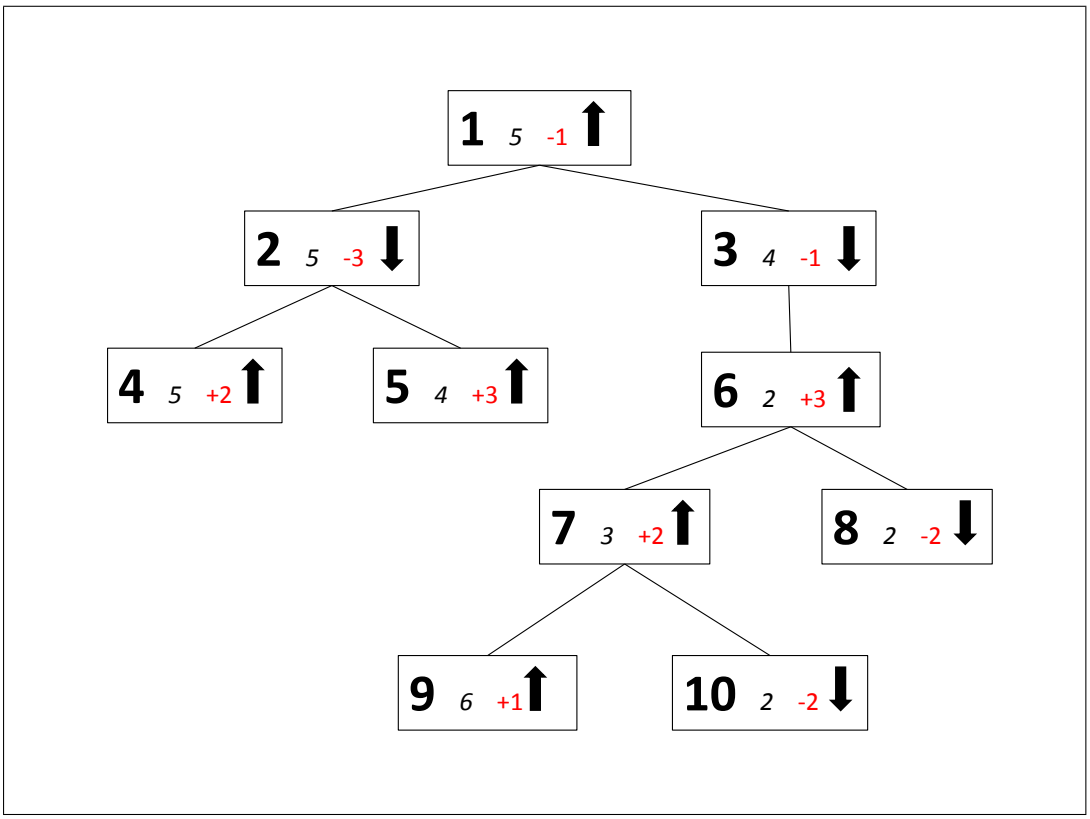

Figure 5.1. Discretization scheme for discretization heuristic Figure 4.2.

Figure 5.1 shows the discretization scheme for the heuristic. Each box represents the items in BOM structure. The number written in bold face is the item ID. The lead times of items are given in Table 5.1 and written italic in item boxes in Figure 5.1. The number in red with a plus or minus sign is the calculated cumulative error of that item. Arrows represent the rounding of processing time that either rounded up or down. Total cumulative error for this case is 20. Figure 5.2 shows the cumulative errors for use of mathematical model (4.2)-(4.10). Mathematical model yields an optimal cumulative error of 19. This decrease in cumulative error originates from rounding of item five that is rounded down causing -4 cumulative error which discretization heuristic does not allow since its absolute value is greater than half of the period length. The rounding heuristic cannot foresee the errors of higher level items whether it may exceed half of the bucket size or not, which results in higher total cumulative errors. This phenomenon is an handicap for the heuristic.

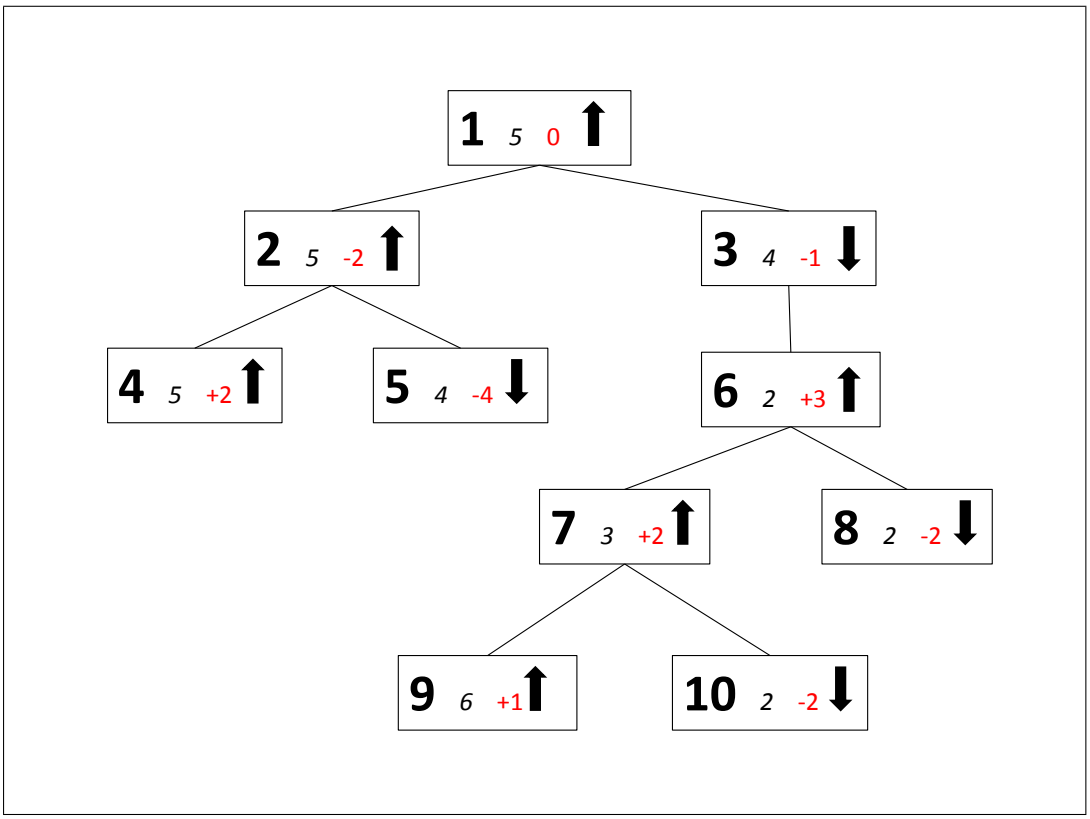

Figure 5.2. Discretization scheme for mathematical model (4.11)-(4.22).

5.1.1. Comparison of Discretization Heuristic and Model on Benchmark Instances

Comparison of models with each other or with a heuristic for different decision parameter settings, benchmark instances are beneficial and widely used tools to measure the effectiveness or solution quality of the mathematical model or a heuristic. In literature, there is a wide range of benchmark instances for lots of different problem type and some of them are compiled in online libraries. In this study, some well-known benchmark instances of JSS problem with different instance sizes are used as well as randomly created instances. These instances are taken from an online library of benchmark instances[15]. The library compiles some articles that studied JSS problem with these instances. There are 82 instances in this library from instance size five machine, five jobs to 10 machine, 50 jobs. We use some of these instances that are relatively small. The instances that are used are listed in Appendix A.

Total 30 instances are used and divided into 3 groups by instance sizes. There is no sufficient benchmark instances for all instance sizes so we created some of them that coded like engXXxY_Z. These created instances has the same parameters as instances taken from the library. All instances have [10,100) range for processing times.XX index shows the number of jobs, Y index shows the number of machines and Z is the instance ID.

In this study, to understand the comparisons between parameter changes, instance size changes or model changes, a useful approach will be used that named performance profile chart [16]. This comparison tool has very beneficial visualization feature that reflects minor changes that are difficult to investigate or identify by only examining data tables.

A performance profile chart categorize the result with the changing factor for different parameter settings. It uses a factor τ that is a proportion of the C_{max} value of the instance with a specific parameter configuration over C_{max} value of the best known solution, which is actually optimum solution of the instance as reported in [17]. In other words, τ is the ratio of the found solution's C_{max} value to the optimum solution. Hence, $\tau = 1$ is the solution that is equal to the optimum solution. To give a sense of the factor τ, Table 5.2 can be useful. Table 5.2 is a small part of the whole result data that only includes $p/C_{max_{opt}} = 0.1$

P(τ) values represent the percentage of times that found C_{max} values are within a specific factor τ of the optimum C_{max} for that instance [17]. Hence, the lines that are closer to the vertical axes on performance profile chart have a better solution quality.

5.1.2. Comparison of Different Bucket Sizes

One of the important aspects of this study is to show how bucket size is related with the solution quality. Solution quality means how much a solution is close to the optimum solution. Bucket size is the basic decision parameter of this study. Intuitively,one expects that solution quality is thought to be proportional to bucket

Table 5.2. τ values for $p/C_{max_{opt}} = 0.1$ for all instances.

Instance	p	C_{max}	$C_{max_{opt}}$	♯ of jobs	♯ of machines	τ
eng6x6_1	4.9	52	49	6	6	1.06
eng6x6_2	5.5	61	55	6	6	1.11
eng6x6_3	5	57	50	6	6	1.14
eng6x6_4	5.2	58	52	6	6	1.12
eng6x6_5	5.1	58	51	6	6	1.14
eng6x6_6	5.4	62	54	6	6	1.15
eng6x6_7	5.9	69	59	6	6	1.17
eng6x6_8	5.5	56	55	6	6	1.02
eng6x6_9	4.8	60	48	6	6	1.25
ft06	5.5	58	55	6	6	1.05
eng10x5_1	57.9	664	579	10	5	1.15
eng10x5_2	7.01	900	701	10	5	1.28
eng10x5_3	61.9	697	619	10	5	1.13
eng10x5_4	54.8	691	548	10	5	1.26
eng10x5_5	62.4	671	624	10	5	1.08
la01	66.6	744	666	10	5	1.12
la02	65.5	899	655	10	5	1.37
la03	59.7	765	597	10	5	1.28
la04	59	815	590	10	5	1.38
la05	59.3	640	593	10	5	1.08
la16	94.5	1271	945	10	10	1.34
la17	78.4	1023	784	10	10	1.30
la18	84.8	1188	848	10	10	1.40
la19	84.2	1052	842	10	10	1.25
orb2	88.8	1220	888	10	10	1.37
orb4	100.5	1432	1005	10	10	1.42
orb5	88.7	1068	887	10	10	1.20
abz5	123.4	1873	1234	10	10	1.52
abz6	94.3	1283	943	10	10	1.36
ft10	93	1378	930	10	10	1.48

size.

To see the behavior of the bucket size over solution quality, there must be a common bucket size parameter to deal with the different $C_{max_{opt}}$ values of the different instances. Source of the different $C_{max_{opt}}$ value is instance size as well as the difference in $C_{max_{opt}}$ values of the same instance size. So, $p/C_{max_{opt}}$ is a useful parameter to investigate the effect of bucket size without including problem size effect and/or makespan of the instances. The only exception is for the bucket sizes that equal to the one. 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1 values are tested for $p/C_{max_{opt}}$ and also for $p/C_{max_{opt}} = 1/C_{max_{opt}}$ which means p=1 is also tested for all instances to prove that our model or heuristic catch the optimum solution that means $\tau = 1$.

Figure 5.3 is based on the results of all the instances. The only parameter is the $p/C_{max_{opt}}$ which takes 0.1, 0.2, 0.4, 0.6, 0.8, 1 as well as $1/C_{max_{opt}}$. Parameter interval is increased because of clear and understandable chart appearance. The Figure 5.3 shows that solution quality generally decreases with the increasing $p/C_{max_{opt}}$ values, but, in a few case solutions of higher $p/C_{max_{opt}}$ values can be better than lower $p/C_{max_{opt}}$ values for larger τ values as marked red in Figure 5.3. This is the expected result of the bucket size change. Figure 5.4, 5.5 and 5.6 are more clear illustrations of the performance profile chart Figure 5.3. These three charts are divided version of the Figure 5.3 in terms of instance size. Differences within instance sizes can be examined in detail.

To give some explanation on the performance profile chart, the following examples may become useful: For $p/C_{max_{opt}} = 0.1$ value in Figure 5.3, 63.3 per cent of all instances can have a better solution quality than $\tau = 1.28$ that means 1.28 times greater than their $C_{max_{opt}}$ value. The worst case for $p/C_{max_{opt}} = 0.1$ is $\tau = 1.51$ which means worst solution quality is 1.51 times higher than the best known result. For 6x6 instances and $p/C_{max_{opt}} = 0.1$ value, 50 per cent of instances can have a solution quality better than $\tau = 1.12$ that shown in Figure 5.4.

In conclusion, there is an inverse relation between bucket size parameter $p/C_{max_{opt}}$ and solution quality in general. Some exceptions occur when bucket size parameter is close to one. When data tables in Appendix A are investigated in detail, there is a few dramatic results in Table A.2 that found solution has an relative gap of 2 per cent for eng6x6_8 instance with a less CPU time.

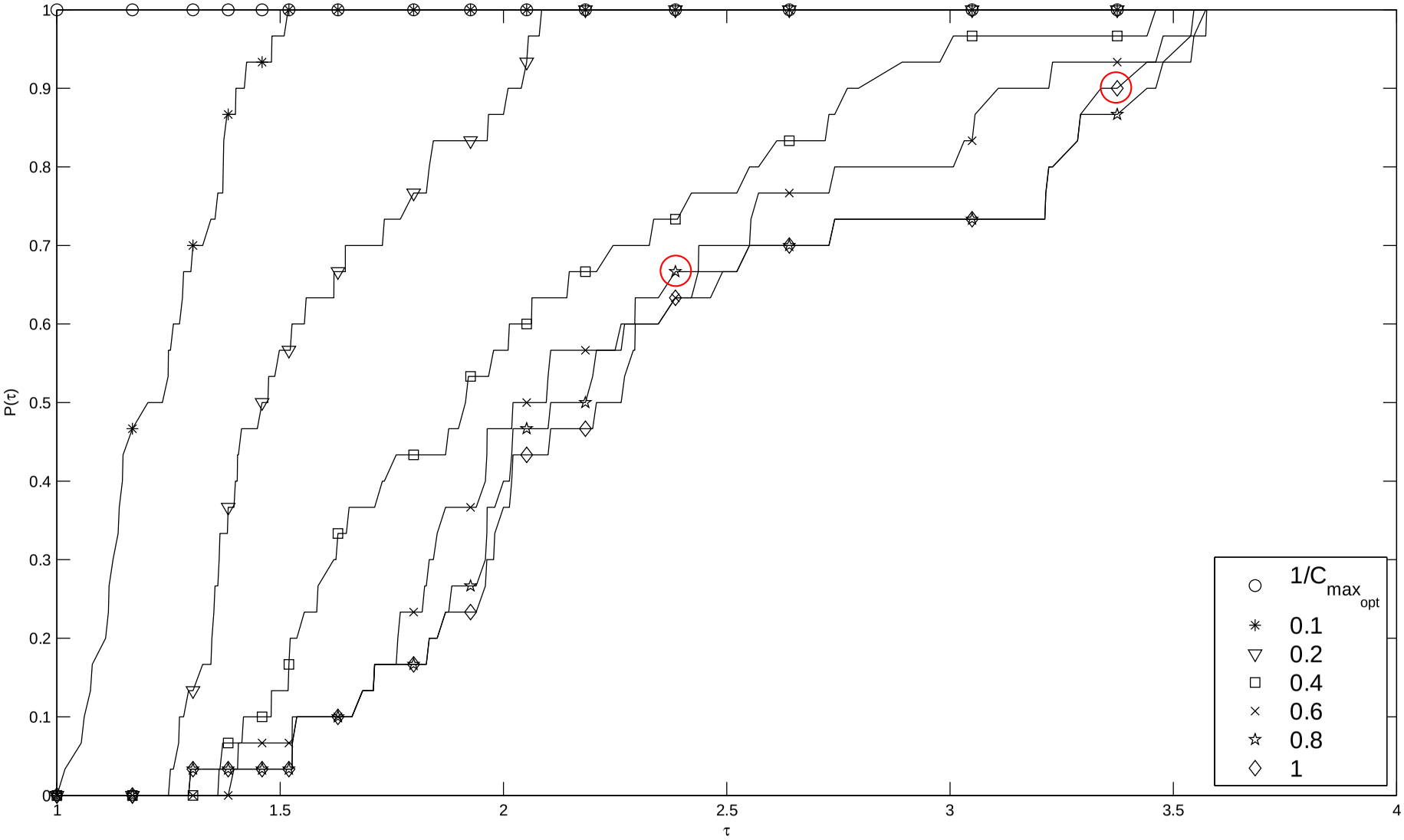

Figure 5.3. Discretization results of overall algorithm (Figure 4.4) for different $p/C_{max_{opt}}$ values.

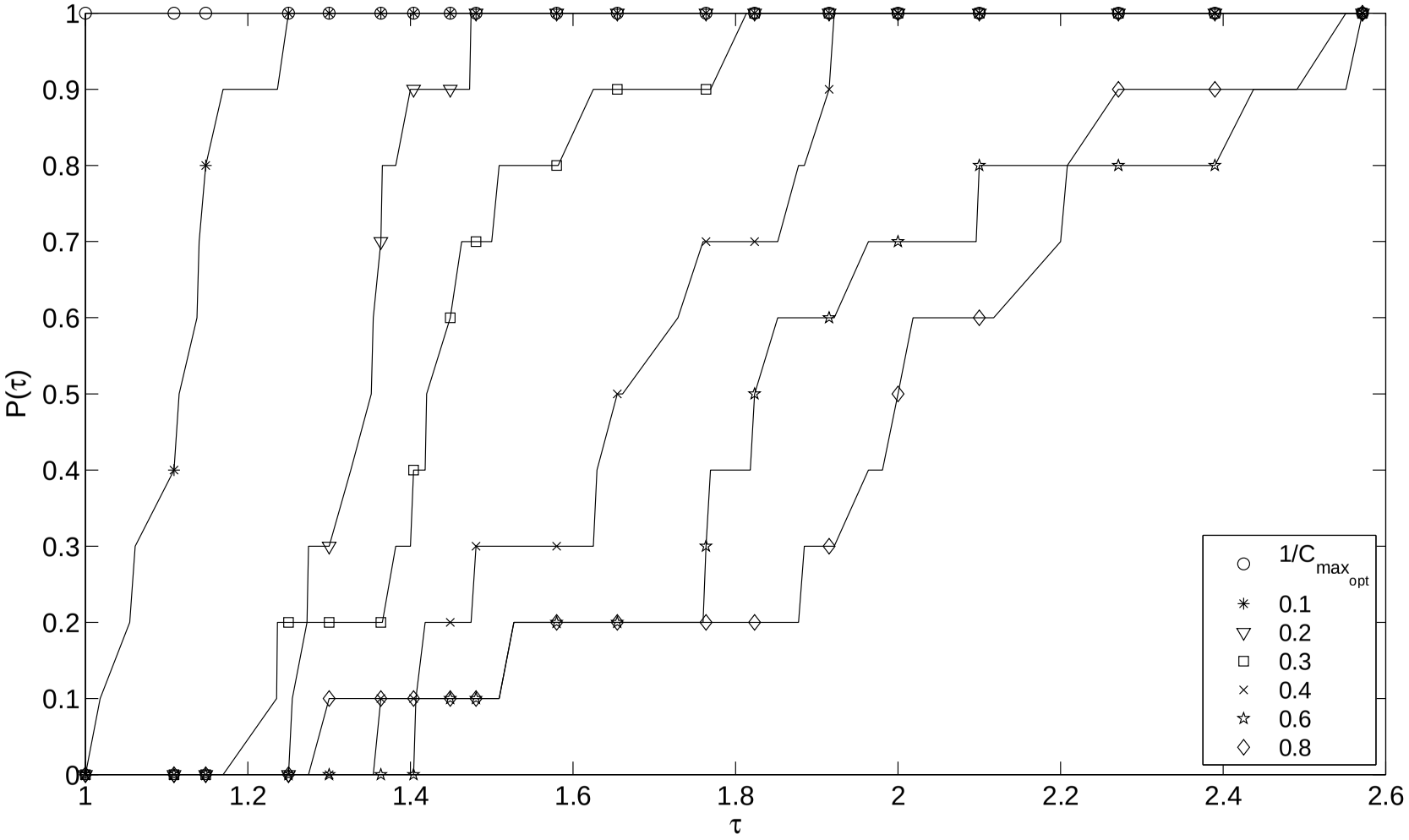

Figure 5.4. Discretization results of overall algorithm (Figure 4.4) for 6x6 instances for different $p/C_{max_{opt}}$ values.

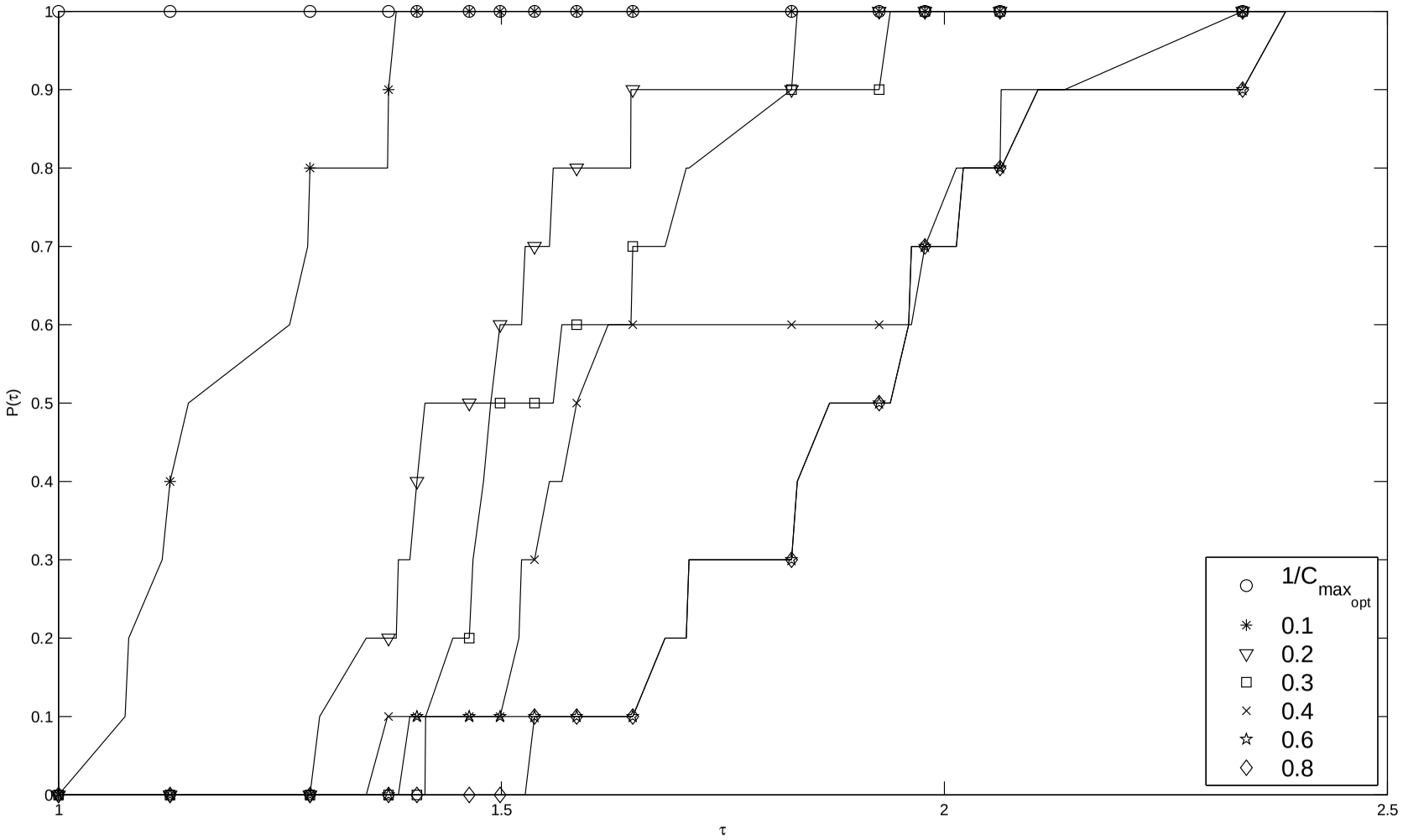

Figure 5.5. Discretization results of overall algorithm (Figure 4.4) for 10x5 instances for different $p/C_{max_{opt}}$ values.

