2017

AN EFFICIENT HEURISTIC TO BALANCE TRADE-OFFS BETWEEN UTILIZATION AND PATIENT FLOWTIME IN OPERATING ROOM MANAGEMENT

Feidi Dang
University of Kentucky, feidi.dang@outlook.com
Author ORCID Identifier:
https://orcid.org/0000-0001-7894-005X

Digital Object Identifier: https://doi.org/10.13023/ETD.2017.471

AN EFFICIENT HEURISTIC TO BALANCE TRADE-OFFS BETWEEN UTILIZATION AND PATIENT FLOWTIME IN OPERATING ROOM MANAGEMENT

THESIS

A thesis submitted in partial fulfillment of the
requirement for the degree of Master of Science in Mechanical Engineering
in the College of Engineering
at the University of Kentucky

By

Feidi Dang

Lexington, Kentucky

Co-Directors: Dr. Wei Li, Assistant Professor of Mechanical Engineering

and Dr. Fazleena Badurdeen, Associate Professor of Mechanical Engineering

Lexington, Kentucky
2017

ABSTRACT OF THESIS

AN EFFICIENT HEURISTIC TO BALANCE TRADE-OFFS BETWEEN UTILIZATION AND PATIENT FLOWTIME IN OPERATING ROOM MANAGEMENT

Balancing trade-offs between production cost and holding cost is critical for production and operations management. Utilization of an operating room affects production cost, which relates to makespan, and patient flowtime affects holding cost. There are trade-offs between two objectives, to minimize makespan and to minimize flowtime. However, most existing constructive heuristics focus only on single-objective optimization. In the current literature, NEH is the best constructive heuristic to minimize makespan, and LR heuristic is the best to minimize flowtime. In this thesis, we propose a current and future deviation (CFD) heuristic to balance trade-offs between makespan and flowtime minimizations. Based on 5400 randomly generated instances and 120 instances in Taillard's benchmarks, our CFD heuristic outperforms NEH and LR heuristics on trade-off balancing, and achieves the most stable performances from the perspective of statistical process control.

Keywords: Operating Room Scheduling, Permutation Flow Shop, Trade-off Balancing, Constructive Heuristic, Makespan, Flowtime.

Feidi Dang

Student's Signature

10/30/2017

Date

AN EFFICIENT HEURISTIC TO BALANCE TRADE-OFFS BETWEEN UTILIZATION AND PATIENT FLOWTIME IN OPERATING ROOM MANAGEMENT

By
Feidi Dang

ACKNOWLEDGEMENTS

I would like to express my sincerest thanks to my advisor Dr. Li. Thanks for his help with my research and thesis writing. His valuable advice and critical thinking have inspired me when I have problem with my research.

I am grateful to the committee members of my thesis defense, Dr. Jawahir and Dr. Badurdeen. Thanks for their critical comments on my work and this thesis. I also would like to thank my research colleagues, Honghan, Vivek, Amin and Xinwei. Thanks for their encouragement. It is my honor to work and study with them.

Finally, I would like to say thanks to my parents, uncles, aunts, brothers, and sister for their constant support and encourage.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter One: Introduction

1.1 Background

In modern industry system, the flow shop is widely applied in production systems to improve the effectiveness and efficiency, such as automotive assembly line. For a flow shop system, there are two important indicators which can be applied to assess system performance: one is utilization, and the other is work-in-process. The utilization impacts the efficiency of the whole system, while the inventory and holding cost are affected by the work-in-process. In order to improve the utilization and reduce the work-in-process, the flow shop scheduling problem attracted many researchers' attention for last several decades. As a decision-making process, scheduling not only plays an important role in manufacturing field, it is also applied in many service industries, such as the transportation(Pinedo, 2012). In the real world, due to the available resources are always limited for a company, schedulers and decision makers should consider how to use the limited resources to finish the activities and meet the requirements. For example, a manufacturer might receive hundreds of orders with different due dates. Usually, for this situation, the decision maker will focus on improving the utilization of the manufacturing system to meet as many of due dates as possible. It means that schedulers need to minimize the idle time and the set-up times for each machine.

