8-1-2014

A Characterization of Open Shop Scheduling Problems using the Hall Theorem and Network Flow

Arunasri Chitti
University of Nevada, Las Vegas, chittia@unlv.nevada.edu

A CHARACTERIZATION OF OPEN SHOP SCHEDULING PROBLEMS USING THE HALL THEOREM AND NETWORK FLOW

By

Arunasri Chitti

Bachelor of Technology, Information Technology
Osmania University, India
2012

A thesis submitted in partial fulfillment
of the requirements for the

Master of Science Degree in Computer Science

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

University of Nevada, Las Vegas
May 2014

UNLV
UNIVERSITY OF NEVADA LAS VEGAS

THE GRADUATE COLLEGE

We recommend the thesis prepared under our supervision by

Arunasri Chitti

entitled

A Characterization of Open Shop Scheduling Problems using the Hall Theorem and Network Flow

is approved in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science
Department of Computer Science

Dr. Wolfgang Bein, Ph.D., Committee Chair

Dr. Ajoy K. Datta, Ph.D., Committee Member

Dr. Lawrence L.Larmore, Ph.D., Committee Member

Dr. Venkatesan Muthukumar, Ph.D., Graduate Faculty Representative

Kathryn Hausbeck Korgan, Ph.D., Interim Dean of the Graduate College

August 2014

ABSTRACT

A Characterization of Open Shop Scheduling Problems using the Hall Theorem and Network Flow

by

Arunasri Chitti

Dr. Wolfgang Bein, Examination Committee Chair

Professor of Computer Science

University of Nevada, Las Vegas

Open shop scheduling problems are combinatorial problems where jobs with certain processing requirements on a number of different machines must be arranged in such a way that objectives related to completion time are optimized. Such problems have applications over a wide spectrum including such as communications, routing and manufacturing.

Many open shop problems are NP-hard but there are a number of special cases which possess polynomial solutions in the case of few machines or few jobs or when preemption of jobs is permitted. Many such solutions are based in the theory of matching or Hall's theorem, or more generally network flow. The primary focus of this thesis is to describe a number of polynomial-time solutions which are constructed using these related concepts and methods.

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. Lawrence L.Larmore for his supervision, advice and guidance throughout my thesis. I convey my special thanks to the graduate coordinator, Dr. Ajoy K. Datta for all his support and advocacy. 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. I extend my gratitude to the Computer Science department for funding my Master's degree. Finally, I would like to thank my parents, sisters and friends for all the support and encouragement they gave me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Scheduling problems are combinatorial problems where jobs with certain processing requirements need to be arranged such that different objectives such as time or deadline are met or optimized under limited resources such as limited number of machines.

The Open shop scheduling has been applied in many areas of manufacturing and communications such as scheduling and wavelength assignment (SWA), routing in optical transpose interconnect system and in satellites and SOC testing.

Open Shop Scheduling problems are frequently combinatorially hard i.e. they are often NP-Hard. Therefore solutions for these problems are not very simple, but there are a number of special cases that do have polynomial solutions and those solutions often come from the theory of matching, network flow and Hall's theorem, which are related concepts and methods.

The primary focus of this thesis is to investigate a number of problems where we have unifying solutions that come from applying matching and network flow techniques. Specifically, we emphasize the use of Hall's theorem as numerous solutions can be derived by applying this theorem directly or iteratively. We also discuss the fact that most open shop problems indeed are not easily solvable as we can show that open shop problem with three machines without preemption is NP-Complete.

The disjunctive graph, a graph with conjunctive and disjunctive edges, is frequently used in shop problems. Most open shop problems are not solvable in polynomial time and then in order to obtain heuristics we use the disjunctive graph.

1.1 Outline

First we begin with the basic scheduling notations that are given in Chapter 2. In Chapter 3 we discuss about a simple linear time algorithm for the two machine makespan non preemptive scheduling problems. In Chapter 4, we turn to Hall's theorem and its application on various open shop scheduling problems. The maximum flow algorithm is demonstrated with an example in Chapter 5. The parallel machine problems and the reduction from parallel machines to open shop problems are discussed in Chapter 6. Chapter 7 discusses in detail about the hard open shop problems and Chapter 8 demonstrates the local neighborhood search and shows how disjunctive graph is useful in searching for the neighbors.

CHAPTER 2

Scheduling Problems and Notations

This chapter provides us with the details about Scheduling and other fundamental aspects of Scheduling.

2.1 Scheduling

Scheduling deals with the arrangement of a given set of jobs, on the given machine such that the provided resources are assigned to the jobs in an optimal way. Scheduling provides us the starting and the completion time for each operation or job.

2.2 Scheduling Problems

According to Peter Brucker [1], a Scheduling problem is defined as following:

Given m machines M_j (j= 1, 2...m) and n jobs J_i (i= 1, 2…n), a schedule now is allocation of one or more time intervals for each job on one or more machines.

The open shop problem is defined below:

Given are m machines denoted by M_1, M_2…M_m and n jobs denoted by J_1, J_2… J_n and each job consists of m operations which are to be processed on the machines. The jth operation of the job J_i must be processed on M_j and the processing time of this operation is given by $p_{ij} \geq 0$. The total processing time for J_i is $\sum_j p_{ij}$ and the total processing time on machine M_j is $\sum_i p_{ij}$. In the open shop problem, the order in which operations are processed is immaterial.

2.3 Terminology in Scheduling

Job

A job is nothing but a set of tasks (operations) and the operations have to be processed on the machines for a particular time interval. A job can be represented as J_i and its operations can be represented as O_{ij}, where i is the job and j is the jth operation of that job. This notation is used throughout the document.

Processing Time

Processing Time is the time that an operation takes when it is processed on a machine.It is denoted by $p_{i,j}$ for an operation O_{ij}. The processing time $p_{i,j}$ on any machine is ≥ 0.

Idle Time

Idle time is that particular time interval at which the machine is idle and there is no task being processed on that machine.

Makespan

Makespan of the schedule is the time at which the last job on the last machines gets completed. It is denoted by C_{max}. It is nothing but the time at which all the jobs have completed processing their operations on the allocated machines.

The Makespan that has minimum value than all the other possible makespan values for a given problem is chosen as the optimal makespan.

Completion Time

The Completion time of a job J_i is the amount of time taken by the job to complete its processing on the last machine. It is denoted by C_i.

The sum of the completion times $\sum C_i$ of a schedule whose value is minimum

among all the other possible $\sum C_i$ for a problem is considered as the optimal $\sum C_i$.

2.4 Classes of Scheduling

Scheduling is classified into various classes and they are represented in the form of notation $\alpha \mid \beta \mid \gamma$, where α denotes the machine environment and β denotes the job characteristics and γ denotes the optimality criteria for the schedule.

2.4.1 Machine Environment

In Scheduling environment, there are different types of machines that are available. Each machine has its own properties associated with it, all these properties altogether constitute to the machine environment. The machine environment is denoted by α. The string α has two strings $\alpha 1$ and $\alpha 2$. In the string α, $\alpha 2$ is the number of machines used in the scheduling. The string $\alpha 1$ belongs to the set {*o*, P, Q, R, PMPM, QMPM, G, X, O, J, and F}. Each element in the set has a different functionality. Several cases are discussed below. Each operation Oij is associated with a set of machines, represented as $\mu_{ij} \subseteq \{M_1 ... M_m\}$, where m denotes the number of machines. O_{ij} may be processed on any of the machines that belong to set μ_{ij}. This property varies depending upon the type of machine chosen.

Case 1: $\alpha 1 = 0$, then we have $\alpha = \alpha 2$. This is called a dedicated machine environment. It is because, 0 represents an empty symbol. Each job has only one operation and it is processed only on one machine.

Case 2: $\alpha 1 \in \{P, Q, \text{and } R\}$, now the jobs are said to run on Parallel machines. Here, a job can have more than one operation and each job can be processed on all the machines in the set M_1 to M_m. This case is subdivided into three parts depending on weather it is P or Q or R [2].

If $\alpha 1 = P$, then the processing times of a job for all the machines that belong to the set μ_{ij} are identical. Therefore, the scheduling is said to be done on Identical Parallel Machines. $P_{ij} = P_i$ for each element in the set μ_{ij}.

If $\alpha 1 = Q$, then each machine is associated with a speed. The speed of the machine is uniform for all the jobs processing on the same machine. Now, the scheduling is said to be performed on Uniform Parallel Machines.

If $\alpha 1 = R$, then every machine is associated with a specific speed and every job is associated with a specific speed for each machine. In this case, the scheduling is said to be performed on Unrelated Parallel Machines.

Case 4: If $\alpha 1$ = PMPM or QMPM, the machines are said to be called Multipurpose Machines [34]. PMPM is that category of machines where each job has identical speed on all the machines and where the speed of a job varies from another. QMPM is that category of machines that will have the same speed associated with the jobs being processed on the same machine. One machine speed may vary from another.

Case 5: If $\alpha 1 \in \{G, X, O, J, F\}$, then each job would be having a set of operations and each operation would be performed on specific machine.

When $\alpha 1 = G$, it is called General Flow Shop. Here, an ordering is associated with the operations of a job to process on a machine. This ordering is nothing but the precedence relation among the operations of a job. Each operation is associated with a priority and based on this; the operation with higher priority will be processed first than the operation with lowest priority. The General Flow Shop can result in Open shop,Flow shop and Job Shop by altering some conditions.

If $\alpha 1$ = J, it is represented as Job Shop. Each job has set of operations O_{ij}.A Precedence relation exists among the operations and is denoted by ->. Given a constraint O_{i1} -> O_{i2}, the operation O_{i2} can start getting processed only after the operation O_{i1} is finished. This relation exists between all the operation $O_{i1,}$ $O_{i2,}$ O_{i3}, ...,O_{ik} , where k is the number of operations of a job [35]. In Job Shop, each job has its own machine sequence.

If $\alpha 1$ = F, it is represented as Flow Shop [36]. Here, the number of operations of a job is same as the number of machines. For each job, there is one operation performed per machine. In Flow Shop, all the jobs follow the same machine sequence. The job sequence may vary from one machine to other machine. Permutation Flow Shop is a special case in Flow Shop where the job sequence is same for all the machines.

If $\alpha 1$ = O, it is represented as Open Shop [28]. It is similar to Flow Shop if a special constraint is associated with it.There is no precedence relation between the operations of a job, i.e.there is no order in which the operations of a job are processed.

2.4.2 Job Characteristics

Job Characteristics shows the characteristics a job is associated and is represented by β. It is a set consisting of six characteristics and is given by { β_1, β_2, β_3, β_4, β_5, β_6 }. Each element in the set is associated with a property.

Case 1: when $\beta = \beta_1$, then preemption is applied. The jobs undergo preemption. It is given as β = pmtn. Preemption in the schedule allows a job to undergo interruption and resume at a later point.

Case 2: when $\beta = \beta_2$, then precedence relation is applied. It is given as β = prec. In this case, jobs are assigned priorities and the job with higher priority is processed first and jobs with lower priority are processed later. The precedence relation is maintained through the schedule. There are various structures where the jobs can be arranged and they are Intree, Outtree, SP-graph, Chains.

If the structure is an Intree, we represent it as β_2 = Intree. Here, every node of the tree has 0 or 1 outgoing connections.

If it is an Outtree, every node in tree will be having 0 or 1 incoming connections. It is represented as β_2 = Outtree. If it is a chain, then every node in the given tree will be connected to only 0 or 1 nodes and is represented as β_2 = chain.

If it is a series parallel graph, then it is represented as β_2 = sp-graph. It can be a base graph with single vertex or it can be a composition of two graphs. In the latter case, the graphs are joined in such a way that the vertices of the two graphs are joined with one another, also the edges are joined in the similar manner. There is another way in which the graphs can be joined i.e by linking the leaf nodes of one graph with the source nodes of other graph.

Case 3: when $\beta = \beta_3$, then release dates (r_i) are associated with each jobs. Release date for a job is the time at which the job's first operation is available for processing. Here, r is the release time and i is the job number. This case is represented as $\beta_3 = r_i$.

