Rahime Şeyma BEKLİ

PARALLEL MACHINE SCHEDULING IN THE FACE OF PROCESSING TIME UNCERTAINTY

AGU 2017

PARALLEL MACHINE SCHEDULING IN THE FACE OF PROCESSING TIME UNCERTAINTY

A THESIS

SUBMITTED TO THE DEPARTMENT OF INDUSTRIAL ENGINEERING

AND THE GRADUATE SCHOOL OF ENGINEERING & SCIENCEOF

ABDULLAH GUL UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Rahime Şeyma BEKLİ

August 2017

PARALLEL MACHINE SCHEDULING IN THE FACE OF PROCESSING TIME UNCERTAINTY

A THESIS

SUBMITTED TO THE DEPARTMENT OF INDUSTRIAL ENGINEERING

AND THEGRADUATE SCHOOL OF ENGINEERING & SCIENCEOF ABDULLAH GUL UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Rahime Şeyma BEKLİ

August 2017

SCIENTIFIC ETHICS COMPLIANCE

I hereby declare that all information in this document has been obtained in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all materials and results that are not original to this work.

Rahime Şeyma BEKLİ

REGULATORY COMPLIANCE

M.Sc. thesis titled **"PARALLEL MACHINE SCHEDULING IN THE FACE OF PROCESSING TIME UNCERTAINTY"** has been prepared in accordance with the Thesis Writing Guidelines of the Abdullah Gül University, Graduate School of Engineering & Science.

Prepared By	Advisor
Rahime Şeyma BEKLİ	Assist. Prof. Dr. Selçuk GÖREN

Head of the Industrial Engineering Graduate Program

Assoc. Prof. Dr. İbrahim AKGÜN

ACCEPTANCE AND APPROVAL

M.Sc.thesis titled **"PARALLEL MACHINE SCHEDULING IN THE FACE OF PROCESSING TIME UNCERTAINTY"** and prepared by Rahime Şeyma BEKLİ has been accepted by the jury in the Industrial Engineering Graduate Program at Abdullah Gül University, Graduate School of Engineering & Science.

21 /08 / 2017

JURY:

Assoc. Prof. Dr. İbrahim AKGÜN :…….

Assist. Prof. Dr. Selçuk GÖREN :…….

Assist. Prof. Dr. Pınar ZARİF TAPKAN :…….

APPROVAL:

The acceptance of this M.Sc. thesis has been approved by the decision of the Abdullah Gül University, Graduate School of Engineering & Science, Executive Board dated ….. /….. / …….. and numbered ……..……. .

….. /….. / …….. Graduate School Dean

Prof. Dr. İrfan ALAN

ABSTRACT

PARALLEL MACHINE SCHEDULING IN THE FACE OF PROCESSING TIME UNCERTAINTY

Rahime Şeyma BEKLİ
MSc. in Industrial Engineering
Supervisor: Assist. Prof. Dr. Selçuk GÖREN
August 2017

Competition in today's business and production world leads the companies to generate schedules that increase productivity and decrease manufacturing cost. However, most of the schedules cannot be executed exactly because of the unexpected disruptions such as machine breakdowns, order cancellations and so forth. In order to develop disruption resistant schedules, robust scheduling subject has gained interest among researchers.

In this study, we consider a parallel machine environment with processing time uncertainty. The performance measure is taken as the completion time of the last job. The uncertainty is modeled by discrete set of scenarios. An integer programming model that can handle small problems is proposed. We observe that this model cannot manage large problems. To alleviate this difficulty, we propose to decrease number of scenarios selected for model. Next, we apply dual decomposition method in order to solve many smaller problems rather than a large problem. Large problems cannot be handled by this method either. This is why; we alter dual decomposition method by relaxing and develop a new heuristic. Also we propose a hybrid tabu search algorithm to solve the large problems.

The results show that, the proposed heuristics; selecting scenario approach and tabu search algorithm perform well for the parallel machine scheduling problems.

Keywords: parallel machine scheduling, robust scheduling, makespan, Dual Decomposition, tabu search algorithm

ÖZET

BELİRSİZ İŞLEM SÜRESİNE TABİ PARALEL MAKİNE ÇİZELGELEMELERİ

Rahime Şeyma BEKLİ
Endüstri Mühendisliği Bölümü Yüksek Lisans
Tez Yöneticisi: Yard. Doç. Dr. Selçuk GÖREN
Ağustos-2017

Günümüz dünyasında iş ve üretim rekabeti, firmaların verimlilik artıran ve imalat maliyetini düşüren çizelgeler üretmesine yol açmıştır. Ancak, üretilen çizelgeler beklenmedik aksaklıklar yüzünden, genellikle amaçlandığı şekilde uygulanamamaktadır. Bu aksaklıklar makine arızalanması, sipariş iptali gibi örneklendirilebilir. Aksaklıklara duyarsız çizelge olan gürbüz çizelgeleme, son yıllarda araştırmacılar arasında önem kazanmıştır.

Bu çalışmada, belirsiz işlem süresine tabi paralel makine ortamı ele alınmıştır. Performans ölçütü son işin bitiş süresi olarak alınmıştır. Belirsizlik, ayrık senaryolar olarak modellenmiş ve küçük boyuttaki problemleri çözebilen bir tam sayılı programlama oluşturulmuştur. Bu model büyük problemleri çözmede sıkıntılıdır. Bu sebeple senaryo sayısını azaltma yaklaşımı denenmiştir. Daha sonra eşiz ayrıştırma yöntemi ile büyük problemlerin çözümü amaçlanmıştır. Bu yöntemi kullanmadaki amaç büyük bir problem çözmek yerine, küçük ama çok sayıda problem çözerek sonuca ulaşmaktır. Ancak bu yöntem de büyük problemlerde istenilen sonuçları vermemiştir. Bu sebeple senaryo sayısı azaltılarak eşiz ayrıştırma yöntemi kullanılmış ve yeni bir sezgisel önerilmiştir. Aynı zamanda bir tabu arama algoritması oluşturulmuştur.

Sonuçlar, önerilen sezgisel algoritmalardan senaryo azaltılması ve tabu arama algoritmalarının paralel makine ortamında iyi sonuçlar verdiğini göstermektedir.

Anahtar kelimeler: paralel makinelerde çizelgeleme, gürbüz çizelgeleme, eşiz ayrışma, tabu arama algoritması

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisor Prof. Selçuk GÖREN for the continuous support of my MSc. study, for his patience and, encouragement. He helped me with his broad knowledge throughout all the stages of this research. His valuable guidance on technical and editorial aspects enabled this thesis to be completed. I want to point out that not just only in this subject of my thesis, Professor GÖREN inspired me to continue my academic studies on probabilistic and stochastic topics of Industrial Engineering.

Besides my advisor, I would like to thank the rest of my thesis committee: Prof. İbrahim AKGÜN and Prof. Pınar Zarif TAPKAN for their insightful comments and support.

Lastly, I would like to thank my family for their unceasing support, my husband for his love and understanding and my son for bringing joy in my life.

I acknowledge that this thesis is supported by TUBİTAK, The Scientific and Technological Research Council of Turkey with project number113M490.

Table of Content

List of Figures

List of Tables

Chapter 1

INTRODUCTION

Scheduling can be defined as a planning activity that assigns resources to particular tasks, and provides answers to questions such as when and how to process. It deals with utilization of the resources to meet the customers' demands in terms of time and quality. The aim is simply to optimize one or more objectives.

For more than fifty years, scheduling has been extensively studied by many researchers [1]. Most of these studies address the deterministic scheduling problems where all the relevant data (processing time, release date, due date and so forth) are known in advance without any uncertainty. However, in real life, productions are subject to many disruptions such as order cancellations, changes in due dates, process time variations, and rush orders, therefore, most of the schedules cannot be executed as they are planned.

The schedule that is yet to be sent to shop floor is called predictive schedule. Predictive schedules are exposed to alterations because of the nature of production environments. At the end of the day, a performed schedule is formed by the changes on predictive schedule. This performed schedule is called the realized schedule. It is not desirable to have significant deviations between the predictive and the realized schedules.

If the disturbances of production environments are not taken into consideration when planning, the generated schedules may perform poorly in practice. To handle the uncertainty in shop floor, different approaches have been proposed by researchers. One of the prominent approaches is called stochastic scheduling approach. Stochastic scheduling is concerned with problems in which at least one parameter is modeled as random variables. If the same scheduling problem has to be solved many times repetitively, the stochastic approach is reasonable as it takes expected realized performance minimization into account. On the other hand, if the schedule is to be solved only once, dealing with the expectation of a performance measure is a risky

approach. Moreover, in real life applications, it is hard to associate probability distributions to the underlying parameters which create additional burden for the stochastic approach. While expected system performance is typically the main concern for stochastic programming approach, the other approach which is called robust scheduling targets to find an acceptable performance [2]. A schedule which is insensitive to disruptions is called a robust schedule. The performance of the realized schedule in terms of the objective(s) is the main concern in robust scheduling [3]. Robust scheduling is studied from reactive and proactive standpoints [4]. In reactive scheduling, one takes actions when an unexpected event occurs which means the revision in schedule is made after the incident. However, in proactive scheduling, the uncertainty is taken into account beforehand. That is why in proactive scheduling, it is important to consider all the probable disruptions in the planning phase of the predictive schedule.

In Sabuncuoğlu and Gören's literature review [4], robustness measure is viewed in two categories. The first category is based on realized performances. The measures in this category can be listed as minimizing the expected performance measure, the worst-case performance (which is also called "absolute robustness" in [2]), the most probable performance and the variance of the realized performance. The second category is based on regret which is called robust deviation measure in [2]. Regret is the difference between the realized objective and the decision maker's predictive schedule's objective. In this category, the robustness measure is taken as minimizing the worst case regret, worst case scenario's regret and most probable scenario's regret.

For a better understanding for robust scheduling approach, consider a parallel machine environment with 5 jobs and 2 machines which are subject to random breakdowns as illustrated in Figure 1. 1. The performance measure is taken as the makespan which is the completion time of the last job.

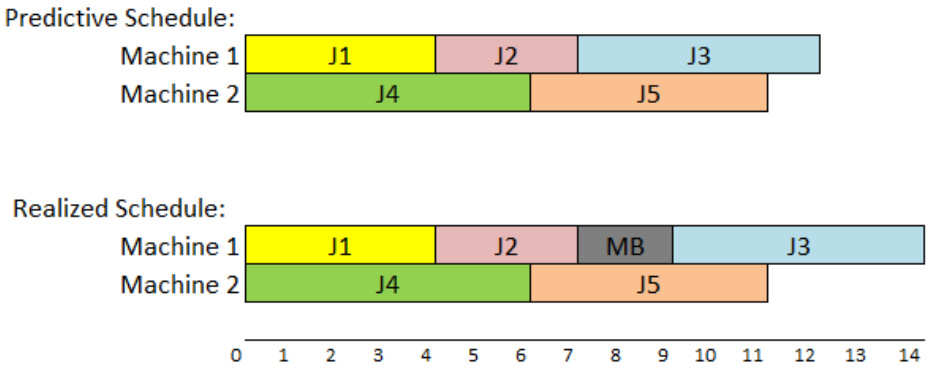

Figure 1. 1 Predictive and Realized Schedule for P2||Cmax Problem

In the predictive schedule, the first three jobs are to be processed in the first machine and the last two jobs are to be processed in the second machine making the makespan 12. In the realized schedule, we understand that a breakdown occurs between time 7-9 at machine 1. The makespan of the realized schedule is 14. From the realized performance perspective, the robustness measure is 14. On the other hand, from the regret perspective, the robustness measure is $14 - 12 = 2$.

In this study, we focus on stochastic proactive scheduling of parallel machine environment aiming to minimize makespan with processing time uncertainty. Deterministic version of this problem has been extensively studied for over 50 years yet efficient exact algorithm has not been found [5]. Thus, considering stochastic parameters to the problem makes our study more challenging but closer to the industrial applications. Hence, we seek optimal or near optimal solutions to the problem and target our methodology to be used by the industry since parallel machine environments are widely seen in real world manufacturing systems [1].

In parallel machine environment, the machines are identical and a number of jobs are considered to be processed. It can be recognized as a generalized single machine environment and also a special case of flexible job shop environment. This is why it is an important environment to consider theoretical approaches. In parallel machine problems, minimizing the completion time of the last job (makespan) is an important objective, since minimizing makespan balances the workload in each machine. This problem is denoted by $Pm||C_{max}$ and proven to be NP-Hard. Even without uncertainty or stochastic nature in the problem, it is impossible to find an optimal schedule in polynomial time unless $P = NP$. With the consideration of uncertainty, it will be even harder to find an optimal policy / schedule that gives a robust allocation for jobs to be

processed. In this thesis, we provide different approaches to solve the $Pm||C_{max}$ problem with uncertain processing times.

The rest of this thesis is organized as follows. In Chapter 2 we review the relevant literature. In Chapter 3, we propose the mathematical model and different heuristic approaches to approximately solve the proposed model. In Chapter 4, we adopt stochastic dual decomposition method proposed by Caroe and Shultz [6] to $Pm||C_{max}$ problem. We decompose the large problems into disjoint sub-problems, and then solve. These sub-problems are much easier to solve due to less variable and constraint numbers. At the end of this chapter, the results are compared with the model provided in Chapter 3. We provide gaps for the objective function values by applying this method with some alterations for large problems. Finally, we construct a new heuristic method that is based on the dual decomposition method. In Chapter 5, another heuristic method (tabu search algorithm) is proposed to handle even larger problems. The problems are solved using the proposed heuristic and the results are compared with the model provided in Chapter 3. In Chapter 6, we make some concluding remarks and point out future research directions.

Chapter 2

LITERATURE REVIEW

In this literature review, we focus on proactive scheduling studies with uncertainties as we study parallel machine environment from the proactive standpoint. To find broad information about stochastic and robust scheduling, we refer the readers to literature review studies [2], [4], [7], [8].

In this chapter, we briefly examine the proactive scheduling in both stochastic and robust studies. Next, we mention the studies in parallel machine scheduling with uncertainties. Finally, we continue with the contribution of this thesis to the existing literature.

2.1 Proactive Scheduling

In proactive scheduling, the decisions are made in the planning phase before the disruptions. We mention here two proactive scheduling approaches that generate solutions (relatively) resistant to changes.

The first approach is stochastic scheduling approach. In stochastic scheduling data uncertainty is represented by probability distributions. Hence, this approach is practical when the information about the data is available [9]. In stochastic scheduling, the aim is mostly to minimize the expected performance measure. In some studies, the probability of having the performance measure below some certain level is also studied and maximized.

Simple priority rules often lead to optimal results in the stochastic problems with exponentially distributed processing times even if the deterministic version is NP-Hard [1]. For instance, for the single machine problems with exponential processing times, weighted shortest expected processing time (WSEPT) rule minimizes the expected weighted tardy jobs [1]. For exponential processing time rate λ_j of job j, the WSEPT rule assigns jobs in the decreasing order of $w_j \lambda_j$ where w represents the weights in the

objective. For more rules that can be applied on stochastic scheduling problems, readers may refer to [1], and [10].

Minority of the studies addressing the processing time uncertainty in stochastic scheduling choose arbitrary distributions, whereas the majority choose practical distributions such as normal distribution [11]. In one of the recent studies, Baker studies single machine environment with jobs having normally distributed processing times. The objective is to minimize cost of tardy and early jobs. With a dominance rule and lower bounds, the author proposes a B&B algorithm to solve the problem. The proposed algorithm can solve up to 20 jobs within an hour.

The second approach to deal with uncertainty is robust scheduling approach. In robust scheduling, the objective functions can be expected performance measure, or regret based. The uncertainty is represented by interval data or scenarios.

In the interval data representation, any uncertain data can take a value within a given continuous interval independently. It is clear that in this representation of uncertainty, the upper bounds of the intervals result in worst case values which constitute the worst case scenario. For a robust problem minimizing the worst case scenario becomes the deterministic version of the problem. This is why in interval data approaches, regret is tried to minimized [2].

In a recent study [12], single machine scheduling with processing time uncertainty is studied to find the schedule that gives the robust (minimum of maximum deviation) total flow time schedule. The authors assume that the processing times are within specific intervals resulting in infinitely many possibilities and model the problem as a shortest path problem in order to find the worst case scenario used in the deviation. Since the model is NP Hard, the authors propose a simulated annealing heuristic method. The results show that the heuristic algorithm can solve large-sized problems up to 200 jobs.

In scenario based representation of uncertainty in robust scheduling, a scenario is defined as a possible realization of uncertain parameters. Suppose that in a problem, a random variable is denoted by x_j where $0 < j \leq r$ and the cardinality of the domain of x_j is finite (i.e. $|D_j| < \infty$). Each element in the Cartesian product $D_1 \times D_2 .. \times D_n$

represents a scenario. There are totally $\prod_{j=1}^{n}|D_j|$ number of scenarios. This method requires enumerating all the possible scenarios, and mostly constructing mixed integer programs that is using the scenarios.

An example to such approach is done by Kasperski and his friends [13]. This time, a permutation flow-shop environment is considered. The number of machines is set to 2 and the uncertain job processing times are modeled via discrete scenario set. The aim is to minimize the maximum regret of makespan. They have considered bounded and unbounded scenario sets for the problem and showed that for the unbounded case, 2-approximation algorithm can be found.

Up to now, we briefly examine the proactive scheduling studies. We continue the literature review with the parallel machine environment which is the production environment studied in this thesis.

2.2 Parallel Machine Scheduling with Uncertainties

There are a few studies addressing robust parallel machine scheduling. The first one to mention here is the study done by Ranjbar et. al. [14]. In their study, the aim is to find a robust schedule that maximizes the probability of having the makespan below a certain level when the processing times are uncertain. The job processing times are assumed to be normally distributed. This is the first study that takes stochastic processing times in a parallel machine environment into account. The authors develop two B&B algorithms along with six dominance rules to solve this problem. The proposed algorithms solved up to 20 jobs for 3, 4, and 5 machines.

A very similar objective function is studied by Alimoradi, Hematian and Moslehi [15] with maximizing the probability of having total flow time below a given level on parallel machines. The processing times of jobs are assumed to be distributed normally. A branch and bound algorithm is proposed along with a lower bound, three upper bounds, and two dominance rules. The algorithm can solve up to 45 jobs for 3, 4, and 5 machines. The authors also apply these algorithms to single machine problems showing that the study also outperforms the previous studies in literature.

In another study [16], parallel machine scheduling with uncertain job processing with interval data is studied. The robustness is defined by the min-max criterion which

gives the minimum maximal deviation of makespan from planned for possible scenarios. In the study, the problem is formulated as a mixed integer programming problem and solved via two exact algorithms by using general iterative relaxation method. A local search and a simulated annealing algorithm are proposed to solve large problems.

Hu, Ng and Qin [17] study a very similar problem with an addition of family set-up times. Each job has a family. Two consecutively scheduled jobs incur a set-up time if they come from different families. The study is done for plastic production, and the set-up times refer to mold changing time for different job families. An artificial bee colony algorithm is proposed as a heuristic approach along with an exact algorithm proposed by Xu et. al. [16], the results are compared and it is concluded that the heuristic algorithm performs well in terms of computational time and objective function.

Xu et. al. [18]. study the same problem with the same definition of uncertainties and robustness measure but minimizing total flow time as the objective function. They transform the problem formulation into a robust single-machine scheduling problem with fewer constraints and variables. It is shown that 2-approximation algorithm gives good results for large problems.

In the study of Kasperski et. al. [19], the uncertainty is modeled via discrete set of scenarios. The first robustness measure is taken as ordered weighted averaging aggregation (OWA). In OWA approaches, the aim is to find a schedule that gives the minimum value of weighted sum of each scenario's makespan. The second robustness approach is Hurwicz criterion which is finding a schedule that minimizes the weighted average of best and worst case scenarios' makespan values. The authors show that, for unbounded scenario sets, no polynomial approximation algorithm can be generated for OWA. For bounded cases, a pseudo-polynomial approximation can be done.

2.3 Contributions

The literature of parallel machine scheduling problem lacks of the following subjects:

1. As stated in Chapter 2.2, the processing time uncertainty is mostly represented by normal distribution or interval data. The only scenario based approach has been done with the aim of minimizing the weighted makespan. The scenario based approach on parallel machine scheduling with minimizing expected makespan is an untouched area.
2. In the literature review study of Li and Ierapetritou [9], the authors emphasized that computational cost is one of the main concerns in stochastic programming.

In our study, we model uncertainty on processing times via discrete set of scenarios in a parallel machine environment. We propose to decompose the stochastic program into several models in order to eliminate the computational cost. That is why we apply “Dual Decomposition Method” proposed by [6]. To the best of our knowledge, this is the first study that investigates impacts of “Dual Decomposition Method” in stochastic parallel machine scheduling problems.

Chapter 3

PROBLEM FORMULATION

In this study, we assume to have m identical parallel machines and n different jobs. The aim is to find the schedule that gives the minimum makespan which is the completion time of the job that is finished the last. The most important aspect of schedules in parallel machine with minimizing makespan is that the job order in machines does not affect the objective function value, this is why only the job allocation to machines are significant.

In our study, it is assumed that the randomness of the problem comes from the processing time uncertainty. The processing times are considered to have discrete triangular distribution, i.e. there are 3 different possible processing times for each job which we can call optimistic, pessimistic and neutral values. We also assume that preemption is not allowed which means that all the running jobs must be continued without interruptions.

3.1. Mathematical Model

The mathematical model without uncertainty which is called Deterministic Parallel Machines Scheduling Problem (DPMSP) is given below:

Mathematical Model of DPMSP:

Indices and Sets:

i represents machines $(1 \leq i \leq m)$

j represents jobs $(1 \leq j \leq n)$

Data:

p_j is the processing time of job j

Decision Variables:

x_{ij} : $\begin{cases} 1 & \text{if job } j \text{ is processed in machine } i \\ 0 & \text{otherwise} \end{cases}$

z : completion time of the last job (makespan)

Deterministic Model Formulation:

$$\min z \tag{3.1}$$

$$s.t.$$

$$z \geq \sum_{j=1}^{n} p_j x_{ij} \qquad \forall i \tag{3.2}$$

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad \forall j \tag{3.3}$$

$$x_{ij} \in \{0,1\} \qquad \forall i,j \tag{3.4}$$

In the model, (3.1) is to minimize makespan. Constraint (3.2) guarantees that z is the completion time of the last job. Constraint (3.3) ensures that each job is processed on only one machine and constraint (3.4) is for binary variables.

DPMSP is an Integer Programming model with $nm + 1$ variables and $m + n + mn$ constraints. This problem can be denoted as $Pm||Cmax$. The $P2||C_{max}$ (minimizing makespan with two identical parallel machine problem) is known to be NP-Hard [1]. As $P2||C_{max}$ is NP-Hard, $Pm||C_{max}$ is also NP-Hard.

For the mathematical model of stochastic parallel machine scheduling problem (SPMSP), scenario based approach for stochastic programming models is adopted. A scenario is defined as a possible realization of each job's processing times. Suppose that p_j in the deterministic model is a random variable and the cardinality of the domain of p_j is finite and equal to 3. (i.e. $|D_j| = 3$ for $\forall j$). The Cartesian product $D_1 \times D_2 .. \times D_n$ represents the scenario set, and each element represents a scenario. Since each job can have 3 different processing times, there are totally $\prod_{j=1}^{n} |D_j| = 3^n$ scenarios. The robustness measure of problem is taken as the expected performance measure (i.e. expected makespan) over the scenarios.

In order to deeply understand the mathematical model of SPMSP and scenario based approach; we first introduce generic stochastic programs with fixed recourse. In general, stochastic programs with recourse have 2 types of decision variables which are first and second stage variables. The first stage variables are the variables whose values should be determined before the uncertain event occurs. They are also called here-and-

now decisions. On the other hand, the second stage variables' values are determined after the uncertain event to make corrective actions [20]. Below, a generic form of stochastic program can be found where x denotes the first and y denotes the second stage variables.

General Stochastic Programs with Fixed Recourse:

$$\min\ z = \{cx + \mathbb{E}_{\zeta}[\min q_j y_j] : (x, y_j) \in S^j\} \tag{3.5}$$

where $S^j = \{(x, y_j) = Ax \leq b, x \in X,$

$$T^j x + W y_j \leq h_j, y_j \in Y\}$$

Here, $c \in \mathbb{R}^{n_1}, b \in \mathbb{R}^{m_1}, A \in \mathbb{R}^{m_1 \times n_1}$ and $W \in \mathbb{R}^{m_2 \times n_2}$ are known. For each random outcome j; $T^j \in \mathbb{R}^{m_2 \times n_1}$, $q_j \in \mathbb{R}^{n_2}, h_j \in \mathbb{R}^{m_2}$ get values accordingly. The expected value in the objective function (3.5) can be written as the summation of probability of each random outcome multiplied by $q_j y_j$ values when the outcomes can be written in discrete events. (i.e. scenarios).

So, the objective function can be replaced by:

$$\min z : \{cx + \sum_{j=1}^{N} p_j q_j y_j : (x, y_j) \in S^j\} \tag{3.6}$$

where p_j denotes the probability of occurrence of each scenario j. This notation is also called "Deterministic Equivalent" formulation.

For our problem SPMSP; we have adopted the same approach. The deterministic equivalent of SPMSP can be found below:

Deterministic Equivalent of SPMSP :

Indices and Sets:

i represents machines $(1 \leq i \leq m)$

j represents jobs $(1 \leq j \leq n)$

s represents scenarios $(1 \leq s \leq 3^n)$

Data:

p_{js} is the processing time of job j under scenario s

α_s is the occurrence probability of scenario s

Decision Variables:

x_{ij} : $\begin{cases} 1 & \text{if job } j \text{ is processed on machine } i \\ 0 & \text{otherwise} \end{cases}$

z_s : completion time of the last job (makespan) at scenario s

Deterministic Equivalent Formulation:

$$\min \sum_{s=1}^{3^n} z_s \alpha_s \qquad (3.7)$$

$s.t.$

$$z_s \geq \sum_{j=1}^{n} p_{js} x_{ij} \qquad \forall i, s \qquad (3.8)$$

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad \forall j \qquad (3.9)$$

$$x_{ij} \in \{0,1\} \qquad \forall i, j \qquad (3.10)$$

In this formulation, the first stage decision variables are x_{ij} and the second stage variables are z_s. The objective function (3.7) is the expected value of makespan over the scenarios. Constraint (3.8) is to calculate makespan values of each scenario, and (3.9) and (3.10) are the same constraints with (3.3) and (3.4) respectively.

A special case of this problem is where $\alpha_1 = 1$ and $\alpha_s = 0$ for $s \neq 1$. We know that this problem is the deterministic version of parallel machine scheduling with minimizing makespan which is NP-Hard. Hence the deterministic equivalent of stochastic parallel machine scheduling is also NP-Hard. Besides, the deterministic model has totally $3^n + nm$ decision variables and $3^n\, m + n + mn$ constraints making the model hard to solve.

3.2. Test Problems

In order to see the performance of the deterministic equivalent of SPMSP, we set up different problems. The problems are generated using the scheme proposed in [21].

First of all, the identical parallel machine number (m) is taken as 2, 3 and 5. The job number (n) is taken as 6,8,10 and 12. The expected values of the each job's processing times are sampled from a discrete uniform distribution with $U(1,100)$. The expected values denote the neutral processing time for each job. To find the pessimistic and optimistic values, 3 different variation calculations are done. The first one is called "low variation", the pessimistic and optimistic values are found by multiplying the probable processing time with 0.8 and 1.2 respectively. Similarly, for the medium level variation, the probable processing time is multiplied with 0.5 and 1.5, and for the high level variation, 0.2 and 1.8 are the multipliers. Here, the aim is to see whether the increase in variation effects the solution time or not. The probabilities of the optimistic, neutral and pessimistic values are taken as .25, .5 and .25 respectively. The probability of a scenario can be easily found by multiplication of each job's respective probability in the scenario. Table 3.2. 1 summarizes the problems:

Expected Processing Time (EPT)	**U[1,100]**
Processing Time Variation	Low: {EPT * 0.8 ,EPT, EPT *1.2} Medium : {EPT * 0.5 ,EPT, EPT *1.5} High : {EPT * 0.2 ,EPT, EPT *1.8}
Total number of jobs (n)	6,8,10,12
Machine number (m)	2,3,5

Table 3.2. 1 Problem set generation

For each problem type, 3 problems are generated. To give an example, for a problem type with 2 machines, 6 jobs and low variance of processing times, 3 different problems are created. Therefore, the problem set has totally 108 different problems.

3.3 Value of Stochastic Solution

In real life, processing time uncertainties are frequently disregarded for practical purposes. In such cases, the processing times are taken as the mean values. To understand if such approach can find the optimal job allocation or not, we fix each job's processing times to the neutral values. We solve the problems with these processing times as in deterministic version of the model. The solution (i.e. job allocation to the machines) is used to find the expected makespan of the deterministic problem. Table

3.3. 1 give the average objective function values of deterministic equivalent model of the problem (direct model) and expected makespan values of deterministic model.

		Direct Model			Deterministic Model		
n	m	Low Var.	Medium Var.	High Var.	Low Var.	Medium Var.	High Var.
6	2	176.36	188.09	200.31	176.36	188.09	200.31
6	3	122.92	137.35	152.38	123.39	137.61	152.59
6	5	90.38	101.62	115.75	91.42	104.56	119.58
8	2	222.79	236.40	250.03	222.79	236.40	250.03
8	3	153.68	170.26	186.87	153.71	170.36	187.05
8	5	102.03	117.64	134.55	102.33	117.91	134.82
10	2	264.02	278.80	293.57	264.02	278.80	293.57

Table 3.3. 1 Average objective function value of direct model and deterministic model

Please note that each cell in the table is the average value of 3 problems that are with the same variation type, machine and job number. All the tables are prepared within the same form. The problems are solved with maximum 4 threads, and 10 hours of time limitation at a computer with 64 GB RAM, AMD Opteron 6276 2.30 GHz Processor. All the calculations of this thesis are done on the same computer.

When we examine Table 3.3. 1, we can see that, the machine number plays an important role for the mean value algorithm. For 2-machine problems, mean value algorithm can find the optimal values whereas for 3 and 5 machine problem types, the average deviation from the optimal solution is up to 3.3% ($= 100\frac{(z-z^*)}{z^*}$ where z is the average objective function value of mean value algorithm and z^* is the optimal average objective function values for the problems).

The difference between the direct model and mean value algorithm is called value of stochastic solution (VSS) in stochastic programming. VSS is at most 9.053 for the third problem of type 6 jobs, 5 machine and high variation where the objective function value is 120.875 and the deviation is 7.5%. Also, the processing times used in the models are symmetrical. In the problems with nonsymmetrical processing times this range is expected to be higher. We can conclude from here that stochastic formulation of the problem becomes significant.

3.4 Alternative Algorithms and Computations

As stated before, the deterministic equivalent model has totally $3^n + nm$ decision variables and $3^n\, m + n + mn$ constraints. Especially an increase in job number results in exponential increase in both decision variables and constraint numbers which make the computations harder.

It is known that, for a problem with two machines and jobs with exponentially distributed processing times or jobs whose processing times are distributed according to a mixture of two exponential distributions; Longest Expected Processing Time (LEPT) rule minimizes the expected makespan [1]. In our problem, we rather have discrete triangular distribution, however we still examine the rule whether it performs well for our problem or not. In LEPT rule, the jobs are ordered decreasingly according to their expected processing time values (i.e. neutral values). The first m jobs are placed into m machines. Whenever a machine frees, the next job is assigned.

Besides LEPT rule, we also examine another algorithm (LEPT2) which takes the same ordering, but places jth job to ith machine where $i = j\ (mod\ m)$.

Table 3.4. 1 gives information about the average objective function values of direct model, LEPT and LEPT2 rules.

		Direct Model			LEPT			LEPT2		
n	m	Low Var.	Medium Var.	High Var.	Low Var.	Medium Var.	High Var.	Low Var.	Medium Var.	High Var.
6	2	176,4	188,1	200,3	176,4	188,1	200,3	189	195	204,9
6	3	122,9	137,4	152,4	122,9	137,4	152,4	133,4	143,9	157,3
6	5	90,4	101,6	115,8	90,4	101,6	115,8	109,1	114,7	126,4
8	2	222,8	236,4	250	223,7	236,8	250,3	230,6	240,1	252,4
8	3	153,7	170,3	186,9	156,5	171,7	187,9	175,5	184,4	197,4
8	5	102	117,6	134,5	102,0	117,6	134,5	124,8	135,6	149,6
10	2	264	278,8	293,6	266,0	279,6	294,1	275,6	284,5	297,3
10	3	NA	NA	NA	182,8	200,2	218,2	199,6	209,3	224,4
10	5	NA	NA	NA	118,5	135,9	154,5	130,6	145,1	162,4
12	2	NA	NA	NA	302,5	317,8	333,4	312,1	322,7	336,6
12	3	NA	NA	NA	207,7	226,6	245,8	225,5	236,2	252,3
12	5	NA	NA	NA	134,0	152,6	172,4	161,3	170,4	185,7

Table 3.4. 1 Objective function values of direct model, LEPT and LEPT2 rules

After carefully examining the data, we can make the following observations. The deviation from the optimal objective function values is up to 3.5% for LEPT (for second problem of 8 jobs, 3 machines and low variation problem type) and 22% for LEPT2 rules. It can be inferred from the table that LEPT rule performs better than LEPT2 although LEPT2 performs well in high variation problems. LEPT rule can find 29 optimal objective function values out of 63 exactly solved problems whereas LEPT2 rule cannot find any. LEPT rule performs almost excellent in 6 jobs problems, however for 8 jobs problems with 2 and 3 machines optimal solutions cannot be found.

Since the results of the LEPT and LEPT2 show that these rules cannot find optimal values in most of the problems, we propose two different heuristic algorithms and compare the results with the direct model. The first algorithm which is called random sampling algorithm (RSA) is randomly selecting 10% of the scenarios (totally $\left\lfloor \frac{3^n}{10} \right\rfloor$) and solving the deterministic equivalent model (3.7-3.10) with these selected scenarios. The main purpose of this algorithm is to decrease the scenario number to make the model analytically tractable. Secondly, we apply another approach which we call expected scenario algorithm (ESA). We consider each job's effect on the scenario set individually. We assume that, at each scenario, only one job can has a random value, and the rest of the jobs have the most probable values. Hence we decrease the number of scenarios to $3n$ since the distribution is symmetrical, we assume these scenarios can represent the original scenario set well.

Table 3.4. 2 gives information about the CPU times of the direct model, RSA and ESA. The problem set is solved with 10 hours duration and maximum 4 threat limits.

CPU Time		Direct Model			RSA			ESA		
n	m	Low Var.	Medium Var.	High Var.	Low Var.	Medium Var.	High Var.	Low Var.	Medium Var.	High Var.
6	2	3,1	3,3	2,7	0,2	0,2	0,2	0,3	0,4	0,3
6	3	5,4	6,9	7,1	0,3	0,4	0,3	0,3	0,5	0,6
6	5	3,4	7,7	8	0,8	0,8	0,6	0,5	0,4	0,6
8	2	134,3	121,4	152,4	3,3	3,6	4,5	0,3	0,5	0,4
8	3	243	235,4	237	9,6	7,7	11,9	0,9	0,9	1
8	5	607	785,7	382,1	18,8	21,7	18,9	1,4	1,9	1,7
10	2	21600,9	30169,3	28716,1	231,2	232,2	276,3	0,7	0,6	0,6
10	3	NA	NA	NA	697,2	692,9	791,5	1,1	1,2	1,1

10	5	NA	NA	NA	1273,6	2030,7	1676,5	1,4	1,7	1,8
12	2	NA	NA	NA	1554,2	1599,9	1814,9	0,9	1,1	1,2
12	3	NA	NA	NA	7652,8	6612,9	9831	1,6	2,1	2,4
12	5	NA	NA	NA	14788,1	18708	23416	2,3	2,7	2,4

Table 3.4. 2 Average CPU times of direct model, RSA and ESA

The values in each cell indicate the average CPU times of the problems with same n, m values and variation type. The CPU times of direct model increase exponentially as expected. After the problem type 10 jobs – 3 machines, it is not possible to solve in ten hours duration limit. Since the 10% of the scenario numbers also increases exponentially, the CPU time reaches to 6.5 hours in RSA. The increase in job and machine numbers do not affect the CPU times of algorithm 2 as they affect direct model and RSA since the scenario numbers in ESA increases in polynomially.

The average objective function values for each problem type are given below in Table 3.4. 3.

		Direct Model			RSA			ESA		
n	m	Low Var.	Medium Var.	High Var.	Low Var.	Medium Var.	High Var.	Low Var.	Medium Var.	High Var.
6	2	176,36	188,09	200,31	177,04	188,58	201,00	176,36	188,09	200,31
6	3	122,92	137,35	152,38	123,43	137,84	153,23	122,92	137,35	152,38
6	5	90,38	101,62	115,75	90,38	101,62	115,75	90,38	101,73	115,75
8	2	222,79	236,40	250,03	222,81	236,69	250,89	222,79	236,40	250,03
8	3	153,68	170,26	186,87	153,90	170,68	187,39	153,71	170,36	187,05
8	5	102,03	117,64	134,55	102,03	117,67	134,88	102,03	117,64	134,55
10	2	264,02	278,80	293,57	264,06	278,86	293,61	264,02	278,80	293,57
10	3	NA	NA	NA	181,93	199,83	217,97	181,91	199,82	217,87
10	5	NA	NA	NA	118,54	135,99	154,47	118,53	136,00	154,61
12	2	NA	NA	NA	301,95	317,63	333,35	301,95	317,62	333,29
12	3	NA	NA	NA	207,18	226,47	245,64	207,17	226,37	245,59
12	5	NA	NA	NA	131,62	151,18	171,48	131,67	151,34	171,38

Table 3.4. 3 Objective Function Values of Direct Model and RSA & ESA

Table 3.4. 3 illustrates that for the exactly solved problems (up to problems with 10 jobs and 3 machines) ESA finds mostly the same average objective function values with the direct model. The gap between ESA and direct model lies between 0 and 1.06%. The same for RSA is between 0 and 1.27%. When the results of the 45 unsolved

problems for RSA and ESA are compared, they have the same objective function value in 18 problems; ESA performs better than RSA in 19 problems and vice versa in 8 problems.

Although we can surely say that direct model gives superior solutions, we cannot distinguish algorithms in terms of objective function values. The solutions generated from RSA and ESA are very close to each other.

		Direct Model			RSA			ESA		
n	m	Low Var.	Medium Var.	High Var.	Low Var.	Medium Var.	High Var.	Low Var.	Medium Var.	High Var.
6	2	3	3	3	2	1	1	3	3	3
6	3	3	3	3	1	1	1	2	2	2
6	5	3	3	3	0	1	2	0	1	2
8	2	3	3	3	2	1	0	3	3	3
8	3	3	3	3	1	0	0	2	1	0
8	5	3	3	3	2	2	1	2	3	3
10	2	3	3	3	2	0	0	3	3	3

Table 3.4. 4 Number of optimal solution found in direct model, RSA and ESA

When we compare the solutions derived from the direct model and algorithms, we can claim that, ESA finds optimal solutions for 75% of the time (47 out of 63 solved problems). It is 33.3% for RSA, which makes ESA better in terms of finding the exact solution.

Even though, ESA has promising solutions in the problem types with 6 and 8 jobs, we are not able to compare the results of the problems sets with 10 and 12 jobs since the exact solutions of these problems cannot be generated. In order to compute results for problem types with 10 or more jobs, we propose to use “Dual Decomposition Method for Stochastic Programs” which is explained in the next chapter.

Chapter 4

STOCHASTIC DUAL DECOMPOSITION

In optimization, one of the main concerns is the time amount needed to solve large scale problems. Generally the increase in the number of the variables and constraints results in significant amount of time. Hence, the problem may become intractable. Over the last six decades, researchers try to decompose the analytically unsolvable problems to alleviate this difficulty.

The first decomposition method to mention here is Bender's decomposition method initially proposed by Benders [22]. In this method, the problem is separated into two problems which are called master and sub-problem. The master problem consists of complicating variables that makes the problem difficult to solve. Without the complicating variables, the problem becomes significantly easier. The sub-problem contains the rest of the variables. First the variables are fixed in the master problem and solved for the sub-problem. At each iteration, a feasibility cut and/or an optimality cut is added to the master problem. Optimal solution is found by solving these two problems iteratively until the objective function values match.

Benders decomposition method is commonly used in stochastic programming problems with recourse which have two variable types, first and second stage. In stochastic programming, Benders decomposition is often called as L-Shaped method as proposed in [23]. In this method, the first stage variables are considered as complicating variables. Therefore, the master problem contains the first stage variables whereas the sub-problem contains the second stage variables.

Integer L-Shaped method for integer stochastic programming problems is proposed by Laporte and Louveaux [24]. In this method, the main difference from the Bender's decomposition is that the first stage variables must be binary. In the method, a

new optimality cut is defined using these binary variables. Again like in the Bender's decomposition method, via the upper and lower bound generation, the exact solution can be found in finite number of iterations.

The other decomposition method to mention here is Dual Decomposition method proposed by Caroe and Shultz [6]. In this method, the original problem is aimed to be separated into sub-problems. Since the number of sub-problems is equal to the scenario number, we employ this method in order to find the exact solution of stochastic parallel machine scheduling problem. In Chapter 4.1, we give brief information and formulations of this method with binary first stage variables.

4.1. Stochastic Dual Decomposition in General for Binary First Stage Variables

Recall the deterministic equivalent of a general stochastic program given in Chapter 3.1.

Generic Deterministic Equivalent of Stochastic Programming Models

$$z = \max\left\{ cx + \sum_{j=1}^{r} p_j q_j y_j : (x, y_j) \in S^j \quad \text{for } j = 1..r \right\} \tag{4.1.1}$$

where $S^j = \{(x, y_j) = Ax \leq b, x \in X,$

$$T^j x + W y_j \leq h_j, y_j \in Y\}$$

In order to decompose the model into sub-problems, the copies of the first stage variables (x) are introduced. As the first stage variables do not depend on scenarios, the values of variable x must be equal to each other in each scenario. This is called "nonanticipativity" in stochastic programming.

Not to violate the nonanticipativity, a new constraint set (4.1.3) is added to the model:

$$\max z = \sum_{j=1}^{r} p_j \, (cx_j + q_j y_j) \; : (x_j, y_j) \in S^j \qquad (4.1.2)$$

$$\text{for } j = 1..r\}$$

$$x_1 = x_2 = \cdots = x_r \qquad (4.1.3)$$

where $S^j = \{(x, y_j) = Ax_j \leq b, x_j \in X,$

$$T^j x + W y_j \leq h_j, y_j \in Y\}$$

Here, since we have nonanticipativity conditions (4.1.3) and the summation of scenario probabilities is equal to 1, we can show that $= \sum_{j=1}^{r} p_j \, cx_j$. Hence, the objective functions do not change. Constraint (4.1.3) has $r - 1$ different equations. If the model have only binary first stage variables (i.e. x) , the nonanticipativity condition can also be satisfied using the following constraint:

$$\left(\sum_{j=2}^{r} a_j\right) x_1 = a_2 x_2 + \cdots + a_r x_r \quad where \quad a_j \in \mathbb{R}^+ \quad \forall j \qquad (4.1.4)$$

Here in (4.1.4), since x_1 is a binary variable, it can only take values of 0 or 1. Let's assume that x_1=0; then in order to have the right hand side of the equation as zero, all x_j's take value of zero. It is similar when x_1=1. With only one constraint, the nonanticipativity can hold.

With this notation, the model obviously has more constraints and variables than the original stochastic program, even though they represent the same problem. Yet, when the nonanticipativity constraint is dualized, Lagrangian relaxation can become a separable problem:

Lagrange Relaxation of Stochastic Program with Binary First Stage Variables:

$$D(\lambda) = \max\left\{\sum_{j=1}^{r} p_j\,(cx_j + q_j y_j) + \lambda\left(\sum_{j=2}^{r} a_j\left(x_1 - x_j\right)\right)\right\} \quad (4.1.5)$$

$$s.t. \qquad Ax_j \le b \qquad \forall j \qquad (4.1.6)$$

$$T^j x_j + W y_j \le h_j \qquad \forall j \qquad (4.1.7)$$

$$x_j \in \{0,1\}, y_j \in Y \qquad \forall j$$

where λ is unbounded.

As Lagrange relaxation models give an upper bound for maximization and lower bound for minimization problems, the model above gives an upper bound to the original model. This model can be separated into independent sub-models. By solving each model and summing the objective function values up, the solution can be found for a given λ.

Below we provide the **sub-model for j=1**.

$$\mathrm{z}_1 = \max\left\{\sum_{j=1}^{r} p_1(cx_1 + q_1 y_1) + \lambda\left(\sum_{j=2}^{r} a_j x_1\right)\right\} \quad (4.1.8)$$

$$s.t. \qquad Ax_1 \le b \qquad (4.1.9)$$

$$T^1 x_1 + W y_1 \le h_1 \qquad (4.1.10)$$

$$x_1 \in \{0,1\}, y_1 \in Y$$

And the **sub-models for $1 < j \le r$:**

$$\mathrm{z}_\mathrm{j} = \max\left\{\sum_{j=1}^{r} p_j\,(cx_j + q_j y_j) - \lambda a_j x_j\right\} \quad (4.1.11)$$

$$s.t. \qquad Ax_j \le b \qquad (4.1.12)$$

$$T^j x_j + W y_j \leq h_j \qquad (4.1.13)$$

$$x_j \in \{0,1\}, y_j \in Y$$

$D(\lambda)$ can be found by $D(\lambda) = \sum_{j=1}^{r} z_j$.

Since $D(\lambda)$ is an upper bound for the original maximization problem, we seek for the lowest upper bound to make the bound tighter. This is called the Lagrangian Dual Problem and it can be represented as:

$$z_{LD} = \min_{\lambda} D(\lambda) \qquad (4.1.14)$$

The λ value that gives z_{LD} can be found via a subgradient algorithm. The subgradient algorithm [25] is adapted for the decomposition method and can be found below:

Subgradient Algorithm for Dual Decomposition Method:

Initialization : $\lambda = \lambda^0$

Iteration k : $\lambda \leftarrow \lambda^k$

Decompose and solve $D(\lambda^k)$ with solution $x(\lambda^k)$ and $z(\lambda^k)$.

$\lambda^{k+1} = \max\{ (\lambda^k - \mu_k . \left(\sum_{j=2}^{r} a_j (x_1 - x_j)\right), 0\}$

$k \leftarrow k + 1$

In this method, it is clear that $\left(\sum_{j=2}^{r} a_j (x_1 - x_j)\right)$ is gradient of the objective function. At each iteration, λ is changed by a step value μ_k in the opposite direction of gradient. There are different methods to calculate μ_k value. In this study, we use

$$\mu_k = \epsilon_k \, [D(\lambda^k) - \overline{D}\,] / \left\| \left(\sum_{j=2}^{r} a_j (x_1 - x_j) \right) \right\|^2$$

Where $\epsilon_k \in (0,2)$ and $\overline{D}$ is an upper bound for z_{LD}. By this method, the algorithm finds the optimal value in finite number of iterations [25].

Although z_{LD} provides a good bound, it doesn't give the exact solution in general. To find the exact solution, a branch-bound algorithm is implemented and can be found below:

Branch and Bound Algorithm For Stochastic Dual Decomposition Method:

Step 1. Set solution of the stochastic problem $\underline{z} = 0$ and denote $\wp$ as the node set in the search tree.

Step 2. Terminate if there is no node in the node set $\wp$. The incumbent solution $\hat{x}$ that yields the objective value $\underline{z}$ is optimal.

Step 3. Select a node P from $\wp$, solve the z_{LD} of P ($z_{LD}(P)$) and delete it from the node set. Go to Step 2 when P is infeasible.

Step 4. If $\underline{z} \geq z_{LD}(P)$, go to Step 2. Else,

i. If the nonanticipativity holds (i.e. solution is feasible), update $\underline{z}$ by $\underline{z} = z_{LD}(P)$ and delete all the problems P' with $z_{LD}(P') \leq \underline{z}$, return back to Step 2.

ii. If the nonanticipativity does not hold, compute average $\bar{x}$, if $0 \leq \bar{x} \leq 0.5$, set $\bar{x}$ as 0, and 1 otherwise. Check the feasibility. If feasible, then compute $z_{LD}(P'')$ and update $\underline{z}$ by $\underline{z} = \max\{\underline{z}, z_{LD}(P'')\}$ and delete all the problems with lower lagrangian dual values than $\underline{z}$, Continue with Step 5.

Step 5. To branch, compute the average $\bar{x}$ of vector x and select a dimension x^i of vector x. Add two new problems with new constraints $x^i \leq \lfloor \overline{x^i} \rfloor$ and $x^i \geq \lfloor \overline{x^i} \rfloor + 1$. Continue with Step 2.

In this branch and bound procedure, the aim is to close the gap via finding better lower bounds by feasible integer solutions and upper bounds by the Lagrangian dual values iteratively.

4.2. Stochastic Dual Decomposition for Robust Parallel Machine Scheduling Problem with Minimizing Makespan

Recall the deterministic equivalent of SPMSP :

Deterministic Equivalence Formulation of SPMSP

Indices and Sets:

i represents machines $(1 \leq i \leq m)$

j represents jobs $(1 \leq j \leq n)$

s represents scenarios $(1 \leq s \leq 3^n)$

Data:

p_{js} is the processing time of job j under scenario s

α_s is the occurrence probability of scenario s

Decision Variables:

$$x_{ij} : \begin{cases} 1 & \text{if job } j \text{ is done in machine } i \\ 0 & \text{otherwise} \end{cases}$$

z_s : completion time of the last job (makespan) at scenario s

Deterministic Equivalent Model Formulation:

$$\min \sum_{s=1}^{3^n} z_s \, a_s \qquad (4.2.1)$$

$$s.t. \qquad z_s \geq \sum_{j=1}^{n} p_{js} x_{ij} \qquad \forall i, s \qquad (4.2.2)$$

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad \forall j \qquad (4.2.3)$$

$$x_{ij} \in \{0,1\} \qquad \forall i, j \qquad (4.2.4)$$

To use the dual decomposition method, first stage variables x_{ij} are copied and changed to x_{ijs}. The following model is equivalent of the previous one. Note that we only change variable x_{ij}, the rest of the data and variables remain the same.

$$x_{ijs} : \begin{cases} 1 & \text{if job } j \text{ is done in machine } i \text{ at scenario } s \\ 0 & \text{otherwise} \end{cases}$$

Model with copied first stage variables:

$$\min \sum_{s=1}^{3^n} z_s \, a_s \tag{4.2.5}$$

$$s.t. \qquad z_s \geq \sum_{j=1}^{n} p_{js} \, x_{ijs} \qquad \forall i, s \tag{4.2.6}$$

$$\sum_{i=1}^{m} x_{ijs} = 1 \qquad \forall j, s \tag{4.2.7}$$

$$\left(\sum_{s=2}^{3^n} a_s \right) x_{ij1} = \sum_{s=2}^{3^n} a_s x_{ijs} \qquad \forall i, j \tag{4.2.8}$$

$$x_{ijs} \in \{0,1\} \qquad \forall i, j, \text{s} \tag{4.2.9}$$

By dualizing the nonanticipativity constraint (4.2.8) with a Lagrange multiplier λ, a Lagrange relaxation (LR) model is obtained. The objective function of this LR is given below; the constraints are the same constraints with (4.2.6), (4.2.7) and (4.2.9)

$$\mathrm{Z_{LR}} = \min \sum_{s=1}^{3^n} z_s \, a_s + \sum_{j=1}^{n} \sum_{i=1}^{m} \lambda_{ij} \left(\sum_{s=2}^{3^n} a_s \left(x_{ij1} - x_{ijs} \right) \right) \tag{4.2.10}$$

Since LR is separable into independent 3^n sub-problems, we decompose it. There are two types of sub-problems for $s = 1$ and $s \neq 1$ which can be found below:

Sub Problem with $s = 1$:

$$Z_{LR_1} = \min a_1 z_1 + \sum_{j=1}^{n} \sum_{i=1}^{m} \lambda_{ij} \, (1 - a_1) x_{ij1} \tag{4.2.11}$$

$$s.t. \qquad z_1 \geq \sum_{j=1}^{n} p_{j1} x_{ij1} \qquad \forall i \tag{4.2.12}$$

$$\sum_{i=1}^{m} x_{ij1} = 1 \qquad \forall j \qquad (4.2.13)$$

$$x_{ij1} \in \{0,1\} \qquad \forall i,j \qquad (4.2.14)$$

Sub Problem with fixed $s \neq 1$:

$$Z_{LR_s} \min a_s z_s - \sum_{j=1}^{n}\sum_{i=1}^{m} \lambda_{ij}\ a_s x_{ijs} \qquad (4.2.15)$$

$$s.t. \qquad z_s \geq \sum_{j=1}^{n} p_{js}\, x_{ijs} \qquad \forall i \qquad (4.2.16)$$

$$\sum_{i=1}^{m} x_{ijs} = 1 \qquad \forall j \qquad (4.2.17)$$

$$x_{ijs} \in \{0,1\} \qquad \forall i,j \qquad (4.2.18)$$

The solution to the LR can be found by the following equation:

$$Z_{LR(\lambda)} = \sum_{s=1}^{3^n} Z_{LR_s} \qquad (4.2.19)$$

As discussed in Chapter 4.1, finding $Z_{LR(\lambda)}$ for a fixed λ value can only give a lower bound for the deterministic equivalent of SPMSP problem. Since we are looking for Lagrangian dual $z_{LD} = \min_{\lambda} Z_{LR(\lambda)}$ the following sub-gradient algorithm is employed:

Pseudo Code for Subgradient Algorithm of Dual Decomposition Method (A):

Initialization : $\lambda_{ij} = \lambda_{ij}{}^{0}$

Iteration k : $\lambda_{ij} \leftarrow \lambda_{ij}{}^{k}$

Decompose and solve $Z_{LR}(\lambda_{ij}{}^{k})$ with solution $x(\lambda_{ij}{}^{k})$ and $z(\lambda_{ij}{}^{k})$.

$\lambda_{ij}{}^{k+1} = \max\{ (\lambda_{ij}{}^{k} - \mu_k . (\sum_{s=2}^{3^n} a_s (x_{ij1} - x_{ijs})), 0\}$

$k \leftarrow k + 1$

where $\mu_k = [\overline{D} - Z_{LR}(\lambda_{ij}{}^k)]/\left\|\left(\sum_{s=2}^{3^n} a_s(x_{ij1} - x_{ijs})\right)\right\|^2$

and the upper bound is taken as $\overline{D} = \sum_{j=1}^{n} p_{j3^n}$ which denotes that all the jobs have the pessimistic values and operated on the same machine. Via this algorithm, a better lower bound can be generated. This lower bound can be used in the branch and bound generated for the dual stochastic decomposition method given below . Via the branch and bound algorithm, the exact solution of the deterministic equivalent of SPMSP can be found.

Dual Decomposition B&B Algorithm Pseudo Code (B):

Step 1. Set solution of the stochastic problem $\underline{z} = +\infty$ and denote $\wp$ as the node set in the search tree

Step 2. Terminate if there is no node in the node set $\wp$. The incumbent solution $\hat{x}$ that yields objective value $\underline{z}$ is optimal.

Step 3. Select a node P from $\wp$, solve the z_{LD} of P ($z_{LD}(P)$) by (A) and delete it from the problem set. Go to Step 2 when P is infeasible.

Step 4. If $\underline{z} \leq z_{LD}(P)$, go to Step 2. Else,

i. If the nonanticipativity holds (i.e. solution is feasible), update $\underline{z}$ by $\underline{z} = z_{LD}(P)$ and delete all the problems P' with $z_{LD}(P') \geq \underline{z}$, return back to Step 2.

ii. If the nonanticipativity does not hold, compute average $\overline{x^i}$, if $0 \leq \overline{x^i} \leq 0.5$, set x^i as 0, and 1 otherwise. Check feasibility. If feasible, then compute objective function z of this new problem and update $\underline{z}$ by $\underline{z} = \min\{\underline{z}, z\}$ and delete all the problems with higher Lagrangian dual values than $\underline{z}$, Continue with Step 5.

Step 5. To branch, compute the average $\overline{x^i}$ values, select the most non-integer $\overline{x^i}$ (the closest one to 0.5) and add two new problems with new constraints $x^i = 0$ and $x^i = 1$. Continue with Step 2.

4.3. Computational Results of Stochastic Dual Decomposition Method

We apply the dual decomposition method for the unsolved problems with 10 jobs. Since we are constructing sub-models according to the number of scenarios and using a slow convergence method –subgradient algorithm, finding solution for the dual decomposition is time consuming. We try solving small problems via the dual decomposition method. Although it can generate the exact solutions, the CPU time needed to solve is much more than the original solution. For instance, the CPU time for the first problem of 6 jobs – 2 machine and low variance problem is 698 sec whereas it is 2,76 sec for the direct model. When we try solving large and unsolved problems via the dual decomposition method, we cannot find exact solutions. This is why we limit the subgradient algorithm to two iterations (rather than finding the Lagrangian Dual) and only solve the first node at the B&B tree. For the same time limit, we compare gaps of the direct model and dual decomposition model in order to see which one gets closer to the optimal value. Gaps are calculated from the formulation

$$\text{Gap} = \frac{\text{Upper Bound} - \text{Lower Bound}}{\text{Lower Bound}}$$

Both direct model and the dual decomposition method are coded in C++ using Cplex 12.6. The time limitation is 10 hour CPU time. 1 thread is used. In Table 4.3. 1 the gaps and CPU times for the dual decomposition and direct model are provided. These experiments are only done for the problems with 10 jobs and 3 and 5 machines. For larger problems, it is not practical to find gap values.

			Direct Model			Dual Decomposition			
n	m	Variation	Time	GAP	Incumbent Solution	Time	GAP	Incumbent Solution	Lower Bound
10	3	Medium	3600	5%	216,67	34308	14%	216,461	186
10	3	Medium	3600	15%	213,64	33763	14%	207,652	177,7
10	3	Medium	3600	8%	175,74	32779	15%	175,303	149,4
10	3	High	3600	5%	236,28	34314	21%	235,561	186,7
10	3	High	3600	75%	528,36	34109	21%	226,603	178,6
10	3	High	3600	25%	198,09	32600	21%	191,65	150,5
10	5	Low	3600	79%	552,36	40836	10%	133,314	119,7
10	5	Low	3600	6%	122,92	41405	6%	122,092	114,5
10	5	Low	3600	79%	445,36	40822	7%	102,948	96,01
10	5	Medium	3600	93%	552,36	43279	19%	148,529	120,4
10	5	Medium	3600	5%	141,18	41739	15%	140,765	120,2
10	5	Medium	3600	47%	187,49	41026	17%	119,074	99,33
10	5	High	3600	75%	552,36	42891	25%	170,767	127,8
10	5	High	3600	35%	220,30	41762	17%	160,734	133,3
10	5	High	3600	75%	445,36	40051	18%	135,897	111

Table 4.3. 1 Gaps for the large problems by dual decomposition and direct model

The results show that, 11 problems out of 15 are resulted in better gaps by using dual decomposition method than direct model. For the same type problems, dual decomposition method almost yields similar gaps. On the other hand, gaps for the direct model fluctuate a lot for the same problem types. See for instance problem type 10 jobs 5 machine and low variation. The gaps yielded from direct model are 79%, 6% and 79%.

From here we can conclude that, for 10 jobs problems, dual decomposition method performs better generally in terms of generating solutions that are closer to the optimal solutions.

4.4. A New Heuristic with Dual Decomposition Method

In Chapter 4.3, it is observed that the computations of the dual decomposition method take more time than the direct model in small problems. In this chapter, we combine this method with the random selection method as applied in Chapter 3 for the RSA to create a new heuristic procedure that can give approximate solutions. In order to do this, 5% of the scenarios ($\left\lfloor \frac{(3^n)}{20} \right\rfloor$) are randomly selected and the dual decomposition

method is applied with these selected scenarios. Table 4.4. 1 gives the summary of the objective function values of the new heuristic method and the direct model of the problems.

		Dual Decomposition Heuristic Method (incumbent solution)			Direct Model		
n	m	Low Var.	Medium Var.	High Var.	Low Var.	Medium Var.	High Var.
6	2	176,83	208,09	201,26	176,36	188,09	200,31
6	3	123,88	142,83	158,82	122,92	137,35	152,38
6	5	90,51	101,62	121,43	90,38	101,62	115,75
8	2	234,52	190,31	202,89	222,79	190,14	201,28
8	3	124,92	138,40	152,34	123,54	137,18	150,84
8	5	83,87	99,86	NA	80,16	93,80	134,55
10	2	265,16	285,06	294,15	264,02	278,80	293,57
10	3	NA	NA	217,61	NA	NA	NA
10	5	NA	143,43	200,28	NA	NA	NA
12	2	309,64	323,06	354,23	NA	NA	NA
12	3	NA	NA	NA	NA	NA	NA
12	5	NA	NA	220,41	NA	NA	NA

Table 4.4. 1 Average objective function values of dual decomposition heuristic method and direct model

The gap between the direct model and dual decomposition heuristic lies between 0% and 6.47% for the solved problems. This interval is in fact larger than the interval observed from the RSA - random selection method in Chapter 3. The reason for a larger interval is that in this method, 5% of the scenarios are selected because of computational reasons, where in RSA, it is 10%. The more the scenarios selected, the closer the gap is generated to the exact solution.

The CPU times of the dual decomposition heuristic method are given in Table 4.4. 2. Up to 8 jobs, only for problems with 8 jobs, 5 machines and low variation, the heuristic method can find solution faster than the direct model with 4.5% gap. For the problems having more than 8 jobs, the heuristic method performs better in terms of CPU times. It can be concluded that, this method can be applied for the problem types 10 job – 2 machine and 12 job – 2 machine. When the problem size is large, the method cannot generate solutions in 3 and 5 machine problems.

		Dual Decomposition Heuristic Method			Direct Model		
n	m	Low Var.	Medium Var.	High Var.	Low Var.	Medium Var.	High Var.
6	2	53,79	254,61	5797,65	3,13	3,34	2,69
6	3	4318,04	678,81	155,11	5,35	6,86	7,05
6	5	5,21	14,03	13,42	3,40	7,69	7,99
8	2	31235,97	1734,49	29467,5	134,30	121,36	152,42
8	3	6001,51	6461,06	1489,18	243,01	235,40	237,03
8	5	525,9	570,45	NA	606,96	785,67	382,06
10	2	8339,92	8488,94	7970,58	21600,93	30169,3	28716,11
10	3	NA	NA	7705,39	NA	NA	NA
10	5	NA	7736,39	8747,97	NA	NA	NA
12	2	23213,93	20515,6	21450,87	NA	NA	NA
12	3	NA	NA	NA	NA	NA	NA
12	5	NA	NA	NA	NA	NA	NA

Table 4.4. 2 Average CPU times of dual decomposition heuristic method and direct model

Via this heuristic method, we are able to generate solutions for small problems; however the CPU time needed is higher than the direct model. For the large problems we are not able to generate solutions. In order to find better objective function values with lower CPU times, we have tried a new heuristic approach in Chapter 5.

Chapter 5

HANDLING LARGE PROBLEMS

Heuristic algorithms can be applied to computationally expensive problems when the approximate solutions are acceptable. They are faster than the exact solution generating algorithms such as branch-bound or branch-cut. However; a heuristic algorithm cannot guarantee to find the exact solution of a problem. A local optimal solution is generated, it may be the global solution or not. The aim in finding such approximate solutions is to find with small deviations to the global optimal solution.

5.1. Tabu Search Algorithm

In our problem, we observe that finding the exact solution is expensive in terms of computational time as it is in NP-Hard problems. We develop a tabu search algorithm (TSA) to find an approximate solution for large problems. TSA has firstly presented in the study of Glover [26]. The algorithm aims to avoid trapping into the local optimal solutions by punishing or restricting some certain moves and allowing the non-improving ones.

There are some basic elements of TSA. The first one to mention is neighborhood structure. A neighbor set of a feasible solution is all the solutions that can be reached within a move. In this algorithm, the most important step is defining moves and neighbor sets. For instance, a move can be swapping the orders of cities to be visited in traveling salesman problem. By using this move, a neighbor set can be described as solutions that have only switched two cities' positions.

A second basic element of a TSA is tabu moves. A tabu move prevents the algorithm from cycling by not allowing going back to the local optimal solution. According to the problem type, a move or the reverse of it is banned in the algorithm

even though it can produce better results. For a better understanding of tabu moves, Figure 5.1. 1 is given below.

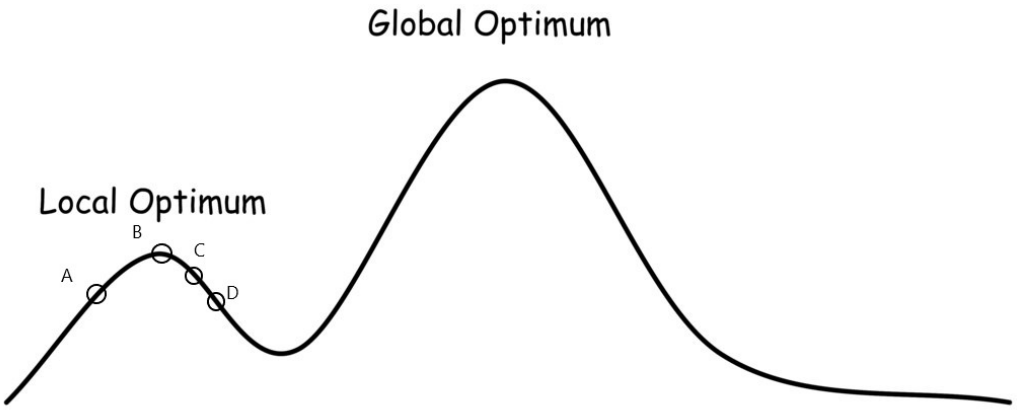

Figure 5.1. 1 Illustration of tabu moves in TSA

In this illustrative example, a move is generated from solution A to solution B as it provides the best solution among the neighbors. At the local optimum solution B, even though staying at B provides the best solution among the neighbors of B, a move must be done among the neighbors and C is selected as the better resulting solution. At solution C, B is clearly in the neighbor set and has a better solution than all the other solutions in the neighbor set of C. However, going back to B is defined as a tabu move and forbidden. This way, the algorithm prevents cycles to the local optimal solutions. From solution C, it is continued with solution D and eventually global optimum can be reached.

The tabus are listed in a tabu list, and after a certain number of iterations, the last tabu is deleted from the tabu list to allow the move. This number is called tabu tenure.

Another basic element of TSA is aspiration criteria. In some cases, even though there is no risk of cycling back to the local optimal solution, a better move is banned because of the tabu list. In these situations, an aspiration criteria is defined to enable the algorithm to go to the better solutions. A well known aspiration criteria is to make the move when the new solution is better than the current best solution even though it is a tabu move.

TSA can be terminated when a specific number of iterations is reached, or when there is no significant improvement for a number of iterations.

A basic TSA for a minimization problem can be found below:

Step 0. Set current best solution $z = +\infty$..

Step 1. Set $k = 1$, Find an initial solution z_1. Update z=min{z, z_1}

Step 2. Find the neighbors $N(z_k)$. Select best one of the neighbors and the aspiration criteria. Update tabu list and z and $k \leftarrow k + 1$.

Step 3. If the termination criterion is met, stop. If not, go to Step 2.

5.2. TSA for Stochastic Parallel Machine Scheduling Problem with Minimizing Makespan

To apply TSA to our problem, we first define moves. There are two candidate moves that could be used in TSA algorithm. The first one is swapping. By swapping, the assignment of two jobs on two different machines will be reversed. For instance, suppose that $x_{i_1 j_1} = 1$ and $x_{i_2 j_2} = 1$; before the swapping, job j_1 and j_2 are scheduled on machines i_1 and i_2 respectively. After the swap operation, job j_1 and j_2 are scheduled on machines i_2 and i_1 respectively. However, by swapping, the layout of the schedule will not be effected and changed. In Figure 5.2. 1, an illustrative example is given. In the upper figure, jobs 1,2, and 3 are processed at machine 1, jobs 4 and 5 are processed at machine 2 and job 6 is processed at machine 3. A swapping operation is done for job 3 and 5. The lower figure gives the schema of the after swapping schedule. Note that after swapping operations, there are 3 jobs in the first, 2 jobs in the second and 1 job in the last machine.

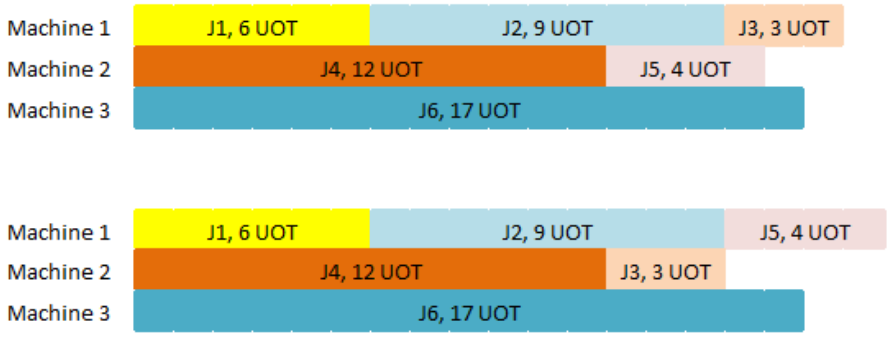

Figure 5.2. 1 The scheduling layout example for before (up) and after (down) swapping

Another candidate move operation is inserting. By inserting, a job is removed from its scheduled machine and put into another one. For a job j and machine i, if $x_{ij} = 1$, after the move, it will be $x_{ij} = 0$, and $x_{ik} = 1$, where $j \neq k$. In our study, we have selected inserting as the first move. At each solution, first the neighbors are found via inserting. The best neighbor is checked if it is a tabu or not. If it is not a tabu move, this solution is locally improved swap moves if it is possible. If the move is a tabu, it is also compared with the current best solution. It is selected if tabu solution is better than the current best solution, then the swapping operations is controlled.

The pseudo code of the algorithm can be found below:

Step 0. Set current best solution $z = +\infty$..

Step 1. Set $k = 1$ and $= 0$, Find an initial solution z_1. Update z=min{z, z_1}

Step 2. Find the neighbors $N(z_k)$ by inserting move.

Step 3. Select the best neighbor

i. If it is not a tabu move, move and check the swap operations. If there is any improvement in swapping, move and update z. Go to Step 4.
ii. If it is a tabu move, compare with the current best solution. If tabu is better than the current best solution, move and update z, check the swap operations. If there is any improvement in swapping, move and update z and go to Step 4. If tabu move is not better than the current best solution, delete the move from the neighbor list, and go back to Step 3.

Step 4. Add the inserting move to the tabu list as the first element, and delete the last element of the tabu list. $k \leftarrow k + 1$.

Step 5. If z is not improved, set $t \leftarrow t + 1$. If z is improved, set $t \leftarrow 0$.

Step 6. If $t \geq T$, terminate. If not, go to Step 2.

where T indicates the maximum allowed iteration number without improvement. The initial solutions are generated by firstly scheduling all the jobs to the first machine and a solution is found. Secondly, all the jobs are randomly allocated to the machines.

Thirdly, LEPT2 rule is assigned. Finally, since ESA algorithm performs well in less than 3 seconds, as the initial solution, jobs are allocated according to the solution generated from ESA.

In the algorithm, two alterations can be done, the first is deciding the tabu list size (tabu tenure) and the second is allowed iteration number to the next improving solution.

To find if there is a dominant starting approach, we solve all the problems with tabu tenure and allowed iteration number equal to 3 and 15 respectively. Table 5.2. 1 gives the average objective function values of different starting approaches. When we examine the table, it can be inferred that ESA outperforms the other starting approaches generally. Out of 36, only in 5 average objective function values ESA could not find a better solution than the other methods. The bald cells are the ones where ESA performs equal or better than the other starting approaches. When we compare the CPU times of the different start approaches, there is no significant dominance relation. This is why we continue our TSA algorithms with different tabu tenure and allowed iteration numbers for ESA start approach.

For the tabu tenure, we examine the solutions with 3, 4 and 5. Since in the inserting moves, the neighbor set has $(m-1)n$ elements, the minimum neighbor set has 6 elements in the 6 jobs- 2 machine problems. This is why we limited the maximum tabu tenure size to 5. The allowed iteration number is taken as 15 and 20.

		All Jobs in Same Machine			Random Assignment			LEPT2 Rule Start			ESA Start			Exact		
n	m	Low Var.	Medium Var.	High Var.	Low Var.	Medium Var.	High Var.	Low Var.	Medium Var.	High Var.	Low Var.	Medium Var.	High Var.	Low Var.	Medium Var.	High Var.
6	2	176,36	188,09	200,31	176,36	188,09	200,31	176,36	188,09	200,31	**176,36**	**188,09**	**200,31**	176,36	188,09	200,31
6	3	122,92	137,35	152,38	122,92	137,35	152,43	122,92	137,35	152,38	**122,92**	**137,35**	**152,38**	122,92	137,35	152,38
6	5	90,38	101,62	115,75	90,38	101,62	115,75	90,38	101,62	115,75	**90,38**	**101,62**	**115,75**	90,38	101,62	115,75
8	2	223,02	236,49	250,09	222,88	236,44	250,05	222,85	236,42	250,04	**222,79**	**236,40**	**250,03**	222,79	236,40	250,03
8	3	153,71	170,26	186,87	153,71	170,28	186,89	153,68	170,26	186,87	153,71	170,28	186,89	153,68	170,26	186,87
8	5	102,03	117,64	134,55	102,81	117,99	134,82	102,03	117,64	134,55	**102,03**	**117,64**	**134,55**	102,03	117,64	134,55
10	2	264,02	278,80	293,57	264,02	278,80	293,57	264,02	278,80	293,57	**264,02**	**278,80**	**293,57**	264,02	278,80	293,57
10	3	182,06	199,88	217,90	182,05	199,96	217,93	181,94	199,89	217,89	**181,91**	**199,82**	**217,87**	NA	NA	NA
10	5	118,53	135,99	154,50	118,53	135,88	154,48	118,53	135,88	154,48	**118,53**	**135,88**	**154,48**	NA	NA	NA
12	2	301,95	317,62	333,29	301,95	317,62	333,29	301,95	317,62	333,29	**301,95**	**317,62**	**333,29**	NA	NA	NA
12	3	207,18	226,34	245,53	207,18	226,37	245,56	207,18	226,36	245,57	**207,17**	226,35	245,57	NA	NA	NA
12	5	131,92	151,47	171,61	132,07	151,29	171,69	131,85	151,43	171,56	**131,62**	**151,27**	**171,32**	NA	NA	NA

Table 5.2. 1 Comparison of different starting approached in terms of average objective function values

Table 5.2. 1 gives the information about the average objective function values of different tabu tenure and allowed iteration numbers. From the table, we see that only for 2 problem types (8 jobs, 3 machines with medium and high variations) TSA algorithms of tabu tenure with 4 and 5 outperforms TSA of tabu tenure with 3.

		15-3 & 20-3			15-4 , 15-5, 20-4 & 20,5		
n	m	Low Var.	Medium Var.	High Var.	Low Var.	Medium Var.	High Var.
6	2	176,36	188,09	200,31	176,36	188,09	200,31
6	3	122,92	137,35	152,38	122,92	137,35	152,38
6	5	90,38	101,62	115,75	90,38	101,62	115,75
8	2	222,79	236,40	250,03	222,79	236,40	250,03
8	3	153,71	170,28	186,89	153,71	170,26	186,87
8	5	102,03	117,64	134,55	102,03	117,64	134,55
10	2	264,02	278,80	293,57	264,02	278,80	293,57
10	3	181,91	199,82	217,87	181,91	199,82	217,87
10	5	118,53	135,88	154,48	118,53	135,88	154,48
12	2	301,95	317,62	333,29	301,95	317,62	333,29
12	3	207,17	226,35	245,57	207,17	226,35	245,57
12	5	131,62	151,27	171,32	131,62	151,26	171,32

Table 5.2. 2 Average objective function values for different tabu list size and allowed iteration numbers

When we compare the CPU times of the TSA algorithms with different tabu tenure size and allowed iteration numbers in Table 5.2. 3, it is clear that there is a significant rise in TSA's with allowed iteration number 20. Since the objective function values are the same for the TSA's 15-4, 15-5, 20-4, 20-5 we restrict ourselves with the choices 15-4 and 15-5. When we compare the CPU times of these TSA's, we can say that, for high variation problems, 15-5 can find the same solution faster than 15-4. For medium variation it is vice-versa.

Since for the large problems, 15-5 performs better generally in terms of CPU times, we select 15-5 as the best TSA algorithm for stochastic parallel machine scheduling problem.

		TSA with 15-3			TSA with 15-4			TSA with 15-5			TSA with 20-3			TSA with 20-4			TSA with 20-5		
n	m	Low Var.	Medium Var.	High Var.	Low Var.	Medium Var.	High Var.	Low Var.	Medium Var.	High Var.	Low Var.	Medium Var.	High Var.	Low Var.	Medium Var.	High Var.	Low Var.	Medium Var.	High Var.
6	2	0,1	0,1	0,1	0,2	0,2	0,2	0,2	0,2	0,2	0,1	0,1	0,1	0,2	0,2	0,2	0,1	0,2	0,2
6	3	0,2	0,2	0,2	0,3	0,2	0,2	0,3	0,3	0,3	0,2	0,2	0,2	0,4	0,4	0,4	0,4	0,4	0,4
6	5	0,3	0,3	0,4	0,3	0,4	0,5	0,4	0,4	0,5	0,3	0,3	0,3	0,6	0,6	0,6	0,6	0,6	0,6
8	2	1,1	1,1	1,1	1,5	1,1	1,3	1,7	1,1	1,1	2,6	2,6	2,6	2,6	2,6	2,6	2,5	2,6	2,6
8	3	1,8	1,9	2,0	2,4	3,2	2,9	1,8	2,4	2,4	4,3	4,5	4,8	3,7	4,3	5,5	2,4	3,0	3,0
8	5	4,2	4,7	4,4	4,1	5,3	5,3	3,0	3,0	3,1	3,8	4,2	4,8	5,8	3,9	6,2	3,8	6,2	3,9
10	2	38	38	41	38	32	36	28	38	36	43	33	42	56	34	34	35	30	36
10	3	57	41	42	50	43	44	64	40	42	66	63	78	62	63	62	66	69	55
10	5	87	121	118	85	82	92	93	99	80	125	95	90	116	96	141	112	123	142
12	2	398	559	560	352	458	732	385	426	485	549	668	466	522	613	1042	579	656	626
12	3	966	829	882	935	944	1019	895	1030	722	1179	1294	1065	1220	966	1003	1327	866	1051
12	5	1672	1794	1989	1321	1281	1409	1285	1389	1172	1802	2320	2511	1977	2372	2572	1874	2368	2589

Table 5.2. 3 Average CPU times of TSA algorithm with different tabu tenure size and allowed number of iterations

Up to this part of the thesis, TSA with tenure size 5 and allowed number of iteration 15 with ESA start performed well. Also, the performance of ESA is remarkable. We check and compare the objective function values and CPU times of these two algorithms in order to find out the best solution method. Table 5.2. 4 gives the average objective function values of these two algorithms with direct model.

		Direct Model			TSA with 15-5 and ESA start			ESA		
n	m	Low Var.	Medium Var.	High Var.	Low Var.	Medium Var.	High Var.	Low Var.	Medium Var.	High Var.
6	2	176,36	188,09	200,31	176,36	188,09	200,31	176,36	188,09	200,31
6	3	122,92	137,35	152,38	122,92	137,35	152,38	122,92	137,35	152,38
6	5	90,38	101,62	115,75	90,38	101,62	115,75	90,38	101,73	115,75
8	2	222,79	236,40	250,03	222,79	236,40	250,03	222,79	236,40	250,03
8	3	153,68	170,26	186,87	153,71	170,26	186,87	153,71	170,36	187,05
8	5	102,03	117,64	134,55	102,03	117,64	134,55	102,03	117,64	134,55
10	2	264,02	278,80	293,57	264,02	278,80	293,57	264,02	278,80	293,57
10	3	NA	NA	NA	181,91	199,82	217,87	181,91	199,82	217,87
10	5	NA	NA	NA	118,53	135,88	154,48	118,53	136,00	154,61
12	2	NA	NA	NA	301,95	317,62	333,29	301,95	317,62	333,29
12	3	NA	NA	NA	207,17	226,35	245,57	207,17	226,37	245,59
12	5	NA	NA	NA	131,62	151,26	171,32	131,67	151,34	171,38

Table 5.2. 4 Average Objective Function Values of TSA and ESA

We can see that, for most of the problems, TSA and ESA find the same objective function value. For the unequal objective function values, TSA slightly outperforms ESA1. The deviation $(100 * \frac{\text{ESA} - \text{TSA 1}}{\text{TSA 1}})$ is at most 0.11%. (The underlined cells indicate that TSA performs better than ESA. The other cells have the same values for TSA and ESA)The CPU times of ESA is at most 3 seconds whereas it reaches 1389 seconds in the large problems in TSA. Here we can conclude that, ESA performs the best in terms of CPU times and it can generate superior schedules for most of the problems.

Chapter 6

CONCLUDING REMARKS AND FUTURE DIRECTIONS

Parallel machine scheduling problem is widely studied by the researchers. However, most of the studies deal with deterministic problems ignoring the disruptions such as processing time variability. In fact these disruptions and uncertainties deteriorate performance of the schedule and must be taken into consideration before the planning phase.

In this thesis, we study parallel machine environment with processing time uncertainty. We assume that the processing time of each job follows a discrete triangular distribution. This problem can be modeled as the deterministic equivalence by the scenario approach. The aim is to find the schedule that minimizes the expected makespan over scenarios. Although it is straightforward to model the problem, the number of scenarios increases exponentially and makes the problem difficult to solve. This is why we have tried different alternative methods for the solution.

We construct different solution algorithms for the beginning. Firstly we have scheduled the jobs according to their expected processing times. The job with the longest processing time is allocated to the first machine; the job with the second longest processing time is allocated to the second machine, and so forth. (This method is called Longest Expected Processing Time) Secondly, we randomly select 10% of the schedules and solved the problems to achieve a feasible solution. Lastly, we assume that in each scenario only one job has an uncertain value and the rest of the jobs have the most probable processing times. We compare the results and conclude that the last algorithm yields better solutions with lower computational time than the others.

In order to find exact solutions of the large-sized problems, we have applied "Dual Decomposition Method" for our study. In our study, for the large sized problem sets we manage to have better gaps for the solution than the direct model. However, even though the results are satisfying, the dual decomposition method is not efficient in terms of time. We also introduce a new heuristic method based on the dual decomposition. We randomly select 5% of the scenarios and applied dual decomposition method with these scenarios. The results show that at most 6% gap to the exact solution is reached via this algorithm.

Lastly, we propose a Tabu Search Algorithm for the study. The results show that, this algorithm generally finds the global optimal solution. Even in the cases of when it cannot find it, the deviation is negligible.

In conclusion, to the best of our knowledge this study is the first study that adopts dual decomposition method for the robust parallel machine scheduling problem with minimizing makespan. This study can be extended by the following research directions:

1. As the number of scenarios increase, it is harder to solve the problems. Can we find a subset of scenario set which can still represents the whole scenario set? Does this subset perform well when the processing time distribution is nonsymmetrical?
2. How does the dual decomposition method perform on other scheduling environments?
3. What will be the effect of the parallel coding on dual decomposition method for parallel machine scheduling?
4. When other disruptions such as machine break down taken into consideration, does the method performs well?

BIBLIOGRAPHY

[1] M. L. Pinedo, *Scheduling Theory, Algorithms and, Systems*, 4th ed. Springer, 2012.

[2] B. Tadayon and J. C. Smith, "Robust offline single-machine scheduling problems," in *Wiley Encyclopedia of Operations Research and Management Science*, J. J. Cochran, Ed. John Wiley & Sons, 2015.

[3] S. Gören, "Generating robust and stable machine schedules from a proactive standpoint," Bilkent University, 2009.

[4] I. Sabuncuoglu and S. Goren, "Hedging production schedules against uncertainty in manufacturing environment with a review of robustness and stability research," *Int. J. Comput. Integr. Manuf.*, vol. 22, pp. 138–157, 2009.

[5] E. Mokotoff, "Parallel machine scheduling problems: A survey," *ÁsiaPacific J. Oper. Res.*, vol. 18, pp. 193–242, 2001.

[6] C. C. Caroe and R. Schultz, "Dual decomposition in stochastic integer programming," vol. 24, pp. 37–45, 1999.

[7] M. Pinedo and L. Schrage, "Stochastic shop scheduling: A survey," in *Deterministic and Stochastic Scheduling*, Springer, 1982, pp. 181–196.

[8] T. Chaari, S. Chaabane, N. Aissani, and D. Trentesaux, "Scheduling under uncertainty : survey and research directions," in *International Conference on Advanced Logistics and Transport*, 2014, pp. 229–234.

[9] Z. Li and M. Ierapetritou, "Process scheduling under uncertainty : Review and challenges," vol. 32, pp. 715–727, 2008.

[10] M. Uetz, "Algorithms for deterministic and stochastic scheduling," Technischen Universitat Berlin, 2001.

[11] K. R. Baker, "Minimizing earliness and tardiness costs in stochastic scheduling," *Eur. J. Oper. Res.*, vol. 236, no. 2, pp. 445–452, 2014.

[12] C. Lu, S. Lin, and K. Ying, "Robust scheduling on a single machine to minimize total flow time," *Comput. Oper. Res.*, vol. 39, pp. 1682–1691, 2012.

[13] A. Kasperski, A. Kurpisz, and P. Zielin, "Approximating a two-machine flow shop scheduling under discrete scenario uncertainty," *Eur. J. Oper. Res.*, vol. 217, pp. 36–43, 2012.

[14] M. Ranjbar, M. Davari, and R. Leus, “Two branch-and-bound algorithms for the robust parallel machine scheduling problem,” *Comput. Oper. Res.*, vol. 39, no. 7, pp. 1652–1660, 2012.

[15] S. Alimoradi, M. Hematian, and G. Moslehi, “Robust scheduling of parallel machines considering total flow time,” *Comput. Ind. Eng.*, vol. 93, pp. 152–161, 2016.

[16] X. Xu, W. Cui, J. Lin, and Y. Qian, “Robust makespan minimisation in identical parallel machine scheduling problem with interval data,” *Int. J. Prod. Res.*, vol. 51, pp. 3532–3548, 2013.

[17] H. Hu, K. K. H. Ng, and Y. Qin, “Robust Parallel Machine Scheduling Problem with Uncertainties and Sequence-Dependent Setup Time,” *Sci. Program.*, pp. 1–13, 2016.

[18] X. Xu, J. Lin, and W. Cui, “Hedge against total flow time uncertainty of the uniform parallel machine scheduling problem with interval data,” *Int. J. Prod. Res.*, vol. 52, pp. 5611–5625, 2014.

[19] A. Kasperski, A. Kurpisz, and P. Zielinski, “Parallel machine scheduling under uncertainty,” in *IPMU*, 2014, no. August.

[20] J. R. Birge and F. Louveaux, *Introduction to Stochastic Programming*. Springer, 2010.

[21] M. L. Fisher, “A dual algorithm for the one machine scheduling problem,” *Math. Program.*, vol. 11, no. 1, pp. 229–251, 1976.

[22] J. F. Benders, “Partitioning procedures for solving mixed-variables programming problems,” *Numer. Math.*, pp. 238–252, 1962.

[23] R. M. Van Slyke and R. Wets, “L-Shaped linear programs with applications to optimal control and stochastic programming,” *J. Appl. Math.*, no. 638–663, 1969.

[24] G. Laporte and V. F. Louveaux, “The integer L-shaped method for stochastic integer programs with complete recourse,” *Oper. Res. Lett.*, pp. 133–142, 1993.

[25] L. A. Wolsey, *Integer Programming*. John Wiley & Sons, 1998.

[26] F. Glover, “Future paths for integer programming and links to artificial intelligence,” *Comput. Oper. Res.*, pp. 533–549, 1986.