According to the definitions of these two terms, the makespan is the time when the last job leaves the manufacturing system. A small value of makespan indicates a good utilization of manufacturing system. For the flowtime, it can be calculated by using total completion of all jobs divide by job numbers. The value of flowtime usually represents the holding cost and work-in-process inventory cost generated by current schedule. Based on the definitions of makespan and flowtime, we can see that the makespan and flowtime are related to utilization and holding cost, respectively. As the utilization and holding cost both are important indicators of the performance of a manufacturing or service system, the schedulers always desire to improve the utilization and minimize the holding cost, especially for a long-time period manufacturing or service planning, such as the operating room scheduling for a hospital.

In hospitals, the operating room can generate more than 40% of its total income, but the operating room also generates the largest proportion of the total cost for a hospital. (Denton, Viapiano, & Vogl, 2007). Therefore, the performance of operating room impacts the profit and service quality of whole hospital.

As the demand increasing in recent years, hospitals have had to improve the efficiency of the operating room system to meet the demand. Generally, there are several common problems for an operating room system, such as the long waiting time for each patient and the idle time of each operating room. For an operating room system, the lateness will not only cause the postponement of other patients, but it will also result in overtime costs. If the operating room in an idle status, it means that the utilization of operating room system is low. To solve these problems, managers of a hospital might set up more operating rooms and buy more instruments or hire more professional employees(Meskens, Duvivier, & Hanset, 2013). However, any of these approaches will increase the budget. Therefore, in recent years, the managers keep searching for some effective methods to improve the utilization of operating room and reduce the patients waiting time.

Before generating the optimization model for the operating room system, we would like to introduce the process of perioperative. A perioperative period consists of three different phases, which are preoperative, intraoperative and postoperative. The first phase is preoperative, the nurses and doctors will do several pre-treatments for patients, such as the registration and some test. Usually, a patient might only stay in this phase for short time. For the intraoperative, patients can receive the operations in the operating room. Based on the processing time for the different type of operations, the time period of intraoperative is longest for these three phases. The last phase is post-operative. A patient will be transferred to the post-anesthesia care unit (i.e. PACU) to receive some recovery treatments after the operations are finished. Therefore, a patient has to go through these three phases and the order is unchangeable.

In addition, according to the opinion of Cardoen(Cardoen, Demeulemeester, & Beliën, 2010), Two types of patients have to be considered when we try to propose a optimization method to improve the performance of operating room. One type is elective patients, and another type is non-elective. For elective patients, such as the patient who has cancer, the surgery for this type of patients is not very urgent. Therefore, doctors and nurses can generate a good planning for this type of patients. However, for the non-elective patients, such as the victims of car accidents. The surgeries for these patients have to be performed immediately. It is extremely difficult for managers to generate a good planning for non-elective patients. In our current research, we only consider the elective patients.

According to Marcon’s work(E. Marcon & Dexter, 2007), the scheduling method can be applied to improve the efficiency of operating rooms. In 2013, Meskens et al(Meskens et al., 2013) reviewed several different models, such as the scheduling model which can be used to optimize the performance of operating rooms. For the scheduling models, the author listed some common assumptions, such as:

1) A surgical cannot be interrupted

2) PACU is always available.

3) One surgeon could only process one patient at the same time.

4) The PACU can serve any type of patients, and so on.

Based on the analysis that we mentioned above, we can see that a perioperative period can be formulated as a three-machine flow shop. The first machine (or stage) can be seen as the preoperative phase, the second machine is the intraoperative phase, and the last machine is the PACU. In order to find out a suitable model of flow shop, we compare two main types of flow shop, such as the traditional flow shop and the hybrid flow shop(Pinedo, 2012) and show as follows:

- Flow shop (*Fm*)

To find out an optimal sequence of jobs for a flow shop scheduling problem, decision makers may consider changing the order of the jobs when these jobs waiting in the buffer to achieve a smaller makespan (i.e. maximum completion time). However, solving a flow shop scheduling problem is very difficult, if the sequence is changeable between two machines. Furthermore, we can always generate an optimal solution for two-machine or three-machine flow shop scheduling problem on makespan minimization objective without changing the order of jobs. A flow shop can be named as permutation flow shop, if the sequence of jobs is not allowed to change between two machines. Under this constraint, the jobs will go through the whole flow line and keep the same order. Moreover, another constraint of flow shop named as no-wait flow shop. For a no-wait flow shop, a job has to go through the whole flow line without waiting in the buffer between two machines.