Case 4: when $\beta = \beta_4$ **,** the jobs are associated with processing times. If the processing times of the operations of the job are 1, then it is called as unit processing requirement.

Case 5: when $\beta = \beta_5$ **,** the jobs are associated with deadlines d_i. Here, d_i represents the deadline for the job i. It is the time when a job has to finish its processing on the machine. It is represented as $\beta_5 = d_i$.

Case 6: when $\beta = \beta_6$ **,** the jobs are processed as batches. Batching is nothing but grouping together a set of jobs and scheduling them back to back with no set up time between them.

There are two types of batching: p-batching and s-batching. In p-batching, the length of the batch is the maximum of the processing times of all the 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. It can be represented as β_6 = p-batch or s-batch.

2.4.3 Optimality Criteria

The Optimality Criteria is represented by γ. It is a cost function associated with every scheduling problem and the aim is to minimize it as much as possible. The following are the cases to be considered.

Case 1: This is known as bottleneck objective function. This is about minimizing the time taken by the last job on the last machine.

Case 2: This is known as sum objective function. This is about minimizing the sum of the completion time of all the jobs.

The other parameters that are used in this optimality criterion are Lateness, Earliness, Tardiness, Deviations and Penalties.

Lateness:

It is the amount of time by which the job gets delayed in finishing its processing. C_i is the Completion time of the job i and d_i is the deadline by which the job i has to

finish processing Lateness is the difference the C_i and d_i values. It is denoted as C_i - d_i.

Earliness:

It is amount of time by which the job gets finished early than the given deadline. C_i is the Completion time of the job i and d_i is the deadline of the job i. The difference between these two will give the earliness and is represented as $E_i = d_i - C_i$, if the difference is greater than 0 else $E_i = 0$.

Tardiness:

It is denoted as T_i for a job i and is same as lateness except that it is true only when L_i greater than 0, otherwise T_i is 0.

Deviations:

There are two kinds of deviations: Squared Deviation and Absolute Deviation.

Squared Deviation: It is the deviation that is calculated as the squared value of lateness and is represented as S_i. S_i given as $(C_i - d_i)^2$.

Absolute Deviation: It is the deviation that is calculated as the absolute value of lateness and is represented as D_i. D_i is given as $| C_i - d_i |$.

Unit Penalty:

It is represented as U_i for a job i and when the value of lateness of a job is greater than 0, a penalty of 1 is associated with that job. Else 0 is associated.

CHAPTER 3

$O2||C_{max}$

Now, we focus on our first Open Shop problem, $O_2||C_{max}$ that is polynomially solvable if there are arbitrary processing times and also if there is no preemption. According to Gonzalez and Sahni[19], the algorithm that solves this is described as follows:

Let A and B be the two machines and a_i be the processing time of job i on machine A and b_i be the processing time of job i on machine B, where i= 1..n.

Consider two sets I and J that are defined below:

$$I = \{i \mid a_i \leq b_i;\ i = 1...n\}$$

$$J = \{i \mid a_i > b_i;\ i = 1\ldots n\}$$

I is the set of those jobs whose processing time on machine B is greater than or equal to on machine A and J is the set of those jobs whose processing time on machine A is greater than on machine B.

We now consider two cases that will give optimal schedules:

$$a_r = \max\{\max\{a_i \mid i \in I\}, \max\{b_i \mid i \in J\}\}$$

An optimal schedule is achieved in the following way:

- On machine A, all the I – r jobs are scheduled in arbitrary order, then the all the jobs in set J are scheduled in arbitrary order and then job r is scheduled.
- On machine B, first job r is scheduled, then I – r jobs are scheduled in the same order as on machine A and then jobs in set J are scheduled in the same order as on machine A.

The optimal schedule is given below:

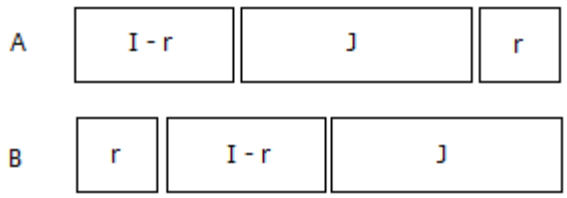

Figure 3.1 Optimal Schedule

Case 2:

$$b_r = \max\{\max\{a_i \mid i \in I\}, \max\{b_i \mid i \in J\}\}$$

An optimal schedule is achieved in the following way:

- On machine A, first job r is scheduled, then jobs in set J – r are scheduled in an arbitrary order, then jobs in set I are scheduled in arbitrary order.
- On machine B, J – r jobs are scheduled in the same order as on machine A and then jobs in set I are scheduled in the same order as on machine A and then job r is scheduled.

The optimal schedule is given below:

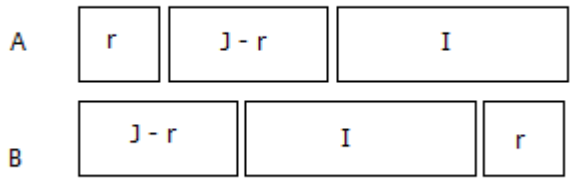

Figure 3.2 Optimal Schedule

Consider the following example:

Given are five jobs J_1, J_2, J_3, J_4, J_5 and two machines A and B.

	A	B
J_1	2	1
J_2	7	6
J_3	2	4
J_4	5	3
J_5	1	3

Table 3.1 To Demonstrate $O_2||C_{max}$

The set I and J are given below:

$$I = \{3, 5\} \text{ and } J = \{1, 2, 4\}$$

Now, we find a_r and b_r values:

a_r is the maximum processing time on machine A from the set I and is given by $a_3 = 2$ and b_r is the maximum processing time on machine B from set J and is given by $b_2 = 6$. We know that b_2 is greater than a_3, therefore our r job is job 2 and we show the optimal schedule using case 2.

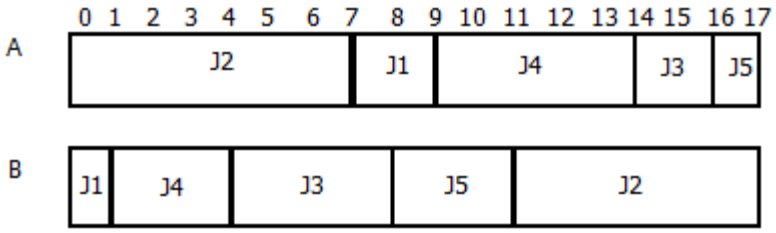

Figure 3.3 To demonstrate $O_2||C_{max}$

CHAPTER 4

Hall's Theorem

Bipartite Graph:

A bipartite graph G given as G = (A, B, E) whose vertices are divided into two disjoint sets A and B such that every edge in E connects a vertex in A to one vertex in B.

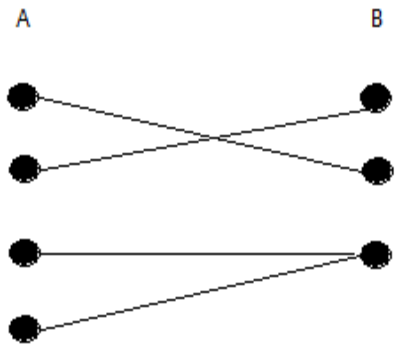

Figure 4.1 Bipartite Graph

Matching:

A matching for a graph G = (V, E) is set of non-adjacent edges, i.e. no two edges share a common.

4.1 Hall's Theorem

Let V_1 and V_2 be the vertex sets of a bipartite graph G where $|V_1| \leq |V_2|$ and E is the edge set. In a Complete Matching, each vertex in the set V_1 is connected by an edge with exactly one vertex in the set V_2.

We define $\Gamma(S)$ as following:

For a subset $S \subset V_1$,

$$\Gamma(S) = \{v \in V_2 : uv \in E \text{ for some } u \in V_1\} \subset V_2.$$

$$\gamma(S) = \{u \in V_1 : uv \in E \text{ for some } v \in V_2\} \subset V_1.$$

Where $\Gamma(S)$ is the image and $\gamma(S)$ is the pre-image.

Theorem:

According to Philip Hall [5]. A bipartite graph G with the vertex sets V_1 and V_2 where $| V_1 | \leq | V_2 |$ contains a complete matching if and only if the following Hall's condition holds:

$$| \Gamma(S) | \geq |S| \text{ for every } S \subset V_1.$$

Proof:

First, we prove that Hall's condition implies the existence of a complete matching.

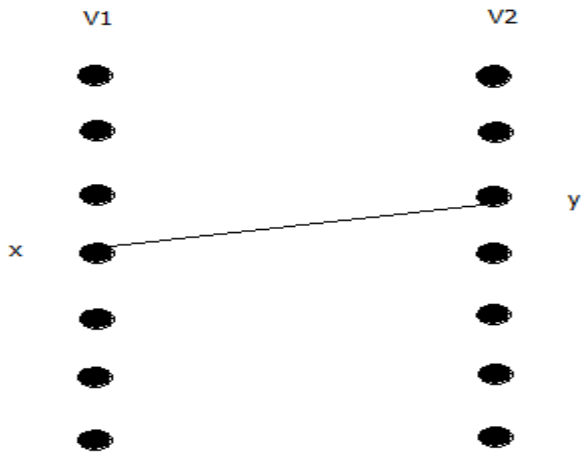

Figure 4.2 To demonstrate Hall's Theorem

Consider an arbitrary vertex $x \in V_1$. Connect the vertex x with its neighbor $y \in V_2$.

Consider the remaining graph with x and y removed which has $G' = V_1 \setminus x$ and V_2 / y and $E - (x, y)$ and $V' = (V_1 / x, V_2 / y)$.

We consider the following two cases.

Case 1:

Hall's theorem holds for this graph G' and therefore recursively get matching for V' and Add(X, Y).

Case 2:

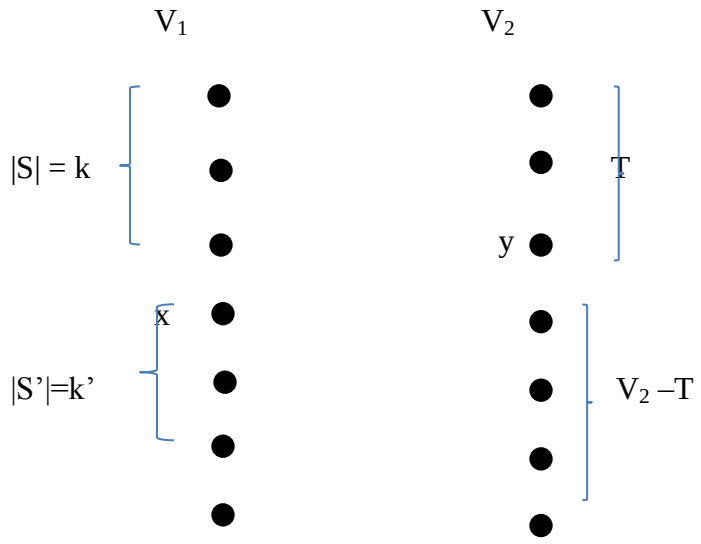

Figure 4.3 To demonstrate Hall's Theorem

Assume that the Hall's condition does not hold for this graph G' but for original graph it did hold.Then ∃ S with | S | = k such that Γ(S) < k.

We divide the sets V_1 and V_2 into two sets and vertex y is included in one of the sets of V_2.

From this follows that Γ(S) = T and | T | = k.

Therefore | S | = |T | = k.

Hall's condition holds for S and T and therefore matching exists. We look at remaining vertices and show that hall's condition holds for $\{V_1 - S, V_2 – T\}$.

Let S' be any subset in the set $V_1 – S$ and | S' | = k'.

We know that | Γ(S ∪ S') | ≥ k + k'.

Γ(S ∪ S') = Γ(S) ∪ Γ(S')

$= T \cup \Gamma(S')$

$= T \uplus \Gamma(S'-T)$

By substituting $|\ \Gamma(S \cup S')| = k + k'$ we get the following:

$K + k' \leq |\ \Gamma(S \cup S')| = |\ T\ | + |\ \Gamma(S'-T)|$

$= k + |\ \Gamma(S'-T)|$

The k on both LHS and RHS get cancelled and the result is given as the following:

$$|\ S'\ | \leq |\ \Gamma(S'-T)|$$

Therefore Hall's condition holds and matching exits.

Finally, for the other direction we observe that a complete matching does not exist if the above condition is not true for any subset of V_1.

Corollary:

A bipartite graph G where $|\ V_2\ | \leq |\ V_1\ |$ contains a complete matching if and only if $|\ \gamma(S)| \geq |S|$ where $S \subset V_2$ by Symmetry .

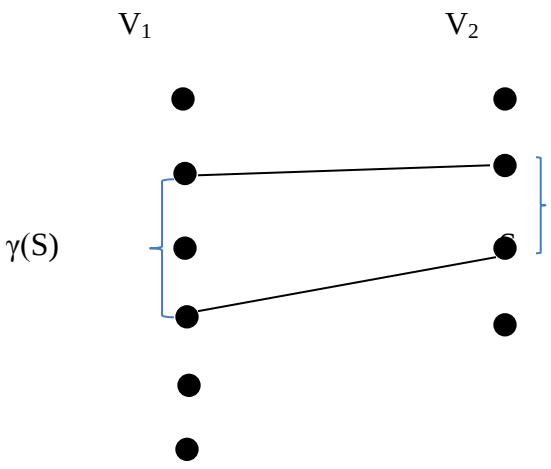

Figure 4.4 To demonstrate Hall's Theorem

4.2 Theorem:

Hall's Theorem can be applied to a bipartite graph G where each vertex has degree equal to D and that graph can be colored using exactly D colors.

Proof:

Consider a bipartite graph G whose vertices have a degree equal to D. The hall condition holds for this graph and therefore matching exists. The graph is colored using D colors by performing matching recursively.

Let us now show this theorem using the following example:

Consider the fully bipartite graph where each vertex is of degree equal to 3.Clearly Hall's property holds for the graph and therefore matching exists.

The coloring of a graph is an assignment of colors to edges such that no two edges have the same color.

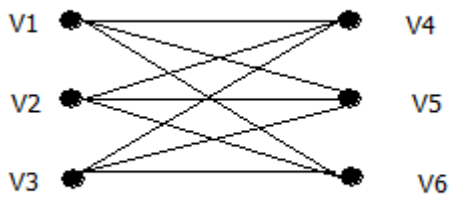

Figure 4.5 To demonstrate D coloring of a Graph

In our first matching, we color the edges in the matching using yellow color.

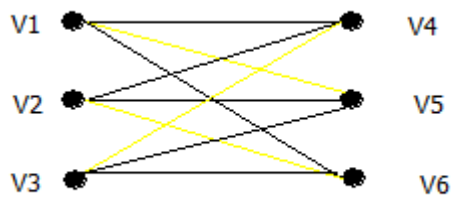

Figure 4.6 To demonstrate D coloring of a Graph

Now, we take out this matching and each vertex in the graph has a degree equal to 2.

The resultant graph is given as:

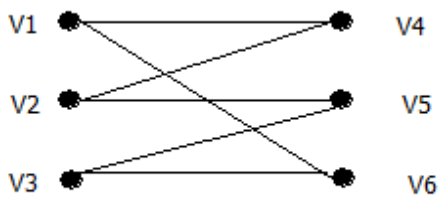

Figure 4.7 To demonstrate D coloring of a Graph

Now, we perform the second matching with the green color.

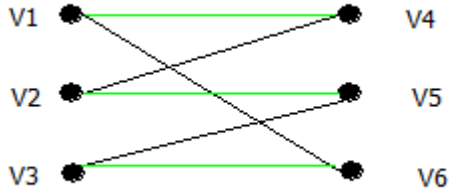

Figure 4.8 To demonstrate D coloring of a Graph

Now, we take out this matching. Each vertex in the graph has a degree equal to 1.

The resultant graph is given as:

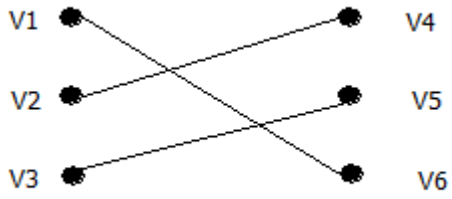

Figure 4.9 To demonstrate D coloring of a Graph

Now, we perform the third matching with the red color.

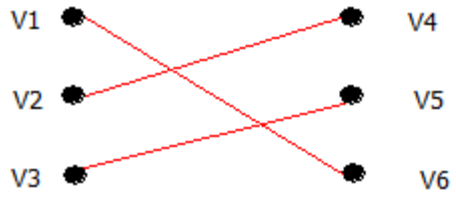

Figure 4.10 To demonstrate D coloring of a Graph

The bipartite graph G having a degree equal to 3 for every vertex is colored with three colors and the resultant graph is given as follows:

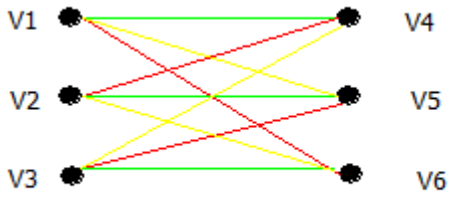

Figure 4.11 To demonstrate D coloring of a Graph

4.3 D-coloring of multi bipartite graph:

Consider a multi bipartite graph G whose vertices have a degree equal to d. The graph can be colored using exactly d colors.

Proof:

Let F be all the edges in the multi bipartite graph which leave S.

Let $||F||$ be the number of edges and $||H||$ be all the edges for $\Gamma(S)$.

we know that $||H|| \geq ||F||$

and $||F|| = d \cdot |S|$, $||H|| = d \cdot |\Gamma(S)|$.

By substituting $||F|| = d \cdot |S|$, $||H|| = d \cdot |\Gamma(S)|$ in the equation $||H|| \geq ||F||$ we get:

$$d \cdot |\Gamma(S)| \geq d \cdot |\Gamma(S)|$$

We know that if $|\Gamma(S)| \geq |\Gamma(S)|$ holds for any subset S of V_1 in the graph, matching exists.

Therefore, a multi bipartite graph G can be colored using exactly d colors.

Let us now apply Hall's theorem on the following open shop problem:

4.4 O | $P_{i,j}$ = 0, 1 |C_{max}:

This is an open shop problem where the processing time of every job on every machine is either 0 or 1 and the objective function of minimizing the makespan.

The open shop scheduling problem with processing times 0 or 1 is called unit processing time open shop problem.

	M_1	**M_2**	**M_3**	**M_4**
J_1	1	1	1	
J_2		1	1	
J_3				1

Table 4.1 To Demonstrate O | $P_{i,j}$ = 0, 1 |C_{max} using Hall's Theorem

The lower bound for C_{max} is defined below:

$$L = \max \{ \max_{i=1}^{n} L_i, \ \max_{j=1}^{n} T_j \}$$

Where T_j is the total time needed on machine M_j and L_i is the length of job i.

Clearly. the lower bound for the above problem is 3. We obtain the following matrix:

In the matrix, M_5 , M_6 , M_7 are dummy machines and J_4, J_5, J_6, J_7 are dummy Jobs.

	M_1	M_2	M_3	M_4	M_5	M_6	M_7
J_1	1	1	1				
J_2		1	1		1		
J_3				1		1	1
J_4	1			1	1		
J_5	1			1		1	
J_6		1			1		1
J_7			1			1	1

Table 4.2 To Demonstrate O | $P_{i,j}$ = 0, 1 |C_{max} using Hall's Theorem

The bipartite graph for the above matrix is given below:

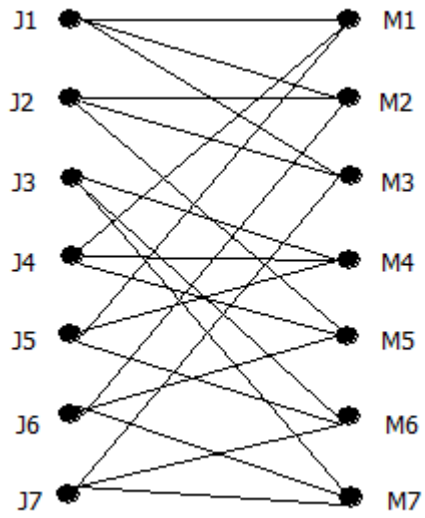

Figure 4.12 To Demonstrate O | $P_{i,j}$ = 0, 1 |C_{max} using Hall's Theorem

Clearly Hall's property holds for the above graph and therefore matching exists.

In our first matching, we color the edges in the matching using blue color.

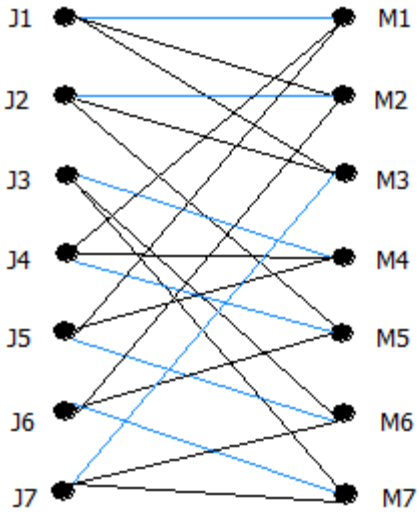

Figure 4.13 To Demonstrate O | $P_{i,j}$ = 0, 1 |C_{max} using Hall's Theorem

Now, we take out this matching and each vertex in the graph has a degree equal to 2.

The resultant graph is given as:

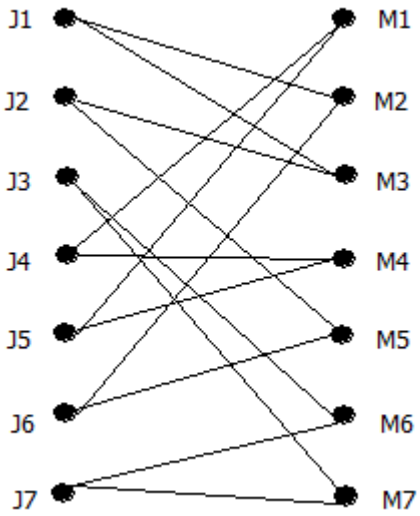

Figure 4.13 To Demonstrate O | $P_{i,j}$ = 0, 1 |C_{max} using Hall's Theorem

The Hall's property holds for the above graph and therefore matching exists.

Now, we perform the second matching with the violet color.

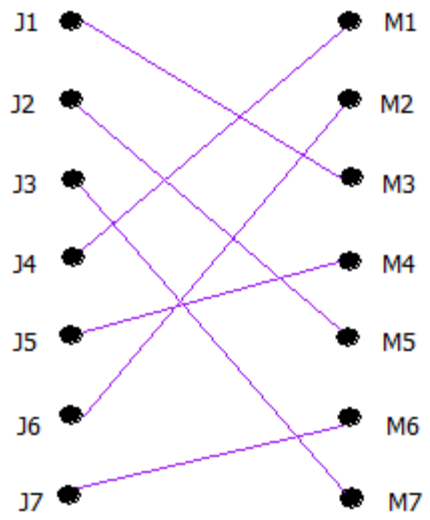

Figure 4.14 To Demonstrate O | $P_{i,j}$ = 0, 1 |C_{max} using Hall's Theorem

Now, we take out this matching. Each vertex in the graph has a degree equal to 1. The resultant graph is given as:

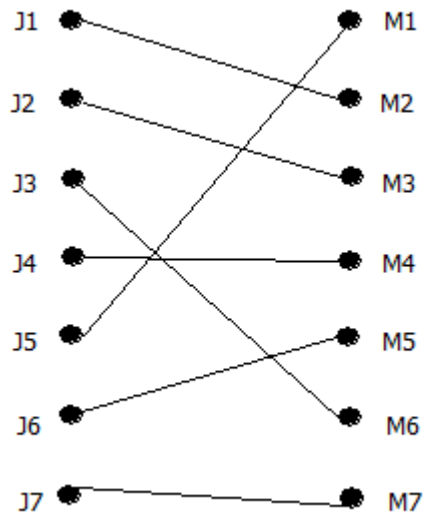

Figure 4.15 To Demonstrate O | $P_{i,j}$ = 0, 1 |C_{max} using Hall's Theorem

The Hall's property holds for the above graph and therefore matching exists.

Now, we perform the second matching with the violet color.

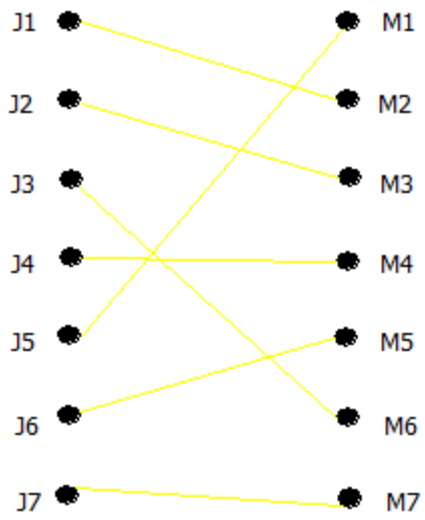

Figure 4.16 To Demonstrate O | $P_{i,j}$ = 0, 1 |C_{max} using Hall's Theorem

The bipartite graph G having a degree equal to 3 for every vertex is colored with three colors and the resultant graph is given as follows:

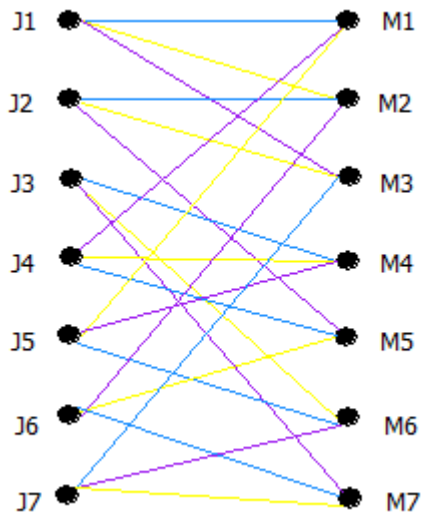

Figure 4.17 To Demonstrate O | $P_{i,j}$ = 0, 1 |C_{max} using Hall's Theorem

Here, the colors represent different time unit.

The Gantt chart is given below:

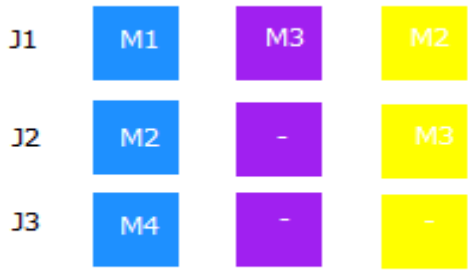

Figure 4.18 To Demonstrate O | $P_{i,j}$ = 0, 1 |C_{max} using Hall's Theorem

At least one arc is eliminated after each step. Let r be the operations with processing time $p_{ij} > 0$, then we have at most O (r) steps. A matching can be calculated in O (r $(n+m)^{0.5}$) steps. Thus the total complexity is O (r^2 $(n+m)^{0.5}$). The complexity can be reduced to O (r^2) due to the fact that in each step the matching from the previous step may be used to calculate the new matching.

4.5 Preemption of Jobs:

Preemption of a job is also called as job splitting. It is allowed on both job and their operations. Preemption is interrupting the processing of a job or an operation and resuming at a later time. Now we see the preemptive version of the make span open shop problem.

O | pmtn | C_{max} :

	M_1	M_2
J_1		2
J_2	1	
J_3	2	1

Table 4.3 To Demonstrate O | pmtn | C_{max} using Hall's Theorem

Clearly, the lower bound for the above problem is 3.

The matrix is given as below:

	M_1	M_2	M_3	M_4	M_5
J_1		2	1		
J_2	1			2	
J_3	2	1			
J_4			2	1	
J_5					3

Table 4.4 To Demonstrate O | pmtn | C_{max} using Hall's Theorem

A multi bipartite graph is drawn below for the above matrix.

The jobs are preempted on the machines so that every node in the graph has a degree equal to the lower bound of the given problem.

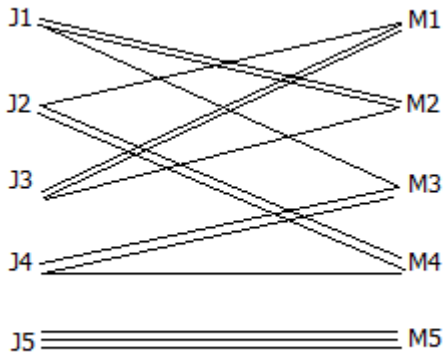

Figure 4.19 To Demonstrate O | pmtn | C_{max} using Hall's Theorem

According to the theorem 4.4, the multi graph can be colored using exactly 3 colors. The first Matching with yellow color is given below.

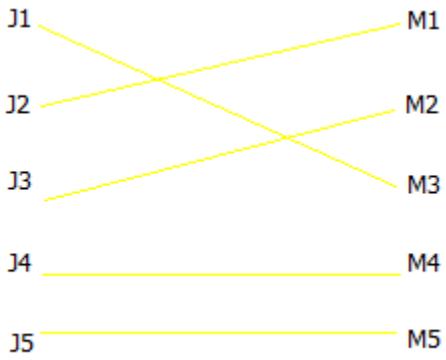

Figure 4.20 To Demonstrate O | pmtn | C_{max} using Hall's Theorem

After removing this matching, the resultant graph is given below. Every node in the graph has a degree equal to 2.

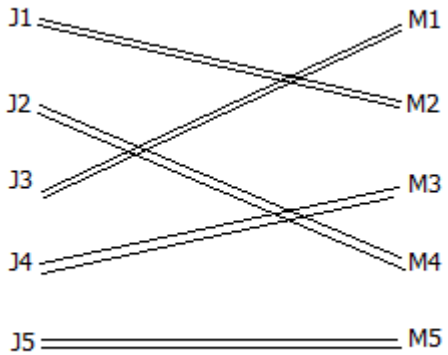

Figure 4.21 To Demonstrate O | pmtn | C_{max} using Hall's Theorem

The second matching with violet color is given below:

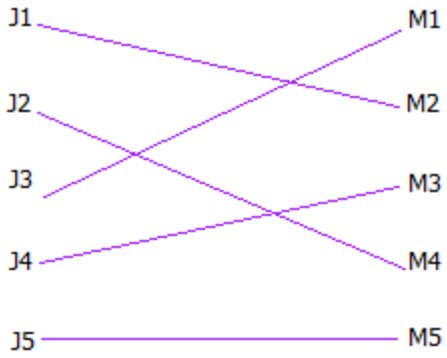

Figure 4.22 To Demonstrate O | pmtn | C_{max} using Hall's Theorem

After removing this matching, the resultant graph is given below. Every node in the graph has a degree equal to 1.

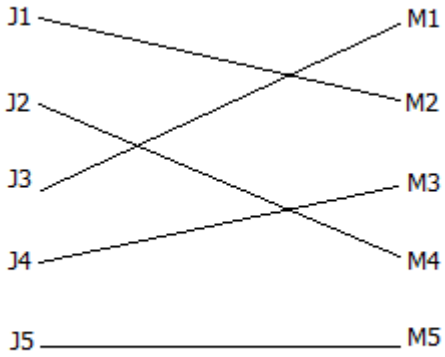

Figure 4.23 To Demonstrate O | pmtn | C_{max} using Hall's Theorem

The third matching with green color is given below:

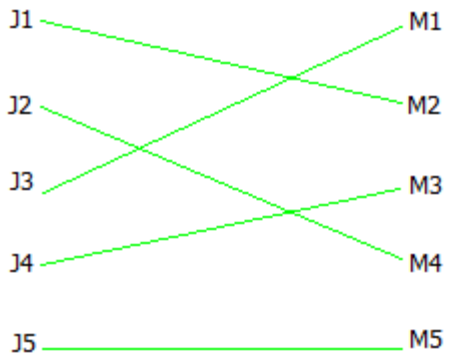

Figure 4.24 To Demonstrate O | pmtn | C_{max} using Hall's Theorem

The resultant graph with all the three matching is given below:

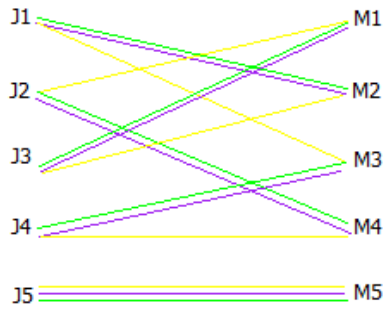

Figure 4.25 To Demonstrate O | pmtn | C_{max} using Hall's Theorem

The Gantt chart for the above problem is given below:

Figure 4.26 To Demonstrate O | pmtn | C_{max} using Hall's Theorem

The processing time of the above problem is given by O (r^2).

Let us consider another problem:

	M_1	M_2	M_3
J_1	1	1	1
J_2		1	1

4.5 Table To Demonstrate Preemption using Hall's Theorem

The lower bound of the above problem is 3. The matrix is given below:

	M_1	M_2	M_3	M_4	M_5
J_1	1	1	1		
J_2		1	1		1
J_3	2			1	
J_4		1		2	
J_5			1		2

4.6 Table To Demonstrate Preemption using Hall's Theorem

The bipartite multi graph is given as follows:

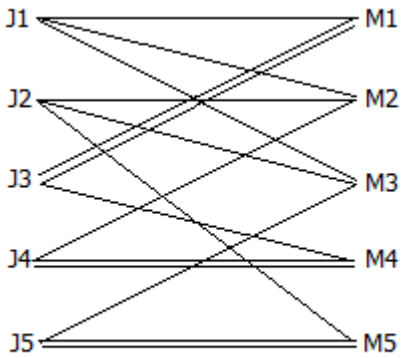

Figure 4.27 To Demonstrate Preemption using Hall's Theorem

The first matching is given below with blue color:

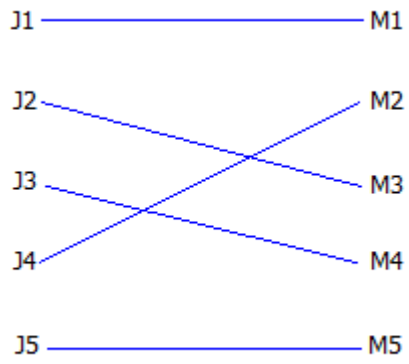

Figure 4.28 To Demonstrate Preemption using Hall's Theorem

Now, we remove the first matching and the resultant graph is given below:

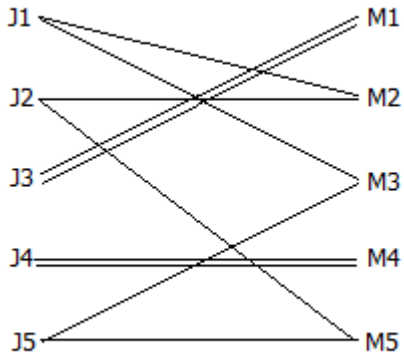

Figure 4.29 To Demonstrate Preemption using Hall's Theorem

The second matching is given with red color as below:

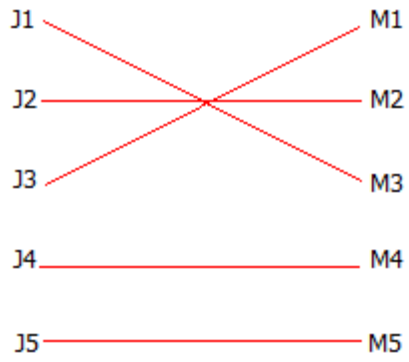

Figure 4.30 To Demonstrate Preemption using Hall's Theorem

Now, we remove the second matching and the resultant graph is given below:

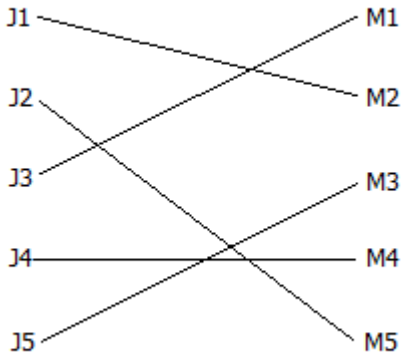

Figure 4.31 To Demonstrate Preemption using Hall's Theorem

The third matching is given below with green color:

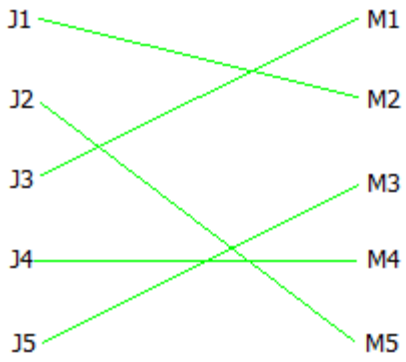

Figure 4.32 To Demonstrate Preemption using Hall's Theorem

The multi bipartite graph with three coloring is given below:

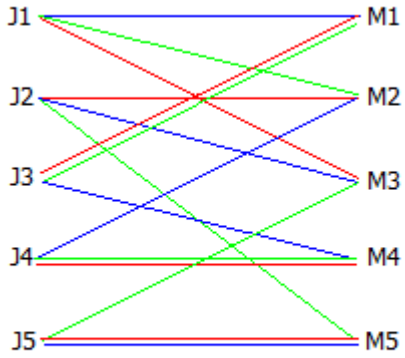

Figure 4.33 To Demonstrate Preemption using Hall's Theorem

The Gantt chart for the multi bipartite graph is given below:

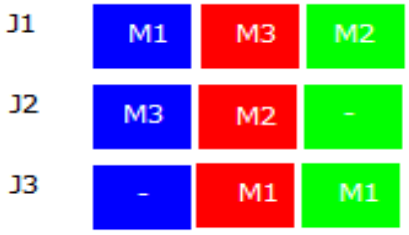

Figure 4.34 To Demonstrate Preemption using Hall's Theorem

CHAPTER 5

Maximum Flow Problem

According to Ford-Fulkerson [31], the max flow problem for single source and single sink is given as follows:

A directed capacitated network (V, E, C) connecting a source and a sink t, where V is the set of vertices in the network, E is the directed edges (i, j) and C is the set of capacities $C_{ij} > 0$ of the edges (i, j). The problem is to determine the maximum flow that can be sent from the source to the sink.

Mathematical Formulation:

- For each edge (i, j) ϵ E, let x_{ij} be the flow that is sent on the edge (i, j).
- For each edge (i, j) ϵ E, the flow is bounded above by the capacity C_{ij} of the edge and is given as below:

$$C_{ij} \geq x_{ij} \geq 0$$

- All the vertices except the source and the sink are transit vertices, i.e. the inflow = the outflow.

$$\sum_{\{k|(k,i)\in E\}} x_{ki} - \sum_{\{j|(i,j)\in E\}} x_{ij} = 0 \quad \text{for all i} \neq \text{s, t}$$

- The objective here is to maximize the flow:

$$\sum_{\{j|(s,j)\in E\}} x_{sj}$$

Max Flow Algorithm:

The algorithm is an iterative method.

- At each iteration, the algorithm searches for a path from source to sink along which it can send a flow. This path is referred as augmenting path.

- Along this augmenting path, the capacities of the edges on this path are adjusted after the flow has been sent.
- The algorithm terminates when an augmenting path is not found.

The capacity of a path is the maximum possible amount we can send along that path.

Let us consider the following example [32]:

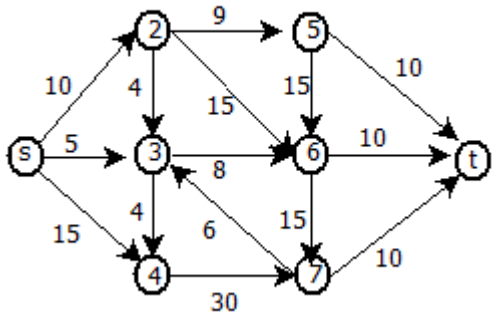

Figure 5.1 To Demonstrate Maximum Flow Problem

We have the maximum flow of 28 for the above graph and the flow is shown below:

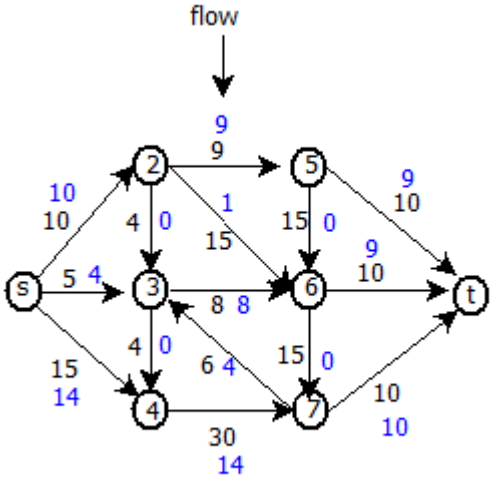

Figure 5.2 To Demonstrate Maximum Flow Problem

A minimum cut is partition (A, B) such that A ϵ source s and B ϵ sink t and the capacity of (A, B) is the sum of the weights of the edges leaving s.

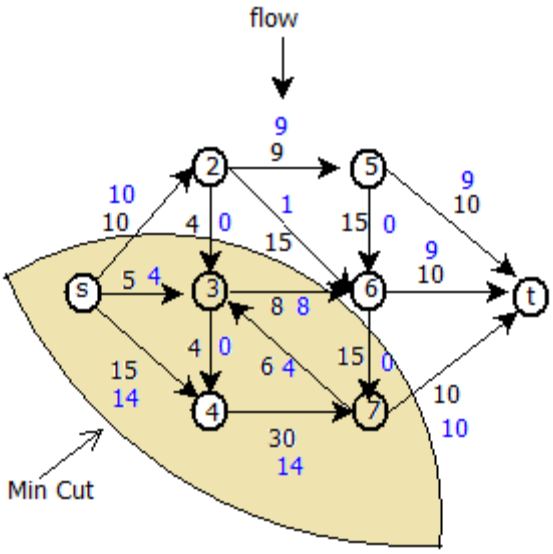

Figure 5.3 To Demonstrate Maximum Flow Problem

In any network, the maximum flow equals the minimum cut.

The time complexity of the algorithm is O (m·max|f|), where m is the number of edges and max|f| is maximum amount of flow.

CHAPTER 6

Reduction from Parallel Machines to Open Shop

6.1 P| pmtn | C_{max}:

Given are five jobs J_1, J_2, J_3, J_4, J_5 and three machines.

According to Peter Brucker[1] when machines are parallel, each operation of a job can be processed on any of the machines.

The lower bound for the above problem is given as follows:

$$\max \{ \max_i p_i , (\sum_{i=1}^{n} p_i) / m \}$$

where m is the number of machines.

A schedule that can meet this lower bound can be achieved in O (n) time. Now, we fill the machines successively. The jobs can be scheduled in any order and the jobs can be split whenever the lower bound is met. If the job is split, then we have to schedule the second part of the job on second machine at zero time.

The machine oriented Gantt chart is given as follows:

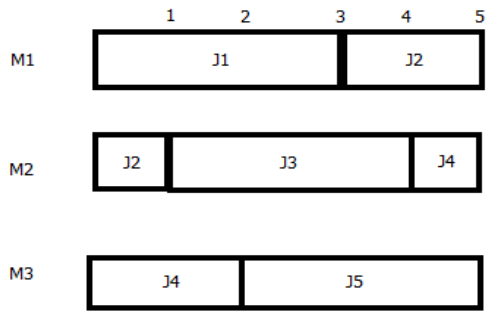

Figure 6.1 To Demonstrate Parallel Machines Problem

The job oriented Gantt chart is given as follows:

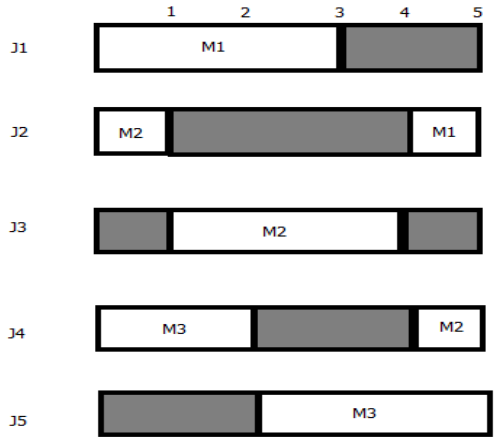

Figure 6.2 To Demonstrate Parallel Machines Problem

Let V_1 be the set consisting of jobs and V_2 be the set consisting of the time slots from 1 to 5.

The following is the bipartite graph of the sets V_1 and V_2:

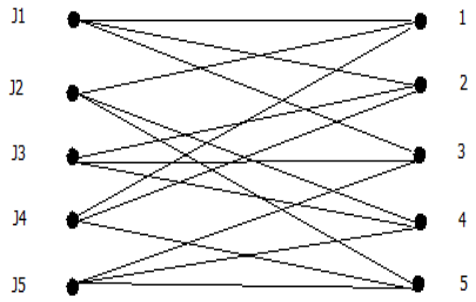

Figure 6.3 To Demonstrate Reduction from Parallel Machines to Open Shop

The bipartite graph satisfies the hall property and therefore by applying hall's theorem matching can be obtained.

The first matching is obtained with blue color and is given as follows:

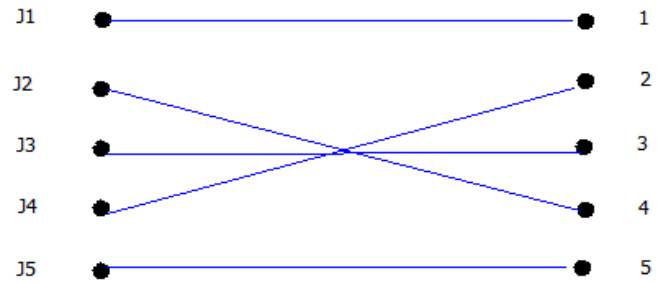

Figure 6.4 To Demonstrate Reduction from Parallel Machines to Open Shop

The graph after taking out the first matching (blue matching) is shown below:

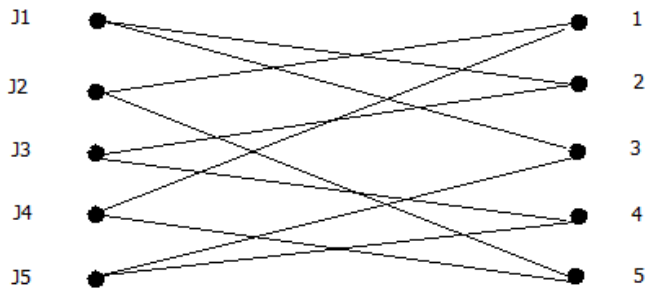

Figure 6.5 To Demonstrate Reduction from Parallel Machines to Open Shop

Each node in the graph has a degree equal to 2. The graph satisfies the hall property and therefore the second matching can be obtained with red color.

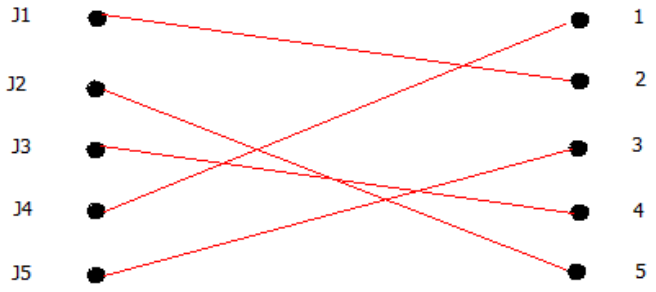

Figure 6.5 To Demonstrate Reduction from Parallel Machines to Open Shop

The resultant graph after taking out the second matching is shown below:

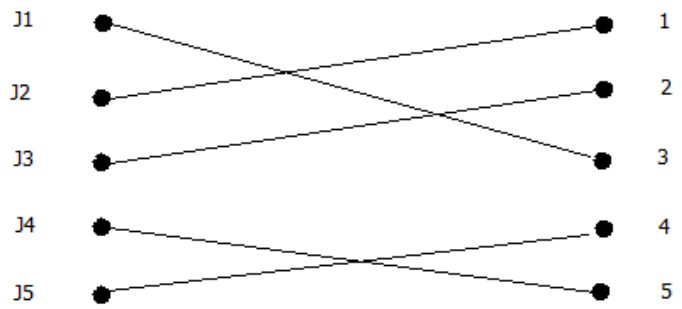

Figure 6.6 To Demonstrate Reduction from Parallel Machines to Open Shop

The above graph satisfies hall property. Each node has a degree equal to 1. Therefore the third matching can be obtained and is shown below:

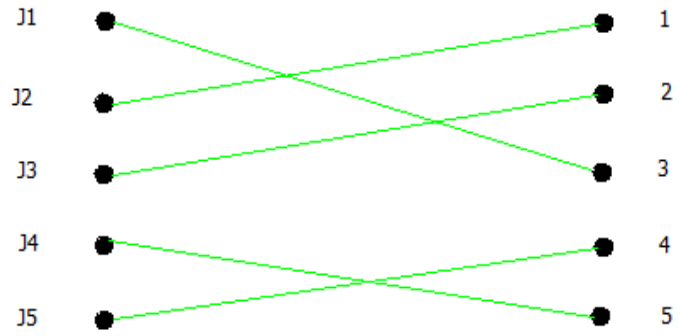

Figure 6.7 To Demonstrate Reduction from Parallel Machines to Open Shop

The resultant graph with all the three matching is given below:

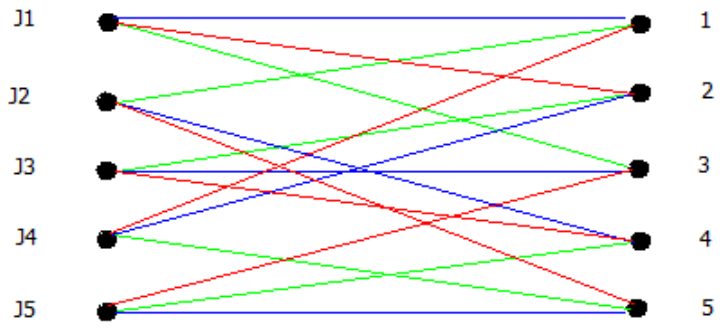

Figure 6.8 To Demonstrate Reduction from Parallel Machines to Open Shop

The Gantt chart for the above example is given below:

The parallel machines problem is reduced to open shop problem using Hall's theorem.

Here, the colors represent the three machines M_1, M_2, M_3. The coloring satisfies the property of scheduling that no job is being processed on more than one machine at any point of time and also a machine does not process more than one job at any given time.

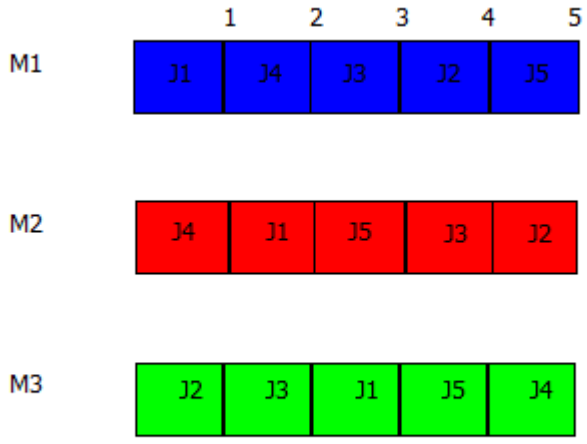

Figure 6.9 To Demonstrate Reduction from Parallel Machines to Open Shop

The processing time of above problem is O (r^2).

6.2 P| pmtn;r_i | L_{max}:

According to Peter Brucker[1], each job is associated with a release time r_i and a deadline d_i where $r_i \leq d_i$. Now, we have to find a preemptive schedule with m identical machines where the maximum lateness is minimized. We first consider that for some given L value, does a schedule exist with $\max_1^n L_i \leq L$?

Also, we want to find out that such a schedule exists if and only if it holds $C_i \leq d_i^L =$ $L + d_i$ for all i = 1. . . n.

Here d_i^L are the modified due dates.

Therefore, all the jobs i must finish processing before the modified due date d_i^L and cannot be Processed before release time r_i. We now know that a job i must be processed in the time interval [r_i, d_i^L]. These are called time windows. We can find a preemptive schedule such that a job i is processed in the time interval [r_i, d_i^L] by reducing to a maximum flow problem in a network. The network is constructed as follows:

Let the release times and deadlines be ordered in the sequence $t_1 \leq t_2 \leq t_3 \leq t_4 \leq t_5 \ldots \leq t_n$. Now, consider the intervals $I_a = [t_a, t_{a+1}]$ and T_a is the length of the interval which is given by $t_{a+1} - t_a$ for a = 1..r-1.A job vertex is associated with every job i and an interval vertex is associated with every Interval. We introduce two vertices s and t and these are called the source and the sink. We now have an arc to each job vertex i form source s and the capacity is p_i and an arc from every interval vertex to sink t with a capacity of mT_a. If a job J_i can be processed in I_a i.e. if $r_i \leq t_a$ and $d_i \leq t_{a+1}$,

then there exists an arc from J_i to I_a. The capacity of this arc is T_a. The network N = (V, A, c) is given as following:

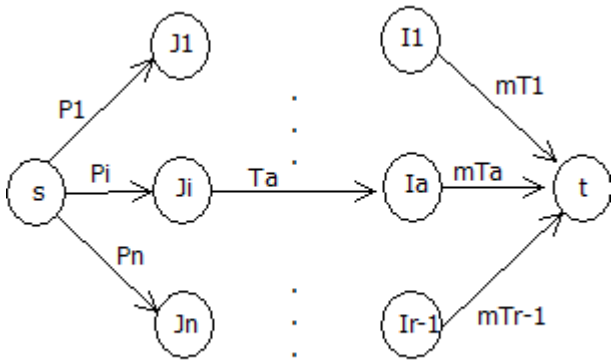

Figure 6.10 To Demonstrate P| pmtn;r_i | L_{max}

The maximum flow of the network is $\sum_{i=1}^{n} p_i$ if there exists a schedule respecting all time windows. Here, the flow x_{ia} on the arc (J_i, I_a) corresponds with the time period in which Job i is processed in the time interval I_a and we have the following:

$$\sum_{a=1}^{r-1} x_{ia} = p_i \text{ for } i = 1,2 \ldots n$$

$$\sum_{i=1}^{n} x_a \leq mT_a \text{ for } a = 1,2 \ldots n$$

Now, each job is completely processed and the total amount of processing time in I_a is at the most mT_a, which is the capacity of m machines.

When all the deadlines are equal to zero then the same is applied for P | pmtn | C_{max}.

The network N can have at most O (n) vertices. The schedule respecting the windows can be constructed in O (n^2) and therefore the windows problem can be solved in O (n^3) time.

P| pmtn;r_j | C_{max}:

Given five jobs and three machines.

The processing times, release times and deadlines are given as follows:

$P_1 = 3, P_2 = 3, P_3 = 3, P_4 = 3, P_5 = 3.$

$r_1 = 0, r_2 = 1, r_3 = 0, r_4 = 2, r_5 = 0.$

$d_1 = 0, d_2 = 0, d_3 = 0, d_4 = 0, d_5 = 0.$

Threshold value L = 5.

Relaxed deadlines: $d_i^L = L + d_i$ for all i=0, 1…n.

$d_1^5 = 5 + 0 = 5,\ d_2^5 = 5 + 0 = 5,\ d_3^5 = 5 + 0 = 5,\ d_4^5 = 5 + 0 = 5,\ d_5^5 = 5 + 0 = 5.$

The ordered sequence of release times and deadlines is given below:

$$0 < 1 < 2 < 5$$

$$t_1 < t_2 < t_3 < t_4$$

The intervals are given below:

$I_1 = [0, 1], T_1 = 1$

$I_2 = [1, 2], T_1 = 1$

$I_3 = [2, 5], T_1 = 3$

The time window diagram:

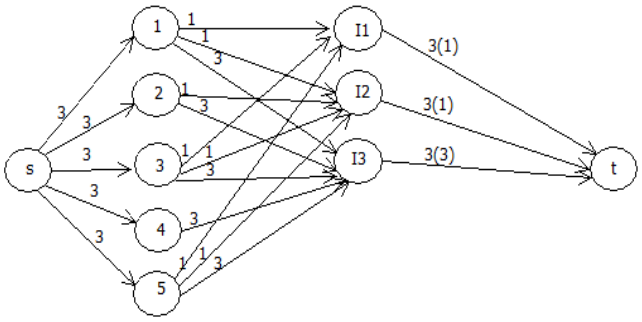

Figure 6.11 To Demonstrate Time Window Diagram

Yes, there exists a feasible schedule for L =5.

The Gantt chart for the problem:

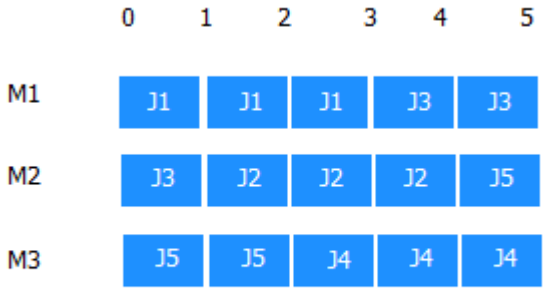

Figure 6.12 To Demonstrate P| pmtn;r_i | L_{max}

The bipartite graph for the above Gantt chart is given below:

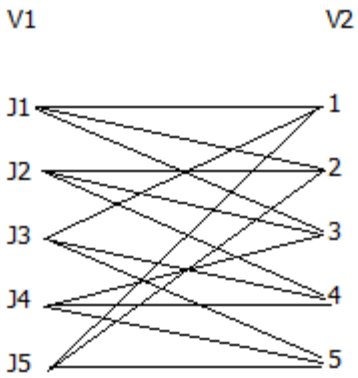

Figure 6.13 To Demonstrate P| pmtn;r_i | L_{max}

The bipartite graph satisfies the hall's property and therefore matching exists.

The first matching is given with blue color:

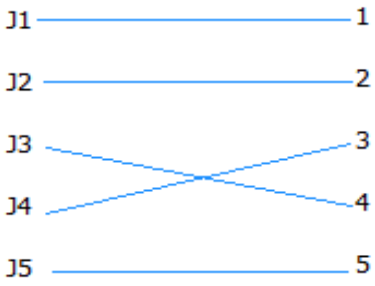

Figure 6.14 To Demonstrate P| pmtn;r_i | L_{max}

The graph without the first matching is given below:

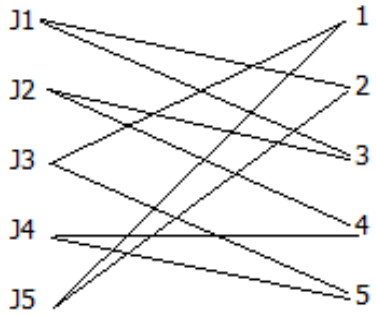

Figure 6.15 To Demonstrate P| pmtn;r_i | L_{max}

The above graph satisfies the hall's property and therefore matching exists. The second matching i.e. Red matching is given below:

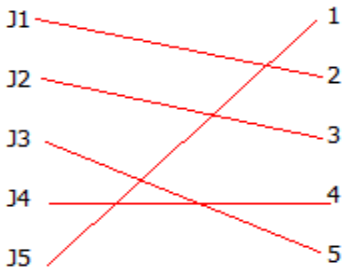

Figure 6.16 To Demonstrate P| pmtn;r_i | L_{max}

The graph without second matching is given below:

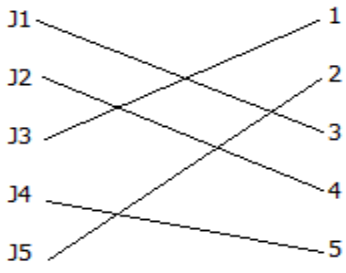

Figure 6.17 To Demonstrate P| pmtn;r_i | L_{max}

The graph satisfies Hall's property and therefore matching exists. The third matching i.e. Green matching is given below:

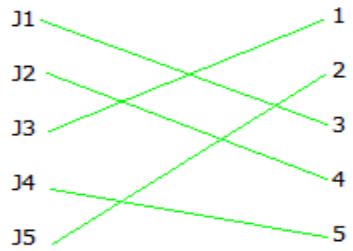

Figure 6.18 To Demonstrate P| pmtn;r_i | L_{max}

The bipartite graph with all the three matching is given below:

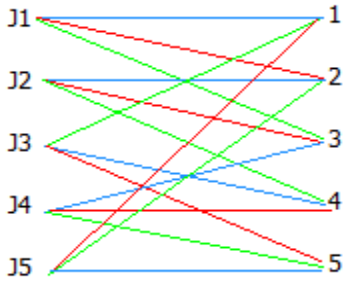

Figure 6.19 To Demonstrate P| pmtn;r_i | L_{max}

The three colors blue, red, green represent the three machines. Now, we reduce the problem to open shop problem. The machine oriented Gantt chart for the open shop problem is given below:

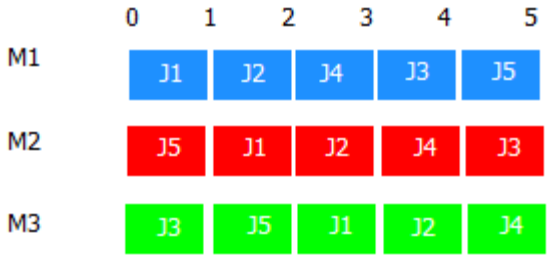

Figure 6.20 To Demonstrate P| pmtn;r_i | L_{max}

The processing time of the above problem is O (n^3).

CHAPTER 7

Hard Open Shop Problems

7.1 $O_3 \mid\mid C_{max}$:

According to Gonzalez and Sahni[19], to prove that the minimum makespan open shop nonpreemptive scheduling problem is NP-hard, we reduce the partition problem to three machine problems.

Given are n integers from 1 to n. Now, the partition problem is to determine whether or not the Integers {1...n} can be partitioned into two sets A_1 and A_2 , in such a way that each set is equal to T⁄2, where T = $\sum_{i=1}^{n} a_i$ (sum of all the integers).

We consider the reduction of three machine non preemptive open shop scheduling problem with 3n + 1 jobs and three machines.

The 3n set of jobs are introduced as follows:

Let jobs $J_1, J_2, J_3, J_4 \ldots J_n$ be the set of jobs on machine M_1 and $k_1, k_2, k_3, k_4 \ldots k_n$ be the set of jobs on machine M_2 and $l_1, l_2, l_3, l_4 \ldots l_n$ be the set of jobs on machine M_3.

J	M_1	K	M_2	L	M_3
J_1	a_1	k_1	a_1	l_1	a_1
J_2	a_2	k_2	a_2	l_2	a_2
J_3	a_3	k_3	a_3	l_3	a_3
.	.	.	.	.	.
J_n	a_n	k_n	a_n	l_n	a_n

Figure 7.1 To Demonstrate $O_3 \mid\mid C_{max}$

Here, a_i is the processing time of each job on each machine.

All the jobs in the set J have nonzero tasks on machine M_1, the jobs in set k have nonzero tasks on machine M_2 and jobs in the set l have nonzero tasks on machine M_3.

All the jobs in the above instance have one nonzero task except for the last job that has nonzero tasks on all the machines.

Let U_{3n+1} be the last job and U_{3n+1} has a processing time of $T/2$ on each machine. Therefore, the job U_{3n+1} has to process without interruptions if there exists a schedule with a makespan of $3T/2$.

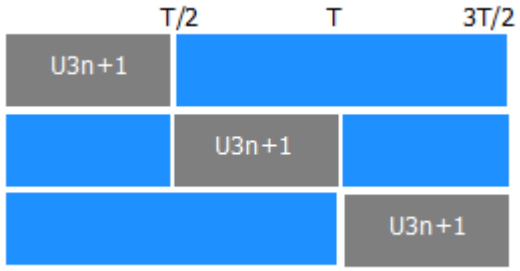

Figure 7.2 To Demonstrate $O_3||C_{max}$

We know that this is a nonpreemptive schedule, therefore the job U_{3n+1} has to be processed from time $T/2$ to time T on one of the machines. From the above Gantt chart we know that there are two disjoint blocks of idle time each having a length of $T/2$ on one machine. Now, these blocks will be used for processing set of n jobs whose processing time corresponds to those sizes in the instance of partition.

The jobs in the set J have to be processed on machine M_1 as shown in the following:

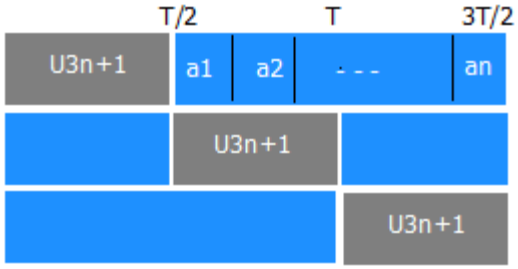

Figure 7.3 To Demonstrate $O_3||C_{max}$

Now, the jobs in set l have to be processed on machine M_3.

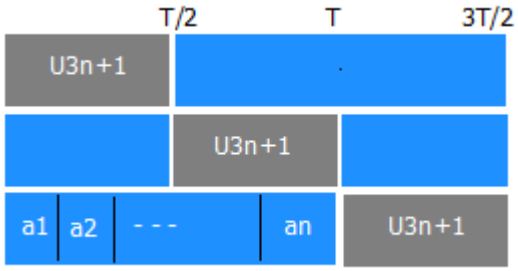

Figure 7.4 To Demonstrate $O_3||C_{max}$

The jobs in the set k are partitioned in such a way that the sum of the processing time of jobs in each set is equal to T/2.

If there exists no partition, then the makespan of 3T/2 cannot be achieved. It is because one of the set among the two sets of jobs of the set k has sum of the processing times less than the other set. This results in a schedule where the makespan extends 3T/2. This is shown below:

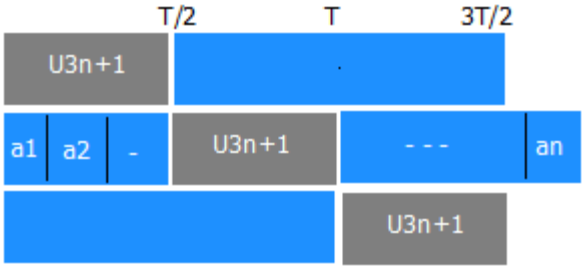

Figure 7.5 To Demonstrate $O_3||C_{max}$

If partition exists, then the makespan remains 3T/2. It is shown below:

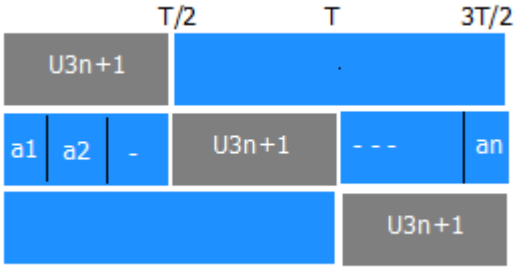

Figure 7.6 To Demonstrate $O_3||C_{max}$

Since partition is required for the makespan to be 3T/2, therefore, the open shop nonpreemptive scheduling problem is NP-hard.

7.2 Disjunctive Graph Model:

According to Peter Brucker[1], a disjunctive graph is a graph containing disjunctive edges. A disjunctive edge connects any two any two nodes from the set V.

Disjunctive Graphs are used to represent certain feasible solutions for Shop problems. The Disjunctive Graph G = (V, C, D) can be defined as follows:

V:

It is the set of nodes representing the operations of all the jobs. There are two special nodes known as source 0 and sink *, both belonging to the set V. Here, each

node is associated with a weight. The source and the sink nodes have a weight of zero, where as other nodes have their processing times as their weights.

C:

It is the set of directed Conjunctive Arcs. These arcs represent the precedence relation that will exist between the operations of the job. The conjunctive arcs also exist between the source and all the operations without predecessor and also between sink and all the operations without successor.

D:

It is set of undirected disjunctive arcs. This disjunctive arc exists between the operations of the Same job which are not connected by conjunctive arcs and also between the pair of operations that are to be processed on the same machine which are not connected by the conjunctive arcs.

Consider the following Open Shop Problem:

Given three jobs J_1, J_2, J_3 and three machines M_1, M_2, M_3.

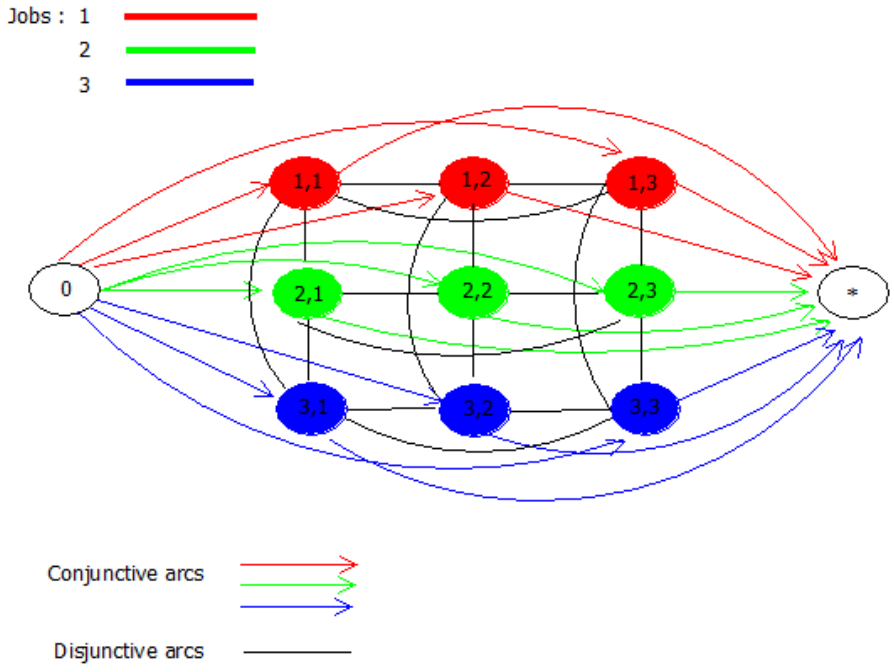

Figure 7.7 To Demonstrate Disjunctive Graph

From the above graph, we know that the operations of jobs 1, 2 and 3 can be processed in an order. Consider J_1, Either the first operation or second operation or the third operation can start processing first. Similarly, the first operation or second operation or third operation is processed at the end. This is represented by the conjunctive arcs for all the jobs. Now, the disjunctive arcs indicate that no machine can process more than one operation at a time and also no job can be processed on more than one machine at any point of time.

Consider the following problem:

	J_1	J_2	J_3	J_4	J_5
M_1	1	1	2	2	3
M_2	1	2	1	3	2

Figure 7.8 To Demonstrate Disjunctive Graph

The Disjunctive Graph for the above open shop problem is given as follows:

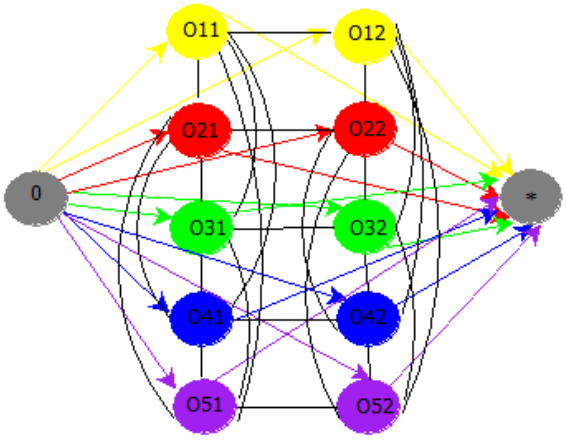

Figure 7.9 To Demonstrate Disjunctive Graph

The disjunctive graph after fixing the disjunctive arcs is given below:

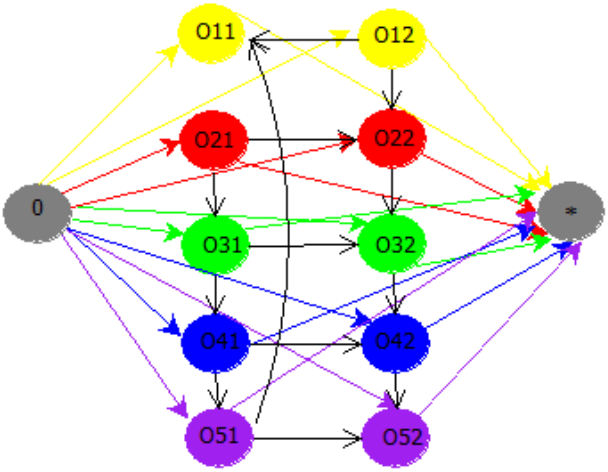

Figure 7.10 To Demonstrate Disjunctive Graph

The Gantt chart to represent the above disjunctive graph is given below:

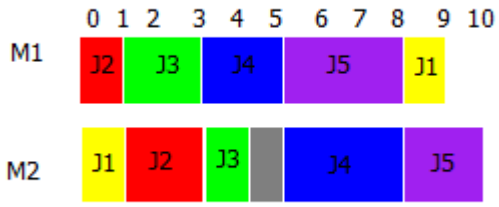

Figure 7.11 To Demonstrate Disjunctive Graph

The C_{max} for the above open shop problem is 10.

Chapter 8

Neighborhood Search

Neighborhood plays an important role in local search. An efficient neighborhood will lead to high quality local optima. The neighborhood is now defined below:

Consider the disjunctive graph of an open shop problem:

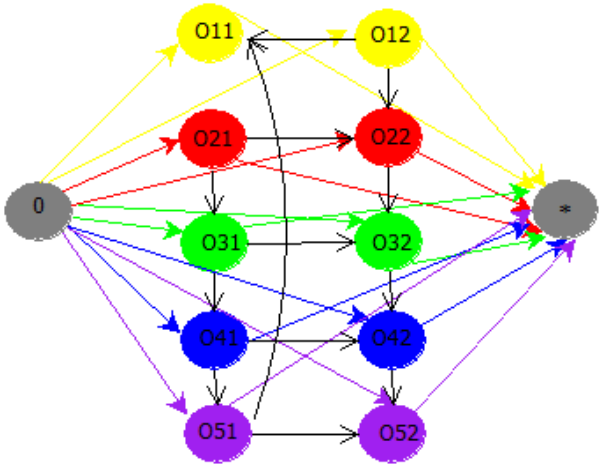

Figure 8.1 To demonstrate Neighborhood for an Open Shop Problem

Let S be a complete selection, i.e. a disjunctive graph with no cycles and also the all the disjunctive edges are fixed. An immediate neighbor of S is defined as: The set of all complete selections that are obtained from the graph S by reversing an edge (i, j) where i and j are processed on the same machine. The neighbor of S is therefore denoted by N (S).

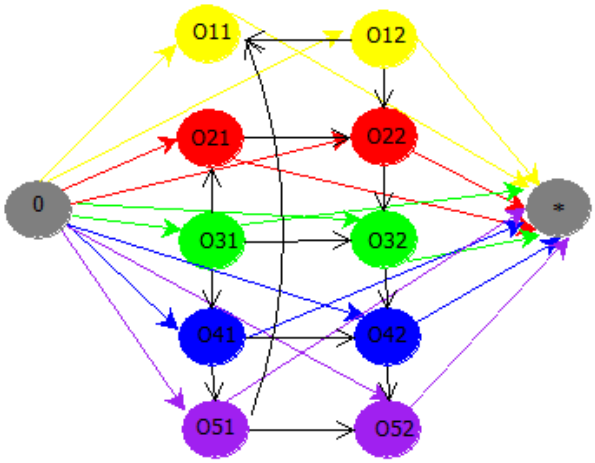

Figure 8.2 To Demonstrate the Neighbor of S

The disjunctive graph in the figure 8.2 is the neighbor of the disjunctive graph in the figure 8.1 and is obtained by reversing the edge (O_{21}, O_{31}).

Local neighborhood search, also known as Hill Climbing, searches for the immediate neighbor in order to obtain a better solution i.e. a solution with a better C_{max}. This algorithm accepts the neighborhood solutions if the solution of the neighbor is better than the current solution. The algorithm recursively performs this search untill it finds a better solution than the current solution. The algorithm terminates when the current solution is our optimal solution. We call this solution as the local optima. The disadvantage of this local neighborhood solution is that the local optima is the best always.

The disjunctive graph is used to show how local neighborhood search works.

We consider the figures 8.1 and figure 8.2 and we know that figure 8.2 is the neighbor of the figure 8.1.

The Gantt chart for the figure 8.2 is given below:

Figure 8.3 To Demonstrate the Neighbor of S

This C_{max} value is worst than the C_{max} value of the figure 8.1. Therefore we do not accept this solution and search for another better solution. This procedure is repeated untill we find the local optimum.

CHAPTER 9

Conclusion and Futurework

Open Shop Scheduling problem is different from other scheduling problems. This shop problem is very closely related to matching. Although, we have surveyed number of problems where Hall technique works, yet there are unexplored problems where matching is applied. The connection between algorithms of parallel machines and open shop are more close than other shop problems. In this thesis, we have learnt flow techinques are useful and seem to be one of the strongest tools available for non-NP Complete scheduling problem.

Bibliography

[1] Peter Brucker, *Scheduling Algorithms*, Fifth Edition, Chapter 1, 6, Springer Publications, March 2007.

[2] L.Michael Pinedo, *Scheduling Theory Algorithms and Systems*, Fourth Edition,Chapter 8, Springer Publishers, January 2012.

[3] Peter Bernus, Jacek Blazewicz and Gunter Schmidt, Michael Shaw, *Handbook of Scheduling: From Theory to Applications*, Chapter 9, Springer Publications, 2007.

[4] M.John Harris, L. Jeffry Hirst and J.Michael Mossinghoff, *Combinatorics and Graph Theory*, second Edition, Chapter 1, Springer Publications, 2008.

[5] Gary Chartrand, Linda Lesniak and Ping Zhang, *Graphs and Digraphs*, Fifth Edition, Chapter 10, CRC Press, October 2010.

[6] Miklos Bona, *A Walk through Combinatorics: An Introduction to Enumeration and Graph Theory*, Third Edition, Chapter 10, World Scientific Publishing, May 2011.

[7] Lih-Hsing Hsu and Cheng-Kuan Lin, *Graph Theory and Interconnection Networks*, Chapter 6, CRC Press, September 2008.

[8] Gary Chartrand and Ping Zhang, *Chromatic Graph Theory*, Chapter 4, CRC Press, September 2008.

[9] S.Armen Asratian, M.Tristan Denley and Roland Haggkvist, *Bipartite Graph and Their Applications*, Chapter 6, Cambridge University Press, August 1998.

[10] R.L Graham, M.Grotschel and L.Lovasz, *Handbook of Combinatorics*, Chapter 3, North Holland, December 1995.

[11] Klaus Jansen and Roberto Solis-Oba, *Approximation and Online Algorithms*, First Edition, Chapter 1, Springer Publications, February 2004.

[12] Ira Gessel and Gian –Carlo Rota, *Classic Papers in Combinatorics,* Birkhauser Boston, May 1987.

[13] Ewa Romanowicz and Adam Grabowski, *The Hall Marriage Theorem,* Univeristy of Bialystok, Volume.12, 2004.

[14] M.M. Cropper, J.L. Goldwasser, A.J.W. Hilton, D.G. Hoffman and P.D. Johnson, *Extending the disjoint –representatives theorems of Hall, Halmos and Vaughan to list multicolorings of graphs*, John Wiley and sons, Volume.33, April 2000.

[15] Bela Bollobas, *Extremal Graph Theory*, Chapter 2, Dover Publications, June 2004.

[16] Eric Schechter, *Handbook of Analysis and its Foundations*, First Edition Chapter 6, Academic Press, October 1996.

[17] L. Mirsky, *Transversal theory: an account of some aspects of combinatorial mathematics*, Academic Press, January 1971.

[18] Jen-Shiang Chen and Jin-Shan Yang, *Model formulations for the machine scheduling problem with limited waiting time constraints*, Taru Publications, Volume.27, 2006.

[19] Teofilo Gonzalez and Sartaj Sahni, *Open Shop Scheduling to Minimize Finish Time,* Association for Computing Machinery, Vol.23, no.4, October 1976.

[20] A. Henry Kierstead, *An Effective Version of Hall's Theorem,* American Mathematical Society, Vol.88, no.1, May 1983.

[21] Ashish Goel, Micheal Kapralow and Sanjeev Khanna, *Perfect Matching via Uniform Sampling in Regular Bipartite Graphs,* ACM, Vol.6, no.2, March 2010.

[22] V. Longani, *Extension of Hall's theorem and an algorithm for finding the (1, n)-Complete Matching,* Thai Journal of Mathematics, Vol.6, no.2, 2008.

[23] A.A. Bolibruch, A.S. Merker'ev and N. Yu Netsvetaev, *Mathematics in St.Petersburg,* American Mathematical Soc., Vol.174.

[24] Rainer Burkard, Mauro Dell'Amico and Silvano Martello, *Assignment Problems,* Chapter 2, Society for Industrial and Applied Mathematics, January 2009.

[25] Francesca Rossi, Peter van Beek and Toby Walsh, *HandBook of Constarint Programming,* First Edition, Chapter 6, Elsevier Science, October 2006.

[26] D.A. Holton, *The Petersen Graph,* Chapter 4, Cambridge University Press, June 1993.

[27] N. Vladimir Sachkov, *Combinatorial Methods in Discrete Mathematics,* Chapter 2, Cambridge University Press, June 2011.

[28] Joseph Y-T. Leung, *Handbook of Scheduling: Algorithms, Models, and Performance Analysis,* Chapter 6, CRC Press, 2004.

[29] Pierre Hansen, Nenad Mladenovic and A. Jose Moreno Perez, *Variable neighborhood search: methods and applications,* Springer Publications, May 2008.

[30] Emile Aarts and Jan Karel Lenstra, *Local Search in Combinatorial Optimization,* Chapter 1, Wiley-Interscience Publications, August 2003.

[31] L. R. Ford and D. R. Fulkerson, *Ford Fulkerson Algorithm,* International Book Market Service Limited, 1954.

[32] Jon Kleinberg and Eva Tardos, *Algorithm Design,* Chapter 7, Addison-Wesley, March 2005.

[33] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, *Introduction to Algorithms*, Third Edition, Chapter 34, The MIT Press Cambridge, Massachusetts London, England, 2009.

[34] Zsuzsanna Vaik, *On scheduling problems with parallel multi-purpose machines*,EGRES Technical Report No. 2005-02, Jan 2005.

[35]A.M.S Zalzala, *Genetic algorithms in engineering systems*, Chapter 7, INSPEC, Incorporated/Institution of Electrical Engineers, January 1997.

[36] R. Panneerselvam, *Production And Operations Management*, Second Edition, Chapter 14, PHI Learning Pvt. Ltd., 2006.

VITA

Graduate College

University of Nevada, Las Vegas

Arunasri Chitti

Degrees:

Bachelor of Technology in Information Technology, 2012

Osmania University

Master of Science in Computer Science, 2014

University of Nevada Las Vegas

Thesis Title: A Characterization of Open Shop Scheduling Problems using the Hall Theorem and Network Flow

Thesis Examination Committee:

Chair Person, Dr. Wolfgang Bein, Ph.D.

Committee Member, Dr. Ajoy K. Datta, Ph.D.

Committee Member, Dr. Lawrence L.Larmore, Ph.D.

Graduate College Representative, Dr. Venkatesan Muthukumar, Ph.D.