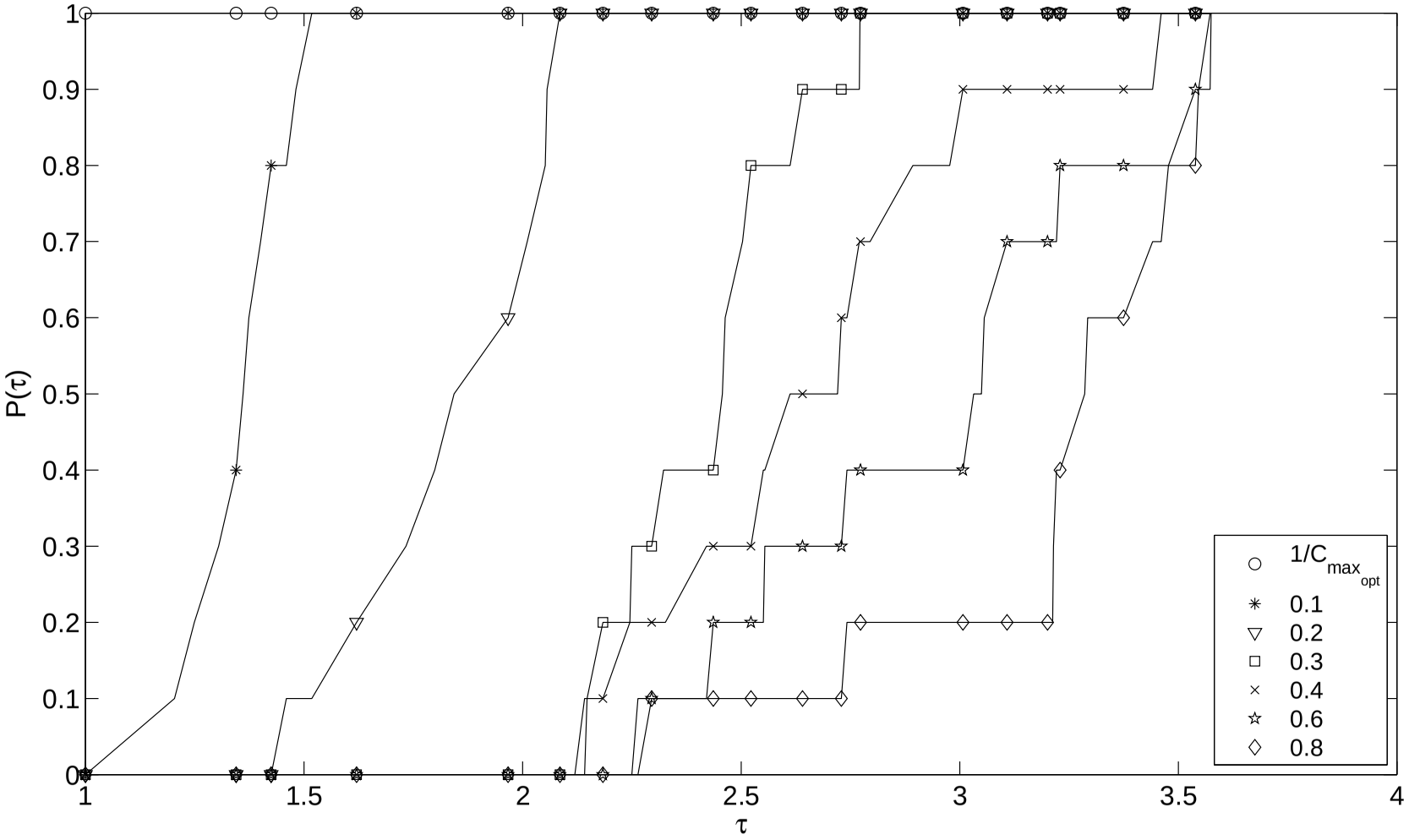

Figure 5.6. Discretization results of overall algorithm (Figure 4.4) for 10x10 instances for different $p/C_{max_{opt}}$ values.

5.1.3. Investigation of Instance Size

Another factor to be investigated is instance size. In this study, there is three benchmark instance sizes from 6x6 to 10x10. To have a comprehensive look on the discretization methodology, effect of instance size on solution quality must also be investigated. To analyze the effect of instance size, three level of $p/C_{max_{opt}}$ value are chosen as 0.1, 0.5 and 0.8 to give a clear understandable perspective. Figure 5.7, Figure 5.8 and Figure 5.9 are performance profile charts of the related bucket size parameters, respectively. These three figures show that instance size matters when $p/C_{max_{opt}}$ values are relatively low as 0.1. The discretization procedure works better when instance size is small. When $p/C_{max_{opt}}$ value increases, the instance size effect disappears gradually. This means our discretization procedure works better for small instances for relatively low $p/C_{max_{opt}}$ values. Decreasing solution quality with increasing instance size effect seems remarkable since one may expect that the algorithm may find better solutions by balancing the errors among operations for longer chains namely bigger instances.

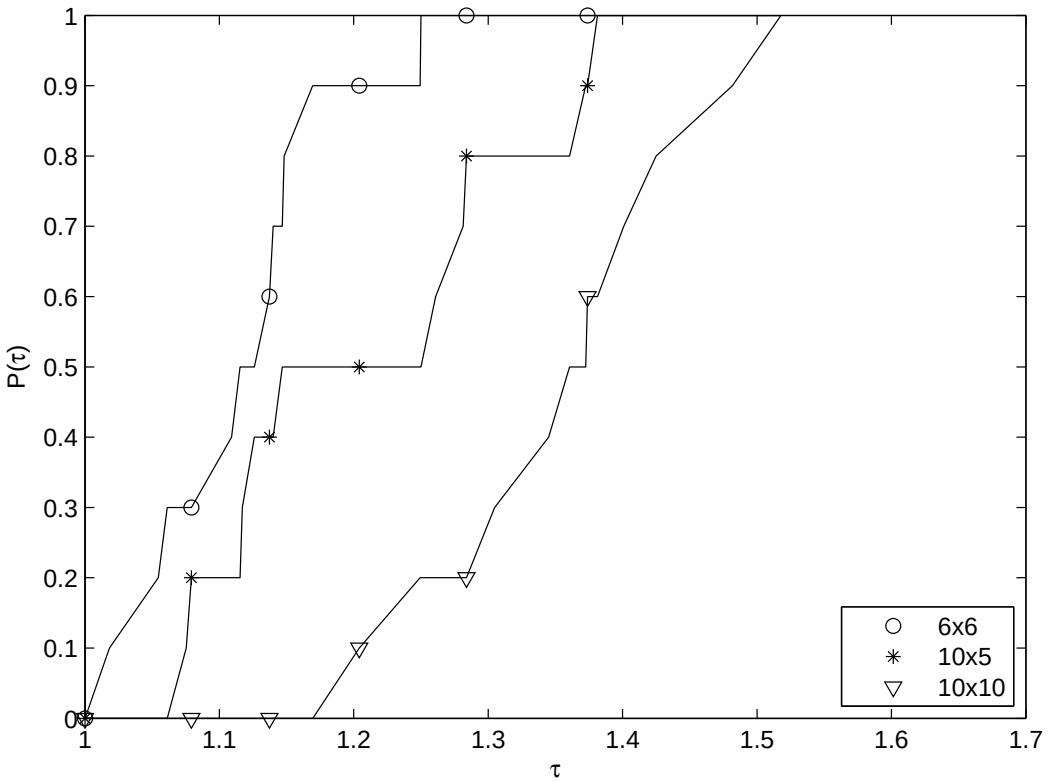

Figure 5.7. Discretization results of overall algorithm (Figure 4.4) for $p/C_{max_{opt}} = 0.1$ for different instance sizes.

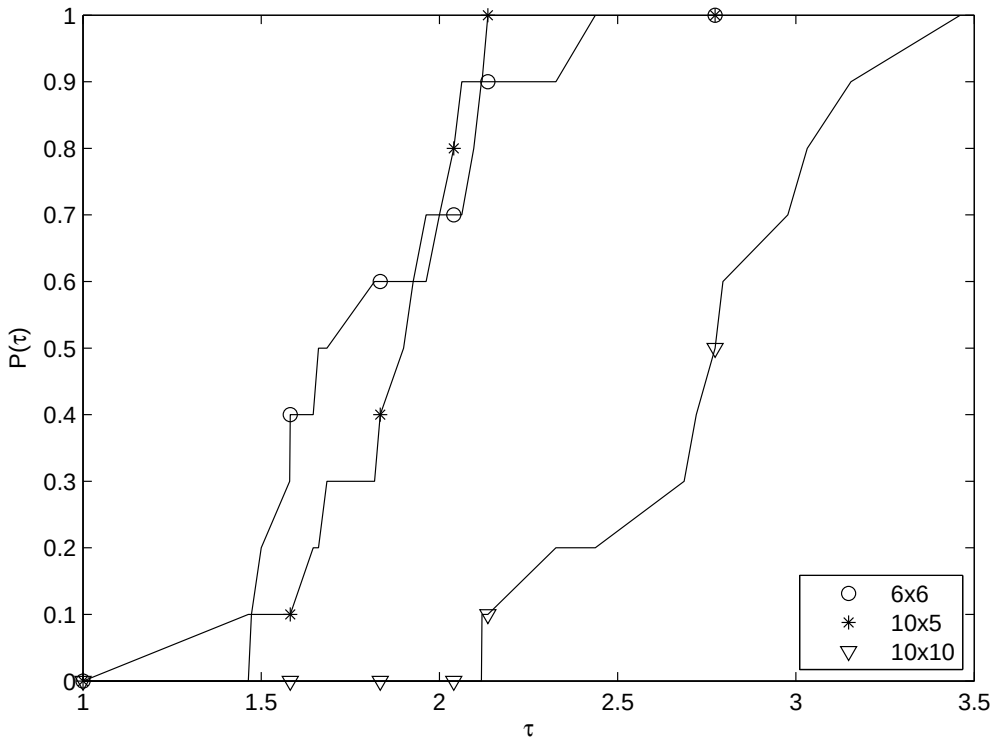

Figure 5.8. Discretization results of overall algorithm (Figure 4.4) for $p/C_{max_{opt}} = 0.5$ for different instance sizes.

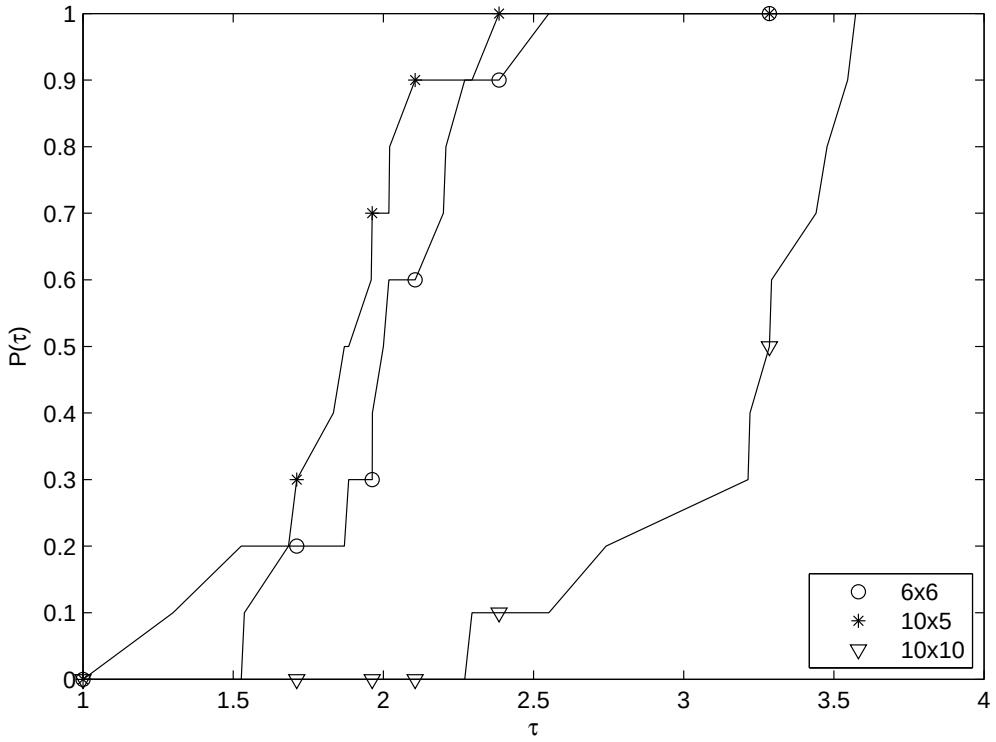

Figure 5.9. Discretization results of overall algorithm (Figure 4.4) for $p/C_{max_{opt}} = 0.8$ for different instance sizes.

5.1.4. CPU Time Analysis

To have better understanding of the results, solution time results are presented. All instances were run on a PC that has 2.2 GHz double core processor, 4Gb RAM. All codes are written with C♯ programming language and compiled with Microsoft Visual Studio 2010 Express running on Windows7 operating system.

CPU time consist of CPU times of discretization model, scheduling model and final sequencing algorithm. CPU times are compared through the used method that is performance profile chart. Differences are hard to determine and investigate from solution time data table. All CPU times are compared with the best found solution time of that instance which is the CPU time of $p/C_{max_{opt}} = 1$. The factor τ is defined and calculated according to this. As seen from the Figure 5.10, CPU times get better with the increasing bucket size. A CPU time is desired to be close to the upper left corner of Figure 5.10, which means its better in terms of CPU time. For example, the indicated data point means that 98 per cent of the instances are solved less than 6.2 times of the best solution time for bucket size factor 0.3. Bucket size effect diminishes gradually for higher ratios of the makespan of the related instance. Note that CPU times are decreasing with the increasing bucket size although the model is not bucket based namely time indexed model. If we use time-indexed model, model gets smaller in terms of variables and constraints and model may have less CPU times with increasing bucket size. Continuous time representation of JSS model does not get smaller in terms of variables and constraints with the increasing bucket size.

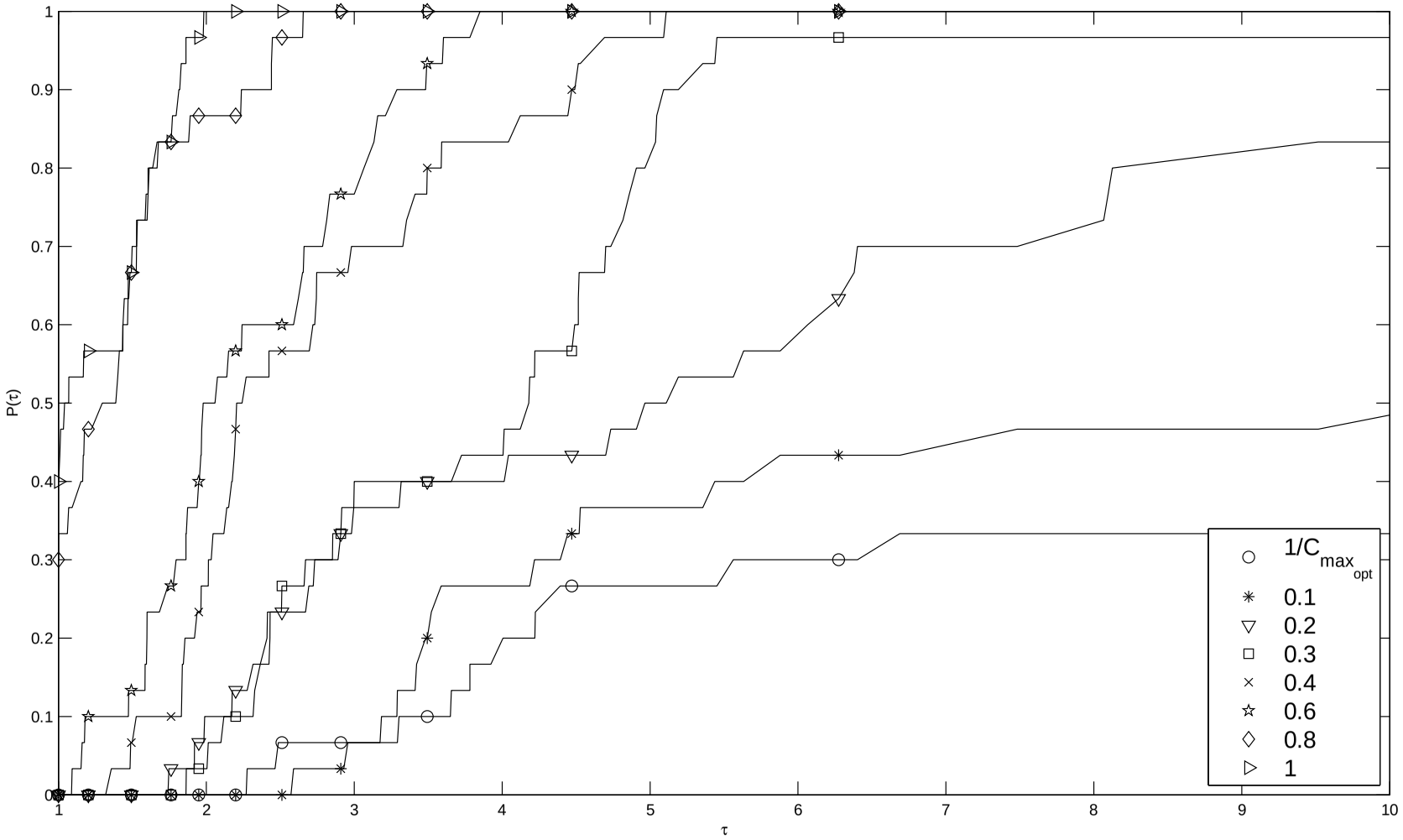

Figure 5.10. CPU time comparison of overall algorithm (Figure 4.4) for different $p/C_{max_{opt}}$ values.

6. CONCLUSIONS AND FURTHER RESEARCH

6.1. Conclusions

In this thesis, we developed a discretization procedure for determination of bucket size for various problems that has an acyclic precedence relations between operations or between items. While doing this, the goal is to minimize sum of errors in lead times of each item or operation. By doing this, we aimed to solve mid-sized job shop scheduling problems with introducing some discretization error for the sake of less solution time. We investigated the relation between solution quality and problem size and concluded that solution quality decreases with the increasing instance size. Another aspect of the study is that bucket size selection affects the solution quality. For small bucket sizes, the error faced is smaller. When the bucket size is increased to higher values as makespan of the problem, the importance of the discretization procedure almost diminishes and continuization heuristic takes place instead. A dramatic result appears in CPU solution times. Although our continuazation algorithm implements continuous-time based JSS model instead of bucket-based (time indexed) model which means model is not smaller, CPU times are decreasing with the increasing bucket size.

6.2. Suggestions for Further Research

To the best of our knowledge, this study is one of the first studies on time bucket discretization and bucket size selection subject since there is little literature about this research area. Hence, there are some open points in this thesis to be studied in detail.

We focused primarily on the effects of bucket size determination especially on JSS problems. The effects of this study can also be studied on project scheduling problem and MRP systems in detail. Especially on one type of project scheduling problem that is Resource Capacitated Project Scheduling Problem (RCPSP), discretization become more complex due to the resources. A distribution of resources within the defined time bucket length is a complicated issue since it can be assumed that resources are

distributed evenly within the time bucket or they can be consumed in the beginning of the period or in a different way. This brings more load to the discretization process.

Rounding model and heuristic can be investigated further. Rounding problem may become an attractive topic itself and its properties can be examined in detail especially time complexity of the problem. NP-hardness of the rounding problem needs to be investigated.

One of the open points of this study is that different objective functions in discretization model can be tested for various purposes such as minimization of cumulative error of only end items. This can lead to fluctuations in cumulative errors of lower level items for the sake of less cumulative errors for end items. The scheduling model used in continuzation process is continuous-time based model where each operation has a starting and finishing time. Hence, discretization process does not effect the number of variables in scheduling model. Bucket-based model can be applied instead of this model to have smaller models and might lead less CPU times. Final sequencing in continuzation process is done by a heuristic, a dispatching algorithm, that only considers SPT of the conflicting operations as a dispatching rule. Other more complicated dispatching rules can be applied to the built heuristic. Another point is that, rounding model can be implemented in scheduling model to give comprehensive decisions to have better discretization and sequencing results.

APPENDIX A: RESULTS OF ALL INSTANCES

Table A.1. τ values for $p/C_{max_{opt}} = 1/C_{max_{opt}}$ for all instances.

Instance	p	C_{max}	$C_{max_{opt}}$	# of jobs	# of machines	τ
eng6x6_1	1	49	49	6	6	1
eng6x6_2	1	55	55	6	6	1
eng6x6_3	1	50	50	6	6	1
eng6x6_4	1	52	52	6	6	1
eng6x6_5	1	51	51	6	6	1
eng6x6_6	1	54	54	6	6	1
eng6x6_7	1	59	59	6	6	1
eng6x6_8	1	55	55	6	6	1
eng6x6_9	1	48	48	6	6	1
ft06	1	55	55	6	6	1
eng10x5_1	1	579	579	10	5	1
eng10x5_2	1	701	701	10	5	1
eng10x5_3	1	619	619	10	5	1
eng10x5_4	1	548	548	10	5	1
eng10x5_5	1	624	624	10	5	1
la01	1	666	666	10	5	1
la02	1	655	655	10	5	1
la03	1	597	597	10	5	1
la04	1	590	590	10	5	1
la05	1	593	593	10	5	1
abz5	1	1234	1234	10	10	1
abz6	1	943	943	10	10	1
ft10	1	930	930	10	10	1
la16	1	945	945	10	10	1
la17	1	784	784	10	10	1
la18	1	848	848	10	10	1
la19	1	842	842	10	10	1
orb2	1	888	888	10	10	1
orb4	1	1005	1005	10	10	1
orb5	1	887	887	10	10	1

Table A.2. τ values for $p/C_{max_{opt}} = 0.1$ for all instances.

Instance	p	C_{max}	$C_{max_{opt}}$	# of jobs	# of machines	τ
eng6x6_1	4.9	52	49	6	6	1.06
eng6x6_2	5.5	61	55	6	6	1.11
eng6x6_3	5	57	50	6	6	1.14
eng6x6_4	5.2	58	52	6	6	1.12
eng6x6_5	5.1	58	51	6	6	1.14
eng6x6_6	5.4	62	54	6	6	1.15
eng6x6_7	5.9	69	59	6	6	1.17
eng6x6_8	5.5	56	55	6	6	1.02
eng6x6_9	4.8	60	48	6	6	1.25
ft06	5.5	58	55	6	6	1.05
eng10x5_1	57.9	664	579	10	5	1.15
eng10x5_2	70.1	900	701	10	5	1.28
eng10x5_3	61.9	697	619	10	5	1.13
eng10x5_4	54.8	691	548	10	5	1.26
eng10x5_5	62.4	671	624	10	5	1.08
la01	66.6	744	666	10	5	1.12
la02	65.5	899	655	10	5	1.37
la03	59.7	765	597	10	5	1.28
la04	59	815	590	10	5	1.38
la05	59.3	640	593	10	5	1.08
abz5	123.4	1873	1234	10	10	1.52
abz6	94.3	1283	943	10	10	1.36
ft10	93	1378	930	10	10	1.48
la16	94.5	1271	945	10	10	1.34
la17	78.4	1023	784	10	10	1.30
la18	84.8	1188	848	10	10	1.40
la19	84.2	1052	842	10	10	1.25
orb2	88.8	1220	888	10	10	1.37
orb4	100.5	1432	1005	10	10	1.42
orb5	88.7	1068	887	10	10	1.20

Table A.3. τ values for $p/C_{max_{opt}} = 0.2$ for all instances.

Instance	p	C_{max}	$C_{max_{opt}}$	# of jobs	# of machines	τ
eng6x6_1	9.8	65	49	6	6	1.33
eng6x6_2	11	69	55	6	6	1.25
eng6x6_3	10	70	50	6	6	1.40
eng6x6_4	10.4	71	52	6	6	1.37
eng6x6_5	10.2	65	51	6	6	1.27
eng6x6_6	10.8	73	54	6	6	1.35
eng6x6_7	11.8	87	59	6	6	1.47
eng6x6_8	11	70	55	6	6	1.27
eng6x6_9	9.6	65	48	6	6	1.35
ft06	11	75	55	6	6	1.36
eng10x5_1	115.8	884	579	10	5	1.53
eng10x5_2	140.2	970	701	10	5	1.38
eng10x5_3	123.8	1019	619	10	5	1.65
eng10x5_4	109.6	1005	548	10	5	1.83
eng10x5_5	124.8	935	624	10	5	1.50
la01	133.2	1038	666	10	5	1.56
la02	131	920	655	10	5	1.40
la03	119.4	844	597	10	5	1.41
la04	118	764	590	10	5	1.29
la05	118.6	799	593	10	5	1.35
abz5	246.8	2427	1234	10	10	1.97
abz6	188.6	1935	943	10	10	2.05
ft10	186	1612	930	10	10	1.73
la16	189	1700	945	10	10	1.80
la17	156.8	1612	784	10	10	2.06
la18	169.6	1705	848	10	10	2.01
la19	168.4	1229	842	10	10	1.46
orb2	177.6	1439	888	10	10	1.62
orb4	201	2096	1005	10	10	2.09
orb5	177.4	1635	887	10	10	1.84

Table A.4. τ values for $p/C_{max_{opt}} = 0.3$ for all instances.

Instance	p	C_{max}	$C_{max_{opt}}$	# of jobs	# of machines	τ
eng6x6_1	14.7	71	49	6	6	1.45
eng6x6_2	16.5	83	55	6	6	1.51
eng6x6_3	15	71	50	6	6	1.42
eng6x6_4	15.6	73	52	6	6	1.40
eng6x6_5	15.3	63	51	6	6	1.24
eng6x6_6	16.2	79	54	6	6	1.46
eng6x6_7	17.7	107	59	6	6	1.81
eng6x6_8	16.5	76	55	6	6	1.38
eng6x6_9	14.4	78	48	6	6	1.63
ft06	16.5	68	55	6	6	1.24
eng10x5_1	173.7	819	579	10	5	1.41
eng10x5_2	210.3	1043	701	10	5	1.49
eng10x5_3	185.7	916	619	10	5	1.48
eng10x5_4	164.4	792	548	10	5	1.45
eng10x5_5	187.2	916	624	10	5	1.47
la01	199.8	1138	666	10	5	1.71
la02	196.5	1197	655	10	5	1.83
la03	179.1	984	597	10	5	1.65
la04	177	1144	590	10	5	1.94
la05	177.9	930	593	10	5	1.57
abz5	370.2	3089	1234	10	10	2.50
abz6	282.9	2020	943	10	10	2.14
ft10	279	2578	930	10	10	2.77
la16	283.5	2322	945	10	10	2.46
la17	235.2	1821	784	10	10	2.32
la18	254.4	2239	848	10	10	2.64
la19	252.6	2124	842	10	10	2.52
orb2	266.4	1998	888	10	10	2.25
orb4	301.5	2476	1005	10	10	2.46
orb5	266.1	1937	887	10	10	2.18

Table A.5. τ values for $p/C_{max_{opt}} = 0.4$ for all instances.

Instance	p	C_{max}	$C_{max_{opt}}$	# of jobs	# of machines	τ
eng6x6_1	19.6	92	49	6	6	1.88
eng6x6_2	22	91	55	6	6	1.65
eng6x6_3	20	88	50	6	6	1.76
eng6x6_4	20.8	77	52	6	6	1.48
eng6x6_5	20.4	98	51	6	6	1.92
eng6x6_6	21.6	88	54	6	6	1.63
eng6x6_7	23.6	113	59	6	6	1.92
eng6x6_8	22	75	55	6	6	1.36
eng6x6_9	19.2	83	48	6	6	1.73
ft06	22	78	55	6	6	1.42
eng10x5_1	231.6	880	579	10	5	1.52
eng10x5_2	280.4	1111	701	10	5	1.58
eng10x5_3	247.6	849	619	10	5	1.37
eng10x5_4	219.2	888	548	10	5	1.62
eng10x5_5	249.6	1288	624	10	5	2.06
la01	266.4	1341	666	10	5	2.01
la02	262	1018	655	10	5	1.55
la03	238.8	1395	597	10	5	2.34
la04	236	1167	590	10	5	1.98
la05	237.2	903	593	10	5	1.52
abz5	493.6	3570	1234	10	10	2.89
abz6	377.2	2836	943	10	10	3.01
ft10	372	2538	930	10	10	2.73
la16	378	2618	945	10	10	2.77
la17	313.6	2000	784	10	10	2.55
la18	339.2	2215	848	10	10	2.61
la19	336.8	2914	842	10	10	3.46
orb2	355.2	1994	888	10	10	2.25
orb4	402	2433	1005	10	10	2.42
orb5	354.8	1905	887	10	10	2.15

Table A.6. τ values for $p/C_{max_{opt}} = 0.5$ for all instances.

Instance	p	C_{max}	$C_{max_{opt}}$	# of jobs	# of machines	τ
eng6x6_1	24.5	98	49	6	6	2.00
eng6x6_2	27.5	81	55	6	6	1.47
eng6x6_3	25	79	50	6	6	1.58
eng6x6_4	26	109	52	6	6	2.10
eng6x6_5	25.5	108	51	6	6	2.12
eng6x6_6	27	81	54	6	6	1.50
eng6x6_7	29.5	98	59	6	6	1.66
eng6x6_8	27.5	100	55	6	6	1.82
eng6x6_9	24	117	48	6	6	2.44
ft06	27.5	87	55	6	6	1.58
eng10x5_1	289.5	953	579	10	5	1.65
eng10x5_2	350.5	1376	701	10	5	1.96
eng10x5_3	309.5	906	619	10	5	1.46
eng10x5_4	274	1005	548	10	5	1.83
eng10x5_5	312	1273	624	10	5	2.04
la01	333	1283	666	10	5	1.93
la02	327.5	1244	655	10	5	1.90
la03	298.5	1275	597	10	5	2.14
la04	295	1217	590	10	5	2.06
la05	296.5	999	593	10	5	1.68
abz5	617	3674	1234	10	10	2.98
abz6	471.5	2533	943	10	10	2.69
ft10	465	2579	930	10	10	2.77
la16	472.5	2865	945	10	10	3.03
la17	392	2133	784	10	10	2.72
la18	424	1973	848	10	10	2.33
la19	421	2914	842	10	10	3.46
orb2	444	2482	888	10	10	2.80
orb4	502.5	3170	1005	10	10	3.15
orb5	443.5	1880	887	10	10	2.12

Table A.7. τ values for $p/C_{max_{opt}} = 0.6$ for all instances.

Instance	p	C_{max}	$C_{max_{opt}}$	# of jobs	# of machines	τ
eng6x6_1	29.4	126	49	6	6	2.57
eng6x6_2	33	84	55	6	6	1.53
eng6x6_3	30	105	50	6	6	2.10
eng6x6_4	31.2	92	52	6	6	1.77
eng6x6_5	30.6	93	51	6	6	1.82
eng6x6_6	32.4	100	54	6	6	1.85
eng6x6_7	35.4	83	59	6	6	1.41
eng6x6_8	33	108	55	6	6	1.96
eng6x6_9	28.8	117	48	6	6	2.44
ft06	33	97	55	6	6	1.76
eng10x5_1	347.4	1083	579	10	5	1.87
eng10x5_2	420.6	1376	701	10	5	1.96
eng10x5_3	371.4	1213	619	10	5	1.96
eng10x5_4	328.8	1005	548	10	5	1.83
eng10x5_5	374.4	1314	624	10	5	2.11
la01	399.6	1140	666	10	5	1.71
la02	393	1324	655	10	5	2.02
la03	358.2	1424	597	10	5	2.39
la04	354	824	590	10	5	1.40
la05	355.8	999	593	10	5	1.68
abz5	740.4	3152	1234	10	10	2.55
abz6	565.8	2931	943	10	10	3.11
ft10	558	2550	930	10	10	2.74
la16	567	2865	945	10	10	3.03
la17	470.4	2532	784	10	10	3.23
la18	508.8	2066	848	10	10	2.44
la19	505.2	2980	842	10	10	3.54
orb2	532.8	2714	888	10	10	3.06
orb4	603	3593	1005	10	10	3.58
orb5	532.2	2008	887	10	10	2.26

Table A.8. τ values for $p/C_{max_{opt}} = 0.7$ for all instances.

Instance	p	C_{max}	$C_{max_{opt}}$	# of jobs	# of machines	τ
eng6x6_1	34.3	115	49	6	6	2.35
eng6x6_2	38.5	84	55	6	6	1.53
eng6x6_3	35	104	50	6	6	2.08
eng6x6_4	36.4	98	52	6	6	1.88
eng6x6_5	35.7	117	51	6	6	2.29
eng6x6_6	37.8	109	54	6	6	2.02
eng6x6_7	41.3	141	59	6	6	2.39
eng6x6_8	38.5	108	55	6	6	1.96
eng6x6_9	33.6	106	48	6	6	2.21
ft06	38.5	121	55	6	6	2.20
eng10x5_1	405.3	1083	579	10	5	1.87
eng10x5_2	490.7	1376	701	10	5	1.96
eng10x5_3	433.3	1213	619	10	5	1.96
eng10x5_4	383.6	1005	548	10	5	1.83
eng10x5_5	436.8	1314	624	10	5	2.11
la01	466.2	1140	666	10	5	1.71
la02	458.5	1324	655	10	5	2.02
la03	417.9	1424	597	10	5	2.39
la04	413	907	590	10	5	1.54
la05	415.1	999	593	10	5	1.68
abz5	863.8	3975	1234	10	10	3.22
abz6	660.1	2988	943	10	10	3.17
ft10	651	2509	930	10	10	2.70
la16	661.5	3189	945	10	10	3.37
la17	548.8	2461	784	10	10	3.14
la18	593.6	2586	848	10	10	3.05
la19	589.4	2705	842	10	10	3.21
orb2	621.6	3088	888	10	10	3.48
orb4	703.5	3217	1005	10	10	3.20
orb5	620.9	2036	887	10	10	2.30

Table A.9. τ values for $p/C_{max_{opt}} = 0.8$ for all instances.

Instance	p	C_{max}	$C_{max_{opt}}$	# of jobs	# of machines	τ
eng6x6_1	39.2	125	49	6	6	2.55
eng6x6_2	44	84	55	6	6	1.53
eng6x6_3	40	65	50	6	6	1.30
eng6x6_4	41.6	98	52	6	6	1.88
eng6x6_5	40.8	102	51	6	6	2.00
eng6x6_6	43.2	109	54	6	6	2.02
eng6x6_7	47.2	134	59	6	6	2.27
eng6x6_8	44	108	55	6	6	1.96
eng6x6_9	38.4	106	48	6	6	2.21
ft06	44	121	55	6	6	2.20
eng10x5_1	463.2	1083	579	10	5	1.87
eng10x5_2	560.8	1376	701	10	5	1.96
eng10x5_3	495.2	1213	619	10	5	1.96
eng10x5_4	438.4	1005	548	10	5	1.83
eng10x5_5	499.2	1314	624	10	5	2.11
la01	532.8	1140	666	10	5	1.71
la02	524	1324	655	10	5	2.02
la03	477.6	1424	597	10	5	2.39
la04	472	907	590	10	5	1.54
la05	474.4	999	593	10	5	1.68
abz5	987.2	3975	1234	10	10	3.22
abz6	754.4	3245	943	10	10	3.44
ft10	744	2550	930	10	10	2.74
la16	756	3105	945	10	10	3.29
la17	627.2	2801	784	10	10	3.57
la18	678.4	2726	848	10	10	3.21
la19	673.6	2986	842	10	10	3.55
orb2	710.4	3088	888	10	10	3.48
orb4	804	3309	1005	10	10	3.29
orb5	709.6	2036	887	10	10	2.30

Table A.10. τ values for $p/C_{max_{opt}} = 0.9$ for all instances.

Instance	p	C_{max}	$C_{max_{opt}}$	# of jobs	# of machines	τ
eng6x6_1	44.1	125	49	6	6	2.55
eng6x6_2	49.5	84	55	6	6	1.53
eng6x6_3	45	65	50	6	6	1.30
eng6x6_4	46.8	103	52	6	6	1.98
eng6x6_5	45.9	102	51	6	6	2.00
eng6x6_6	48.6	109	54	6	6	2.02
eng6x6_7	53.1	134	59	6	6	2.27
eng6x6_8	49.5	126	55	6	6	2.29
eng6x6_9	43.2	106	48	6	6	2.21
ft06	49.5	137	55	6	6	2.49
eng10x5_1	521.1	1083	579	10	5	1.87
eng10x5_2	630.9	1376	701	10	5	1.96
eng10x5_3	557.1	1213	619	10	5	1.96
eng10x5_4	493.2	1005	548	10	5	1.83
eng10x5_5	561.6	1314	624	10	5	2.11
la01	599.4	1140	666	10	5	1.71
la02	589.5	1324	655	10	5	2.02
la03	537.3	1424	597	10	5	2.39
la04	531	907	590	10	5	1.54
la05	533.7	999	593	10	5	1.68
abz5	1110.6	3975	1234	10	10	3.22
abz6	848.7	3245	943	10	10	3.44
ft10	837	2550	930	10	10	2.74
la16	850.5	3105	945	10	10	3.29
la17	705.6	2617	784	10	10	3.34
la18	763.2	2726	848	10	10	3.21
la19	757.8	2986	842	10	10	3.55
orb2	799.2	3088	888	10	10	3.48
orb4	904.5	3309	1005	10	10	3.29
orb5	798.3	2036	887	10	10	2.30

Table A.11. τ values for $p/C_{max_{opt}} = 1$ for all instances.

Instance	p	C_{max}	$C_{max_{opt}}$	# of jobs	# of machines	τ
eng6x6_1	49	125	49	6	6	2.55
eng6x6_2	55	84	55	6	6	1.53
eng6x6_3	50	65	50	6	6	1.30
eng6x6_4	52	103	52	6	6	1.98
eng6x6_5	51	102	51	6	6	2.00
eng6x6_6	54	109	54	6	6	2.02
eng6x6_7	59	134	59	6	6	2.27
eng6x6_8	55	126	55	6	6	2.29
eng6x6_9	48	106	48	6	6	2.21
ft06	55	137	55	6	6	2.49
eng10x5_1	579	1083	579	10	5	1.87
eng10x5_2	701	1376	701	10	5	1.96
eng10x5_3	619	1213	619	10	5	1.96
eng10x5_4	548	1005	548	10	5	1.83
eng10x5_5	624	1314	624	10	5	2.11
la01	666	1140	666	10	5	1.71
la02	655	1324	655	10	5	2.02
la03	597	1424	597	10	5	2.39
la04	590	907	590	10	5	1.54
la05	593	999	593	10	5	1.68
abz5	1234	3975	1234	10	10	3.22
abz6	943	3245	943	10	10	3.44
ft10	930	2550	930	10	10	2.74
la16	945	3105	945	10	10	3.29
la17	784	2617	784	10	10	3.34
la18	848	2726	848	10	10	3.21
la19	842	2986	842	10	10	3.55
orb2	888	3088	888	10	10	3.48
orb4	1005	3309	1005	10	10	3.29
orb5	887	2036	887	10	10	2.30

REFERENCES

1. Pochet, Y. and L. A. Wolsey, *Production Planning by Mixed Integer Programming*, Springer Science+Business Media, Inc., New York, pp. 1-47, 2006.

2. Helber, S., "Lot Sizing in Capacitated Production Planning and Control Systems", *OR Spektrum*, Vol. 17, pp. 5-18, 1995.

3. Riezebos, J., *Design of a Period Batch Control Planning Sysyem for Cellular Manufacturing*, Ph.D. Thesis, University of Groningen, Netherlands, 2001.

4. Riezebos, J., "Time Bucket Length and Lot-splitting Approach", Research Report, University of Groningen, Netherlands, 2002.

5. Riezebos, J. and G. J. C. Gaalman, "Lot Splitting and Time Bucket Size in Planning Systems Design", In: Flexible Automation and Intelligent Manufacturing, Dublin, 2001.

6. Brandimarte, P. and A. Villa Eds, *Modeling Manufacturing Systems: From Aggregate Planning to Real-Time Control*, Springer, Berlin, pp. 5-23, 1999.

7. Maravelias, C. T. and I. E. Grossmann, "New General Continuous-Time State-Task Network Formulation for Short-Term Scheduling of Multipurpose Batch Plants", *Industrial Engineering Chemistry Research*, Vol. 42, pp. 3056-3074, 2003.

8. Lin, J. T. and Y.-Y. Chen, "A Multi-site Supply Network Planning Problem Considering Variable Time Buckets - A TFT-LCD Industry Case", *International Journal of Advanced Manufacturing Technologies*, Vol. 33, pp. 1031-1044, 2007.

9. Stadtler, H., *Supply Chain Management and Advanced Planning*, Third Edition, Springer, Berlin, pp. 210,213, 2005.

10. Suerie, C. and H. Stadtler, "The Capacitated Lot-Sizing Problem with Linked Lot

Sizes", *Management Science*, Vol. 49, No. 8, pp. 1039-1054, 2003.

11. Timpe, C. H. and J. Kallrath, "Optimal Planning in Large Multi-site Production Networks", *European Journal of Operational Research*, Vol. 126, pp. 422-435, 2000.

12. Chou, Y.-C. and I.-H. Hong, "Methodology for Product Mix Planning in Semiconductor Foundry Manufacturing", *IEEE Transactions on Semiconductor Manufacturing*, Vol. 13, No. 3, pp. 278-285, 2000.

13. Riezebos, J., "Time Bucket Length and Lot-splitting Approach", *International Journal of Production Research*, Vol. 42, No. 12, pp. 2325-2338, 2004.

14. Šeda, M., "Mathematical Models of Flow Shop and Job Shop Scheduling Problems", *International Journal of Mathematical, Computational, Physical and Quantum Engineering*, Vol. 1, No. 7, pp. 295-300, 2007.

15. J. E. Beasley, *OR-Library*, 1990, http://people.brunel.ac.uk/ mastjjb/jeb/orlib/files/jobshop1.txt(Accessed: 20 April 2015).

16. Dolan, E. D. and J. J. Moré, "Benchmarking Optimization Software with Performance Profiles", *Mathematical Programming*, Vol. 91, No. 2, pp. 201-213, 2002.

17. Gören, M. and Z. C. Taşkın, "A Column Generation Approach for Evaluating Delivery Efficiencies of Collimator Technologies in IMRT Treatment Planning", *Physics in Medicine and Biology*, Vol. 60, No. 5, 2015.