- Hybrid flow shop (HFS)

Hybrid flow shop (HFS) is another type of flow shop. Compare to the traditional flow shop, a hybrid flow shop contains *m* stages and each stage has *k* parallel machines. For the hybrid flow shops, there are some common settings (Pinedo, 2012; Ruiz & Vázquez-Rodríguez, 2010): a) The stages number is larger than 2 (i.e. $m \geq 2$). b) The machine number of each stage is at least 1 (i.e. $k \geq 1$). c) The order of jobs can be changed between two stages and so on.

According to the assumptions of scheduling model for operating room and the settings of permutation flow shop, we can see that a general operating room system can be modeled as a permutation flow shop. In 2006, Marcon and Dexter(Eric Marcon & Dexter, 2006) showed that the simple sequencing rules, such LPT (longest processing time first) and SPT (shortest processing time first), can reduce the patient numbers in the waiting list.

In 2007, Denton examined the impact of scheduling on the patient waiting list and idle time of the operating room (Denton et al., 2007). Furthermore, according to the work of Cardoen (Cardoen et al., 2010), there are several common measurements to evaluate the performance of the operating room system, such as the utilization which can be defined as the workload divided by total processing time, patient waiting time (i.e. the waiting list) and makespan. Usually, in order to improve the profit and service quality, the wait time of patients (i.e. work-in-process) should be decreased, and the utilization need to be increased.

From the flow shop scheduling perspective, as we discussed before, the maximum completion time (C_{max}) can be related to the utilization of an operating room scheduling problem. The patient flowtime can be represented by the flowtime (i.e. $\sum C_{j,m}$, the total completion time of j patients on last stage). The waiting time of next (i.e. $(j+1)^{th}$) patient can be reduced, if we minimize the patient flowtime of the patients before this patient. From the literature in recent years, many researchers focus on the flow shop scheduling problems, and generate significant contributions to improve the performance of a production system, such NEH and LR. However, lots of studies focus on the single objective (i.e., makespan minimization or flowtime minimization.). For multi-objective optimization problem, part of them focus on solving the related objective, such as the makespan and lateness or tardiness. However, for a flow shop scheduling problem, the makespan and flowtime are two fundamental criteria for flow shop scheduling. Moreover, as the growing of computational capacity of computers, researchers prefer to apply the evolutionary algorithm, such as the GA (Genetic Algorithm) and SA (Simulation Annealing), to achieve multi-objective optimization. However, the computation time is very large.

1.2 Motivations

Currently, the existing heuristics mainly focus on the single objective optimization, such as the makespan or flowtime minimization criteria. After the Johnson's algorithm was proposed to generate the optimal solution for two-machine flow shop scheduling problem, many heuristics for the *m*-machine problem were developed by creating several virtual two-machine flow shop problems and applying Johnson's algorithm to minimize the makespan. However, Johnson's algorithm is only suitable for 2-machine flow shop with makespan criteria. Therefore, since the efficiency and effectiveness of NEH were approved, its framework was widely applied in constructive heuristic development. Furthermore, since the job selection scheme of NEH heuristic depends on the objective function, NEH framework is also suitable for flowtime minimization objective.

However, for the real-world problems, to evaluate a manufacturing products system or service system, such as the operating room system, the decision-makers not only need to optimize the system performance on a single objective, they also need to consider the others. Moreover, for the existing evolutionary multi-objective algorithm, their performance is good enough, but the computational time is not acceptable. Another problem is that most of these multi-objective heuristics and algorithms focus on the related objectives, such as makespan and tardiness. However, the tardiness time and the number of tardy jobs are related to the job due date. Furthermore, the most fundamental objectives of a flow shop are maximum completion time (i.e. makespan or *Cmax*) and total completion time (i.e. flowtime denote as $\sum C_j$), and other objectives can be derived out from these two objectives.

1.3 Difficulties and Challenges

Although the flow shop scheduling problem has been researched for several decades, there are some challenges need to be handled. If we only consider the one-machine flow shop scheduling problem, we can easily find out the optimal solution by applying LPT and SPT rule to minimize the makespan and flowtime, respectively. In 1954, Johnson(Johnson, 1954) developed an exact method to obtain an optimal solution for 2-machines and 3-machines permutation flow shop on maximum completion time minimization objective. It is likely for us to assume that the jobs should be sequenced by increasing order of processing time. However, it easy to see that the processing times of one job on each machine are likely not less than the processing time of another job on corresponding machines. According to Garey's work in 1976(Garey, Johnson, & Sethi, 1976), it has been proved that the flow shop scheduling problem is NP-complete, which means that the exact optimal solution is too difficult to find within a polynomial time. In addition, for the minimization of makespan and minimization of flowtime, these two objectives are inconsistent with each other(W. Li, Mitchell, & Nault, 2014). It means that minimize one objective does not always can minimize another one.

Moreover, there are two important measurements to evaluate the performance of heuristics and algorithms, which are effectiveness and efficiency. The effectiveness can be described as the relative deviation from the optimal solutions, and the computational complexity of a heuristic can be used to evaluate the efficiency. For many algorithms, they improved the quality of solutions (i.e. effectiveness), but the computational complexity of heuristic is increased (i.e. the efficiency is decreased.). Therefore, we desire to develop a new constructive heuristic to generate the solution with high solution quality and acceptable computation complexity.

1.4 Contribution

The contribution of our work that presented in this thesis are listed as follows. First, we developed a new method to generate the lower bound and upper bound for the completion time. Based on this lower and upper bound generation method, we proposed a new initial sequence scheduling method, which depends on the deviation of actual completion time from the lower and upper bound, to balance the trade-off between makespan and flowtime. Furthermore, in the initial sequence generation method, we are not only considered the impact of current job, but we also take the effect of unscheduled on our objectives into account. For the trade-off balancing objective, we model the trade-off between two coupled deviations by a factor α for each job on each machine (i.e. operation level). Moreover, we also generate a model to balance the trade-off at line level, which means the trade-off between makespan and flowtime for the whole flow line.

In our heuristic, the job insertion method is applied to improve the quality of initial sequence. For the job insertion phase, we developed a new normalized evaluation function to determine which partial sequence should be selected. In order to justify the performance of our heuristic, the case studies are carried out on small-scale and large-scale cases. The results show that our proposed heuristic can achieve better performance on trade-off balancing objective. For the single objective optimization, our heuristic also outperforms the existing heuristics with same computational complexity. Furthermore, we applied our proposed heuristic to solve the operating room scheduling problem for the UK healthcare. The utilization of operating room is increased, and the patient flowtime is reduced when our heuristic is applied.

In addition, current existing heuristics are designed to solve the scheduling problem without considering the performance for a long-time period planning. However, in our work, we applied the statistical process control (SPC) to evaluate the long-time period performance of our heuristic based on the dataset from UKHC. The performance of proposed heuristic is more stable with a higher solution quality than the method which is used by UKHC.

1.5 Structure of this thesis

The structure of this thesis is organized as follows:

In the Chapter Two, we present the literature review for current status of flow shop scheduling. In this section, we review existing heuristics for both single objective and multi-objective optimization in permutation flow shop. Furthermore, we also review the basic concept of statistical process control.

In Chapter Three, we show the problem description of the permutation flow shop scheduling problem, and generate a Gantt chart to explain the calculation method of completion time. Then, we present a new initial sequence generation method for trade-off balancing objective. Furthermore, a job insertion method with new evaluation scheme is generated to improve the solution quality of initial sequence.

In Chapter Four, the results of case studies are provided. We compare our CFD heuristic with other existing heuristics on the single objective and trade-off balancing objective. The case studies are carried on small-scale and large-scale (i.e. Taillard's benchmark) database. Moreover, we also applied the CFD heuristic on UK Healthcare database and the results are presented.

In Chapter Five, the conclusions are summarized, and the future work is discussed.

Chapter Two: Literature Review

As the classical flow shop scheduling problem, there are thousands of publications and research results for makespan minimization (denote as $Fm|prmu|Cmax$) and flowtime minimization (denote as $Fm|prmu|\Sigma C_j$)(Graham, Lawler, Lenstra, & Kan, 1979) on permutation flow shop scheduling problem. Currently, in many companies and service industries, the LPT and SPT dispatching rule are widely applied. However, these two simple dispatching rules can only obtain the optimal solutions for makespan and flowtime objectives on one-machine permutation flow shop scheduling problem. According to Johnson's work in 1954(Johnson, 1954), the optimal solution can be generated for makespan objective on 2-machine flow shop. In Johnson's algorithm, we find out the minimum processing time among all the jobs on two machines. If the minimum processing time occurs on the second machine, the corresponding job will be scheduled to the last position. Otherwise, allocate the job to the first location of the sequence. Then, delete the job from the unscheduled jobs, and repeat this procedure until there is no job left. However, Garey(Garey et al., 1976) proved the *m*-machines permutation flow shop scheduling problem is NP-complete. Which means that it is difficult to find the optimal solution within the polynomial time. Therefore, researchers start to develop the heuristics and algorithm to solve the flow shop scheduling problem within an acceptable computation time.

In this chapter, we reviewed several existing heuristics and algorithms for flowtime and makespan minimization objectives. In general, there are two different types of method to solve the scheduling problems: the first one is the exact method, such as the enumeration method and Branch & Bound method. For example, a branch-and-bound algorithm for tardy jobs minimization in a 2-machine flow shop with release date was proposed by Abouei et al(Abouei Ardakan, Hakimian, & Rezvan, 2013) in 2013. However, these exact methods cannot be applied to the large-size problem, because of the unacceptable computation time. Another type is the approximate method, which includes the heuristics and meta-heuristics and so on. For the approximate method, one can generate the solutions that close to the optimal results within a short computation period. Obviously, the approximate methods are more suitable for solving the real-world problems. In this chapter, the literature review is classified as three different types based on the different objectives. Moreover, the evolutionary algorithms, such as the genetic algorithm (GA), are also reviewed briefly.

2.1 Makespan objective

The makespan minimization for permutation flow shop scheduling problem has been proved to be *NP*-complete for an *m*-machine flow shop (Rand, 1977). From Johnson's algorithm (Johnson, 1954), the optimal solution of makespan can be obtained with $O(n*\log n)$ for two-machine flow shop. After Johnson's algorithm was developed, there are many heuristics were developed based on the concept of Johnson' algorithm. These heuristics solve the scheduling problems by creating several virtual 2-machines problems, and then Johnson's algorithm was applied to solve these 2-machines problems.

Campbell *et al* proposed CDS heuristic (Campbell, Dudek, & Smith, 1970), which m machines were regrouped as (m-1) artificial two-machines flow shops. Then, apply Johnson's algorithm to solve these (m-1) two-machine flow shop problems. Therefore, (m-1) candidate solutions can be obtained. Then calculate the makespan (i.e. Cmax) and the sequence with minimum makespan is selected as the final solution.

In 1965, a heuristic is proposed by Palmer based on the concept of '*slop index*' (Palmer, 1965), the solution is generated by decreasing order of the value of SI_j, where the $SI_j = -\sum_{i=1}^{m}[m-(2*i-1)]*t_{j,i}/2$. However, there are several works have proved that the Palmer's algorithm is not effective.

Gupta (J. N. Gupta, 1971) proposed a revised function of *SI*, and the author showed that the newly proposed heuristic obtained better performance than Palmer's. The new index function of SI can be defined as:

$$SI_j = e_j/\min\{t_{j,k}+t_{j,(k+1)}\}, (1 \leq \mathrm{k} \leq \mathrm{m}-1)$$

where

$$e_j = \begin{cases} 1 & if \quad t_{j,1} < t_{j,m} \\ -1 & if \quad t_{j,1} < t_{j,m} \end{cases}$$

Then, scheduling the jobs follow the non-ascending order of SI values. In this work, the case study was carried out, and Gupta proved that the new heuristic can provide better performance on makespan minimization than Palmer's.

The NEH heuristic was proposed by Nawaz *et al* in 1983 (Nawaz, Enscore, & Ham, 1983). NEH heuristic has two different phases. Phase.I: an initial sequence is generated by sorting jobs according to the non-increasing order of total processing times on all machines. The total processing time can be computed by:

$$p_j = \sum_{i=1}^{m} p_{j,i}\,, \text{where } j = 1 \dots n$$

In the second phase, select first two jobs from the initial sequence to create a partial sequence with minimum makespan value. Then, insert the next jobs from the initial sequence into all possible locations of current partial sequence and select the partial sequence with minimum makespan. Repeat the second phase until all jobs are removed from the initial sequence.

Furthermore, Taillard's proposed a modified NEH heuristic in 1990(Taillard, 1990). In Taillard's work, the new heuristic reduced the computational complexity of NEH from $O(n^3m)$ to $O(n^2m)$ without sacrifice the quality of the final solutions. However, this "speed-up" method was designed to solve the makespan minimization problem. For the flowtime minimization, this "speed-up" procedure does not work(J. M. Framinan, Leisten, & Rajendran, 2003).

Since the NEH heuristic was developed, many newer heuristics and algorithms were developed according to the framework of NEH heuristic. For these newer heuristics and algorithm, they generated the initial sequence(s) first, and the constructive method (i.e. insertion method) is applied to generate the final solution. In order to obtain an initial solution, some simple sequencing rules can be used, such as the ascending or descending order of total processing time (i.e. SPT and LPT rule).

According to the structure of NEH, the quality of the final solution is very likely related to the goodness of the initial sequence. In 2003, Framinan(J. M. Framinan et al., 2003) evaluated 177 different initial orders to identify which initial sequence could obtain the best performance for makespan, idle time and flowtime minimization objectives. Based on the results that presented in Framinan's work, the original NEH heuristic is the best heuristic for makespan objective among 177 candidate heuristics. According to the Framinan's work, we can say that the strength of NEH depends on the order of which job is selected to be inserted during the second phase.

As the good effectiveness and efficiency of NEH heuristic framework, researchers start to find out other objective functions that can be applied in the final sequence construction phase, such as the *idle time*. In the latter of this thesis, we use the definition of idle time that proposed by King and Spachis(King & Spachis, 1980) and show as follows:

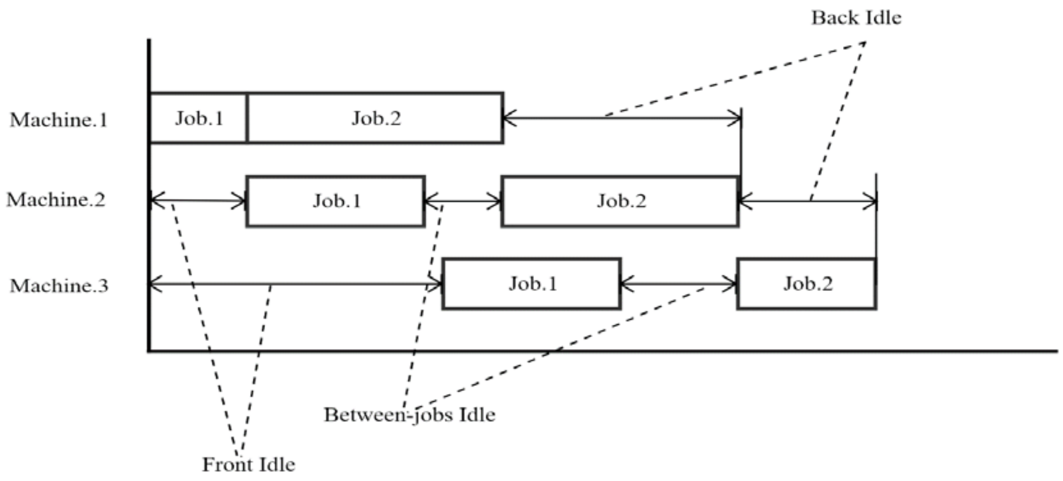


Figure 2.1 Different type of idle time

From the Figure 2.1, we can see that the makespan can be calculated by sum up the total processing time of all jobs and total idle time on the last machine. Therefore, there are several heuristics and algorithms were developed by replacing the original objective function in phase II of NEH by minimization of idle time. For example, the heuristic which is proposed by Sarin and Lefoka(Sarin, 1992) (denote as SL). In SL heuristic, the initial sequence is also generated by following the descending order of total processing time of each job on all machines, which is same as the NEH heuristic. However, in the sequence construction stage, the job which could generate the minimum idle time on the last machine will be selected to append to the partial sequence. The specific steps of SL heuristic are shown as follows:

Step.1:
- Generate the initial sequence following the descending order of total processing time, which is $TP_j = \sum_{i=1}^{m} p_{j,i}$
- Divide the jobs into two sets, one is scheduled jobs set {S} and another is unscheduled jobs set $\{U\}$.

Step.2
- Select each job in $\{U\}$ to append to partial sequence (i.e. $\{S\}$). Then calculate the idle time on the last machine by

$$IT = \sum_{i=1}^{i=m-1} p_{1,i} \ for\ |S| = 1$$

$$IT = max\{C_{|S|,i-1} - C_{|S|-1,i}, 0\}$$

If the idle time of all candidate partial sequences are greater than 0, then select the one with minimum IT.

In Sarin and Lefoka's work, they used an indicator (denote as SLI), which can be defined as $\text{SLI} = SL\ makespan/NEH\ makespan$, to compare the effective of the SL and NEH heuristic. According to the results, the solutions generated by SL heuristic when the machine numbers are not large (i.e. machine number is less than 100). However, when the number of machines is large than 100, the performance of NEH is better than SL heuristic.

Chakraborty and Laha(Chakraborty, 2007), in 2007, proposed a revised heuristic (denote as CL) based on the NEH. The initial sequence generation method is same as NEH heuristic. However, the job insertion phase was modified, and we presented the whole process of CL heuristic as follows:

Step.1: Generate the initial sequence by following the descending order of total processing time.

Step.2: The first 4 jobs are selected from the initial sequence and enumerate all possible sequence of these 4 jobs to generate (4!=24) candidate partial sequences. Then select the best k (a parameter in this heuristic.) 4-job sequence from these partial sequences. Set z=5.

Step.3: Select the z^{th} job from the initial sequence and insert to *z* possible position of *k* partial sequences. Then select the best k sequences from (*z***k*) z-jobs partial sequence.

Step.4: Set *z*=z+1, and if *z*>*n*, then choose the best sequence from the *k* *n*-jobs sequences as the final solution.

The author claimed that the proposed heuristic yield better performance than original NEH heuristic. However, the whole heuristic is based on the framework of original NEH. Moreover, the computational complexity is same as NEH, but CL heuristic has to evaluate more sequence than NEH. In 2012, Singhal et al(Singhal, Singh, & Dayma, 2012) proposed a heuristic which is very similar with CL heuristic, but they did not provide any computational experiment results except a specific numerical example.

In 2016, Li proposed a lever concept and applied this concept to solve the permutation flow shop scheduling problem(W. Li, Freiheit, & Miao, 2016). In their work, three sequencing methods were proposed and named as SBL (without applying the lever concept), SBLL, SLL. In SBLL, the impact of the idle time on bottleneck machine (M_B) was considered. The larger idle time created on the bottleneck, the worse solution will be obtained for makespan objective. