Scheduling jobs on two uniform parallel machines to minimize the makespan

Sandhya Kodimala
University of Nevada, Las Vegas, kodimalasandhya@gmail.com

Follow this and additional works at: http://digitalscholarship.unlv.edu/thesesdissertations

Part of the Theory and Algorithms Commons

Repository Citation
Kodimala, Sandhya, "Scheduling jobs on two uniform parallel machines to minimize the makespan" (2013). UNLV Theses, Dissertations, Professional Papers, and Capstones. 1850.
http://digitalscholarship.unlv.edu/thesesdissertations/1850

This Thesis is brought to you for free and open access by Digital Scholarship@UNLV. It has been accepted for inclusion in UNLV Theses, Dissertations, Professional Papers, and Capstones by an authorized administrator of Digital Scholarship@UNLV. For more information, please contact digitalscholarship@unlv.edu.
SCHEDULING JOBS ON TWO UNIFORM PARALLEL MACHINES TO MINIMIZE THE MAKESPAN

by

Sandhya Kodimala

Bachelor of Technology (B.Tech.)
Jawaharlal Nehru Technological University, India
2011

A thesis submitted in partial fulfillment of
the requirements for the

Master of Science in Computer Science

School of Computer Science
Howard R. Hughes College of Engineering
The Graduate College

University of Nevada, Las Vegas
May 2013
We recommend the thesis prepared under our supervision by

Sandhya Kodimala

entitled

Scheduling Jobs on Two Uniform Parallel Machines to Minimize the Makespan

be accepted in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science
School of Computer Science

Dr. Wolfgang Bein, Committee Chair

Dr. Ajoy K Datta, Committee Member

Dr. Lawrence L. Larmore, Committee Member

Dr. Venkatesan Mutukumar, Graduate College Representative

Thomas Piechota, Ph.D., Vice President of Research and Graduate Studies and Dean of the Graduate College

May 2013
Abstract

The problem of scheduling $n$ independent jobs on $m$ uniform parallel machines such that the total completion time is minimized is a NP-Hard problem. We propose several heuristic-based online algorithms for machines with different speeds called $Q_2||C_{max}$. To show the efficiency of the proposed online algorithms, we compute the optimal solution for $Q_2||C_{max}$ using a pseudo-polynomial algorithms based on dynamic programming method. The pseudo-polynomial algorithm has time complexity $O(n \cdot T^2)$ and can be run on reasonable time for small number of jobs and small processing times. This optimal offline algorithm is used to benchmark the efficiency of the proposed online algorithms.
Acknowledgements

First and foremost, I would like to thank my committee chair, Dr. Wolfgang Bein for the support he has extended to me throughout my work towards this thesis, and for his informed guidance and advice. I would like to express my gratitude to Dr. Doina Bein for her supervision, advice and guidance throughout my thesis. I extend my sincere gratitude to Dr. Ajoy K Datta, for all the motivation, support and encouragement he gave, to pursue my masters at UNLV. I would like to thank Dr. Lawrence L.Larmore, Dr. Venkatesan Muthukumar and Dr. Ajoy K Datta for serving my committee and reviewing my thesis and I am grateful to Dr. Lawrence L.Larmore, Edward Jorgenson and Lee Misch for all the support they gave me to work as a teaching assistant. Finally, I would like to thank my parents, sisters and friends for all the support and encouragement they gave me. This thesis is dedicated to my parents, Venkanna and Padma Kodimala, who have always supported me in my endeavors, always given me the strength and encouragement to follow my dreams.

Sandhya Kodimala

University of Nevada, Las Vegas
May 2013
# Contents

Abstract iii

Acknowledgements iv

Contents v

List of Tables vii

List of Figures ix

Chapter 1  Introduction 1
  1.1  Introduction to Scheduling ........................................... 1
  1.2  Outline ................................................................. 1

Chapter 2  Background 2
  2.1  Scheduling ............................................................ 2
     2.1.1  Gantt Chart ..................................................... 2
  2.2  Problem Classification ............................................... 3
     2.2.1  Machine Environment .......................................... 3
     2.2.2  Job Characteristics ............................................. 4
  2.3  Optimality Criteria .................................................. 6
  2.4  Makespan ............................................................... 7

Chapter 3  Literature Review 9
  3.1  Types of Algorithms .................................................. 9
  3.2  Time Complexity ....................................................... 11

Chapter 4  Implementation 12
  4.1  Problem Description .................................................. 12
## List of Tables

4.1 Job processing times ................................................................. 14
4.2 Algorithm3 job set-1 ................................................................. 18
4.3 Algorithm3 job set-2 ................................................................. 18
4.4 Algorithm3 Job set-3 ................................................................. 19
4.5 Algorithm4 Job set-1 ................................................................. 21
4.6 Algorithm4 Job set-2 ................................................................. 21
4.7 Algorithm4 Job set-3 ................................................................. 22
4.8 Job processing of next input ....................................................... 23
4.9 Job processing of next input ....................................................... 23
5.1 Dynamic Approach for job set ..................................................... 29
5.2 Job Set-1 .................................................................................. 29
5.3 Makespan of all the algorithms .................................................... 30
6.1 Job Set1 .................................................................................. 31
6.2 Job set2 processing times ............................................................ 33
6.3 Job set2 processing times ............................................................ 34
6.4 Job Set-1 .................................................................................. 35
6.5 Makespan of all the algorithms .................................................... 36
6.6 Jobset-1 data .......................................................................... 37
6.7 Makespan of all the algorithms .................................................... 38
6.8 Average rate of algorithms .......................................................... 38
6.9 Makespan of all the algorithms .................................................... 39
6.10 Makespan of all the algorithms ................................................... 39
6.11 Average rate of algorithms .......................................................... 40
6.12 Jobset-3 Rate ......................................................................... 40
List of Figures

2.1 Machine-oriented Gantt chart ................................................. 2
2.2 Job-oriented Gantt Chart .................................................. 3

4.1 Algorithm1 Gantt chart ...................................................... 15
4.2 Algorithm2 Gantt chart ...................................................... 17
4.3 Algorithm3 Gantt chart for job set-1 ...................................... 18
4.4 Algorithm3 Gantt chart for job set-2 ...................................... 19
4.5 Algorithm3 Gantt chart for job set-3 ...................................... 19
4.6 Algorithm4 Gantt chart for job set-1 ...................................... 21
4.7 Algorithm4 Gantt chart for job set-2 ...................................... 21
4.8 Algorithm4 Gantt chart for job set-3 ...................................... 22
4.9 Algorithm5 Gantt chart for job set-1 ...................................... 23
4.10 Algorithm5 Gantt chart for job set-2 ..................................... 24

6.1 Algorithm3 Gantt chart for Job Set1 ...................................... 32
6.2 Algorithm4 Gantt chart for Job Set1 ...................................... 32
6.3 Algorithm5 Gantt chart for Job Set1 ...................................... 32
6.4 machine-oriented Gantt chart for job set-2 ............................... 32
6.5 machine-oriented Gantt chart for job set-2 ............................... 33
6.6 machine-oriented Gantt chart for job set-2 ............................... 33
6.7 machine-oriented Gantt chart for job set-3 ............................... 33
6.8 machine-oriented Gantt chart for job set-3 ............................... 34
6.9 machine-oriented Gantt chart for job set-3 ............................... 34
6.10 Algorithm1-comparison job set-1 ........................................ 35
6.11 Algorithm2-comparison job set-1 ........................................ 35
6.12 Algorithm3-comparison job set-1 ........................................ 35
6.13 Algorithm4-comparison job set-1 ............................................. 35
6.14 Algorithm5-comparison job set-1 ............................................. 36
Chapter 1

Introduction

In this chapter, we will discuss the fundamental concepts of the scheduling and detailed outline about the rest of the thesis. In Section 1.1, a basic introduction to the scheduling is given, and Section 1.2, gives an outline of the thesis and discusses the scope and purpose of this thesis.

1.1 Introduction to Scheduling

Scheduling is really important, in the field where the resources have to be assigned to perform an activity in a given period of time. One such industry is manufacturing and service industry. Schedules are concerned with the allocation of resources optimally to a certain task or activities over a period of time.

1.2 Outline

In Chapter 2, we will discuss the basic concepts of scheduling, different types of scheduling environments and parameters used to measure the optimality criteria of the schedule. In Chapter 3, we will give an overview of the type of algorithms present, discuss in detail about their usage and its importance. We also explain about the NP-Hard, NP-Complete and Pseudo Polynomial time complexities. In Chapter 4, we will illustrate the implementation of different offline and online algorithms, and we will further explain about their working using sample job sets. In Chapter 5, we introduce a pseudo polynomial algorithm, which uses a dynamic programming procedure to obtain an optimal makespan, and we also discussed its implementation in this chapter. In Chapter-6, we have tested the efficiency of the online algorithms using the optimal makespan obtained by dynamic programming procedure. Finally, in Chapter 7, we have concluded the thesis and future research work are discussed.
Chapter 2

Background

In this chapter, we will review some of the fundamental concepts of scheduling. Section 2.1 discusses representation of schedule using Gantt charts. Section 2.2 illustrates about the different scheduling characteristics, such as machine environment and job characteristics. We will further discuss the optimality criteria used to measure the efficiency of the schedule. Finally, in Section 2.3, we will discuss makespan and an algorithm to calculate makespan of a given schedule.

2.1 Scheduling

Let us suppose that we have \( n \) jobs and \( m \) machines. \( J_i \) represents the jobs where \( i = 1, 2, \ldots, n \), and \( M_i \) representing machines where \( i = 1, 2, \ldots, m \). Jobs \( (J_i) \) have to be processed on machines \( (M_i) \) such that each machine can process at most one job at a given time, and each job can be processed on just one machine. Allocation of jobs to machines is called Scheduling [1].

2.1.1 Gantt Chart

The schedules of jobs on machines are represented using Gantt charts. Gantt chart is either machine oriented or job oriented charts. For example, if two machines and five jobs are given, the schedule could look like the one in Figure 2.1. In another Figure 2.2, we have four machines and two jobs whose operations can be split among several machines.

<table>
<thead>
<tr>
<th></th>
<th>J1</th>
<th>J3</th>
<th>J4</th>
</tr>
</thead>
<tbody>
<tr>
<td>J2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 2.1: Machine-oriented Gantt chart
### 2.2 Problem Classification

The scheduling problem is defined in terms of three parameters as \( \alpha \mid \beta \mid \gamma \) depending on the job, machine or scheduling characteristics [2]. Here, \( \alpha \) describes the machine environment, \( \beta \) describes the job environment and \( \gamma \) describes the optimality criteria.

#### 2.2.1 Machine Environment

According to Graham [3], machine environment is qualified by a string \( \alpha = \alpha_1\alpha_2 \), where \( \alpha_1 \) takes the values \( \alpha_1 \in \{o,P,Q,R,PM,PM,QM,PM\} \) and \( \alpha_2 \) denotes the number of machines. \( P_{ij} \) represents the \( i_{th} \) processing time of the job \( J_j \).

**Dedicated Machines**

The symbol ‘o’ denotes an empty symbol. If \( \alpha_1 = o \), then \( \alpha = \alpha_2 \). When \( P_{ij} = P_i \) and \( \alpha_1 = o \), jobs must be processed on a dedicated machine. Let \( O_{i1},...O_{in} \) be operations associated with \( J_i \) and \( O_{ij} \) which can be processed on any of set of machines \( \mu_{ij} \in M_1,M_2...M_m \). In the dedicated machines \( \mu_{ij} \) is equal to a set of machines since all machines are identical.

**Parallel Machines**

When \( \alpha_1 \in P,Q,R \) [4], then jobs can be processed on any of the \( n \) machine \( M_1,...M_n \). When \( \alpha_1 = P \), then the machines are called identical parallel machines. So job \( J_i \) with the processing time \( P_{ij} \) on the machine \( M_j \) will have \( P_{ij} = P_i \). When \( \alpha_1 = Q \), then the machines are called as uniform parallel machines. Then the processing time \( P_{ij} \) of job \( J_i \) on machine \( M_j \) will be \( P_{ij} = P_i/S_j \), where \( S_j \) is the speed of the machine \( M_j \).

When \( \alpha_1 = R \), then the machines are called unrelated parallel machines. The processing time \( P_{ij} \) of job \( J_i \) on the machine \( M_j \) will be \( P_{ij} = P_i/S_{ij} \), where \( S_{ij} \) is the speed of machine, \( M_j \) which is dependent on job \( J_i \). Parallel machine are discussed detailed in Section 2.2.
Multi-purpose Machines

Let $O_{ij}$ be the operation associated with $J_i$. If this operation can be processed on any machine then this type of scheduling machines, are called multi-purpose machines [5]. If $\alpha_1 = PMPM$, then they are multi-purpose machine with identical speeds. If $\alpha_1 = QMPM$, then they are multi-purpose machine, with an uniform speeds.

Machine Count $\alpha_2$

In a machine environment, the number of machines is denoted by $\alpha_2$. If $\alpha_2$ is a positive integer value, then it denotes the number of machines available. For example, when $\alpha_2 = 3$ then we have three machines available for the jobs to be processed on these machines. If the number of machines is arbitrary and varying, then we denote $\alpha_2 = o$.

2.2.2 Job Characteristics

According to Brucker [2], job characteristics are defined by a set $\beta$, which contains six elements $\beta_1, \beta_2, \beta_3, \beta_4, \beta_5$ and $\beta_6$.

$\beta_1$ Preemption

Preemption means the processing of a job can be interrupted and job processing can be later resumed even on other machines. There is no limit to the number of times jobs can be interrupted and resumed. If job preemption is allowed, we represent $\beta_1 = pmtn$. If there is no preemption, we do not specify $\beta_1$ in job characteristics.

$\beta_2$ Precedence Relation

If job $J_i$ is processed and completed before starting processing $J_k$ then such a precedence is represented as $J_i \rightarrow J_k$. These precedence relations is represented by an acyclic directed graph $G = (V, A)$, where $V = 1, \ldots, n$ corresponds with the jobs, $(i, k) \in A$. An acyclic digraph is a directed graph containing no directed cycles; it is also known as a directed acyclic graph or a "DAG." Every finite acyclic digraph has at least one node of outdegree 0.

Scheduling problems with the restricted precedences are given by chains, an intree, an outtree, a tree or a series-parallel directed graph. According to Baker [1], for a node, the number of head endpoints adjacent to a node is called the indegree of the node and the number of tail endpoints adjacent to a node is called the outdegree of the node. An outdegree and indegree is called "branching factor" in trees. We denote $\beta_2 = intree$ so an acyclic directed graph $G$ is a rooted tree. An acyclic
directed graph $G$ has an outdegree which is most one for every vertex, So in an intree all the arrows are directed towards the root. Similarly $\beta_2 = \text{ outtree}$, then an acyclic directed graph $G$ will have an indegree which is most one for every vertex, so we have all arrows away from the root. If an acyclic directed graph $G$ is either an intree or an outtree then $\beta_2$ is represented as $\beta_2 = \text{ tree}$.

$\beta_2 \in \text{ sp - graph, chain}$

A chain in a graph model is defined as a chain graph which may have both directed and undirected edges, but without any directed cycles. A set of chains is called tree when it as outdegree and indegree of each vertex, at most one degree. We reIf $\beta_2 = \text{ chain}$, then $G$ is a set of chains.

Series-parallel graphs [6] are related to trees. According to Brucker [1], a graph is called as sp-graph is built by means of the following rules. Let graph $G_1 = (V_1, A_1)$ Any graph consisting of a single vertex, is called sp-graph, is know as a Base graph. The graph which is formed from $G_1$ and $G_2$ by joining the vertex sets and arc sets in sp-graph. It is represented as $G = (V_1 \cup V_2 \cup A_1 \cup A_2)$ and is called Parallel composition. The graph $G = (V_1 \cup V_2 \cup A_1 \cup A_2 \cup T_1 \times S_2)$, $G_1$ and $G_2$ are similar to parallel composition graph. Additionally, arcs $(t,s)$ are added to graph $G$, where $t \in T_1$, $T_1$ belongs to $G_1$ and $s \in S_2$, $S_2$ is set of source in graph $G_2$, Such a graph is called Serial composition, so $\beta_2 = \text{ sp-graph}$ we represent it this way only when $G$ is a serial parallel graph. If there is no such precedence relations between jobs, then $\beta_2$ is not represented with any notation.

$\beta_3 \in r_i$

Release time, is the time when a job had to be release from processing on a machine irrespective of whether job had completed its processing on machine or not. If $\beta_3 = r_i$ then the job $J_i$ has release time $r_i$. If they are no release dates then we ignore denoting $\beta_3$ in job characteristics representation $\beta$.

$\beta_4 \in P_{ij}$

Let job $J_i$, with the operation $O_{ij}$, has a processing requirement $P_{ij}$. $\beta_4$ represents the job operations or the processing times of the jobs to be processed on the machines. If each job has a unit processing time associated with it then we represent the processing time as $P_{ij} = P_i$. As $\beta_4$ represents the processing time we represents it as $\beta_4 = P_i$. 

5
\(\beta_5 \in d_i\)

Some of the jobs will have deadline \(d_i\) and these jobs have to be processed on machines no later than their deadline \(d_i\). If we have any such deadline for the jobs then we represent it in the job characteristics as \(\beta_5 = d_i\). If there is no such deadline then we don’t include \(\beta_5\) in the job characteristics representation.

\(\beta_6 \in s -\text{batch, p - batch}\)

A batch means a set of jobs which have to be processed together on a machine. In batching problem, group of jobs need to be scheduled together on a single machine. They are two type of batching problems, p-batching and s-batching. In p-batching the length of the batch is the maximum among the processing times of all jobs in the batch. In s-batching the length of the batch is the sum of the processing times of the jobs in the batch. \(\beta_6\) of the job characteristics is represented as \(\beta_6 = p\)-batch or \(\beta_6 = s\)-batch according to p-batching or s-batching problem respectively. If there is no such batching problem then \(\beta_6\) is not indicated in the job characteristics.

### 2.3 Optimality Criteria

The time taken by the job to complete its processing on a machine is called finishing time. Every job \(J_i\) have a finishing time \(C_i\). The cost of finishing the job \(J_i\) is denoted as \(f_i(C_i)\). The optimality criteria are represented as \(\gamma\), which is a performance measurement. According to Graham’s [3] notation they are two types of the total cost functions.

\[
f_{\max}(C) = \max f_i(C_i) | i = 1, .., n
\] (2.1)

and

\[
\Sigma f_i(C) = \Sigma_{i=1}^{n} f_i(C_i)
\] (2.2)

Equation 2.1 and 2.2 represents the cost functions. The optimality criteria \(\gamma\) is represented as \(\gamma = f_{\max}\) or \(\gamma = \Sigma f_i\). The main goal of the objective function is to minimize the makespan and total flow time in a schedule.

**Makespan**

\[
\text{Makespan} = \max C_i | i = 1, .., n
\] (2.3)

**Total Flow time**

\[
\text{Sum of Flowtime} = \Sigma_{i=1}^{n} C_i | i = 1, .., n
\] (2.4)
Some of the objective functions in the optimality criteria are weighted flow time, lateness, earliness, tardiness, absolute deviation, square deviation and unit penalty

**Tardiness**

Tardiness is defined as the time taken by the job after its due date to complete its processing on the machine. Tardiness for a job $J_i$, is represented as $T_i = \max \{ 0, C_i - d_i \}$

**Lateness**

Lateness [7] is a category used to find whether the job is completed ahead of the schedule or on, or before schedule. Lateness is represented as $L_i = C_i - d_i$

**Earliness**

Earliness is defined as the time left after processing the job. Earliness occurs when job completes its processing before it’s deadline. Earliness is represented as $E_i = \max \{ 0, d_i - C_i \}$

**Unit Penalty**

Unit penalty is the penalty imposed on the job $J_i$, if it had not completed its processing before the deadline $d_i$. Unit penalty is represented as $U_i = 0$ if the job is processed before the deadline, $U_i = 1$ if the job is processed after the deadline.

**Absolute Deviation**

Absolute deviation is represented as $A_i = |C_i - d_i|$

**Squared Deviation**

Squared deviation is represented as $S_i = |C_i - d_i|^2$

**2.4 Makespan**

They are set of jobs $J_i$, where $i = 1,2,3, n$. These jobs have either identical processing time or different processing time and the task is that, we need to assign these jobs $J_i$ on a set of machines $M_i$, indexed by $i = 1,2,3, n$, so that completion time is minimized.
The completion time is also known as makespan of the schedule. Minimizing the makespan is nothing but minimizing the total processing time of the schedule. Makespan is a measure of efficiency. Makespan algorithm is used to minimize the total completion time, by scheduling the jobs, in such a way that they are processed on the best available machine so that the total completion time is minimized.

**Makespan Algorithm**

Step-1: Jobs are listed in an arbitrary order.
Step-2: Machines available, to process the jobs are listed in an arbitrary order.
Step-3: The first job is allocated to the first available machine.
Step-4: The second job is allocated to the next available machine.
Step-6: The next job is allocated to the machine that has the least amount of work load.
Step-7: Terminate the algorithm when all the jobs are scheduled on the machines.
Chapter 3

Literature Review

In Section 3.1, we will discuss different types of algorithms available and various approaches to solve these types of problems. In Section 3.2, we will discuss competitive analysis, which is a performance analyzer. In later Sections, we will study about the basic definitions, notations and concepts of the parallel machine environment, which we will discuss further in this thesis. We will further review the time complexities of the algorithms, which minimize the completion time or make span of the schedule.

3.1 Types of Algorithms

Algorithms are divided into three types, based on input type available to an algorithm in solving a problem. Sanlaville [8] has proposed three different types of algorithm.
1. Offline Algorithm
2. Online Algorithm
3. Nearly Online Algorithm

An Offline Algorithm

An offline algorithm has the entire input available at the beginning of the algorithm. In real life, it is hard to find examples of offline algorithms since the entire input in not available at the beginning of the processing the jobs. Optimal algorithms are in general offline algorithms.

In today’s world, it is difficult to know the entire input and it is hard to find algorithms which gives an optimal solution, without having any prior knowledge of the future jobs to be processed.
To solve many of current problems, we need algorithms which will process its input, even though it
does not have any prior knowledge of the future input needed to the process. This requirement had led to the existence of the online algorithms.

**Online Algorithm**

Today, in all the real time applications, entire input will not be available from the beginning. For example, scheduling orders in a coffee shop, entire orders won’t be available from the beginning and orders have to served in such a way that, the shop earns maximum profit. The orders are considered as an online input since we are not aware of the future orders which are yet to come.

According to Allan Borodin [9], in an online algorithm, input does not arrive as a batch but it arrives as a sequence of input portions and the algorithms have to schedule the jobs according to the input arrived. In an online algorithm, future input will be available at any given time. No assumptions are made about the input stream. The input to the algorithm is given as a sequence of requests $\sigma = \sigma(1), \sigma(2), \sigma(3)…\sigma(m)$ These requests must be served in the order of their occurrence. The request $\sigma(t^1)$ had to be completed at time $t^1$, where $t^1 > t$.

**Nearly Online Algorithm**

Nearly online algorithms are which are neither offline algorithms nor online algorithms. Nearly online algorithm process the input, knowledge of future input is partial, but the entire future won’t be available. At a given time $t$, we know the available input, but in nearly online algorithm we will know the future input but not the entire input. The future input will be known at a time $t’$.

In real time, we may have knowledge of the entire future input but having the knowledge of next available input in available time $t’$ is similar to online algorithms. To find an optimal solution for these algorithms is equally complex when compared to online algorithms.

**Competitive Analysis**

In competitive analysis [10], we get the competitive ratio to analyses the efficiency of the algorithm. Competitive ratio is the ratio between the cost of function by the algorithm to the cost of function by the optimal algorithm. In general offline algorithm, is the algorithms which gives the optimal solution for scheduling jobs.

The performance of the online algorithm is usually evaluated using competitive analysis. The online algorithm OLG is compared with an offline algorithm OPT. According to Susanne OPT that knows the entire request sequence $\sigma$. Let OLG($\sigma$) and OPT($\sigma$) denote the cost occurred by OLG an OPT algorithms. Algorithm OLG is called $C_0$-competitive if there exists a
constant $b$ such that,

$$\text{ALG}(\sigma) \geq b.\text{OPT}(\sigma)$$  \hspace{1cm} (3.1)

### 3.2 Time Complexity

#### NP-Hard

NP-hard (non-deterministic polynomial-time hard) [11], in computational complexity theory, is a class of problems that are, informally, "at least as hard as the hardest problems in NP". A problem $H$ is NP-hard if and only if there is an NP-complete problem $L$ that is polynomial time Turing-reducible to $H$ (i.e., $L \leq_T H$). In other words, $L$ can be solved in polynomial time by an oracle machine with an oracle for $H$. Informally, we can think of an algorithm that can call such an oracle machine as a subroutine for solving $H$, and solves $L$ in polynomial time. If the subroutine call takes only one step to compute. NP-hard problems may be of any type: decision problems, search problems, or optimization problems.

#### NP-Complete

In computational complexity theory, the complexity class NP-complete (abbreviated NP-C or NPC) [12] is a class of decision problems. A decision problem $L$ is NP-complete if it is in the set of NP problems and also in the set of NP-hard problems. The abbreviation NP refers to "nondeterministic polynomial time."

#### Pesudo Polynomial

In computational complexity theory, a numeric algorithm runs in pseudo-polynomial time if its running time is polynomial in the numeric value of the input (which is exponential in the length of the input, its number of digits). An NP-complete problem with known pseudo-polynomial time algorithms is called weakly NP-complete. An NP-complete problem is called strongly NP-complete if it is proven that it cannot be solved by a pseudo-polynomial time algorithm unless P=NP. The strong/weak kinds of NP-hardness are defined analogously.
Chapter 4

Implementation

This chapter consists of three parts. The first describes the problem and some of the fundamental constraints and assumptions made to describe the problem. The second section deals with the implementation of the two algorithms which accepts an offline input and try to provide an optimal solution to the above stated problem. Finally, the third section deals with implementation of the three algorithms which accepts an online input and has various constraints on the knowledge of the next job to be scheduled.

4.1 Problem Description

We consider the problem of scheduling n independent jobs, ( Ji ) indexed by i = {1, 2...n} on two machines, machine-1 and machine-2. The machine-1 and machine-2 are related and uniform, and the machine-1 is twice as speed as machine-2. The objective is to minimize the maximum completion time and produce a minimum makespan.

Job characteristics

The job characteristics are non-preemption, no precedence relations and finally there is no release date and deadline on the jobs to be scheduled. The job ( Ji ) consists of the processing time \( P_i \).

Machine Environment

The machine environment is a uniform parallel machine.

\[ P_{ij} = P_i/S_i \]

\( S_j \) : speed of machine \( M_j \)

So if job \( J_i \) is processed on machine-1 then the processing time is \( P_i \) and the processing time on
the machine-2, is $2^* P_i$

**Notations**

$J_i$ : represents the $i^{th}$ job of total $n$ jobs

$P_{ik}$ : represents the processing time on the machine $M_k$, where $k = 1, 2$

$M_1$ : represents the machine-1

$M_2$ : represents the machine-2

### 4.2 An Offline Algorithm

For an offline algorithm [13], the entire input is available from the beginning, and the algorithm has to produce an output using these input values. In this Section, we will propose an offline algorithm, for the problem stated in Section 4.1. In Section 4.2.1, we will deal with an offline algorithm, in which next available job is scheduled on the machine, which becomes idle at the time of scheduling the jobs.

Finally, in the Section 4.1.2, deals with an offline algorithm, in which the highest processing jobs are scheduled on the faster machine and the less processing time jobs are scheduled on the slower machine.

#### 4.2.1 Algorithm 1

In this algorithm, jobs are scheduled on the first available machine. When both machines are available preference, is given to the machine, which takes the least time to process the job. If we have 10 jobs with the processing times 8, 9, 3, 2, 4, 5, 7, 1, 6, 3 then the job processing times on the machine-1 are $P_{11} = 8$, $P_{21} = 9$, $P_{31} = 3$, $P_{41} = 2$, $P_{51} = 4$, $P_{61} = 5$, $P_{71} = 7$, $P_{81} = 1$, $P_{91} = 6$, $P_{101} = 3$.

Then the job processing times on the machine-2 are $P_{12} = 16$, $P_{22} = 18$, $P_{32} = 6$, $P_{42} = 4$, $P_{52} = 8$, $P_{61} = 10$, $P_{71} = 14$, $P_{81} = 2$, $P_{91} = 12$, $P_{101} = 6$.

**Calculating Makespan**

**Initialization:**

makespan1 = 0, makespan2 = 0

counter1 : represents the total processing times of the jobs scheduled on the machine-1

counter2: represents the total processing times of the jobs scheduled on the machine-2

**Procedure:**

1. Allocate the first job in the list to the faster machine and increment the counter1 by the processing
time of the job.

2. Check which machine is idle and allocate the next job to the available idle machine and then increment the respective counter variable, by the processing time of the job.

3. If both machines are available, then schedule the next available job on the faster machine

4. Stop the procedure, when all jobs are scheduled.

Under this method, the jobs are scheduled according to the order they have arrived and no priority is given to any jobs. So if, we have five jobs (n = 10), with the processing times as 8, 9, 3, 2, 4, 5, 7, 1, 6, 3 with respect to the machine-1. At beginning, both machines (M1 and M2) are available, and the jobs, j1 and j2 are scheduled on the machines, M1 and M2 respectively.

The load on the machines M1 and M2 are 8 and 18, since P11 = 8 and P22 = 18 and the variables, counter-1 and counter-2 are incremented by 8 and 18 respectively. To schedule the third job J3, we have M1 available, since the counter-1 value is less compared to the counter-2 value. Now the load on the machine M1, after processing job J3, having processing time as P13 = 3 is,

\[ P_{11}(8) + P_{13}(3) = 11 \]

The counter-1 value, after incrementing by processing time of J3 is 13. We schedule the job J4, having the processing time P14 = 2, on M1 since the counter-1 value is less compared to the counter-2 value. Now the load on the machine M1, is

\[ P_{11}(8) + P_{13}(3) + P_{14}(2) = 13 \]

We schedule, job J5 and J6 on the machine-1 M1, since the counter-1 value is less compared to the counter-2 value. Now the load on the machine M1 is

\[ P_{11}(8) + P_{13}(3) + P_{14}(2) + P_{15}(4) + P_{16}(5) = 22 \]

Since the counter-1 value is greater than the counter-2 value, we will schedule the job J7, on the machine-2 M2. Now the load on M2 is

\[ P_{22}(18) + P_{27}(14) = 32 \]
The counter-2 value, after incrementing by the processing time of the job $J_7$ is 32. We schedule the job $J_8$ on the machine-1 since the counter-1 value is less compared to the counter-2 value. Now the load on the $M_1$ is

$$P_{11}(8) + P_{13}(3) + P_{14}(2) + P_{15}(4) + P_{16}(5) + P_{18} = 23$$

The counter-1 value, after incrementing by the processing time of the job $J_7$ is 23. Finally, the jobs $J_9$ and $J_{10}$ are scheduled on $M_1$. Now the load on the machine $M_1$ is

$$P_{11}(8) + P_{13}(3) + P_{14}(2) + P_{15}(4) + P_{16}(5) + P_{18} + P_{19}(6) + P_{101}(3) = 32$$

The counter-1 value, after incrementing, by the processing time of the jobs $J_9$ and $J_{10}$ is 32. Makespan of the schedule will be given by

$$\max \{ \text{counter-1, counter-2} \}$$

Since both counter-1 and counter-2 have equal value, makespan of the schedule is 32.

4.2.2 Algorithm2

In this algorithm, we sort the jobs in the increasing order of their processing time on the machine-1 and schedule the longer jobs on the machine-1 and shorter jobs on the machine-2 irrespective of whether the other machine is available or not. If the number of jobs $n$, to be scheduled are even then $n/2$ jobs, having a longer processing times are scheduled on the faster machine, and the other $n/2$ jobs are scheduled on the slower machine. If the number of jobs $(n+1)$, to be scheduled are odd then $(n+1)/2$ jobs, having a longer processing times are scheduled on the faster machine, and the other $n/2$ jobs are scheduled on the slower machine. We sort the jobs($J_i$) where $i = 1, 2, ..., n$, in the decreasing order of their processing times on the machine-1.

Calculating Makespan

Initialization:

makespan1 = 0, makespan2 = 0

counter1: represents the total processing time of the jobs scheduled on the machine-1

counter 2: represents the total processing time of the jobs scheduled on the machine-2
Sort the jobs in the decreasing order of their processing time on the machine-1.

Procedure:

if ( Jobs == n ) then

The jobs( $J_i$), where $i=1,2...,n/2$ are scheduled on $M_1$.

counter-1 = counter-1 + $\sum_{i=1}^{n/2} P_i$

The next available jobs( $J_i$), where $i = n/2,...,n$ are scheduled on $M_2$

counter-2 = counter-2 + $\sum_{i=n/2}^{n} P_i$

end if

if ( Jobs == n+1 ) then

The jobs( $J_i$), where $i=1,2...,((n/2)+1)$ are scheduled on $M_1$.

counter-1 = counter-1 + $\sum_{i=1}^{((n/2)+1)} P_i$

The next available jobs( $J_i$), where $i = (n/2)+1,...,n$ are scheduled on $M_2$

counter-2 = counter-2 + $\sum_{i=((n/2)+1)}^{n} P_i$

end if

if ( counter-1 $\geq$ counter-2 ) then

Makespan = counter-1

else

Makespan = counter-2

end if

Let us take the job set represented in the Table 4.1. The processing times of the 10 jobs with respect to the machine-1 are $P_{11}=8$, $P_{21}=9$, $P_{31}=3$, $P_{41}=2$, $P_{51}=4$, $P_{61}=5$, $P_{71}=7$, $P_{81}=1$, $P_{91}=6$, $P_{101}=3$. The job processing times with respect to the machine-2 are $P_{12}=16$, $P_{22}=18$, $P_{32}=6$, $P_{42}=4$, $P_{52}=8$, $P_{62}=10$, $P_{72}=14$, $P_{82}=2$, $P_{92}=12$, $P_{102}=6$.

Under this method, the jobs are scheduled according to the order they have arrived and no priority is given to any jobs. At the beginning of the method, both machines ($M_1$ and $M_2$) are available, and the decreasing order of jobs is $J_2, J_1, J_7, J_9, J_6, J_5, J_{10}, J_3, J_4, J_8$. The jobs ($J_2, J_1, J_7, J_9, J_6$) having processing times ($P_{21} = 9, P_{11} = 8, P_{71} = 7, P_{91} = 6, P_{61} = 5$) are scheduled on the machine-1. Increment the counter-1 value, by the processing time of the jobs scheduled on the machine-1. Now the counter-1 value is 35. The jobs ($J_5, J_{10}, J_3, J_4, J_8$) having processing times ($P_{52} = 8, P_{102} = 6, P_{32} = 6, P_{42} = 4, P_{82} = 2$) are scheduled on the machine-2. Increment the counter-2 value, by the processing time of the jobs scheduled on the machine-2. Now the counter-2 value is 26. Since the counter-1 value is greater than the counter-2 value, makespan of the schedule is 35.
4.3 An Online Algorithms

The entire input is not available at the beginning of the algorithm. The best methods are obtained when entire input available but in real life we have mostly on-line problem. We stated three algorithms to schedule \( n \) jobs. These algorithms take \( n \) jobs, as an online input and schedule them on either machine-1 or machine-2. In Section 4.3.1, an algorithm, which takes the jobs to be scheduled as an online input, is described.

In Section 3.3.2, Describe an algorithm, which also deals with an online input, but if we have knowledge of next two available jobs, we schedule these incoming jobs in such a way that we will minimize the completion time and get a minimum make span. Section 3.3.3 Describe an algorithm which works in the same way as the algorithm stated in Section 3.3.2. We also consider that, is it worth enough to wait and schedule the succeeding job on the faster machine.

Representation

In the online algorithms, since we are not aware of the future input, we will pause the processing of jobs as we do not have any more jobs be schedule at that particular time. We will calculate the current make span of the schedule, and we will resume from the pause. if we get any new jobs to be scheduled.

4.3.1 Algorithm3

When a job \( J_i \) need to be scheduled, and both machines are available then the job \( J_i \) is scheduled on the faster available machine among the both machines. Otherwise, \( J_i \) is scheduled on the next available machine. So if, we have five jobs with the processing times 8, 9, 3, 2, 4. The processing times of these jobs on machine-1 are \( P_{11} = 8, P_{21} = 9, P_{31} = 3, P_{41} = 2, P_{51} = 4 \). The job processing times on the machine-2 are \( P_{12} = 16, P_{22} = 18, P_{32} = 6, P_{42} = 4, P_{52} = 8 \).
Calculating Make span

Case 1: When both the machines are available, and the jobs $J_i$ and $J_{i+1}$ need to be scheduled.

\[
\text{if } ( P_{i1} \geq P_{(i+1)1} ) \text{ then }
\]

$J_i$ is scheduled on the faster machine.

$J_{i+1}$ is scheduled on the slower machine

else

$J_{i+1}$ is scheduled on the faster machine

$J_i$ is scheduled on the slower machine

Case 2: When one machine is available and the job $J_i$ need to be scheduled.

Schedule the job $J_i$ on the available machine.

At the beginning, both machines $M_1$ and $M_2$ are available, so the first job $J_1$ with the processing time $P_{11}$, is scheduled on the machine-1 and the second job $J_2$ with the processing time $P_{22}$ is scheduled on the machine-2. The counter-1 and counter-2 values are increment by $P_{11} = 8$ and $P_{22} = 18$ respectively. The jobs $(J_3, J_4, J_5)$ with the processing times ($P_{31} = 3, P_{41} = 2, P_{51} = 4$) are scheduled on the machine-1 since the machine-2 is available only after processing these jobs. The counter-1 value is increment, by the total processing times of these jobs. Now the counter-1 value is 17. The current makespan of the schedule is 18. Since Algorithm-3 takes an online input, it has paused its job processing momentarily till it gets another job to be
scheduled. So, if we have two jobs, with the processing times on machine-1 as $P_{61}=4$ and $P_{71}=3$ and the processing times on the machine-2 as $P_{62}=8$ and $P_{72}=6$. The job $J_6$ is scheduled on the machine-1 since the counter-1 value is 17, and the counter-2 is 18. The counter-1 value incremented to 22 after scheduling $J_6$. The job $J_7$ is scheduled on the machine-2 since the counter-1 value is greater than the counter-2 value, and now the counter-2 value is incremented to 32. The current makespan is 32.

The counter-1 value is lower when compared to the counter-2 value, so the job $J_8$ is scheduled on the machine-1 and counter-1 value is incremented to 23. Similarly, jobs $J_9$ and $J_{10}$ are scheduled on machine-1 since the counter-1 value is lower compared to the counter-2 value. The counter-1 value is incremented by $P_{91} = 6 + P_{101} = 3$, and now the counter-1 value is 32. The current makespan of the schedule is 32.

### 4.3.2 Algorithm 4

When a job $J_i$ need to be scheduled, and both machines are available $J_i$ is compared with $J_{i+1}$ and the job with longer processing time is scheduled on machine-1. It again checks for the scheduling of
shorter processing time job. When one machine is available \( M_k \), \( K \in 1,2 \). Algorithm-4 check is it worth waiting for the other machine to become available. If it is not worth waiting, then we will go ahead and schedule the job on the available machine. At time \( t \), job \( J_i \) need to be scheduled. The machine \( M_k \) is available, and the machine \( M_{\bar{k}} \) is not available, but we know that machine \( M_{\bar{k}} \) will be available at time \((t1 + \Delta t_i)\), where \( t1 \) is the total processing time of all the jobs, which are scheduled on the machine \( M_{\bar{k}} \). If the job \( J_i \) is scheduled on the machine \( M_k \) then it is terminated at the time,

\[
T_1 = (t + \Delta t_i + \text{Processing time on machine} M_k)
\]  

(4.1)

If we schedule job \( J_i \), on the machine \( M_k \) then it have to be terminated at the time.

\[
T_2 = (t2 + \text{Processing time of job on} M_k)
\]  

(4.2)

Where \( t2 \) is the total processing time of all the jobs, which are scheduled on the machine \( M_k \). If \( T_1 \geq T_2 \) then \( J_i \) is scheduled on the machine \( M_k \), otherwise on the machine \( M_{\bar{k}} \).

**Calculating Makespan**

Step 1: If \( M_1 \) is available and \( M_2 \) is not available, but \( M_2 \) becomes available at \( t + \Delta t \)

- If \( J_i \) is scheduled on machine \( M_1 \) then
  - \( J_i \) will be completed at \( t + P_{i1} \)
- If \( J_i \) waits and will be scheduled on the machine \( M_2 \) then
  - \( J_i \) will be completed at \( t + \Delta t + P_{i2} \)

To decide what to do with \( J_i \) we compare \( t + P_{i1} \) ? \( t + \Delta t + P_{i2} \). whatever is shorter we decide for choosing that machine \( t + P_{i1} \) ? \( t + \Delta t + 2P_{i1} \)

Step 2: If \( M_2 \) is available and \( M_1 \) is not available. By extrapolation we compare

\[
t + P_{i2} \? t + \Delta t + P_{i1} \\
 t + 2* P_{i1} \? t + \Delta t + P_{i1}
\]

Case a) \( P_{i1} > \Delta t \Rightarrow t + P_{i2} > t + \Delta t + P_{i1} \) then schedule \( J_i \) on machine \( M_1 \)

Case b) \( P_{i1} < \Delta t \Rightarrow t + P_{i2} < t + \Delta t + P_{i1} \) then schedule \( J_i \) on machine \( M_2 \)

Lets us consider five jobs whose processing times are as follows 8, 9, 3, 2, 4 and the processing times of these jobs on the machine-1 are \( P_{11} = 8, \ P_{21} = 9, \ P_{31} = 3, \ P_{41} = 2, \ P_{51} = 4 \). The job processing times on the machine-2 are \( P_{12} = 16, \ P_{22} = 18, \ P_{32} = 6, \ P_{42} = 4, \ P_{52} = 8 \). Since both machines, are available and the job \( J_1 \) with the processing time \( P_{11} = 8 \), is scheduled on the machine-1 \( M_1 \).

Next job \( J_2 \) with the processing time \( P_{21} = 9 \) needs to be scheduled. If job \( J_2 \) is scheduled on the machine-1, then the load on the machine-1 will be \( 8+9 = 16 \). If the job \( J_2 \) is scheduled on the
Table 4.5: Algorithm4 Job set-1

<table>
<thead>
<tr>
<th>Job</th>
<th>machine-1</th>
<th>machine-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>

Figure 4.6: Algorithm4 Gantt chart for job set-1

machine-2 then the load on the machine-2 will be 18. So, according to algorithm4, it is more optimal
to schedule the job \( J_2 \) on the machine-1 than on the machine-2. Similarly, the jobs \( J_3, J_4 \) and \( J_5 \)are scheduled on the machine-2, since its load upon scheduling these jobs is lower when compared
to the current load on the machine-1. Load on the machine-2 is \( P_{32}(6) + P_{42}(4) + P_{52}(8) = 18 \).

Load on the machine-1 is \( P_{11}(8) + P_{21}(9) = 17 \). If we have two jobs, with the processing times on
machine-1 as \( P_{61}=4 \) and \( P_{71}=3 \) and on the machine-2 as \( P_{62}=8 \) and \( P_{72}=6 \). The current load on

Table 4.6: Algorithm4 Job set-2

<table>
<thead>
<tr>
<th>Job</th>
<th>machine-1</th>
<th>machine-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>14</td>
</tr>
</tbody>
</table>

Figure 4.7: Algorithm4 Gantt chart for job set-2

the machine-1, so the job \( J_6 \) with the processing time \( P_{61} = 5 \) is scheduled on the machine-1. So,
the current load on machine-1 is 18 + \( P_{61}(5) = 23 \). The next available job \( J_7 \) with the processing
time \( P_{71} = 7 \) needs to be scheduled. If we schedule \( J_7 \) on the machine-1, we will have the load as
follows. Load on the machine-1 is \( P_{11}(8) + P_{21}(9) + P_{61}(5) + P_{71}(7) = 29 \). If we schedule the
job \( J_7 \) on the machine-2 then the load on the machine-2 is \( P_{32}(6) + P_{42}(4) + P_{52}(8) + P_{72}(14) = 32 \). So, the job \( J_7 \) is scheduled on the machine-1, and the current load on the machine-1 is 29.

If we have three more jobs, with the processing times on machine-1 as \( P_{81} = 2 \), \( P_{91} = 6 \),
\( P_{101} = 3 \) and on the machine-2 as \( P_{82} = 4 \), \( P_{92} = 12 \), \( P_{102} = 6 \). If job \( J_8 \) need to be scheduled on
Table 4.7: Algorithm4 Job set-3

<table>
<thead>
<tr>
<th>Job</th>
<th>machine-1</th>
<th>machine-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>6</td>
</tr>
</tbody>
</table>

Figure 4.8: Algorithm4 Gantt chart for job set-3

either machine-1 or machine-2. If it is scheduled on the machine-1 then the load is $P_{11}(8) + P_{21}(9) + P_{61}(5) + P_{71}(7) + P_{81}(1) = 30$. If it is scheduled on the machine-2 load is $P_{32}(6) + P_{42}(4) + P_{52}(8) + P_{81}(2) = 20$. So the job $J_8$, is scheduled on the machine-2, and the current load on the machine-1 and machine-2 are 29 and 20 respectively. If job $J_9$, is scheduled on either machine-1 or machine-2 then the load on the machine-1 while be $P_{11}(8) + P_{21}(9) + P_{61}(5) + P_{71}(7) + P_{91}(6) = 35$ and load on the machine-2 while be $P_{32}(6) + P_{42}(4) + P_{52}(8) + P_{81}(2) + P_{92}(12) = 32$. So the job $J_9$ is scheduled on the machine-2, and the current load on the machine-1 and machine-2 are 29 and 32 respectively. If job $J_{10}$ need to be scheduled on either machine-1 or machine-2 then the load on the machine-1, and the machine-2 while be $P_{11}(8) + P_{21}(9) + P_{61}(5) + P_{71}(7) + P_{101}(3) = 32$ and $P_{32}(6) + P_{42}(4) + P_{52}(8) + P_{81}(2) + P_{92}(12) + P_{102}(6) = 38$ respectively. So, the job $J_{10}$ is scheduled on the machine-1, and the current load on the machine-1 and machine-2 are 32 and 32 respectively. The current makespan of the schedule is 32.

4.3.3 Algorithm5

Algorithm5 is similar to algorithm4 but in the case, when only one machine is available, we do not schedule the job $J_i$ on the faster available machine. Instead, we check for the availability of the next job $J_{i+1}$ and if both jobs are available, then we compare the job $J_i$ with the job $J_{i+1}$ and schedule the job with the highest processing time on the machine-1 and slowest processing time on the machine-2. If we have 5 jobs with the processing times 8,9, 3, 2, 4 and the processing times of these jobs on the machine-1 are $P_{11} = 16, P_{21} = 18, P_{31} = 6, P_{41} = 4, P_{51} = 8$. The job processing times on the machine-2 are $P_{12} = 16, P_{22} = 36, P_{32} = 12, P_{42} = 4, P_{52} = 8$.

At the beginning, both machines are available and job $J_1$, processing time is compared with job $J_2$, processing time and the job with the highest processing time are scheduled on the machine-1 and the job with the lowest processing time is scheduled on the machine-2. Since $P_{21} > P_{11}$, $J_1$ is scheduled on the machine-2, and $J_2$ is scheduled on the machine-1. The load on the machine-2 is
Table 4.8: Job processing of next input

<table>
<thead>
<tr>
<th>Job</th>
<th>machine-1</th>
<th>machine-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>

$P_{12}(16) = 16$. The load on the machine-1 is $P_{21}(9) = 9$ though the load on the machine-1 is less the job $J_3$ is not scheduled on the machine-1. The job $J_3$ is compared with the next job $J_4$. Since $P_{31} > P_{41}$, $J_3$ is scheduled on the machine-1. The load on the machine-1 is $P_{21}(9) + P_{31}(3) = 12$.

To schedule job $J_4$, it is compared with $J_5$, and though the machine-1 is available, we will check for the load on the machines. If job $J_4$ is scheduled on the machine-1 and the machine-2 then the load on the machine-1 will be $P_{21}(9) + P_{31}(3) + P_{41}(2) = 14$ and the load on the machine-2 will be $P_{12}(16) + P_{41}(4) = 20$. So the job $J_4$ is scheduled on the machine-1. Finally, job $J_5$ is also scheduled on the machine-1 since, the load on the machine-1 will be less, when we schedule job $J_5$ on the machine-1.

Figure 4.9: Algorithm5 Gantt chart for job set-1

Table 4.9: Job processing of next input

<table>
<thead>
<tr>
<th>Job</th>
<th>machine-1</th>
<th>machine-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>14</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>6</td>
</tr>
</tbody>
</table>

Let us consider five more jobs $J_6$, $J_7$, $J_8$, $J_9$ and $J_{10}$ with the processing time on the machine-1 as $P_{61} = 5$, $P_{71} = 7$, $P_{81} = 2$, $P_{91} = 6$, $P_{101} = 3$ and the processing time on the machine-2 as $P_{62} = 10$, $P_{72} = 14$, $P_{82} = 4$, $P_{92} = 12$, $P_{102} = 6$ The jobs $J_6$ and $J_7$ are scheduled on the machine-1, the load on the machine-1 after processing $J_6, J_7$ is less compared to load on the machine-2 if these jobs are processed on the machine-2. The load on the machine-1 is
\[ P_{21}(9) + P_{31}(3) + P_{41}(2) + P_{51}(4) + P_{61}(5) + P_{71}(7) = 30 \]

and the load on the machine-2 is \( P_{12}(16) = 16 \). Next job \( J_8 \), with processing time \( P_{81} = 1 \), have to be scheduled. Next job \( J_9 \) is available, and we compare \( J_8 \) with the job \( J_9 \), since \( P_{81} < P_{91} \), we schedule job \( J_9 \) on machine-1 and job \( J_8 \) is scheduled on the machine-2. The load on the machine-1 is \( P_{21}(9) + P_{31}(3) + P_{41}(2) + P_{51}(4) + P_{61}(5) + P_{71}(7) + P_{91}(6) = 36 \) and the load on the machine-2 is \( P_{12}(16) + P_{82}(2) = 18 \). Next job \( J_{10} \), with the processing job \( P_{101} \) have to be scheduled. If \( J_{10} \) is scheduled on the machine-1 then the load on the machine-1 will be \( P_{21}(9) + P_{31}(3) + P_{41}(2) + P_{51}(4) + P_{61}(5) + P_{71}(7) + P_{91}(3) + P_{101}(3) = 39 \). If the job \( J_{10} \) is scheduled on the machine-2, then the load on the machine-2 will be \( P_{12}(16) + P_{82}(2) + P_{102}(6) = 24 \). As the load on the machine-2 is less compared to load on the machine-1 so the job \( J_{10} \) is scheduled on the machine-2. The current makespan of the schedule is 36.

![Figure 4.10: Algorithm5 Gantt chart for job set-2](image)

### Calculating Makespan

**Case 1:**

M1 and M2 are available

If \( P_{i1} \leq P_{(i+1)1} \)

Schedule \( J_i \) on \( M_2 \) and \( J_{(i+1)} \) on \( M_1 \)

else

Schedule \( J_i \) on \( M_1 \) and \( J_{(i+1)} \) on \( M_2 \)

**Case 2:**

Only M1 is available

If \( P_{i1} \leq P_{(i+1)1} \)

Schedule \( J_{(i+1)} \) on M1

If we schedule \( J_i \) on M1 then

\( J_i, J_{i+1} \) will be terminated after
\[ T = P_{i1} + P_{(i+1)1} \]  \hspace{1cm} (4.3)

If we schedule \( J_{i+1} \) on M-2 then
\( J_i, J_{i+1} \) will be terminated after

\[ T^1 = \text{Max}(P_{(i+1)1}, \Delta t + 2 \times P_{i1}) \]  \hspace{1cm} (4.4)

if \( T \leq T^1 \) then
\( J_i \) is scheduled on \( M_1 \)
else on \( M_2 \)

**Case 3:**

Only M2 is available
If \( P_{i1} \leq P_{(i+1)1} \) then
Schedule \( J_i \) on \( M_2 \) and decide on \( J_{i+1} \) same as in case-2
else
Schedule \( J_{i+1} \) on \( M_2 \) and decide on \( J_i \) same as in case-2
Chapter 5

Dynamic Programming Approach

In this subsection, we propose a pseudo polynomial algorithm for solving $Q_2||C_{\text{max}}$ problem.

5.1 Problem Description

In $Q_2||C_{\text{max}}$ problem, one has two independent machines, machine-1 and machine-2 and $n$ jobs $j = 1,...,n$. Each of these jobs can be processed either on the machine-1 or machine-2. If job $J_i$ is processed on the machine-1 (machine-2), then the processing time is $P_i(2P_i)$. We have to assign the jobs to the machines such that the makespan is minimized.

5.2 Dynamic Programming Approach

Notations

All the notations used in the dynamic programming procedure [14].

$n$: the total number of jobs

$J_{ij}$: the job $J_i$ on the machine $M_j$

$P_i$: the processing time of the job $J_i$

$t_1$: the total processing time of all jobs assigned to the machine $M_1$

$t_2$: the total processing time of all jobs assigned to the machine $M_2$

$T$: the upper bound of the makespan

$$T = \max\{\sum_{j=1}^{n} P_j, \sum_{j=1}^{n} (2P_j)\}$$  \hspace{1cm} (5.1)
In this dynamic programming procedure, we use the indicator variable $I(j, t_1, t_2)$ to compute makespan, for $j = 1, 2 \ldots, n$ and
\[ 0 \leq t_1, t_2 \leq T \]
\[(5.2)\]

**Initialization**

This procedure is initialized by setting values to the indicator variables $I(1, t_1, t_2)$.

\[
\text{if}(\ t_1 \geq P_i \ || \ t_2 \geq (2 \ast P_i)) \text{ then} \\
I(1, t_1, t_2) = 1; \\
\text{else} \\
I(1, t_1, t_2) = 0; \\
\]

Where $P_i$, $2 \ast P_i$ are the processing times of the job $J_i$ on the machine-1 and the machine-2 respectively.

**Procedure**

This procedure is done recursively assuming that all the values $I(j-1, t_1, t_2)$ are known.

\[
\text{for (} \ j = 1, 2 \ ..., n \ \text{)} \\
\text{if (} I(j-1, \ \max \ { ( t_1 - P_j), 0 } , \ t_2) = 1 \text{ or if(} I(j-1, \ t_1, \ \max \ { ( t_2 - (2 \ast P_j)), 0 } ) = 1 \text{) } I(1, t_1, t_2) = 1; \text{else} \ I(1, t_1, t_2) = 0; \\
\]

While calculating $\max \ { ( \ t_1 - P_j ), 0 }$ we will make sure there is no possibility of scheduling anything for a negative time value.

\[
\text{if (} \ t_1 \geq P_i \text{) then} \\
I(j-1, \ \max \ { ( t_1 - P_j), 0 } , \ t_2) = 0; \\
\text{Similarly while calculating } \max \ { ( t_2 - 2 \ast P_j), 0 } \text{) } \\
\text{if (} \ t_2 \geq P_i \text{) then} \\
I(j-1, \ t_1, \ \max \ { ( t_2 - 2 \ast P_j), 0 } ) = 0; \\
\]

**Optimal Makespan**

The optimal makespan is given by the smallest $t$ with $I(n, t, t) = 1$. 

27
Time Complexity

The time complexity of the algorithm is $O(n \cdot T^2)$, where $n$ is the total number of jobs, and $T$ is the upper bound for the makespan.

5.3 Example

Example1

In this Section, we will see the working of the dynamic programming approach, to obtain an optimal makespan for the jobs $J_i$, where $i=1,2,3$.

Using the initialization method, mentioned in Section 5.2, we get the $I(0,t1,t2)$ as follows,

$$
\begin{bmatrix}
0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{bmatrix}
$$

$I(1,t1,t2)$ matrix is as follows

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{bmatrix}
$$
$I(2,t_1,t_2)$ is as follows

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 
\end{bmatrix}
$$

We get the above matrix, upon following the procedure mention the section 5.1. **Optimal makespan** of the jobset in the Table 5.1 is given by the last matrix $I(2,t_1,t_2)$. The smallest $t$ with $I(n, t, t) = 1$, gives the optimal makespan. In the above matrix, $I(2,3,3) = 1$ when $t = 3$, so the optimal makespan is 3.

**Table 5.1: Dynamic Approach for job set**

<table>
<thead>
<tr>
<th>Job</th>
<th>Processing time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

**Example2**

Comparing the results generated by the algorithms proposed in the chapter-4 with the optimal makespan.

**Table 5.2: Job Set-1**

<table>
<thead>
<tr>
<th>Job</th>
<th>Processing time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
</tr>
</tbody>
</table>
Table 5.3: Makespan of all the algorithms

<table>
<thead>
<tr>
<th>Name of Algorithm</th>
<th>Makespan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm1</td>
<td>32</td>
</tr>
<tr>
<td>Algorithm2</td>
<td>35</td>
</tr>
<tr>
<td>Algorithm3</td>
<td>32</td>
</tr>
<tr>
<td>Algorithm4</td>
<td>32</td>
</tr>
<tr>
<td>Algorithm5</td>
<td>36</td>
</tr>
</tbody>
</table>

Optimal makespan generated by dynamic programming is 32

In this case optimal makespan is near to the makespan generated by the proposed algorithms
Chapter 6

Results Evaluation

A different set of jobs, are used to evaluate the performance of the algorithms, proposed in the Chapter 4. A set of jobs has to scheduled on the machines, such that the total completion time of jobs is minimized. This set of jobs will demonstrate the effectiveness of the proposed algorithms by comparing their makespan.

In this chapter, we will further discuss the performance of the algorithms based on the number of jobs with the different processing time, available in sets. In addition, we also compare the makespan generated by the proposed algorithms against the optimal solution obtained using dynamic programming procedure. Dynamic programming procedure gives the optimal makespan for a set of job. It is one of the best benchmark to compare our results.

6.1 Performance Measurement of Algorithms

Job Set1

Let's consider the following job set 1, 2, 2, 3, 5, 8, 13, 21

<table>
<thead>
<tr>
<th>Job</th>
<th>Processing time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>13</td>
</tr>
<tr>
<td>8</td>
<td>21</td>
</tr>
</tbody>
</table>
Figures 6.1, 6.2, 6.3 represent the Gantt chart of the jobs, which are scheduled by the algorithm3, algorithm4 and algorithm5 respectively. Makespan generated by these algorithms are 37, 50 and 35. This experiment shows that algorithm3 and algorithm5 work efficiently, when the job processing times increase gradually.

**Job Set2**

Let's consider the following job set 1, 3, 5, 7, 9, 11, 13, 15

Figures 6.4, 6.5 and 6.6 represents, the Gantt chart of the jobs, which are scheduled by the algorithm3, algorithm4 and algorithm5 respectively. Makespan generated by these algorithms are
Table 6.2: Job set 2 processing times

<table>
<thead>
<tr>
<th>Job</th>
<th>Processing time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>11</td>
</tr>
<tr>
<td>7</td>
<td>13</td>
</tr>
<tr>
<td>8</td>
<td>15</td>
</tr>
</tbody>
</table>

Machine 1

<table>
<thead>
<tr>
<th>J_1(1)</th>
<th>J_2(3)</th>
<th>J_3(5)</th>
<th>J_4(9)</th>
<th>J_5(11)</th>
<th>J_6(15)</th>
</tr>
</thead>
</table>

Machine 2

<table>
<thead>
<tr>
<th>J_7(17)</th>
<th>J_8(26)</th>
</tr>
</thead>
</table>

Figure 6.5: machine-oriented Gantt chart for job set-2

Machine 1

<table>
<thead>
<tr>
<th>J_1(3)</th>
<th>J_2(7)</th>
<th>J_3(11)</th>
<th>J_4(9)</th>
<th>J_5(15)</th>
</tr>
</thead>
</table>

Machine 2

<table>
<thead>
<tr>
<th>J_7(17)</th>
<th>J_8(26)</th>
</tr>
</thead>
</table>

Figure 6.6: machine-oriented Gantt chart for job set-2

54, 34 and 45. This results show that algorithm 4 works efficiently, when the job processing times increase by a constant factor. In the job set-2 the processing times of the jobs are increased by a constant factor 2.

Job Set 3

Let’s consider the following job set 2, 3, 5, 7, 11, 13, 17, 19, 23

<table>
<thead>
<tr>
<th>Machine-1</th>
<th>J_1(2)</th>
<th>J_3(5)</th>
<th>J_5(11)</th>
<th>J_6(13)</th>
<th>J_8(19)</th>
<th>J_9(23)</th>
</tr>
</thead>
</table>

Machine-2

| J_2(6) | J_4(14) | J_7(34) | —       |

Figure 6.7: machine-oriented Gantt chart for job set-3

Figures 6.7, 6.8 and 6.9 represents the Gantt chart of the jobs, which are scheduled by the algorithm 3, algorithm 4 and algorithm 5 respectively. The makespan generated by these algorithms are 73, 76 and 94.
Table 6.3: Job set2 processing times

<table>
<thead>
<tr>
<th>Job</th>
<th>Processing time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>11</td>
</tr>
<tr>
<td>6</td>
<td>13</td>
</tr>
<tr>
<td>7</td>
<td>17</td>
</tr>
<tr>
<td>8</td>
<td>19</td>
</tr>
<tr>
<td>9</td>
<td>23</td>
</tr>
</tbody>
</table>

This results show that algorithm3 works efficiently, when the job processing times increase by a constant factor. In the job set-2 the processing times of the jobs are increased randomly.

6.2 Comparison between the Algorithms and the Optimal Solution

In this Section, we compare the results of the proposed algorithms, with an optimal solution generated by a dynamic programming strategy, proposed in the Chapter-5. This Section is used to estimate the effectiveness of the proposed algorithms.

Job Set-1

Let's consider the following job set \{ 1, 1, 2, 2, 1, 3, 1, 2, 3, 1, 2, 1, 1, 1 \}

Optimal makespan, given by dynamic programming procedure for the above job set is 17.

Figures 6.10, 6.11, 6.12, 6.13, 6.14 represents the Gantt chart of the jobs, which are scheduled on the algorithm1,algorithm2,algorithm3,algorithm4 and algorithm5. Makespan generated by these algorithms are 17,18,17,17 and 17 respectively. Except, algorithm2 all other algorithms give an optimal solution for the job set-1.
Table 6.4: Job Set-1

<table>
<thead>
<tr>
<th>Job</th>
<th>Processing time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
</tr>
</tbody>
</table>

![Figure 6.10: Algorithm1-comparison job set-1](image1)

Figure 6.10: Algorithm1-comparison job set-1

![Figure 6.11: Algorithm2-comparison job set-1](image2)

Figure 6.11: Algorithm2-comparison job set-1

![Figure 6.12: Algorithm3-comparison job set-1](image3)

Figure 6.12: Algorithm3-comparison job set-1

![Figure 6.13: Algorithm4-comparison job set-1](image4)

Figure 6.13: Algorithm4-comparison job set-1

Figures 6.10, 6.11, 6.12, 6.13 and 6.14 represents the Gantt chart of the jobs, which are scheduled on the algorithm1,algorithm2,algorithm3,algorithm4 and algorithm5. Makespan generated
by these algorithms are 17, 18, 17, 17 and 17 respectively. Except, algorithm2 all other algorithms give an optimal solution for the job set-1.

**Comparision results for jobset-1**

We consider 20 jobsets of size $n = 5$, to compare the results of the proposed algorithms with the optimal solution. Table 6.6 represents the jobset-1 data. Table 6.7 represents the makespan of the algorithms. Table 6.8, represents the average of the ratios $r_i$ calculated using $r_i = m_i/m_{opt}$.

In the jobset-1, algorithm4 performance was better comparing to all other algorithms.

**Comparision results for jobset-2**

We consider 20 jobsets of size $n = 10$, to compare the results of the proposed algorithms with the optimal solution. For the sample 20 jobsets, considered for the testing the algorithm, we calculate the makespan and calculated makespans are represented in Table 6.9. Table 6.10, represents the average of the ratios $r_i$ calculated using $r_i = m_i/m_{opt}$. In the jobset-2, algorithm5 performance was better comparing to all other algorithms.

**Comparision results for jobset-3**

We consider 20 jobsets of size $n = 15$, to compare the results of the proposed algorithms with the optimal solution. For the sample 20 jobsets, considered for the testing the algorithm, we calculate the makespan and calculated makespans are represented in Table 6.11. Table 6.12, represents the average of the ratios $r_i$ calculated using $r_i = m_i/m_{opt}$. In the jobset-3, algorithm4 performance was better comparing to all other algorithms.

<table>
<thead>
<tr>
<th>M1</th>
<th>J2</th>
<th>J3</th>
<th>J4</th>
<th>J5</th>
<th>J6</th>
<th>J7</th>
<th>J8</th>
<th>J9</th>
<th>J10</th>
<th>J11</th>
<th>J12</th>
<th>J13</th>
<th>J14</th>
<th>J15</th>
</tr>
</thead>
<tbody>
<tr>
<td>M2</td>
<td>11</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>17</td>
</tr>
</tbody>
</table>

Figure 6.14: Algorithm5-comparision job set-1
Table 6.6: Jobset-1 data

<table>
<thead>
<tr>
<th>Job</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
</tr>
</thead>
<tbody>
<tr>
<td>J1</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td>J2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>J3</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>J4</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>J5</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>J6</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>J7</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>J8</td>
<td>6</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>J9</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>J10</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>J11</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>J12</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>J13</td>
<td>8</td>
<td>9</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>J14</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>J15</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>J16</td>
<td>9</td>
<td>10</td>
<td>13</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>J17</td>
<td>24</td>
<td>50</td>
<td>10</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>J18</td>
<td>40</td>
<td>50</td>
<td>10</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>J19</td>
<td>10</td>
<td>9</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>J10</td>
<td>10</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>J21</td>
<td>5</td>
<td>9</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Comparision results for jobset-4

We consider 20 jobsets of size \( n = 20 \), to compare the results of the proposed algorithms with the optimal solution. For the sample 20 jobsets, considered for the testing the algorithm, we calculate the makespan and calculated makespans are represented in Table 6.13. Table 6.14, represents the average of the ratios \( r_i \) calculated using \( r_i = m_i/m_{opt} \). In the jobset-4, algorithm4 performance was better comparing to all other algorithms.

Comparision results for jobset-5

We consider 20 jobsets of size \( n = 25 \), to compare the results of the proposed algorithms with the optimal solution. For the sample 20 jobsets, considered for the testing the algorithm, we calculate the makespan and calculated makespans are represented in Table 6.15. Table 6.16, represents the average of the ratios \( r_i \) calculated using \( r_i = m_i/m_{opt} \). In the jobset-4, algorithm4 performance was better comparing to all other algorithms.
Table 6.7: Makespan of all the algorithms

<table>
<thead>
<tr>
<th></th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>A4</th>
<th>A5</th>
<th>OT</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>30</td>
<td>28</td>
<td>26</td>
<td>26</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>14</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>19</td>
<td>18</td>
<td>16</td>
<td>17</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>21</td>
<td>22</td>
<td>18</td>
<td>18</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>12</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>13</td>
<td>14</td>
<td>14</td>
<td>16</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>10</td>
<td>10</td>
<td>11</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>19</td>
<td>24</td>
<td>21</td>
<td>22</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>21</td>
<td>17</td>
<td>17</td>
<td>18</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>12</td>
<td>14</td>
<td>11</td>
<td>11</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>16</td>
<td>18</td>
<td>17</td>
<td>20</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>32</td>
<td>24</td>
<td>26</td>
<td>24</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>84</td>
<td>100</td>
<td>74</td>
<td>60</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>100</td>
<td>90</td>
<td>80</td>
<td>80</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>20</td>
<td>18</td>
<td>18</td>
<td>18</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>21</td>
<td>26</td>
<td>20</td>
<td>26</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>17</td>
<td>18</td>
<td>14</td>
<td>16</td>
<td>14</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.8: Average rate of algorithms

<table>
<thead>
<tr>
<th></th>
<th>sum</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>25.2129931</td>
<td>1.200618719</td>
</tr>
<tr>
<td>A2</td>
<td>24.81743</td>
<td>1.181782381</td>
</tr>
<tr>
<td>A3</td>
<td>25.21299</td>
<td>1.200618571</td>
</tr>
<tr>
<td>A4</td>
<td>22.67282</td>
<td>1.079658095</td>
</tr>
<tr>
<td>A5</td>
<td>23.60579</td>
<td>1.124085238</td>
</tr>
</tbody>
</table>

Comparision results for jobset-6

We consider 20 jobsets of size n = 30, to compare the results of the proposed algorithms with the optimal solution. For the sample 20 jobsets, considered for the testing the algorithm, we calculate the makespan and calculated makespans are represented in Table 6.17. Table 6.18, represents the average of the ratios $r_i$ calculated using $r_i = m_i/m_{opt}$. In the jobset-5, algorithm5 performance was better comparing to all other algorithms.
Table 6.9: Makespan of all the algorithms

<table>
<thead>
<tr>
<th></th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
<th>M5</th>
<th>OPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>14</td>
<td>151</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>20</td>
<td>23</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>14</td>
<td>16</td>
<td>14</td>
<td>16</td>
<td>14</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>15</td>
<td>16</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>32</td>
<td>30</td>
<td>32</td>
<td>28</td>
<td>28</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>47</td>
<td>46</td>
<td>36</td>
<td>37</td>
<td>36</td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>59</td>
<td>46</td>
<td>47</td>
<td>46</td>
<td>46</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>24</td>
<td>23</td>
<td>23</td>
<td>22</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>22</td>
<td>19</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>109</td>
<td>157</td>
<td>109</td>
<td>110</td>
<td>110</td>
<td>109</td>
<td></td>
</tr>
<tr>
<td>341</td>
<td>480</td>
<td>341</td>
<td>383</td>
<td>426</td>
<td>341</td>
<td></td>
</tr>
<tr>
<td>341</td>
<td>480</td>
<td>341</td>
<td>341</td>
<td>352</td>
<td>341</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>50</td>
<td>42</td>
<td>42</td>
<td>40</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>12</td>
<td>14</td>
<td>12</td>
<td>13</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>43</td>
<td>33</td>
<td>35</td>
<td>33</td>
<td>33</td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>162</td>
<td>200</td>
<td>150</td>
<td>114</td>
<td>113</td>
<td></td>
</tr>
<tr>
<td>220</td>
<td>250</td>
<td>220</td>
<td>220</td>
<td>200</td>
<td>200</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.10: Makespan of all the algorithms

<table>
<thead>
<tr>
<th></th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
<th>M5</th>
<th>OPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>14</td>
<td>151</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>20</td>
<td>23</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>14</td>
<td>16</td>
<td>14</td>
<td>16</td>
<td>14</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>15</td>
<td>16</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>32</td>
<td>30</td>
<td>32</td>
<td>28</td>
<td>28</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>47</td>
<td>46</td>
<td>36</td>
<td>37</td>
<td>36</td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>59</td>
<td>46</td>
<td>47</td>
<td>46</td>
<td>46</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>24</td>
<td>23</td>
<td>23</td>
<td>22</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>22</td>
<td>19</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>109</td>
<td>157</td>
<td>109</td>
<td>110</td>
<td>110</td>
<td>109</td>
<td></td>
</tr>
<tr>
<td>341</td>
<td>480</td>
<td>341</td>
<td>383</td>
<td>426</td>
<td>341</td>
<td></td>
</tr>
<tr>
<td>341</td>
<td>480</td>
<td>341</td>
<td>341</td>
<td>352</td>
<td>341</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>50</td>
<td>42</td>
<td>42</td>
<td>40</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>12</td>
<td>14</td>
<td>12</td>
<td>13</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>43</td>
<td>33</td>
<td>35</td>
<td>33</td>
<td>33</td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>162</td>
<td>200</td>
<td>150</td>
<td>114</td>
<td>113</td>
<td></td>
</tr>
<tr>
<td>220</td>
<td>250</td>
<td>220</td>
<td>220</td>
<td>200</td>
<td>200</td>
<td></td>
</tr>
</tbody>
</table>
Table 6.11: Average rate of algorithms

<table>
<thead>
<tr>
<th>Alg</th>
<th>sum</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>21.6405797</td>
<td>1.082028985</td>
</tr>
<tr>
<td>A2</td>
<td>34.0569</td>
<td>1.702845</td>
</tr>
<tr>
<td>A3</td>
<td>21.64058</td>
<td>1.082029</td>
</tr>
<tr>
<td>A4</td>
<td>21.06262</td>
<td>1.053131</td>
</tr>
<tr>
<td>A5</td>
<td>20.54021</td>
<td>1.0270105</td>
</tr>
</tbody>
</table>

Table 6.12: Jobset-3 Rate

<table>
<thead>
<tr>
<th>Average rate of algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
</tr>
<tr>
<td>A2</td>
</tr>
<tr>
<td>A3</td>
</tr>
<tr>
<td>A4</td>
</tr>
<tr>
<td>A5</td>
</tr>
</tbody>
</table>

Table 6.13: Jobset-4 Rate

<table>
<thead>
<tr>
<th>Average rate of algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
</tr>
<tr>
<td>A2</td>
</tr>
<tr>
<td>A3</td>
</tr>
<tr>
<td>A4</td>
</tr>
<tr>
<td>A5</td>
</tr>
</tbody>
</table>

Table 6.14: Jobset-5

<table>
<thead>
<tr>
<th>Average rate of algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
</tr>
<tr>
<td>A2</td>
</tr>
<tr>
<td>A3</td>
</tr>
<tr>
<td>A4</td>
</tr>
<tr>
<td>A5</td>
</tr>
</tbody>
</table>

Table 6.15: Jobset-6

<table>
<thead>
<tr>
<th>Average rate of algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
</tr>
<tr>
<td>A2</td>
</tr>
<tr>
<td>A3</td>
</tr>
<tr>
<td>A4</td>
</tr>
<tr>
<td>A5</td>
</tr>
</tbody>
</table>
Chapter 7

Conclusion and Future work

In this study we have considered the problem of scheduling $n$ independent jobs on two uniform parallel machines with an objective to minimize the make span ($Q_2||C_{max}$). A few efficient algorithms have been developed to solve large-scale and practical problems in scheduling. These algorithms performance is tested using different set of jobs. Their efficiency to schedule $n$ independent jobs, to produce an optimal make span is tested by comparing their respective make spans with the optimal make span generated by a pseudo polynomial procedure. The time complexity of this dynamic programming approach is $O(n\ T^2)$, so this approach can be adopted when the number jobs and total processing times are not very large. Further we can improve the efficiency of the proposed algorithms and the time complexity of the dynamic programming procedure can be further enhanced.
Bibliography


Vita

Graduate College
University of Nevada, Las Vegas

Sandhya Kodimala

Degrees:
Masters of Science in Computer Science 2013
Las vegas, Nevada

Thesis Title: Scheduling Jobs on Two Uniform Parallel Machines to Minimize the Makespan

Thesis Examination Committee:
Chairperson, Dr. Wolfgang Bein, Ph.D.
Committee Member, Dr. Ajoy K Datta, Ph.D.
Committee Member, Dr. Lawrence L.Larmore, Ph.D.
Graduate Faculty Representative, Dr. Venkatesan Mutukumar, Ph.D.