ABSTRACT

HAN, XIAOTIAN. Allocating Manpower with Absolute Partial Labor Flexibility to Optimize a Lower Bound on L_{max} in Job Shop Scheduling Problems. (Under the direction of Dr. Thom J. Hodgson.)

This research examines dual resource constrained job shop scheduling problems considering skill-limited labor as the second constraint, with the objective of minimizing maximum lateness. Absolute partial labor flexibility (APLF) is defined and modeled: a skill matrix is introduced to illustrate the skill structure of the workers in a job shop; and a worker allocation matrix is used to keep information on the number of workers in each worker group assigned to each machine group. The problem of determining whether a worker allocation is feasible under a given APLF situation is then formulated as an Allocation Feasibility Determination (AFD) problem. Two existing LP based methods that can be used to solve the AFD problem are discussed. A new method that searches for worker reassignment from a feasible allocation to gain the allocation being tested (the PSP Method) is developed, where the worker reassignment is realized by a new search algorithm—the Path Searching Procedure (PSP). Then the Lower Bound Search Algorithm (Lobo, 2011) is extended to work under the APLF situation, using the PSP Method to test the feasibility of each allocation, so that it will still find the allocation with the optimal lower bound on L_{max}. Experimental results indicate that the flexibility performance of a job shop is affected by complex factors, most strongly by skill structures and the symmetry of the job shop. If the average number of machine groups that the workers in each worker group can operate is greater than or equal to 4 out of 10, or if the staffing level of the job shop is 90% or above, then the job shop will have the same performance as the same one with complete labor flexibility.

Allocating Manpower with Absolute Partial Labor Flexibility to Optimize a Lower Bound on Lmax in Job Shop Scheduling Problems

by
Xiaotian Han

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

Industrial Engineering

Raleigh, North Carolina

2012

APPROVED BY:

_____	_____
Dr. Thom J. Hodgson Committee Chair	Dr. Kristin A. Barletta
_____	_____
Dr. Russell E. King	Dr. James R. Wilson

DEDICATION

To my father, Min Han, and my mother, Jie Cong

To all who take time to read this thesis

BIOGRAPHY

Xiaotian Han was born in 1986 in Lanzhou, China. He started his undergraduate study in 2005 in Nanjing, China, which is 1500 miles away from his hometown, and received his Bachelor of Science in Applied Physics from Nanjing University in 2009. Xiaotian then relocated to Raleigh, North Carolina, which is 15000 miles away from his hometown, as an international graduate student in Department of Industrial and Systems Engineering in North Carolina State University.

Xiaotian is a very talented clarinet player and a popular singer.

ACKNOWLEDGEMENTS

I would like to thank the following people:

- Zhengzhong Liu
- Dr. Thom Hodgson
- Dr. Kristin Thoney
- Dr. Russell King
- Dr. James Wilson
- Dr. Benjamin Lobo
- Zhuowei Wang
- Yingying Wang

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1 Introduction

Sequencing jobs in most job shop environments is an NP-Hard problem (Lenstra *et al*., 1977). During the past few decades, considerable effort has been spent on finding ways of getting optimal or near-optimal solutions (Mellor, 1966; Graham *et al*., 1979; Emmons, 1987; Baker and Scudder, 1990; Potts and Strusevich, 2009). The majority of this literature has dealt with scheduling in job shops constrained by machines only. There are several assumptions for this single resource constrained job shop (Lobo, 2011):

- Each job has a specific routing through the machines.
- There are no machine breakdowns.
- No preemption is allowed.
- Transportation time between different machines and set-up time on machines is negligible (one can view the processing time of a job as already having the set-up time included).
- A machine can only process one job at a time, and a job can only be processed on one machine at a time.

However, in the real world, a job shop is typically dual resource constrained or even multi-resource constrained. The number of workers available to operate the machines is often far less than the number of machines in the job shop (Gargeya and Deane, 1996).

In the recent work presented by Lobo (2011), a dual resource constrained (DRC) job shop scheduling problem was studied. In their study, machines are grouped together according to their different functions, and each worker is assigned to a specific machine group to operate

the machines within it. The additional assumptions that transform the single resource constrained job shop into a dual resource constrained job shop are (Lobo, 2011):

- There are fewer workers than machines in the job shop.
- A machine group must have at least one worker assigned to it; otherwise it cannot process any jobs.
- A machine takes one worker to operate it, and a worker cannot operate more than one machine at a time.
- The allocation of workers, once determined, is static.
- Every worker can be assigned to any machine group in the job shop (complete labor flexibility, CLF).

However, the last assumption is often not true: workers are usually not equally trained, and as a result, each worker will only be able to operate the machines in a subset of the machine groups. Workers with the same skills can be grouped together in a worker group. This type of system is termed a partial resource (labor) flexibility system (Daniels *et al*., 2004). We define partial labor flexibility (PLF) to be the situation where every worker in the job shop can work on more than one type of machine but is either unable to operate all types of machines or unable to operate the machines with equal skillfulness. Therefore, partial labor flexibility can be divided into two subtypes: absolute partial labor flexibility and relative partial labor flexibility. Absolute partial labor flexibility (APLF) is a type of partial labor flexibility in which each of the workers is either fully trained to operate a certain machine (group) or unable to operate that machine (group) at all. Relative partial labor flexibility (RPLF) is a type of partial labor flexibility in which workers may operate machines with different skill

levels. The 100% level is “capable” while the 0% is “unable” in terms of the absolute flexibility case. A new worker could have a 30% skill level while a veteran could have an 80% skill level.

The single resource constrained job shop scheduling problem with the objective of minimizing L_{max}, denoted as $N/M/L_{max}$, is NP-Hard. Hodgson *et al*. (1998) used a heuristic approach to solve this problem. Lobo (2011) added workers (labor) to the $N/M/L_{max}$ job shop scheduling problem. He developed a procedure to find the allocation of workers to machine groups in the job shop that optimizes a lower bound on L_{max}, where the Virtual Factory, a heuristic scheduler developed by Hodgson *et al*. (1998) was used to generate a schedule for a given worker allocation. However, a feasible allocation under the complete flexibility assumption may not be feasible in the partial labor flexibility situation.

In this thesis, a dual resource constrained job shop scheduling problem with absolute partial labor flexibility (APLF) is investigated. All the assumptions used in Lobo (2011), except for “every worker can be assigned to any machine group in the job shop” will be applied. The following additional assumptions regarding the partial flexibility issue will also be made:

- Each worker has a static skill pattern, i.e. this worker can only be assigned to a subset of all the machine groups, which does not change over time.
- A certain worker (worker group) is either unable to operate machines in a machine group completely, or able to operate machines in a machine group with the same level of skillfulness as any other worker (worker group) who can operate machines in that machine group.

The minimization of L_{max} is again used as the performance measure in this research, as it addresses the needs of on-time completion of jobs required by most companies (Ovacik and Uzsoy, 1996).

In this thesis, the APLF job shop scheduling problem is modeled, and the problem of determining whether a worker allocation is feasible in the APLF problem settings is defined as an Allocation Feasibility Determination (AFD) problem. A procedure that searches for feasible worker movement path and solves the AFD problem, the Path Searching Procedure (PSP), is developed. The Lower Bound Search Algorithm (LBSA) is extended to work in the APLF situation so that the optimal lower bound on L_{max} can still be found. Then a series of experiments are conducted to compare the optimal lower bounds obtained for the APLF job shop with the ones obtained for the CLF job shop.

In Chapter 2, a literature review and some background research are presented. In Chapter 3, new terms and definitions required by the AFD problem are introduced, the path searching procedure and two traditional methods that can be used to solve the AFD problem are discussed, and the revised lower bound search algorithm that finds the allocation with the optimal lower bound in APLF situation is developed. Chapter 4 presents the experimental results of the performance of lower bounds. Finally, conclusions and future research possibilities are presented in Chapter 5.

Chapter 2 Literature Review

In Section 2.1, literature on dual-resource constrained systems with partial labor flexibility is reviewed. The Virtual Factory approach to solving the $N/M/L_{max}$ problem is presented in Section 2.2, and previous work on finding the allocation that optimizes the lower bound on L_{max} using complete labor flexibility is briefly described in Section 2.3.

2.1 Dual-Resource Constrained Systems with Partial Labor Flexibility

Treleven and Elvers (1985) defined a dual constrained job shop as "one in which shop capacity may be constrained by machine and labor capacity or both. This situation exists in shops that have equipment that is not fully staffed and machine operators who are capable of operating more than one piece of equipment". They also claimed that workers "may be transferred from one work center to another (subject to skills restrictions) as the demand dictates. When a work center is fully staffed, transfer of additional operators to that center will not increase capacity".

Felan *et al*. (1993) studied how labor flexibility and staffing levels affect the performance of a job shop. A homogeneous workforce (i.e. the skills of each worker are stable) is considered, which is consistent with the assumptions made in this research. The labor flexibility is obtained by enabling each worker to work in multiple departments in the job shop. Four levels of worker flexibility (levels 1 through 4) and four levels of staffing (50%, 60%, 70%, 80%) were tested. A level *i* flexibility corresponds to each worker being able to work in *i* different departments. Given there are 10 machines in each department, a 70%

staffing level stands for each department having 7 workers assigned to it. They used mean flow time, average tardiness, and percentage of jobs tardy as performance measures, which "represent the primary drivers for measuring manufacturing performance in many organizations" (Felan *et al*., 1993).

Their experiments showed that when the worker flexibility level increased given a fixed staffing level or the staffing level increased given a fixed worker flexibility level, the number of transfers increased, and the mean flow time, average tardiness and percentage of tardy jobs decreased. When the worker flexibility level and the staffing level both increased, all incremental changes in the number of transfers, the mean flow time, average tardiness, and percentage of tardy jobs respectively diminished. Based on those results in the WIP and due-date performance, they concluded that a job shop (as they modeled it) with 60% staffing level and a medium worker flexibility level (level 2 or 3) is the optimal job shop.

"Although Felan *et al*. (1993) allow workers to transfer between departments, they are concerned with finding the optimal staffing level and workforce flexibility level combination, rather than optimizing the allocation of workers to departments" (Lobo, 2011). Moreover, though partial labor flexibility was considered in their study, all workers are assumed to have the same level of flexibility once the worker flexibility level of the job shop is determined, which is a restricted case rather than a general case. There was no formulation of which departments each worker can be assigned to. Thus one cannot find the allocation of different workers among departments in the job shop.

Daniels and Mazzola have published a series of papers on flow shop scheduling with resource flexibility since 1993. In the early research, they assumed complete resource

flexibility. They claimed that "resource flexibility can have a significant impact on the quality of a schedule", and "some resources are fundamentally flexible in that they are capable of migrating to processing centers as needed" (Daniels and Mazzola, 1994). "Labor flexibility can be achieved by cross-training workers to develop the skills required to perform different tasks associated with multiple processing centers" (Daniels and Mazzola, 1994). They termed the problem the flexible-resource flow shop scheduling (FRFS) and formulated it as a mixed integer program. They developed both optimal and heuristic methods to find the solution to the mixed integer program.

Daniels *et al*., (2004) did another study which extended their previous research on resource flexibility to a wider application—flow shop scheduling with absolute partial resource flexibility. In that paper, they defined resource flexibility as "the ability to dynamically reallocate units of resource from one stage of a production process to another in response to shifting bottlenecks" (Daniels *et al*., 2004). They termed the problem the "flow shop rescheduling with partial resource flexibility problem" (FSPRF).

To formulate this partial flexibility problem, Daniels *et al*., (2004) introduced a very useful structure to keep the skill flexibility information of a flow shop—the skill matrix. Let W be the set of all workers, and M be the set of all stations (machines). Also let $w=|W|$ and $m=|M|$. Then the worker-station skill matrix S is defined as a $w \times m$ matrix with element

$$s_{hj,h\in W,j\in M} = \begin{cases} 1, & \text{if worker } h \text{ is trained to staff station } j; \\ 0, & \text{otherwise.} \end{cases}$$

Furthermore, they defined a skill matrix to be feasible when each column and row of that matrix have at least one element NOT equal to zero. To better evaluate a skill matrix, they

introduced flexibility metric, *FM*, which simply computes the portion of "1" elements in the skill matrix. Mathematically, it can be presented as

$$FM = \left(\sum_{i \in W} \sum_{j \in M} s_{ij}\right)/wm. \quad (2.1)$$

Other useful metrics defined by Daniels *et al*. (2004) are the station-balance metric and station worker-balance metric. The station-balance metric considers the difference between the station which has the largest number of workers capable to operate and the station which has the smallest number of capable workers. The station worker-balance metric splits the workers across stations which they can operate, based on the station-balance metric.

A specially designed ideal skill matrix structure defined by Daniels *et al*. (2004) is the *k*-chain. It is an $m \times m$ matrix where type i worker can work k consecutive work stations, i through $i+k-1$, where $0<k<m$. For any $i'>m$, $i' \in [i, i+k-1]$, the corresponding work station i' is made as work station $i'-m$. These chain structures satisfy the principles on the benefits of manufacturing process flexibility studied by Jordan and Graves (1995).

In the study of Jordan and Graves (1995), they considered how much flexibility is needed and how to configure flexibility for plants. The two principles they claimed were 1) "limited flexibility (i.e., each plant builds only a few products), configured in the right way, yields most of the benefits of total flexibility (i.e., each plant builds all products)" and 2) "limited flexibility has the greatest benefits when configured to chain products and plants together to the greatest extent possible", where a chain was defined to be "a group of products and plants which are all connected, directly or indirectly, by product assignment decisions" (Jordan and Graves 1995). They went on to state that "within a chain, a path can be traced from any product or plant to any other product or plant via the product assignment

links". Their experimental results illustrated that fewer (i.e. less links added) and longer (i.e. more nodes connected) chains lead to more effective flexibility. This manufacturing flexibility configuration with plants and products is consistent with the worker skill flexibility configuration with workers (worker groups) and stations (machine groups). Therefore, the 2-chain structure as defined by Daniels *et al*., (2004) is the best single chain because it connects all workers (worker groups) and all stations (machine groups) with the fewest links.

Daniels *et al*., (2004) dynamically assigned absolutely partially flexible workers in a flow shop and studied the impact of that on the scheduling performance assuming that "the number of workers is equal to the number of work stations". In that flow shop setting, machines and workers are not grouped together, and the flow shop is fully stuffed. Thus, they did not need to consider a numerical allocation of workers among machine groups.

In summary, though there is a good deal of comprehensive research on both job shop and flow shop scheduling problems considering labor as well as the labor flexibility, based on our knowledge, there is no research on finding the optimal worker allocation in job shop scheduling with partial labor flexibility with the objective of minimizing $L_{\max}$. Most of the studies on partial labor flexibility focused on how different flexibility levels affect system performance, but none of them designed a detailed procedure of finding a numerical allocation of partially flexible workers in a job shop.

2.2 Virtual Factory solution for $N/M/L_{\max}$ Problem

The Virtual Factory (Hodgson *et al*., 1998) is a transient, deterministic simulation of a job shop, where the factory is run from its current state until all jobs are completed. The first step

of the simulation is to determine the initial due date for each job *i* at each machine *m*. The latest possible time that a job can finish on machine *m* and still satisfy its final due date is termed $\text{slack}_{i,m}$, and is computed as

$$\text{slack}_{i,m} = d_i - \sum_{j \in m_i^+} p_{ij}. \tag{2.2}$$

Carroll (1965) found that slack did not perform well as a sequencing rule according to experiments, because queuing time is not considered when computing slack. Therefore, the Virtual Factory procedure incorporates queuing times into a revised slack measure to sequence jobs.

To estimate the queuing times, the Virtual Factory uses an approach proposed by Vepsalainen and Morton (1988)—successive approximations using deterministic simulation. The job shop is first simulated using $\text{slack}_{i,m}$ as a sequencing rule. Queuing times are then recorded for each job at each machine visited. Equipped with the queuing times (q_{ij}), a "revised" slack can be calculated as

$$\text{slack}'_{i,m} = d_i - \sum_{j \in m_i^+} p_{ij} - \sum_{j \in m_i^{++}} q_{ij} \tag{2.3}$$

where m_i^{++} is the set of all operations after machine *m* on the routing machine list for job *i* except the immediate subsequent operation. The simulation is run again using the revised slack from the previous iteration as the sequencing rule. This procedure will be repeated until the queuing times steady—usually 3 to 10 iterations—or the lower bound is achieved. The procedure can solve industrial scale problems in seconds.

To evaluate the quality of a schedule generated by the Virtual Factory for the N/M/$L_{\max}$ problem, a comparison of the $L_{\max}$ value of the schedule is made with a lower bound on $L_{\max}$.

A well-known and useful method to calculate a lower bound was first developed by Carlier and Pinson (1984). Define the earliest possible start time for each job *i* on machine *m* as

$$\mathrm{ES}_{i,m} = \sum_{j \in m_i^-} p_{ij} \tag{2.4}$$

where m_i^- is the set of all operations previous to machine *m* on the routing machine list for job *i*, and p_{ij} denotes the processing time of job *i* on machine *j*. Secondly, compute the latest possible finish time for each job *i* on machine *m* as

$$\mathrm{LF}_{i,m} = d_i - \sum_{j \in m_i^+} p_{ij} \tag{2.5}$$

where d_i is the due date of job *i*, and m_i^+ is the set of all operations after machine *m* on the routing machine list for job *i*. Then $\mathrm{ES}_{i,m}$ can be treated as the effective release time ($r_{i,m}$) for job *i* on machine *m*. And similarly, $\mathrm{LF}_{i,m}$ can be interpreted as the effective due date for job *i* on machine *m*. Therefore, we can calculate the lower bound for every machine *m*, LB_m, as a $N/1/L_{\max}|r_{i,m}$ problem. The lower bound for the $N/M/L_{\max}$ problem, LB, is then

$$\mathrm{LB} = \max_{m=1,\ldots,M}\{\mathrm{LB}_m\}. \tag{2.6}$$

This lower bound can be computed very quickly by the help of the relaxation suggested by Baker and Su (1974), which allows preemption of a job in process whenever a job with a more imminent due date becomes available. This lower bound is effective because there are *M* opportunities to get a tight bound. This method was used as the method of obtaining a lower bound in the Virtual Factory.

Extensions to the Virtual Factory include: Hodgson *et al*. (2000) generated schedules that identify and prioritize critical jobs; Weintraub *et al*. (1999) identified the best alternative process plans incorporated with tabu search; Hodgson *et al*. (2004) and Thoney *et al*. (2002) extended the Virtual Factory to consider multiple factories; Zozom *et al*., (2003) optimized

the release time of jobs; Schultz *et al*. (2004) improved solution quality by incorporating simulated annealing; and Lobo (2011) transformed the Virtual Factory into a dual resource constrained job shop.

2.3 $N/\{M,W\}_{js}/L_{\max}$ Problem with Complete Labor Flexibility

2.3.1 Optimizing the Lower Bound on $L_{\max}$

In a dual resource constrained (DRC) system, how workers are allocated to each machine group may affect the performance of the job shop. As noted in Chapter One, Lobo (2011) developed a procedure to find the allocation of workers to machine groups in the job shop that optimizes a lower bound on $L_{\max}$. The problem was defined as a $N/\{M,W\}_{\text{js}}/L_{\max}$ job shop scheduling problem. $W = \sum_i^S w_i$, where S represents the number of machine groups, and w_i is the number of workers assigned to each machine group. An allocation was denoted as $\vartheta = [w_1, \dots, w_S]$. The basic idea was to first compute the lower bound on $L_{\max}$ for each machine group and then find the maximum value over all machine groups. For a given allocation ϑ, the lower bound on $L_{\max}$ for the job shop is

$$\text{LB}_{\vartheta} = \max_{i=1,\dots,S}\{\text{LB}_{\text{mg}_i}(w_i)\}. \tag{2.7}$$

They defined the $N/\{M,W\}_{\text{mg}}/L_{\max}|r_{j,m}$ job shop scheduling problem for each machine group. The local release time and effective due-date for each job is determined using Equations (1.1) and (1.2).

Finding $\text{LB}_{\text{mg}}(L_{\max})$ was formulated as a series of network flow problems, each testing to see if a certain $L_{\max}$ value could be achieved. The formulation was based on an approach developed by Labetoulle *et al*. (1984) for minimizing $L_{\max}$ on preemptive parallel machines.

Lobo (2011) added worker nodes to the network to effectively constrain the number of machines operated in each time period.

They used the out-of-kilter algorithm to determine whether the network flow formulation was feasible using a given trial $L_{\max}$ value. A series of network flows were solved starting with an $L_{\max}$ value that allowed a feasible network, and decreasing the $L_{\max}$ value by one unit until the network was infeasible. Since this network flow formulation allows for preemption, the smallest $L_{\max}$ value that allows a feasible network is a lower bound on $L_{\max}$.

To find an initial feasible $L_{\max}$ value for a machine group, a constructive algorithm was developed based on the method of solving the $N/1/L_{\max}|r_{j,m}$ problem with preemption that considered worker availability. The main idea is that for each time *t*, the algorithm chooses the w_i machines with the largest $L_{\max}$ values to be operated so that the $L_{\max}$ value of the machine group can be minimized. Using the network flow model and the constructive algorithm, the lower bound on $L_{\max}$ for any job shop with a given worker allocation ϑ can be determined.

Lobo (2011) proved that "the lower bound on $L_{\max}$ for a machine group is a discrete, monotonic, non-increasing function of the number of workers assigned to the machine group". He designed a search algorithm, named the Lower Bound Search Algorithm (LBSA) to find an allocation that minimizes the lower bound on $L_{\max}$ for the job shop. The algorithm starts with any given allocation ϑ. It first finds the set of constraining machine groups:

$$K = \arg\max_{1 \le l \le S}\{\mathrm{LB}_{\mathrm{mg}_l}\big(w_l(\vartheta)\big)\}. \tag{2.8}$$

If any machine group in *K* has as many workers as machines in that group, the algorithm stops with termination CONDITION 1: "no additional workers can be assigned to this

Algorithm 1 The Lower Bound Search Algorithm (LBSA)

Set $\lambda = 0$ TC=2
Determine an initial allocation ϑ^0.
1.
Determine $\mathrm{LB}_{\vartheta^\lambda} = \max_{1\le l\le S}\left\{\mathrm{LB}_{\mathrm{mg}_l}\left(w_l\left(\vartheta^\lambda\right)\right)\right\}$
Find $K = \mathrm{argmax}_{1\le l\le S}\left\{\mathrm{LB}_{\mathrm{mg}_l}\left(w_l\left(\vartheta^\lambda\right)\right)\right\}$
If $w_k\left(\vartheta^\lambda\right) = M_{\mathrm{mg}_k}$ for any $k \in K$
TC=1 STOP. ϑ^λ MINIMIZES LB($L_{\max}$) FOR THE JOB SHOP.
Else
Choose $k \in K$.
Set $F = K$
For $z \in \{1, \dots, S\}\backslash F$
If $w_z\left(\vartheta^\lambda\right) = 1$
$F \leftarrow F \cup \{z\}$
2.
If $\{1, \dots, S\}\backslash F \neq \emptyset$
Find $J = \mathrm{argmin}_{1\le l\le S, l\notin F}\left\{\mathrm{LB}_{\mathrm{mg}_l}\left(w_l\left(\vartheta^\lambda\right)\right)\right\}$
Choose $j \in J$
If $\mathrm{LB}_{\mathrm{mg}_j}\left(w_j\left(\vartheta^\lambda\right)\right) > \mathrm{LB}_{\mathrm{mg}_k}\left(w_k\left(\vartheta^\lambda\right)\right)$
$F \leftarrow F \cup \{j\}$
GO TO 2.
Else
$\vartheta^{\lambda+1} = \vartheta^\lambda$
$\vartheta^{\lambda+1}(j) = \vartheta^\lambda(j) - 1$
$\vartheta^{\lambda+1}(k) = \vartheta^\lambda(k) + 1$
If $\vartheta^{\lambda+1} \in \{\vartheta^0, \dots, \vartheta^\lambda\}$
$F \leftarrow F \cup \{j\}$
GO TO 2.
Else
$\lambda = \lambda + 1$
GO TO 1.
Else
STOP. ϑ^λ MINIMIZES LB($L_{\max}$) FOR THE JOB SHOP.

machine group". Otherwise, one machine group is picked from *K* as the one the algorithm needs to reassign an additional worker. The machine groups which cannot have worker unassigned from them for any reason are put into a set *F*. These reasons include: 1) the machine group is in set *K*; 2) the machine group has only one worker; 3) the machine group will have a lower bound on $L_{\max}$ greater than the current lower bound after one worker is unassigned; 4) the allocation that is realized after unassigning one worker from the machine group is one of the allocations the algorithm has already tried. One machine group is picked from set *J* as the machine group the algorithm needs to unassign a worker from, where

$$J = \arg\min_{1 \le l \le S, l \notin F} \{\mathrm{LB}_{\mathrm{mg}_l}(w_l(\vartheta))\} \tag{2.9}$$

and *F* is initialized with the machine groups that satisfy the first and second reasons noted above. Finally, the machine groups in *J* that satisfied the third and fourth reasons noted above are put into set *F*; and one machine group that is proved to have worker to be unassigned is used to realize the reassignment with the objective machine group picked from *K*. After these four steps, an iteration is completed. The algorithm continues to search for improvement to the LB until termination CONDITION 2 is satisfied: "it is not possible to reassign a worker to the constraining machine group from any of the *S*-1 other machine groups".

To conclude their study on the lower bound searching procedure, they proved that "the Lower Bound Search Algorithm (LBSA) is an optimal algorithm" (the LBSA will find an allocation with the minimum LB value).

2.3.2 Optimality Criteria and Performance Evaluation

A more recent paper presented by Lobo (2011) addressed and solved three issues following their previous study on optimizing the lower bound on $L_{\max}$. First, they built up a set of

optimality criteria which helps determine whether the allocation found by LBSA with the minimum lower bound—allocation ϑ^* is optimal. They defined the optimal allocation to be the allocation that allows the Virtual Factory to generate the schedule that has the minimum possible $L_{\max}$ value for the $N/\{M,W\}_{js}/L_{\max}$ job shop. Second, they introduced a method for evaluating the performance of an allocation by computing the probability distribution. Finally, they developed three heuristic searching strategies to find allocations which yield a smaller $L_{\max}$ value than that corresponding to the minimum lower bound allocation found by LBSA. They were used when the allocation ϑ^* neither satisfied the optimality criteria nor gave statistical confidence that it was the optimal allocation.

Chapter 3 Optimizing the Lower Bound on L_{max} over Feasible Allocations

In Section 3.1, the Allocation Feasibility Determination (AFD) problem is defined and formulated, and notation and concepts useful in the discussion of this new problem are defined. In Section 3.2, two existing methods that can be used to solve the AFD problem are introduced. A new method named the Path Searching Procedure is developed and discussed in Section 3.3. In Section 3.4, the Lower Bound Search Algorithm (LBSA) is extended to the absolute partial labor flexibility (APLF) situation by using the Path Searching Procedure as the methodology to determine the feasibility of allocations.

3.1 New Problem Settings

As an extension to the study of Lobo (2011), this thesis will transform the Lower Bound Search Algorithm to find the allocation with the optimal lower bound for any given APLF situation. For each worker allocation proposed by the LBSA, we need to determine whether it is feasible in that APLF situation and to justify the answer. We first model this feasibility. There are some basic terms used in the following discussion:

- mg_j – machine group j
- wg_i – worker group i
- M – the set of all machine groups
- W – the set of all worker groups
- $S = |M|$ – the number of machine groups
- $T = |W|$ – the number of worker groups

- $\vartheta = (w_1(\vartheta),\ w_2(\vartheta), \dots, w_S(\vartheta))$ – a given worker allocation
- $\pi = (\pi_1,\ \pi_2, \dots, \pi_T)$ – the number of workers in each worker group

3.1.1 Skill Matrix and Worker Allocation Matrix

A skill matrix is an *S* by *T* matrix which is used to record the skillset information of each worker group, where each row represents a worker group and each column represents a machine group. The element of the matrix, $s_{i,j}$, equals 1 if the workers in worker group *i* are trained to operate the machines in machine group *j*; it equals 0 otherwise. Note that the skill matrix is feasible if and only if there is no row or column whose elements are all zero (Daniels *et al*., 2004). The combination of a skill matrix and the worker distribution vector, π, uniquely defines a certain APLF situation.

The number of workers in each worker group that are allocated to each machine group is kept in the worker allocation matrix [*A*], where each element $a_{i,j}$ represents the number of workers in worker group *i* that are assigned to machine group *j*. $a_{i,j}$ is defined as a feasible element if the corresponding element $s_{i,j}$ in the skill matrix equals 1, as infeasible elements otherwise. $a_{i,j}$'s can only have nonnegative integer values.

Table 1 Skill Matrix and the Associated Worker Allocation Matrix

$wg_i \backslash mg_j$	1	2	3	4	5
1	1	1	0	0	1
2	0	1	0	1	0
3	0	0	1	1	0
4	1	0	0	0	1
5	0	1	1	1	0

$wg_i \backslash mg_j$	1	2	3	4	5
1	$a_{1,1}$	$a_{1,2}$			$a_{1,5}$
2		$a_{2,2}$		$a_{2,4}$	
3			$a_{3,3}$	$a_{3,4}$	
4	$a_{4,1}$				$a_{4,5}$
5		$a_{5,2}$	$a_{5,3}$	$a_{5,4}$	

We use an arbitrary APLF structure with 5 machine groups and 5 worker groups (S=5, T=5) as an example. The skill matrix and the worker allocation matrix associated with it are shown in Table 1 above.

3.1.2 From Allocation to Allocation Matrix

A given worker allocation is feasible in a given APLF situation defined by a skill matrix and a worker distribution vector if and only if we can find nonnegative integer values for all feasible elements in the associated worker allocation matrix, so that the sums of the values of the feasible elements of each row in the matrix equal the numbers of workers in the corresponding worker groups, and the sums of the values of the feasible elements of each column equal the numbers of workers assigned to the corresponding machine groups. Specifically, given the skill matrix s and the vector of the number of workers in each worker group π, we can claim a certain worker allocation ϑ as feasible if and only if we can find $a_{i,j}$ for all $(i,j) \in \{(i,j)|s_{i,j} = 1\}$, such that

$$\begin{cases} \sum_{j:\, s_{i,j}=1} a_{i,j} = \pi_i, \ i = 1 \text{ to } T \\ \sum_{i:\, s_{i,j}=1} a_{i,j} = w_j(\vartheta), \ j = 1 \text{ to } S \\ a_{i,j} \in \mathbf{Z}, \ a_{i,j} \geq 0 \end{cases} \tag{3.1}$$

To better illustrate the allocation feasibility determination problem, we again use the skill structure in Table 1 as an example. We arbitrarily make π = (5, 5, 5, 5, 5); and we design an arbitrary worker allocation ϑ = (3, 6, 7, 4, 5). The problem of determining feasibility of this worker allocation is then illustrated as Example 1.

Example 1, allocation feasibility determination (AFD): Find nonnegative integer values for all $a_{i,j}$'s in Table 2 so that the feasibility condition is satisfied.

Table 2: Worker Allocation Matrix with Unknown Feasible Elements Values

$wg_i \backslash mg_j$	1	2	3	4	5	π_i
1	$a_{1,1}$	$a_{1,2}$			$a_{1,5}$	5
2		$a_{2,2}$		$a_{2,4}$		5
3			$a_{3,3}$	$a_{3,4}$		5
4	$a_{4,1}$				$a_{4,5}$	5
5		$a_{5,2}$	$a_{5,3}$	$a_{5,4}$		5
$w_j(\vartheta)$	3	6	7	4	5	

We need to find a nonnegative integer solution for the following system of equations:

- $a_{1,1}+a_{1,2}+a_{1,5}=5$; $a_{2,2}+a_{2,4}=5$; $a_{3,3}+a_{3,4}=5$; $a_{4,1}+a_{4,5}=5$; $a_{5,2}+a_{5,3}+a_{5,4}=5$; $a_{1,1}+a_{4,1}=3$; $a_{1,2}+a_{2,2}+a_{5,2}=6$; $a_{3,3}+a_{5,3}=7$; $a_{2,4}+a_{3,4}+a_{5,4}=4$; $a_{1,5}+a_{4,5}=5$. (3.2)

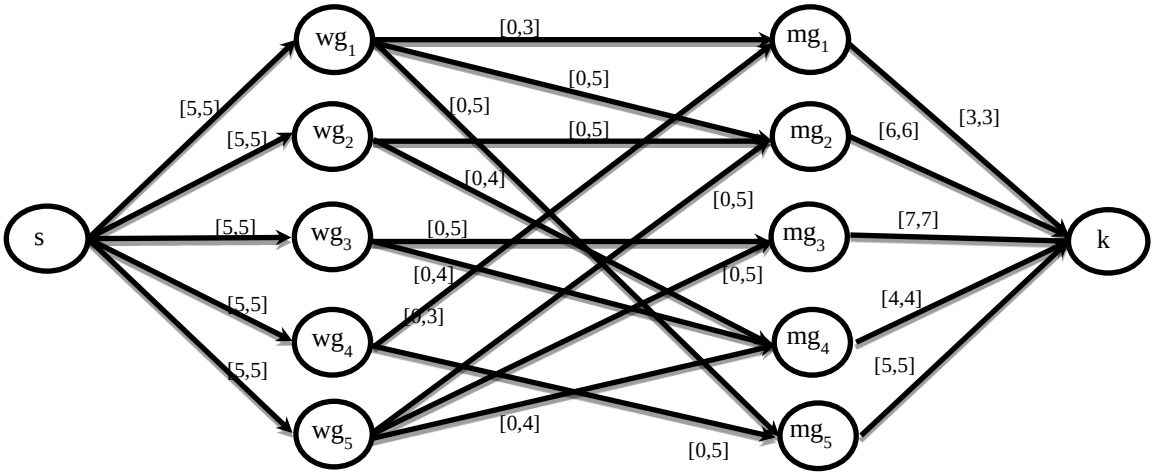

Figure 1 Network Flow Model of Example 1

An equivalent model is the network flow model depicted in Figure 1. In this network flow graph, the wg_i nodes represent worker groups and the mg_j nodes represent machine

groups. There is an arc from wg_i to mg_j if $s_{i,j}$=1. The arcs connecting the worker group nodes and the source node must have a flow equal to the number of workers in each worker group. The arcs connecting the machine group nodes and the sink node must have a flow equal to the given worker allocation. Solving the AFD problem is equivalent to deciding whether a feasible flow for the network can be found.

3.2 Traditional Methods to Solve the AFD Problem

There are at least two traditional methods to solve the AFD problem: 1) formulate as a transportation problem with prohibited routes and homogeneous costs; 2) use the out-of-kilter method to determine a feasible flow.

If we make the feasible elements (routes) as variables, make the feasibility condition (3.1) as constraints, we actually have defined a feasible region for an integer linear programming problem. The feasible region standard form polyhedron of the IP formulation of Example 1 is shown in Table 3.

Table 3 Polyhedron of the IP Formulation of Example 1 Represented in Matrix

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	b
1	1	1										5
			1	1								5
					1	1						5
							1	1				5
									1	1	1	5
1							1					3
	1		1						1			6
					1					1		7
				1		1					1	4
		1						1				5

Because of the special structure of this IP model, which is actually a transportation problem with prohibited routes, we can use the transportation method to solve it. Make worker groups as suppliers, machine groups as customers, number of workers in each worker group as the supplies of each supplier, and the number of workers assigned to each machine group as demands of each customer. Each pair of worker group and machine group connected by an arc has a route between them with consistent cost of 1; each of other pairs has a prohibited route between them with a cost of a big positive number *M*. Then the cost matrix can be written as in Table 4.

Table 4 Cost Matrix of the Transportation Problem Associated with Example 1

Cost	1	2	3	4	5	Supply
1	1	1	*M*	*M*	1	5
2	*M*	1	*M*	1	*M*	5
3	*M*	*M*	1	1	*M*	5
4	1	*M*	*M*	*M*	1	5
5	*M*	1	1	1	*M*	5
Demand	3	6	7	4	5	

The corresponding transportation problem is to determine the shipping amount for each pair of supplier and customer so that the total cost is minimized. Any solution, with total cost smaller than *M*, yields a feasible flow for the network in Figure 1. This transportation problem can be solved by the simplex method for transportation problems. For this particular problem, the transportation method took 6 iterations to find a solution.

Another way of finding a feasible flow (circulation) of a network flow problem is the out-of-kilter method. The out-of-kilter method tries to obtain a feasible flow, i.e. a circulation

and a set of node numbers that satisfy the kilter condition, by making change either in the circulation or in the node numbers. We apply Minty's painting theorem to make the changes. There are four steps of the method:

- Step 0: set the circulation to 0, set all node numbers to 0;
- Step 1: painting & labeling;
- Step 2: change in circulation, GO TO Step 1;
- Step 3: change in node numbers, GO TO Step 1.

Because there is no cost associated with each arc, c_{ij}=0 for all i, j, the change in node numbers w_i is calculated as

$$\epsilon = \min_{i,j}\{c_{ij} - (w_j - w_i) | (i,j) \in R \cup Y\} \equiv 0 \tag{3.3}$$

where R is the set of red arcs and Y is the set of yellow arcs according to Minty's painting theorem. This means the node numbers will never change after initialized as 0, in another word, the yellow-red cycle can be ignored. Therefore the out-of-kilter algorithm in this case consists of only the first three steps.

In addition to (3.3), we have

$$w_j - w_i = c_{ij} \equiv 0, \forall (i,j) \in Y \cup G \tag{3.4}$$

where G is the set of green arcs. Therefore, the change in circulation, δ, is calculated as

$$\delta_1 = \min_{i,j}\{u_{ij} - x_{ij} | (i,j) \in Y^+ \cup G^+\}$$

$$\delta_2 = \min_{i,j}\{x_{ij} - l_{ij} | (i,j) \in Y^- \cup G^-\}$$

$$\delta = \min\{\delta_1, \delta_2\} \tag{3.5}$$

where x_{ij} is arc number, l_{ij} is the lower bound, and u_{ij} is the upper bound of arc (i,j), sets with "+" means arc numbers in those sets need to be increased and sets with "-" means arc numbers in those sets need to be decreased (reverse arc direction).

The out-of-kilter algorithm found a feasible flow for the network of Example 1 using 8 iterations.

3.3 Worker Reassignment and Path Searching Procedure

In the following sections, we will introduce a more intuitive approach. Given a worker allocation matrix with arbitrary values on its feasible elements such that the feasibility condition for rows is satisfied, move workers within each row among machine groups until, the feasibility condition for columns, the desired worker allocation, is satisfied. Once the feasibility condition for rows is satisfied, how many workers of each type we have in the job shop and where different types of workers are located are illustrated clearly in a matrix form. Then the adjustment of the distribution of workers in each row can be interpreted more intuitively. This is one of the practical advantages of this method.

Without the constraints associated with the columns, the worker allocation matrix can be initialized easily. For instance, the worker allocation matrix for Example 1 can be initialized as shown in Table 5. From the table we can see that the current worker allocation is inconsistent with the desired worker allocation: one worker should be unassigned from machine group 1, two extra workers should be assigned to machine group 3, two workers should be unassigned from machine group 4, and one extra worker should be assigned to machine group 5. To realize these worker reassignments, a new algorithm was designed.

Table 5 Initial Worker Allocation Matrix for Example 1

$wg_i \backslash mg_j$	1	2	3	4	5	π_i
1	1	2	X	X	2	5
2	X	3	X	2	X	5
3	X	X	3	2	X	5
4	3	X	X	X	2	5
5	X	1	2	2	X	5
Current $w_j(\vartheta)$	4	6	5	6	4	
Desired $w_j(\vartheta)$	3	6	7	4	5	

3.3.1 Definitions

Since the total number of workers is constant, the worker reassignments are realized in pairs, i.e. every time a machine group has a worker assigned to it, there must be another machine group that has a worker unassigned from it. We define a reassignment request as a pair of worker reassignments that need to be achieved together. Thus for each reassignment request, we define a source machine group (the source) as the machine group from which one worker needs to be unassigned, and an objective machine group (the objective) as the machine group to which one worker needs to be assigned. A reassignment request is represented as [*p*, *q*], where *p* is the index of the source machine group and *q* is the index of the objective machine group.

A reassignment request may be realized by one or more worker movements in series. The statement that one worker can be moved from machine group *p* to machine group *q* is equivalent to the one that machine group *p* has one worker to donate to machine group *q*. Workers can only be moved horizontally. In Table 5, for example, [1,5] can be realized simply by moving one worker in worker group 1 once, while [1,3] needs at least two steps

(move one worker in worker group 1 from machine group 1 to machine group 2, then move one worker in worker group 5 from machine group 2 to machine group 3). We define a path for a given worker reassignment request as a set of ordered potential movements of workers in the worker allocation matrix that accomplish the request.

The search algorithm must be able to find a path for any reassignment request if one exists. Along with the source machine group and the objective machine group, we define a target machine group (the target) as an intermediary machine group from which a path is found to the objective machine group. The objective machine group is taken as the first target machine group.

3.3.2 Development of Path Searching Procedure

The basic idea of path searching is to start with the objective machine group and search backward for target machine groups until some target machine group, including the objective machine group, to which one worker can be moved from the source machine group is found. The algorithm should terminate either when a path for the original reassignment request is found or when all machine groups except the source are tested and no worker can be moved from the source to any of those machine groups. The algorithm can be briefly described as in the following four steps:

- Step 1: make the objective as the target and put it into the set of targets.
- Step 2: check if the source has a worker to donate to the target: if yes, done; otherwise, go to step 3.

- Step 3: search for a machine group that is not in the set of targets and has proper worker to donate to the target: if found, set this machine group as the new target and put it into the set of targets, go to step 2; otherwise, go to step 4.
- Step 4: set the previous target as the target: if this machine groups is not the objective, go to step 3; otherwise, the algorithm terminates in failure.

There are some set and variable definitions used in the Path Searching Procedure (PSP):

- Wg_j – the set of indices of the worker groups that can operate mg_j.
- Mg_i – the set of indices of the machine groups that worker from wg_i can operate.
- $|Wg_j|$ – the number of worker groups in Wg_j.
- $|Mg_i|$ – the number of machine groups in Mg_i.
- J – the set of indices of all target machine groups that are in the final path.
- K – the set of indices of all worker groups that the final path goes through corresponding to the machine groups in J.
- U – the set of indices of the worker groups that the algorithm chose in Wg_i's corresponding to all mg_i's in J.
- V – the set of indices of the machine groups that the algorithm chose in Mg_i's corresponding to all wg_i's in K.
- G – the set of indices of target machine groups the algorithm has already found that the source doesn't have a worker to move to.

We present the PSP algorithm used for coding as Algorithm 2.

Algorithm 2 can be used to realize reassignment requests. We use the worker allocation matrix of Example 1 to illustrate how Algorithm 2 works. Table 5 shows that some of the

Algorithm 2 (PSP) Finding Path and Tracing Worker Movements for [p, q].

Part 1:

1. Set $j=q$, $J_1=q$, $m=1$, $n=1$.
2. Set $u=1$, $v=1$.
   Find out if mg_p has worker of a worker group in Wg_j.
   If yes, say worker of $wg_{k'} \in Wg_j$
   $J_{n+1}=p$, $K_n=k'$. **DONE, GO TO Part 2;**
   Else
   $G_m=j$, GO TO 3.
3. Choose the u^{th} worker group in Wg_j, say wg_k, search for the v^{th} machine group in Mg_k other than mg_j that has worker.
   If found, say $mg_{j'}$
   If j' is not in G
   $j=j'$, $J_{n+1}=j'$, $K_n=k$, $U_n=u$, $V_n=v$, $m=m+1$, $n=n+1$, GO TO 2;
   Else if $v<|Mg_k|$
   $v=v+1$, GO TO 3;
   Else if $u<|Wg_j|$
   $u=u+1$, $v=1$, GO TO 3;
   Else if $n>1$
   GO TO 4;
   Else
   THE ALGORITHM FAILED TO ACCOMPLISH [*p, q*].
4. $n=n-1$, $J_{n+1}=0$, $j=J_n$, $k=K_n$.
   If $V_n<|Mg_k|$
   $v=V_n+1$, $u=U_n$, GO TO 3;
   else if $U_n<|Wg_j|$
   $u=U_n+1$, $v=1$, GO TO 3;
   else if $n>1$
   GO TO 4;
   else
   THE ALGORITHM FAILED TO ACCOMPLISH [*p, q*].

Part 2:

For $i=1$ to n
$j_1=J_{i+1}$, $j_2=J_i$, $k=K_i$,
$a_{k,j1}=a_{k,j1}-1$, $a_{k,j2}=a_{k,j2}+1$.

reassignment requests that potentially achieve the desired worker allocation are [1, 3], [4, 3], [4, 5], etc. We only discuss [1, 3] because the PSP works similarly on any other reassignment requests. The step-by-step process that the PSP uses to realize a worker reassignment request [1, 3] is described as follows and illustrated in Table 6 and Table 7.

Table 6 Achieve Reassignment Request [1, 3] by the PSP – The Backward Path

$wg_i \backslash mg_j$	1	2	3	4	5	π_i
1	1 ←	2	X	X	2	5
2	X	3 ←	X	2	X	5
3	X	X	3 →	2	X	5
4	3	X	X	X	2	5
5	X	1	2	2	X	5
$w_j(\vartheta)$	4	6	5	6	4	

Table 7 Achieve Reassignment Request [1, 3] by the PSP – Result Matrix

$wg_i \backslash mg_j$	1	2	3	4	5	π_i
1	0	3	X	X	2	5
2	X	2	X	3	X	5
3	X	X	4	1	X	5
4	3	X	X	X	2	5
5	X	1	2	2	X	5
$w_j(\vartheta)$	3	6	6	6	4	

p=1, q=3.

Part 1:

1. Set $j=J_1=q=3$. m=1, n=1.

2. Set u=1, v=1. Wg_j={3, 5}. mg_p does not have a worker in wg_3 or wg_5. $G_m=j$.

3. The u^{th} worker group in Wg_j is wg_3. k=3. Mg_k={3, 4}. The v^{th} machine group in Mg_k other than mg_j that has worker is mg_4, j'=4. 4 is not in set G, so j= j'=4, J_{n+1}= j', K_n=k, U_n=u, V_n=v, m=m+1=2, n=n+1=2.
4. Set u=1, v=1. Wg_j={2, 3, 5}. mg_p does not have a worker in wg_2, wg_3 or wg_5. G_m=j.
5. The u^{th} worker group in Wg_j is wg_2. k=2. Mg_k={2, 4}. The v^{th} machine group in Mg_k other than mg_j that has worker is mg_2, j'=2. 2 is not in set G, so j= j'=2, J_{n+1}= j', K_n=k, U_n=u, V_n=v, m=m+1=3, n=n+1=3.
6. Set u=1, v=1. Wg_j={1, 2, 5}. mg_p has a worker in wg_1, k'=1. J_{n+1}=p, K_n= k'.

Part 2:

$a_{3,4}$=$a_{3,4}$-1=1, $a_{3,3}$=$a_{3,3}$+1=4; $a_{2,2}$=$a_{2,2}$-1=2, $a_{2,4}$=$a_{2,4}$+1=3; $a_{1,1}$=$a_{1,1}$-1=0, $a_{1,2}$=$a_{1,2}$+1=3.

3.3.3 Properties of the Path Searching Procedure

Proposition 1: Given that at least one path exists for a given reassignment request, it is impossible that Algorithm 2 terminates in failure.

Proof. The Path Searching Procedure actually consists of a series of sub-problems. Every time the Algorithm sets a machine group as target, it starts to look at a new sub-problem. Assume the reassignment request is [p, q]. When the algorithm makes machine group j as the target, it is actually dealing with [p, j] with a worker allocation matrix excluding machine groups in set G.

Define Sub(n) as a formal sub-problem which represents any sub-problems with n machine groups not in G yet. The PSP starts from Sub(S). No matter what is the associated target, Sub(n) only has three possible outcomes:

- Searching DONE, if the source has proper worker to donate to the target
- Sub(n-1), if a machine group other than the source that is not in G has proper worker to donate to the target, the target will be put into G
- Sub(n), if none of the machine groups that are not in G has proper worker to donate to the target, the previous target will be set as target

Then we use mathematical induction to prove Proposition 1 is true.

Step 1: Start from n=1, Sub(1).

When there is only one machine group not in G and the algorithm did not finish in success, all machine groups except the source and the current target have been made as target at least once before, and the source does not have proper worker to donate to any of those targets. On the other hand, a path from the current target to the objective has been found. And this last "unvisited" machine group must be the source. If there is a path for this reassignment request, the source must have proper worker to donate to at least one of the other machine groups. In summary, the source must have a worker to donate to the current target. This means the algorithm successfully find a path for the request. The above analysis proves the base: Proposition 1 is true for n=1.

Step 2: Assume Proposition 1 is true when n=k. When there are k machine groups not in G, if there is a path, Algorithm 2 will find one. PSP at Sub(k) will always find a path if one exists.

Step 3: Prove that Proposition 1 is still true when n=k+1, Sub(k+1).

Now the statement to prove is: if there is a path, Sub(k+1) will either finish in success, or go to Sub(k). Since the first two possible outcomes of Sub(k+1) satisfy the statement, we

only need to show that in the third case, Sub(k+1) finally will still result in Sub(k) if there is a path.

In the third case, the source does not have proper worker to donate to any of the m-k-1 machine groups including the current target and the objective in G. None of the machine groups not in G have a proper worker to donate to the current target. Then Sub(k+1) stays as Sub(k+1) with the previous target. If there is a path, there must be at least one machine group not in G that has a proper worker to donate to some of the machine groups in G. After several steps, the PSP at Sub(k+1) with some previous target will find the machine group not in G that has a proper worker to donate to the target because the PSP searches every machine group not in G if needed. And this machine group not in G, once found, can be made as a new target because a path has been found from any machine group in G to the objective. Then Sub(k+1) changes to Sub(k). So we proved that if a path exists, Sub(k+1) will either finish in success, or result in Sub(k).

All in all, we proved that Proposition 1 is always true for any n=1 to S.

Proposition 1 discusses the property of the PSP in one single iteration, where the PSP deals with a certain reassignment request. It is easy to see that given a feasible allocation ϑ, if after reassignment [p, q] the resulted allocation ϑ' is also feasible, there is at least one path exist for reassignment request [p, q]. Therefore, Proposition 1 guarantees that the PSP will find a path for a reassignment request if the worker allocation after the reassignment is feasible. This proposition also implies the following Lemma when the algorithm is applied for multiple iterations.

Lemma: If the PSP can achieve $[p_{n+1}, q_{n+1}]$ when Path A was selected for $[p_n, q_n]$, it can still achieve $[p_{n+1}, q_{n+1}]$ when Path B was selected for $[p_n, q_n]$; and vice versa.

In the statement, $[p_k, q_k]$ represents the reassignment request for iteration k, and Path A and Path B are two different possible paths in iteration n.

Proof. Before iteration n, Path A and Path B both exist for $[p_n, q_n]$. If the PSP can achieve $[p_{n+1}, q_{n+1}]$ after Path A was selected in iteration n, i.e. there is a path for $[p_{n+1}, q_{n+1}]$ in iteration n+1, after reassignment $[p_{n+1}, q_{n+1}]$ in iteration n+1, the resulted allocation is proved to be feasible. When Path B was selected in iteration n, the reassignment request is still $[p_{n+1}, q_{n+1}]$ in iteration n+1, and we know that the allocation after this reassignment is feasible. Therefore, the PSP can still find a path for $[p_{n+1}, q_{n+1}]$.

If the PSP cannot achieve $[p_{n+1}, q_{n+1}]$ after Path A was selected in iteration n, the allocation after reassignment $[p_{n+1}, q_{n+1}]$ will be infeasible. Therefore, even if Path B was selected in iteration n, the PSP will still work on a request that leads to an infeasible allocation, which will terminate in failure.

Proposition 2: If the PSP can achieve $[p_{n+k}, q_{n+k}]$ when Path A was selected for $[p_n, q_n]$, it can also achieve $[p_{n+k}, q_{n+k}]$ when Path B was selected for $[p_n, q_n]$ (k=1, 2 …); and vice versa.

Proof. If each one of the requests $[p_{n+1}, q_{n+1}]$, $[p_{n+2}, q_{n+2}]$, $[p_{n+3}, q_{n+3}]$, …, $[p_{n+k}, q_{n+k}]$ can be achieved by the PSP in order when Path A was selected for $[p_n, q_n]$. According to the Lemma, if Path B was selected, the PSP can also achieve $[p_{n+1}, q_{n+1}]$. After each iteration, since the resultant allocations are all known to be feasible, there exists a path for each request; and no matter which path is selected in each iteration, the PSP can always achieve the next request. Finally, the PSP will be able to achieve $[p_{n+k}, q_{n+k}]$.

If the PSP cannot achieve $[p_{n+k}, q_{n+k}]$ when Path A was selected for $[p_n, q_n]$, without loss of generality, we assume $[p_{n+k}, q_{n+k}]$ is the first reassignment request after $[p_n, q_n]$ that is not achievable. Then we know that the allocation after reassignment $[p_{n+k}, q_{n+k}]$ is infeasible. Therefore, no matter which path was selected in iteration n, the PSP cannot achieve $[p_{n+k}, q_{n+k}]$ by any means.

Proposition 2 guarantees that when the PSP works on any given reassignment request in any given iteration, whichever path is used will not affect the achievability of any other reassignment request in the future iterations. This important property supports the methodology that the algorithm will stop once it finds a path and achieve the request.

3.3.4 Solve the AFD Problem with the PSP

Continuing the discussion at the beginning of Section 3.3, the method developed to solve an AFD problem can be summarized as follows. For any allocation feasibility determination problem, after the worker allocation matrix is initialized, there may be a difference between the worker allocation associated with the initial worker allocation matrix and the worker allocation being tested. Reassignment requests that may eliminate the difference are arbitrarily decided. Then the PSP is used to justify and achieve those requests; one request per iteration. If some of the requests cannot be achieved, the PSP will continue to try other possible requests until the difference is eliminated or no other reassignment request that could contribute to the elimination can be achieved. If the difference is finally eliminated, the worker allocation being examined is proved to be feasible in the corresponding APLF situation. Otherwise, this allocation is infeasible in this situation. We name this new method as the PSP Method.

Given the initial worker allocation matrix in Table 5, we can solve the AFD problem for allocation (3, 6, 7, 4, 5) by 3 iterations of the PSP. We arbitrarily make [1, 3], [4, 3], [4, 5] as the reassignment requests for each iteration respectively. We already have the worker allocation matrix after [1, 3] achieved shown in Table 7. Now we present the matrix after the second and the third iteration as in Table 8 and Table 9.

Notably, the PSP Method took only 3 iterations to solve Example 1, which is less than the transportation method (6), and the out-of-kilter algorithm (8).

Table 8 Worker Allocation Matrix after Reassignment Request [4, 3] Achieved by the PSP

$wg_i \backslash mg_j$	1	2	3	4	5	π_i
1	0	3	X	X	2	5
2	X	2	X	3	X	5
3	X	X	5	0	X	5
4	3	X	X	X	2	5
5	X	1	2	2	X	5
$w_j(\vartheta)$	3	6	7	5	4	

Table 9 Worker Allocation Matrix after Reassignment Request [4, 5] Achieved by the PSP

$wg_i \backslash mg_j$	1	2	3	4	5	π_i
1	0	2	X	X	3	5
2	X	3	X	2	X	5
3	X	X	5	0	X	5
4	3	X	X	X	2	5
5	X	1	2	2	X	5
$w_j(\vartheta)$	3	6	7	4	5	

3.4 The Lower Bound Search Algorithm in an APLF situation

The reason for us to introduce and develop the allocation feasibility determination methods, as discussed at the beginning of this Chapter, is that we want to make the LBSA work in the APLF situation. The PSP Method and the out-of-kilter algorithm have an obvious advantage over the transportation method when embedded into the LBSA. In the LBSA, the only one worker allocation that needs to be tested as a complete AFD problem is the initial allocation ϑ^0. And this AFD problem only needs to be solved once. Most of the time, the LBSA is trying to reassign one worker from one machine group to another machine group based on a known feasible allocation. The PSP and the out-of-kilter algorithm could directly test the feasibility of the reassignment and realize it in a single iteration, while the transportation method will have to solve complete problems. This is because the worker allocation matrix (the network flow) will be updated after each iteration of the PSP (the out-of-kilter algorithm), thus every time the LBSA proposes a worker reassignment, the PSP (the out-of-kilter algorithm) only needs to run one iteration based on the worker allocation matrix (the network flow) obtained after the previous reassignment to decide if the reassignment request (the new flow) is achievable (feasible). In this research, the PSP Method is used as the allocation feasibility determination method for the LBSA. An experiment that compared the running times of the PSP Method and the out-of-kilter algorithm is discussed in Chapter 4, Section 4.4.

The LBSA for the APLF situation using the PSP Method as the allocation feasibility determination method is presented as Algorithm 3. The initial worker allocation selected by Lobo (2011) will be tested first. If this allocation is feasible, the LBSA will continue with it.

Algorithm 3 The Lower Bound Search Algorithm in Absolute Partial Labor Flexibility Situation Using the Path Searching Procedure to Determine Allocation Feasibility.

Set $\lambda = 0$ TC=2
Determine an initial allocationϑ^0.
Solve AFD(ϑ^0). Update ϑ^0.
1.
Determine $\mathrm{LB}_{\vartheta^\lambda} = \max_{1\le l\le S}\left\{\mathrm{LB}_{\mathrm{mg}_l}\left(w_l\left(\vartheta^\lambda\right)\right)\right\}$
Find $K = \mathrm{argmax}_{1\le l\le S}\left\{\mathrm{LB}_{\mathrm{mg}_l}\left(w_l\left(\vartheta^\lambda\right)\right)\right\}$
If $w_k\left(\vartheta^\lambda\right) = M_{\mathrm{mg}_k}$ for any $k \in K$
TC=1 STOP. ϑ^λ MINIMIZES LB($L_{\max}$) FOR THE JOB SHOP.
Else
Choose $k \in K$.
Set $F = K$
For $z \in \{1, \dots, S\}\backslash F$
If $w_z\left(\vartheta^\lambda\right) = 1$
$F \leftarrow F \cup \{z\}$
2.
If $\{1, \dots, S\}\backslash F \neq \emptyset$
Find $J = \mathrm{argmin}_{1\le l\le S, l\notin F}\left\{\mathrm{LB}_{\mathrm{mg}_l}\left(w_l\left(\vartheta^\lambda\right)\right)\right\}$
Choose $j \in J$
If $\mathrm{LB}_{\mathrm{mg}_j}\left(w_j\left(\vartheta^\lambda\right)\right) > \mathrm{LB}_{\mathrm{mg}_k}\left(w_k\left(\vartheta^\lambda\right)\right)$
$F \leftarrow F \cup \{j\}$, GO TO 2.
Else
$\vartheta^{\lambda+1} = \vartheta^\lambda$
$\vartheta^{\lambda+1}(j) = \vartheta^\lambda(j) - 1$
$\vartheta^{\lambda+1}(k) = \vartheta^\lambda(k) + 1$
If $\vartheta^{\lambda+1} \in \{\vartheta^0, \dots, \vartheta^\lambda\}$
$F \leftarrow F \cup \{j\}$, GO TO 2.
Else
If PSP(j,k) succeeds /*using the Path Searching Procedure to realize [j, k]
$\lambda = \lambda + 1$, GO TO 1.
Else
$F \leftarrow F \cup \{j\}$, GO TO 2.
Else
STOP. ϑ^λ MINIMIZES LB($L_{\max}$) FOR THE JOB SHOP.

If this allocation is infeasible, according to the discussion in Section 3.3.4, the PSP Method must terminate at an allocation that is closest to the initial allocation. We name this allocation as the alternative initial allocation. Because of the optimality of the LBSA, we can use the alternative initial allocation to start Algorithm 3. As in the LBSA, if any one of the constraining machine groups is already fully staffed, the termination CONDITION 1 will be satisfied. In Part 2 of Algorithm 3, the PSP is used to check if the proposed worker reassignment request [*j*, *k*] is achievable. If a request is achieved, the allocation will be updated and Algorithm 3 moves on to the next iteration. If a request cannot be achieved, the source *j* will be put into the set of failure; and Algorithm 3 searches for other reassignments. As in the original LBSA, when all machine groups other than the constraining machine group are in the failure set, termination CONDITION 2 will be satisfied.

Proposition 3: Algorithm 3 finds a feasible allocation ϑ^* that satisfies $\mathrm{LB}_{\vartheta^*} \leq \mathrm{LB}_{\vartheta}$ for all feasible allocation ϑ.

Proof. We can prove Proposition 3 by extending the proof conducted by Lobo (2011) on the proposition that "the Lower Bound Search Algorithm finds an allocation ϑ^* that satisfies $\mathrm{LB}_{\vartheta^*} \leq \mathrm{LB}_{\vartheta}$ for all allocation ϑ". Algorithm 3 terminates in allocation ϑ^* only if termination CONDITION 1 or CONDITION 2 is satisfied. If Algorithm 3 terminated because CONDITION 1 was satisfied, the lower bound on $L_{\max}$ of the selected constraining machine group is already as small as possible. Obviously, there is no other allocation that yields a better lower bound. If Algorithm 3 terminated because CONDITION 2 was satisfied, for any selected constraining machine group *k*, any worker reassignment [*j*, *k*] will satisfy at least one of the following conditions:

a) There is only one worker assigned to machine group j;

b) The lower bound on $L_{\max}$ of machine group j after one worker unassigned from it is larger than the current lower bound on $L_{\max}$ of machine group k;

c) Machine group j is a constraining machine group itself;

d) Reassignment request [j, k] cannot be achieved by the PSP.

Lobo (2011) already proved that it is impossible to find an allocation Ω that yields a smaller lower bound if any [j, k] satisfied condition a), b), and c). If the only machine groups that do not satisfy condition a), b), or c) satisfy condition d), a feasible allocation Ω that yields a smaller lower bound still cannot exist, because all allocations that yield smaller lower bounds, the allocations after those reassignments, are proved to be infeasible in the given APLF situation. Therefore, allocation ϑ^* yields the smallest lower bound on $L_{\max}$ in all feasible allocations.

Proposition 3 implies that in the absolute partial labor flexibility problem settings, Algorithm 3 can be used to optimize the lower bound on $L_{\max}$ of the job shop. It is worth pointing out that Algorithm 3 though finds the allocation which yields the optimal lower bound on $L_{\max}$, the schedule generated by the Virtual Factory from that allocation may or may not be the best schedule in terms of the actual $L_{\max}$ value.

Chapter 4 Computational Experiment

4.1 Experimental Design

Algorithm 2 and Algorithm 3 are coded in C++ and embedded in the Virtual Factory. The goal of this experiment is to test the difference in the lower bound on $L_{\max}$ achievable when there is not complete flexibility. Because this experiment is an extension of the experiment of testing the performance of the lower bound by Lobo (2011), the experimental job shop is the same with some additions. There are 80 machines and 10 machine groups in the job shop, where each machine group has 8 machines. The total number of jobs processed is 1200. Two different job shop types (symmetric and asymmetric) are used. Eight different due date ranges are tested: from 200 to 3000, with an increment of 400. Seven different staffing levels are applied: 30% (24), 40% (32), 50% (40), 60% (48), 70% (56), 80% (64), and 90% (72). All processing times are discrete random variables ranged from 1 to 40 with a uniform distribution. The number of operations of each job ranging from 6 to10, and at most 3 operations are in one machine group.

There are 10 worker groups; the skill matrix is a 10×10 matrix. And we assume there is at least one worker in each group. The number of workers in each worker group is assumed to be equal or as close to equal as possible. For example, if the total number of workers is 56, the number of workers in each worker group will be as (6, 5, 6, 5, 6, 6, 5, 6, 5, 6) or another array that only contains 5 and 6. We use the flexibility metric (*FM*) introduced by Daniels *et al*. (2004) to measure the degree of labor flexibility of the job shop. Job shop performance is tested for 3 different *FM*s: 0.2, 0.3, and 0.4. The skill matrices corresponding to each *FM* are

generated following two patterns. One pattern is the *k*-chain structure, i.e. *k* equals 3 when *FM* equals 0.3 in this case. The other pattern is the "irregular" structure which is arbitrarily designed for comparison test. The irregular structure is defined as any skill structure that is not following a known pattern and is close to a random 1/0 matrix. In the real world, different job shops could have different irregular skill structures depending on how the job shop is configured and what types of workers are involved in the job shop. However, our experiment cannot cover all the possibilities though the methodologies developed in the previous chapters are capable of solving the problem with all possible skill structures. To generate a set of irregular structures in a relatively consistent way, we can set all elements of the skill matrix to "1"s first and then change some of the "1"s to "0"s in some structured way. The procedure that is used in this experiment to generate the irregular structure is illustrated in Appendix A. Both two types of skill matrix will be tested and compared to the job shop performance with complete labor flexibility. Therefore, in total, we will test 6 absolute partial labor flexibility (APLF) skill structures and the complete labor flexibility (CLF) structure. All skill matrices used in the experiment are presented in the Appendix B.

According to the type of job shop and due-date range, randomly generated problems are defined. For each problem type with each staffing level, each *FM* value, and each type of skill structure, 50 randomly generated problems are solved.

4.2 Experimental Results

The experimental results are summarized in the graphs presented in the following sections. In the first two sets of graphs, the average minimum lower bounds on $L_{\max}$ for the APLF skill structures (LB_{APLF}) are compared with the average minimum lower bounds on $L_{\max}$ for the

CLF situation (LB_{CLF}) in each due-date range at each staffing level. Each line is a plot of LB_{APLF} - LB_{CLF} as a function of the due-date range and represents a specific skill structure as noted in the legend. This value is used to represent the performance of the job shop with a particular combination of job shop type, due-date range, staffing level, *FM* value, and skill structure type. The closer the value is to zero, the better the performance of the job shop is in the certain APLF situation. For each job shop type and each staffing level, only three skill structures, (*FM* 0.2, irregular), (*FM* 0.3, irregular), and (*FM* 0.2, *k*-chain), possibly have nonzero LB_{APLF} - LB_{CLF}. In other words, for all other skill structures, the lower bounds found are always equal to the ones found in CLF situation. Therefore, we only present the performance of these three skill structures in the first two sets of graphs in sections 4.2.1 and 4.2.2. The third set of graphs in Section 4.2.3 look at how the average performance varies by skill structure, where the performance is averaged by the due-date ranges and plotted for each staffing level. All performances are presented in units of the average processing time (20.5).

4.2.1 Performance vs. Due-date Range: Symmetric Job Shop

In the symmetric job shop, each machine group sees, on average, 10% of the work load that passes through the job shop. The graphs illustrating symmetric job shop performances are shown in Figure 2 to Figure 9. Based on these graphs, we can make the following observations.

Generally, the performance is getting better when the staffing level increases, i.e. the higher the staffing level is, the smaller the average value of LB_{APLF}-LB_{CLF} is for each skill structure. However, close scrutiny shows that this positive correlation between the performance and the staffing level also has its volatility. From a 30% staffing level to a 50%

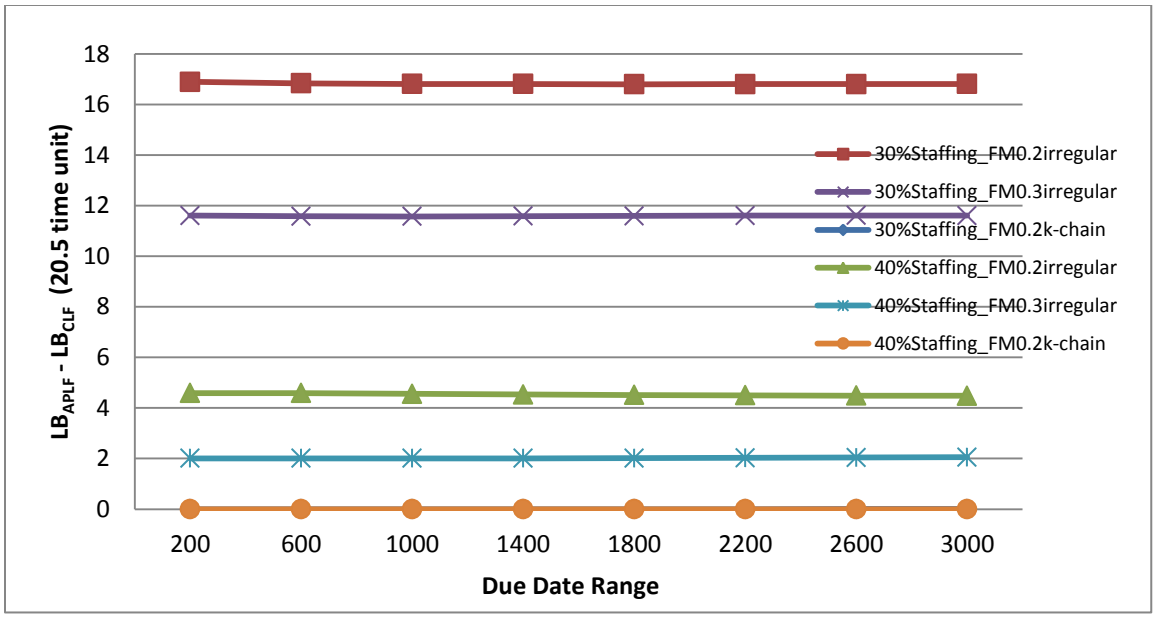

Figure 2 Performance of Absolute Partial Labor Flexibility to Complete Labor Flexibility, Symmetric Job Shop, 80 Machines, 30% and 40% Staffing

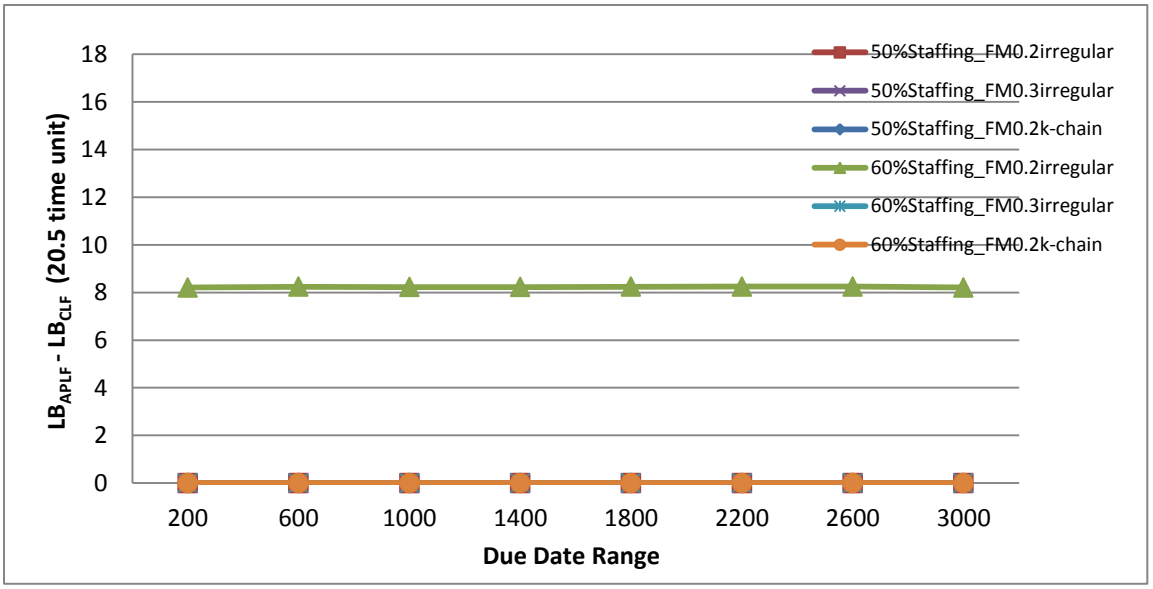

Figure 3 Performance of Absolute Partial Labor Flexibility to Complete Labor Flexibility, Symmetric Job Shop, 80 Machines, 50% and 60% Staffing

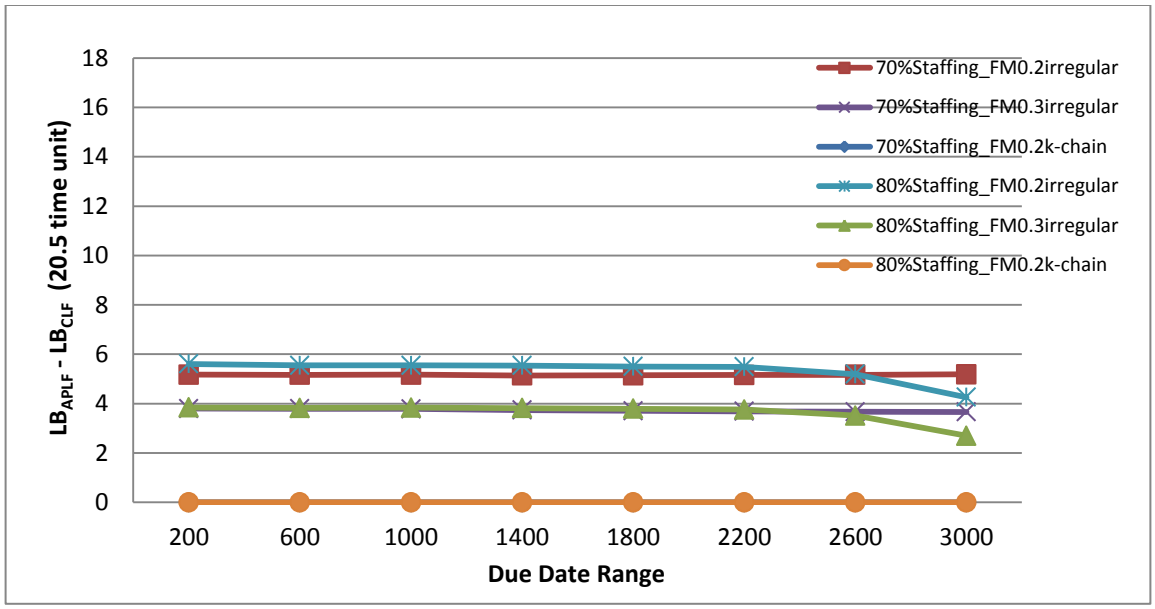

Figure 4 Performance of Absolute Partial Labor Flexibility to Complete Labor Flexibility, Symmetric Job Shop, 80 Machines, 70% and 80% Staffing

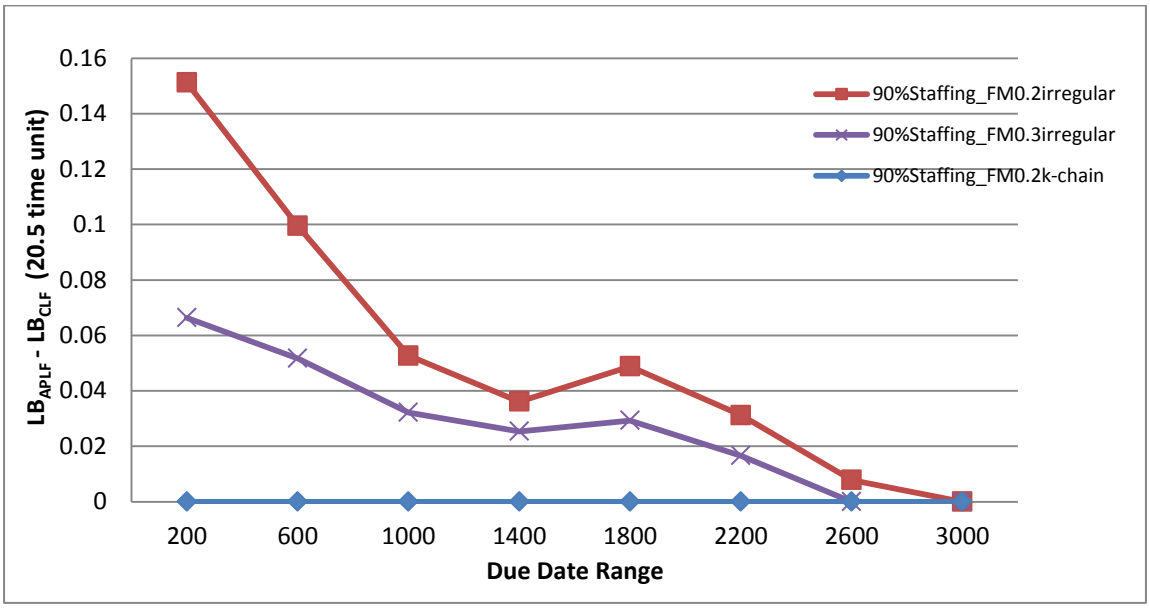

Figure 5 Performance of Absolute Partial Labor Flexibility to Complete Labor Flexibility, Symmetric Job Shop, 80 Machines, 90% Staffing, Micro-scaled

staffing level, the average value of LB_{APLF} - LB_{CLF} decreases as the staffing level increases. At the 50% staffing level, all average difference values equal to zero. But start from the 60% staffing level, the average value of LB_{APLF} - LB_{CLF} for the (*FM* 0.2, irregular) structure jumped to a value higher than the one with the same structure at the 40% staffing level; and the average value of LB_{APLF} - LB_{CLF} for the (*FM* 0.3, irregular) structure jumped at the 70% staffing level. At the 80% staffing level, the values of the two structures jumped again. At all but 80% staffing level, for each skill structure, LB_{APLF} - LB_{CLF} is almost stable as the due-date range changes. At the 80% staffing level, there is a significant decreasing trend from a due-date range of 2600 to 3000. At the 90% staffing level, for all skill structures, LB_{APLF} - LB_{CLF} is equal to or very close to zero. But a micro-scaled graph of the 90% staffing level shows that the performance is getting better as the due-date range increases.

4.2.2 Performance vs. Due-date Range: Asymmetric Job Shop

In the asymmetric job shop, the average work load routed through each machine group is no longer equal. Without loss of generality, machine groups 1 through 4 have greater probabilities of a job being routed through them, and these probabilities are summarized in Table 10. We can see the performances of asymmetric job shop in Figure 10 to Figure 17.

Table 10 Probability of a Job Being Routed through a Machine Group

Machine Group	1	2	3	4	5	6	7	8	9	10
Probability	0.14	0.14	0.14	0.10	0.8	0.8	0.8	0.8	0.8	0.8

The scale of the value of LB_{APLF} - LB_{CLF} is more than 10 times greater than that for the symmetric job shop. This tells us that given the skill structures we used, the performance of

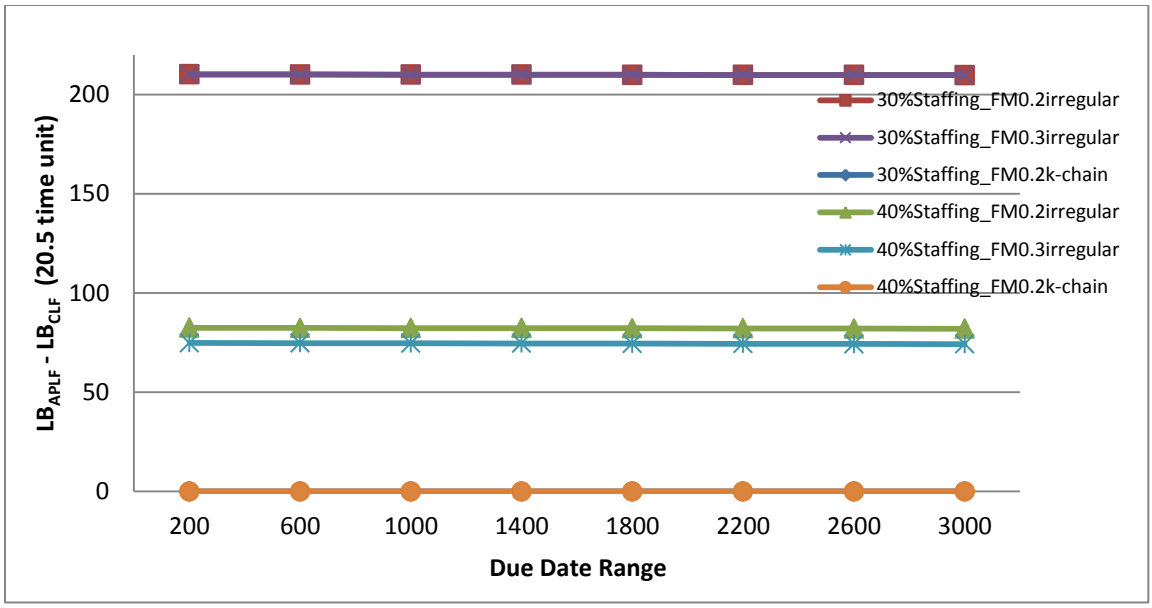

Figure 6 Performance of Absolute Partial Labor Flexibility to Complete Labor Flexibility, Asymmetric Job Shop, 80 Machines, 30% and 40% Staffing

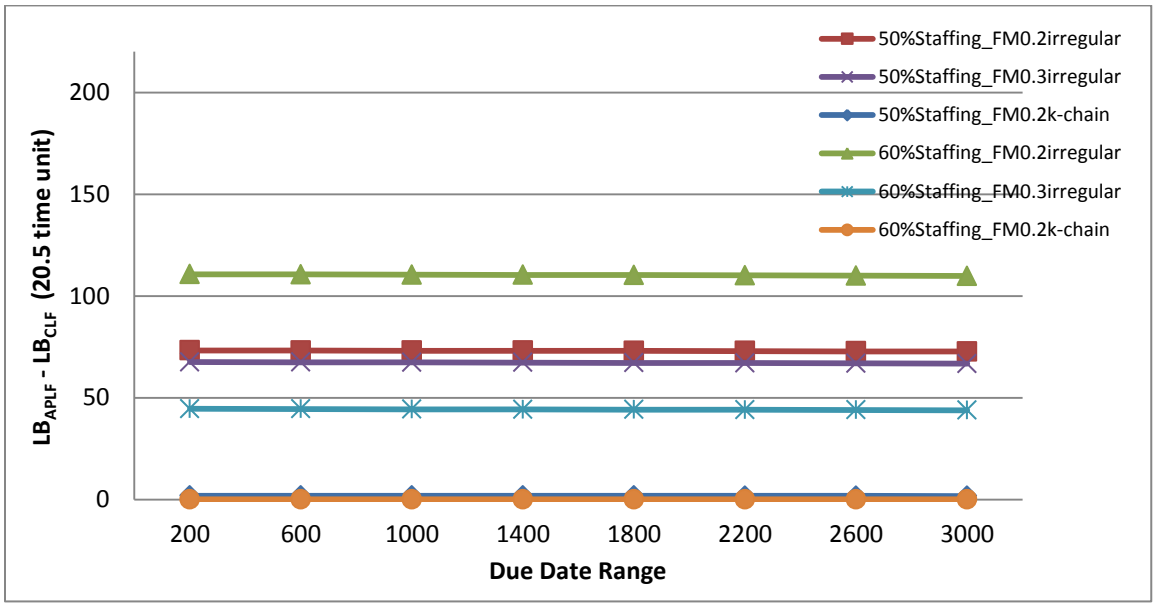

Figure 7 Performance of Absolute Partial Labor Flexibility to Complete Labor Flexibility, Asymmetric Job Shop, 80 Machines, 50% and 60% Staffing

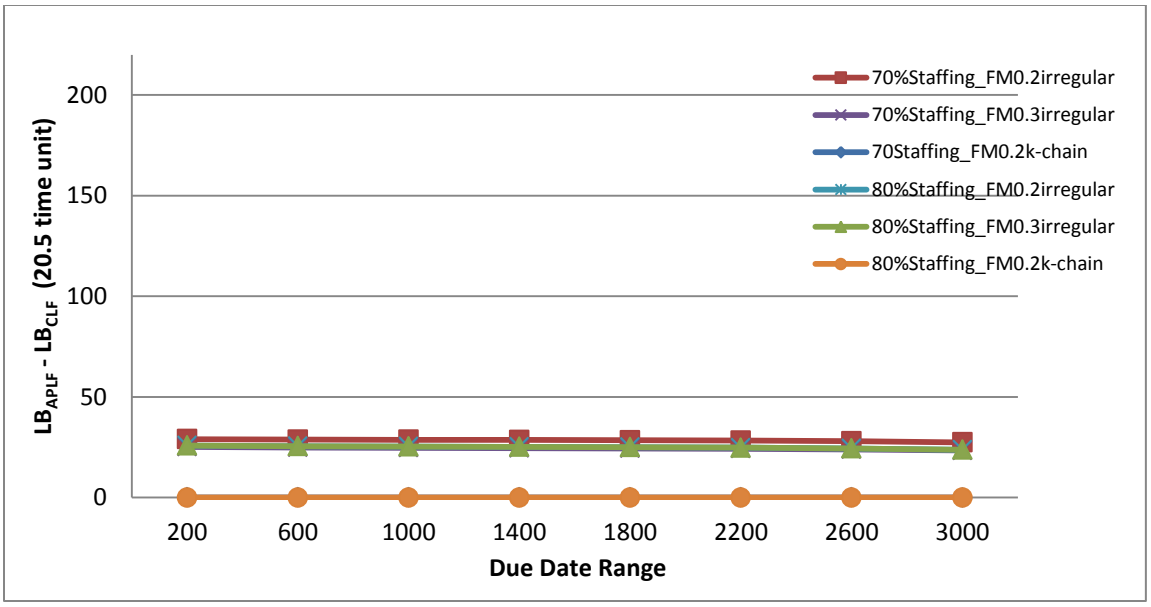

Figure 8 Performance of Absolute Partial Labor Flexibility to Complete Labor Flexibility, Asymmetric Job Shop, 80 Machines, 70% and 80% Staffing

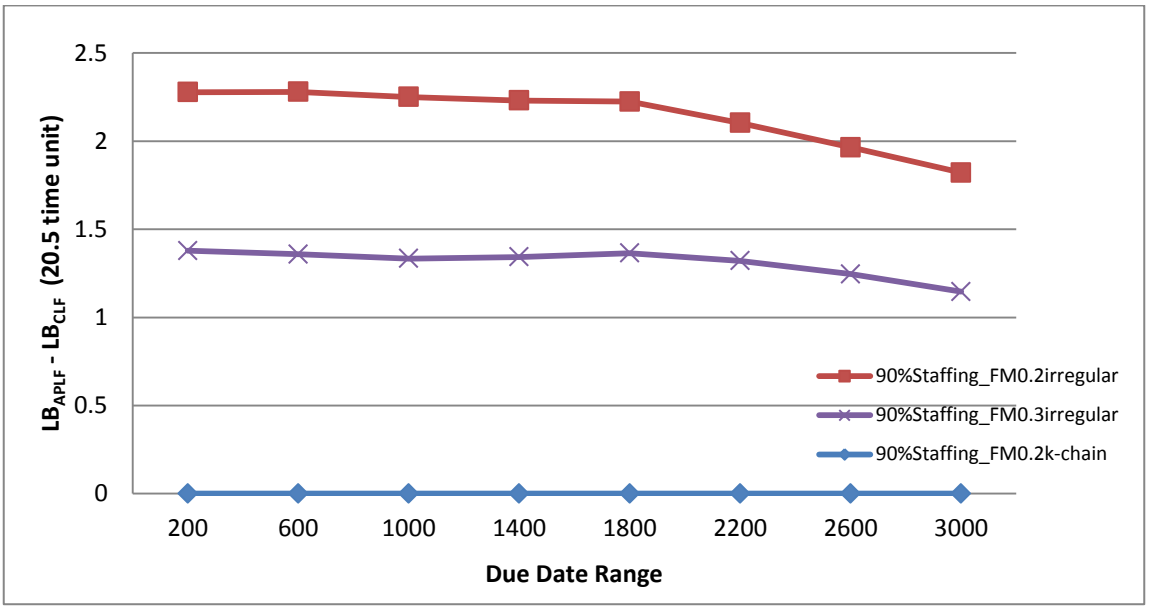

Figure 9 Performance of Absolute Partial Labor Flexibility to Complete Labor Flexibility, Asymmetric Job Shop, 80 Machines, 90% Staffing, Micro-scaled

the asymmetric job shop in APLF situation is much worse than the one of the symmetric job shop. The biggest difference is over 200 times as large as the unit processing time (30% staffing). Generally, the average performance is getting better when the staffing level increases, i.e. the higher the staffing level is, the smaller the average value of LB_{APLF}-LB_{CLF} is for each skill structure. However, close scrutiny shows that this positive correlation also has its volatility. From a 30% staffing level to a 50% staffing level, the average value of LB_{APLF} - LB_{CLF} decreases as the staffing level increases. At the 60% staffing level, the average value of LB_{APLF} - LB_{CLF} for the (*FM* 0.2, irregular) structure jumped to a value higher than the one with the same structure at the 40% staffing level; and the average difference for the (*FM* 0.3, irregular) structure also slightly jumped at the 80% staffing level. At the 90% staffing level, for all skill structures, the average values of LB_{APLF} - LB_{CLF} are equal to or very close to zero. In all staffing levels, for each skill structure, LB_{APLF} - LB_{CLF} is almost stable as the due-date range changes. A micro-scaled graph of the 90% staffing level shows that there is a slight trend that the performance is getting better as the due-date range increases.

4.2.3 Performance vs. Skill Structure

Since for both the symmetric job shop and the asymmetric job shop, LB_{APLF} - LB_{CLF} is stable as the due-date range changes for a given staffing level and a given skill structure, we can use the mean value of LB_{APLF} - LB_{CLF} to represent the average performance of the job shop with each combination of staffing level and skill structure. Then all staffing levels can be drawn in one graph. Figure 18 and Figure 19 show the plots of the average performance as a function of the change of the skill structure, in which each line represents a staffing level. The skill

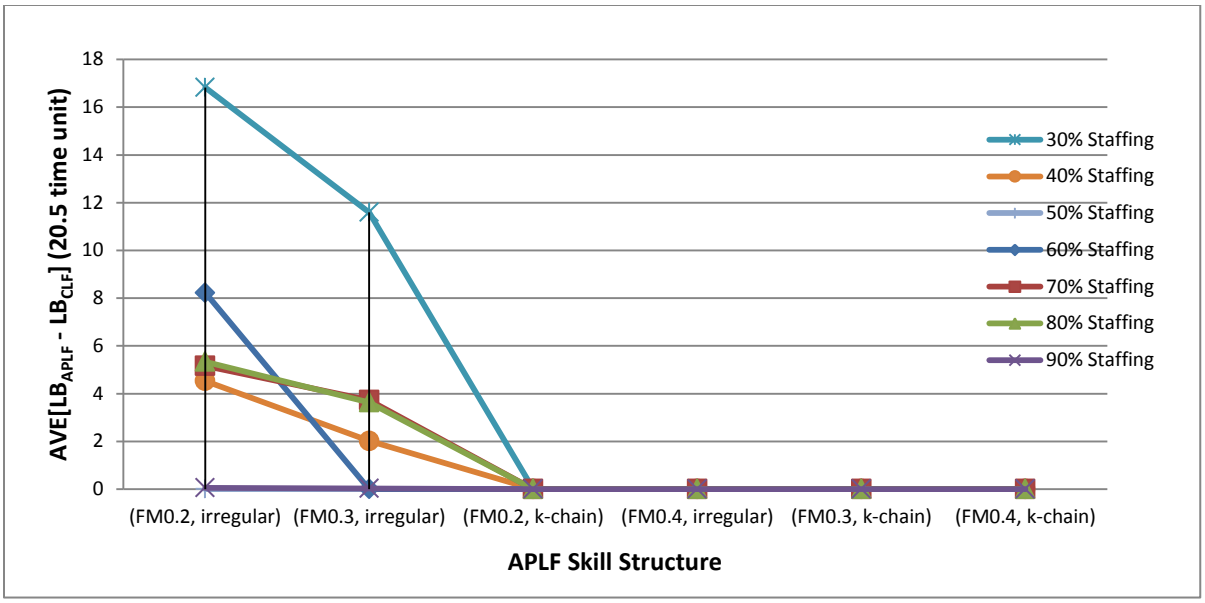

Figure 10 Average Performance of Absolute Partial Labor Flexibility to Complete Labor Flexibility over Due-Dates, Symmetric Job Shop, 80 Machines

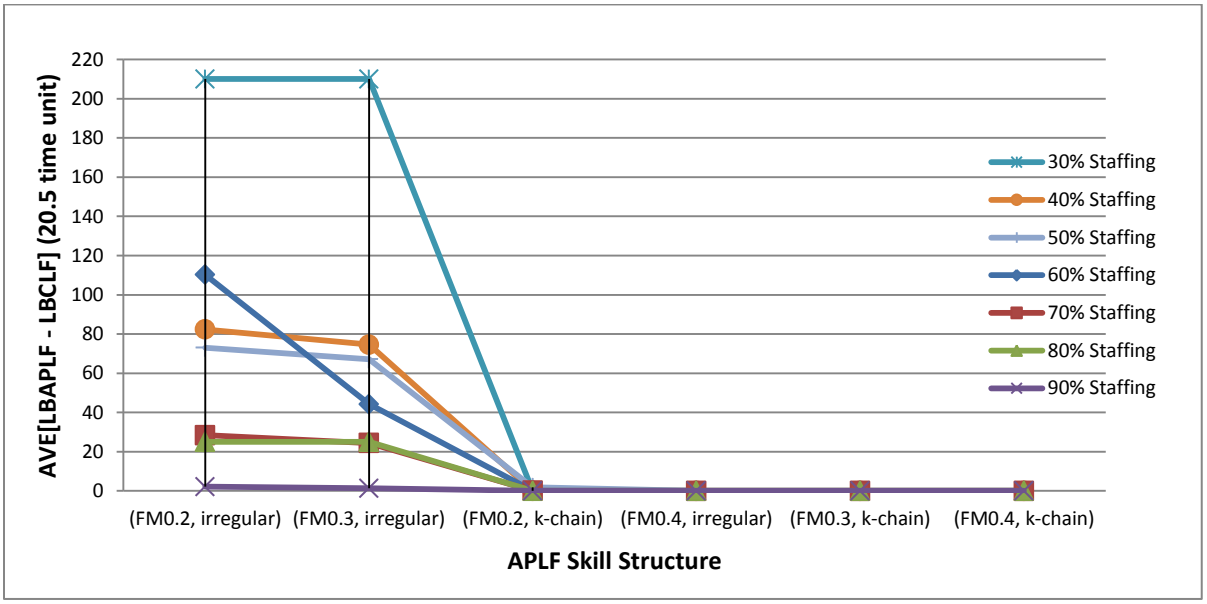

Figure 11 Average Performance of Absolute Partial Labor Flexibility to Complete Labor Flexibility over Due-Dates, Asymmetric Job Shop, 80 Machines

structures are ordered as the performance of flexibility improves in this experiment. We can see that for all staffing levels, the job shops with a skill structure (*FM* 0.4, irregular), (*FM* 0.3, *k*-chain), and (*FM* 0.4, *k*-chain) have the same performance as a complete labor flexibility job shop; the job shop with (*FM* 0.3, irregular) has better performance than the one with (*FM* 0.2, irregular), while it has worse performance than the one with (*FM* 0.2, *k*-chain).

4.3 Overall Results and Discussion

Skill structure is a crucial factor impacting the performance of the lower bound found in the APLF problem settings. When the *FM* is greater than or equal to 0.4, performance is as good as those of the CLF situations. When the FM is equal to 0.3 or 0.2, the performance of the *k*-chain structure is much better than the performance of the random structure. In particular, the performance of the (*FM* 0.3, *k*-chain) is as good as that of the CLF situation; and the performance of the (*FM* 0.2, *k*-chain) is much better than that of the (*FM* 0.3, irregular). Possible reasons for this result are the followings. The number of machine groups that the workers in each worker group can be assigned to is 2 (3) in both two types of skill structures with *FM* = 0.2 (0.3). Each machine group can have workers in 2 (3) worker groups assigned to it with *k*-chain structures. However, with irregular structures, though the mean number of worker groups in which workers can be assigned to a given machine group is also 2 (3), some of the machine groups can only have workers in 1 worker group assigned to it. These machine groups dramatically reduce the overall labor flexibility of the job shop. Thus the value of *FM* cannot fully represent the flexibility level of a skill structure.

The job shop type is also a significant factor. On average, the performance of the asymmetric job shop with skill structures (*FM* 0.2, irregular) and (*FM* 0.3, irregular) is over

10 times worse than the performance of the symmetric job shop. However, this large difference may not be a general characteristic but caused by a coincidence. In the asymmetric job shop generated, machine group 2 and machine group 3 each have a high probability (0.14) of having jobs routed through it. But in the (*FM* 0.2, irregular) skill matrix, machine group 2 and machine group 3 each can only have workers in 1 worker group assigned to it; in the (*FM* 0.3, irregular) skill matrix, machine group 2 can only have workers in 1 worker group assigned to it, and machine group 3 can only have workers in 2 worker groups assigned to it. These two machine groups will be constraint machine groups frequently. Based on Table 10, it is reasonable to expect that if we rearrange the columns of the skill matrix of these two irregular structures so that the first four machine groups each can have workers in more worker groups (more than $T \times FM$) assigned to it, the average performance will be improved significantly.

The staffing level affects the performance with an unclear pattern. When there are more workers in the job shop, the average lower bound is guaranteed to be smaller. But the experimental results show that sometimes as the staffing level increases, the difference between the lower bound found with some APLF skill structures and the lower bound found in the CLF situation will increase. A possible explanation is that the lower bound found in the CLF situation decreases faster than the lower bound found with those APLF skill structures in those cases.

Finally, the due-date range, though is a crucial factor when testing the performance of the lower bound found by the LBSA to the Virtual Factory $L_{\max}$ value, is almost negligible

when analyzing the performance of the lower bound found in APLF situation to the lower bound found in CLF situation.

In summary, the flexibility performance of a job shop in terms of the lower bound of L_{max} is affected by complex factors. Given a job shop containing 10 machine groups and 10 worker groups, if the average number of machine groups the workers in each worker group can be assigned to is greater than or equal to 4 ($FM \geq 0.4$), the LBSA will find the same optimal lower bound as in the complete labor flexibility situation. On the other hand, if the staffing level is greater than or equal to 90%, it also makes little sense to worry about the effect of the partial flexibility. From a labor training point of view, if the probabilities of jobs being routed through each machine group are known, the skill structures which ensure the machine groups with heavier workloads possibly have workers in more worker groups assigned to them are preferred; if the probabilities of jobs being routed through each machine group are unknown, *k*-chain structure is a good choice.

All discussions above are based on the assumption that the workers are equally distributed among all worker groups. The performance of the job shop without this assumption has not been studied in this thesis.

4.4 Comparison of Running Times

Finally, we tested the computational efficiency of the PSP Method and the out-of-kilter algorithm when used as the allocation feasibility determination method for the LBSA. The running times of the LBSA were recorded and used as the measurement. We conducted the experiment on one arbitrarily selected case: asymmetric job shop, 60% staffing level, 1800

due date range, and the (*FM* 0.2, irregular) skill structure. The same 50 random generated problems were solved using both two methods.

The experiments were run on a Lenovo Think Pad T400 computer with Intel ® Core™ 2 Duo CPU (T9600, 2.80GHz), 4.00 GB RAM, and Windows 7 Professional 64-bit Operating System. The C++ codes were compiled in MS Visio Studio 2010 Ultimate.

The LBSA with both the PSP Method and the out-of-kilter algorithm found the same optimal lower bound on L_{max} for all 50 problems. The statistics of the running time of the LBSA are summarized in Table 11 and a point by point comparison is presented in Figure 12. The PSP Method has a marginally larger standard deviation than the out-of-kilter algorithm. However, it is significantly more efficient in terms of the average LBSA running time.

Table 11 Statistics of the Running Times of the LBSA Using the Two AFD Methods

AFD Method	Mean (s)	Standard Deviation (s)
PSP Method	74.66	14.28
out-of-kilter	91.64	13.13

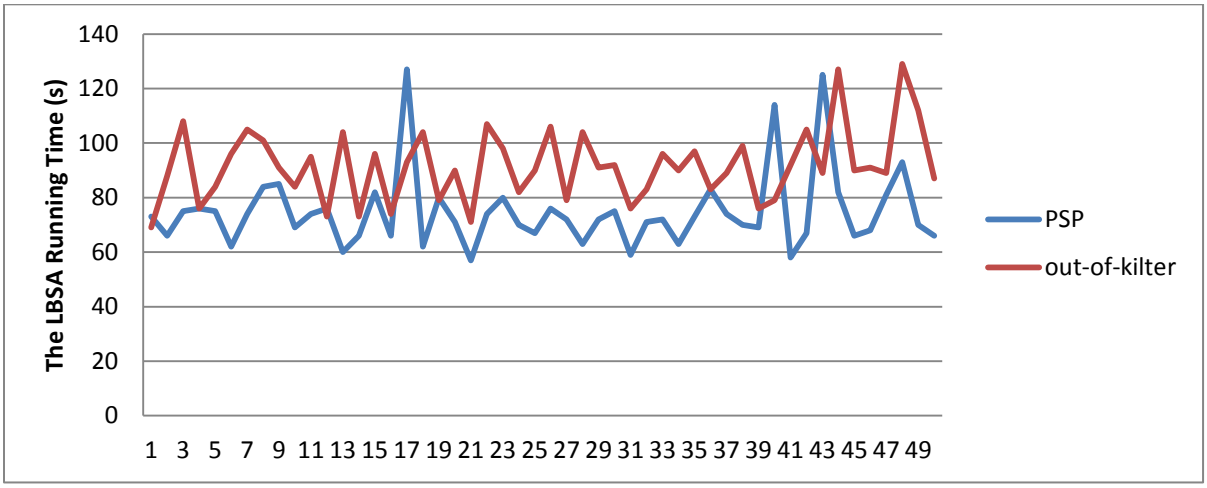

Figure 12 The Running Times of the LBSA Using the Two AFD Methods

Chapter 5 Conclusions

5.1 Summary

This research focuses on finding the optimal allocation of skill-limited workers in a job shop so that the lower bound on $L_{\max}$ is optimized, i.e. the lower bound is minimized; it is based on the Lower Bound Search Algorithm (LBSA) presented by Lobo (2011) that determines the lower bound for a given worker allocation and searches for the allocation which leads to the optimal lower bound. To formulate the absolute partial labor flexibility (APLF) situation, workers with same skills were grouped as in a worker group, and the worker allocation matrix was introduced to illustrate the number of workers in each worker group assigned to each machine group. To determine if each allocation proposed by the LBSA is feasible, two existing linear programming based methods—the transportation method and the out-of-kilter algorithm—were discussed. A new methodology was developed, which initializes the worker allocation matrix with a feasible allocation and then reallocate workers among machine groups to get the allocation desired. This worker reallocation was achieved by a new search algorithm named here as the Path Searching Procedure (PSP). The optimality of each reallocation can be ensured because the PSP was proved to be able to realize the reallocation if possible and because a different choice of path would not cause a difference in terms of the existence of other paths in future iterations. The method based on the PSP that determines the feasibility of a worker allocation (the PSP Method) solved the Allocation Feasibility Determination (AFD) problem in fewer iterations than the other two existing methods for some particular problems. Then the PSP Method was embedded into the LBSA. This LBSA,

adapted to the absolute partial labor flexibility situation, was proved to be optimal, i.e. it will find the optimal lower bound with any feasible initial allocation.

In the experiments, the optimal lower bound found with skill limit constraints (in absolute partial labor flexibility situations) was compared with the optimal lower bound found without skill limit constraints (in complete labor flexibility situation); the value of LB_{APLF}-LB_{CLF} was presented for different job shop types, staffing levels, skill structures, and due-date ranges. We concluded that globally, the value of LB_{APLF}-LB_{CLF} will decrease as the average number of machine groups that the workers in each worker group can operate increases, or as the staffing level increases, and will stay relatively constant as the due-date range changes. If the average number of machine groups that the workers in each worker group can operate is greater than or equal to 4 out of 10 ($FM \geq 0.4$), then LB_{APLF}-LB_{CLF} is equal to 0. When the staffing level is greater than or equal to 90%, the average value of LB_{APLF}-LB_{CLF} is also relatively close to 0. Locally, the job shop type and the particular skill structure (matrix) used interact with each other and greatly affect the value of LB_{APLF}-LB_{CLF}. If the skill structure matches the distribution of the workloads among machine groups, i.e. the machine groups that have more loads can be operated by workers in more worker groups, then the value of LB_{APLF}-LB_{CLF} will be significantly reduced. Otherwise, this difference can be as big as 200 times of the unit processing time. In cases where the job shop type is unclear, the *k*-chain structure is a recommended worker skill pattern if possible.

5.2 Future Research

There are some extensions to the current model. In this thesis, for each *FM* value, we only tested one irregular skill structure. Currently, all skill structures are 10×10 matrices and the

number of workers in each worker group is equal to or as close as possible to the mean number of workers per group. We are interested in determining the best skill structure without constraints on the number of worker groups given the value of *FM* and finding the optimal distribution of workers among worker groups according to the skill structure and the staffing level. And since it is a reasonable assumption in the real world that increasing either the *FM* value or the staffing level of a job shop will bring more cost, we may try to find a balance between these two when hiring workers. In other words, given a target lower bound, we are interested in finding the best strategy (*FM* value, skill structure, staffing level, and worker distribution among worker groups) that minimizes the labor cost. Moreover, when the size of the job shop is huge, i.e. the skill matrix is 30×30 or larger, it may be more efficient to first rearrange rows and columns of the skill matrix to form blocks along the diagonal of the skill matrix. Then we could develop a block method to assign workers to the worker allocation matrix by blocks.

Recall that there are two types of partial labor flexibility: absolute partial labor flexibility and relative partial labor flexibility. This thesis only addresses the issue of allocating workers with absolute partial labor flexibility. If workers have different levels of efficiency operating a machine, the current model does not work. To push the model closer to the realities of industry, we are interested in building a new model to determine the best allocation which minimizes the lower bound on $L_{\max}$ or even minimizes the $L_{\max}$ value in a job shop in relative partial labor flexibility situation.

REFERENCES

Baker, K., and Scudder, G. (1990) Sequencing with earliness and tardiness penalties: a review. *Operations Research*. **38**, 22-36

Baker, K. and Su, Z. (1974) Sequencing with due-dates and early start times to minimize maximum tardiness. *Naval Research Logistics Quarterly*, **21**(1), 171-176.

Carlier, J. and Pinson, E. (1984) An algorithm for solving the job shop problem. *Management Science*, **30**(9), 164-176.

Carroll, D. (1965) *Heuristic Sequencing of Single and Multiple Component Jobs*. Ph.D. thesis, MIT, Cambridge, MA.

Daniels, R and Mazzola, J. (1994) Flow shop scheduling with resource flexibility. *Operations Research*, **42**, 504-522.

Daniels, R., Mazzola, J., and Shi, D. (2004) Flow shop scheduling with partial resource flexibility. *Management Science*, **50**, 658-669.

Emmons, H. (1987) Scheduling to a common due date on parallel processors. *Naval Research Logistics*, **34**, 803-810

Felan, J., Fry, T., and Philipoom, P. (1993) Labour exibility and sta_ng levels in a dual-resource constrained job shop. *International Journal of Production Research*, **31**(10), 2487-2506.

Gargeya, V. and Deane, R. (1996) Scheduling research in multiple resource constrained job shops: A review and critique. *International Journal of Production Research*, **34**(8), 2077-2097.

Graham, R., Lawler, E., Lenstra, J., and Rinnooy Kan, A. (1979) Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, **5**, 287-326.

Hodgson, T., Cormier, D., Weintraub, A., and Zozom, A. (1998) Satisfying due dates in large job shops. *Management Science*, **44**(10), 1442-1446.

Hodgson, T., Joines, J., Roberts, S., Thoney, K., and Wilson, J. (2000a) Satisfying due-dates in large job-shops: Characterizing ‘industrial' test problems. Unpublished.

Hodgson, T., King, R., Thoney, K., Stanislaw, N., Weintraub, A., and Zozom, A. (2000b) On satisfying due-dates in large job shops: Idle time insertion. *IIE Transactions*, **32**, 177-180.

Hodgson, T., Melendez, B., Thoney, K., and Trainor, T. (2004) The deployment scheduling analysis tool (DSAT). *Mathematical and Computer Modelling*, **39**(6-8), 905-924.

Jordan, W. and Graves, S. (1995) Principles of the benefits of manufacturing process flexibility. *Management Science*. **41**, 577-594

Labetoulle, J., Lawler, E., Lenstra, J., and Rinnooy Kan, A. (1984) *Progress in Combinatorial Optimization*, Academic Press, chap. Preemptive scheduling of uniform machines subject to release dates. 245-261.

Lobo, B. (2011) *Allocating Manpower to Minimize Lmax in a Job Shop*. Ph.D. dissertation, NCSU, Raleigh, NC.

Lenstra, J., Rinnooy Kan, A., and Brucker, P. (1977) Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, **1**, 343-362.

Mellor, P. (1966) A review of job shop scheduling. *Operational Research Society*, **17**(2), 161-171.

Ovacik, I. and Uzsoy, R. (1996) Decomposition methods for scheduling semiconductor testing facilities. *International Journal of Flexible Manufacturing Systems*, **8**(4), 357-387.

Potts, C. and Strusevich, V. (2009) Fifty years of scheduling: A survey of milestones. *Journal of the Operational Research Society*, **60**(S1), S41-S68.

Schultz, S., Hodgson, T., King, R., and Thoney, K. (2004) Minimizing Lmax for large-scale, job-shop scheduling problems. *International Journal of Production Research*, **42**(23), 4893-4907.

Thoney, K., Hodgson, T., King, R., Taner, M., and Wilson, A. (2002) Satisfying due-dates in large multi-factory supply chains. *IIE Transactions*, **34**(9), 803-811.

Treleven, M. and Elvers, D. (1985) An investigation of labor assignment rules in a dual-constrained job shop. *Journal of Operations Management*, **6**, 51-68.

Vepsalainen, A. and Morton, T. (1988) Improving local priority rules with global lead-time estimates: A simulation study. *Journal of Manufacturing & Operations Research*, **1**, 102-118.

Weintraub, A., Cormier, D., Hodgson, T., King, R., Wilson, J., and Zozom, A. (1999) Scheduling with alternatives: A link between process planning and scheduling. *IIE Transactions*, **31**(11), 1093-1102.

Zozom, A., Hodgson, T., King, R., Weintraub, A., and Cormier, D. (2003) Integrated job release and shop-oor scheduling to minimize WIP and meet due-dates. *International Journal of Production Research*, **41**(1), 31-45.

APPENDICES

APPENDIX A Generate Irregular Structures

Table 12 (*FM* 0.8, irregular) Structure, *S*=5, *T*=5

$wg_i \backslash mg_j$	1	2	3	4	5	n^{th} "1" changed
1	0	1	1	1	1	diag
2	1	0	1	1	1	diag
3	1	1	0	1	1	diag
4	1	1	1	0	1	diag
5	1	1	1	1	0	diag

Table 13 (*FM* 0.6, irregular) Structure, *S*=5, *T*=5

$wg_i \backslash mg_j$	1	2	3	4	5	n^{th} "1" changed
1	0	1	0	1	1	2
2	1	0	1	1	0	4
3	0	1	0	1	1	1
4	1	1	0	0	1	3
5	1	0	1	1	0	2

Table 14 (*FM* 0.4, irregular) Structure, *S*=5, *T*=5

$wg_i \backslash mg_j$	1	2	3	4	5	n^{th} "1" changed
1	0	1	0	1	0	3
2	1	0	0	1	0	2
3	0	0	0	1	1	1
4	1	1	0	0	0	3
5	0	0	1	1	0	1*

* If changed the 2nd "1" to "0", there would be no "1" in column 3, which is unacceptable.

APPENDIX B Skill Structures Used in Experiments

Table 15 Skill Matrix of the (*FM* 0.4, *k*-chain) Structure

$wg_i \backslash mg_j$	1	2	3	4	5	6	7	8	9	10
1	1	1	1	1	0	0	0	0	0	0
2	0	1	1	1	1	0	0	0	0	0
3	0	0	1	1	1	1	0	0	0	0
4	0	0	0	1	1	1	1	0	0	0
5	0	0	0	0	1	1	1	1	0	0
6	0	0	0	0	0	1	1	1	1	0
7	0	0	0	0	0	0	1	1	1	1
8	1	0	0	0	0	0	0	1	1	1
9	1	1	0	0	0	0	0	0	1	1
10	1	1	1	0	0	0	0	0	0	1

Table 16 Skill Matrix of the (*FM* 0.3, *k*-chain) Structure

$wg_i \backslash mg_j$	1	2	3	4	5	6	7	8	9	10
1	1	1	1	0	0	0	0	0	0	0
2	0	1	1	1	0	0	0	0	0	0
3	0	0	1	1	1	0	0	0	0	0
4	0	0	0	1	1	1	0	0	0	0
5	0	0	0	0	1	1	1	0	0	0
6	0	0	0	0	0	1	1	1	0	0
7	0	0	0	0	0	0	1	1	1	0
8	0	0	0	0	0	0	0	1	1	1
9	1	0	0	0	0	0	0	0	1	1
10	1	1	0	0	0	0	0	0	0	1

Table 17 Skill Matrix of the (*FM* 0.2, *k*-chain) Structure

$wg_i \backslash mg_j$	1	2	3	4	5	6	7	8	9	10
1	1	1	0	0	0	0	0	0	0	0
2	0	1	1	0	0	0	0	0	0	0
3	0	0	1	1	0	0	0	0	0	0
4	0	0	0	1	1	0	0	0	0	0
5	0	0	0	0	1	1	0	0	0	0
6	0	0	0	0	0	1	1	0	0	0
7	0	0	0	0	0	0	1	1	0	0
8	0	0	0	0	0	0	0	1	1	0
9	0	0	0	0	0	0	0	0	1	1
10	1	0	0	0	0	0	0	0	0	1

Table 18 Skill Matrix of the (*FM* 0.4, irregular) Structure

$wg_i \backslash mg_j$	1	2	3	4	5	6	7	8	9	10
1	0	1	0	0	0	1	0	1	0	1
2	1	0	0	1	0	1	0	0	0	1
3	0	0	0	1	1	0	0	1	0	1
4	1	0	1	0	0	0	1	1	0	0
5	0	0	1	1	0	0	1	0	1	0
6	1	0	0	0	1	0	1	0	1	0
7	1	0	1	1	0	0	0	0	1	0
8	0	0	0	1	0	1	0	0	1	1
9	1	1	0	0	0	1	0	1	0	0
10	0	0	1	1	1	0	1	0	0	0

Table 19 Skill Matrix of the (*FM* 0.3, irregular) Structure

$wg_i \backslash mg_j$	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	1	0	1	0	1
2	1	0	0	1	0	0	0	0	0	1
3	0	0	0	1	0	0	0	1	0	1
4	1	0	1	0	0	0	1	0	0	0
5	0	0	0	1	0	0	1	0	1	0
6	1	0	0	0	1	0	0	0	1	0
7	1	0	0	1	0	0	0	0	1	0
8	0	0	0	1	0	1	0	0	1	0
9	0	1	0	0	0	1	0	1	0	0
10	0	0	1	1	0	0	1	0	0	0

Table 20 Skill Matrix of the (*FM* 0.2, irregular) Structure

$wg_i \backslash mg_j$	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	1	0	0	0	1
2	0	0	0	1	0	0	0	0	0	1
3	0	0	0	1	0	0	0	1	0	0
4	1	0	0	0	0	0	1	0	0	0
5	0	0	0	0	0	0	1	0	1	0
6	1	0	0	0	1	0	0	0	0	0
7	1	0	0	0	0	0	0	0	1	0
8	0	0	0	0	0	1	0	0	1	0
9	0	1	0	0	0	1	0	0	0	0
10	0	0	1	0	0	0	1	0	0	0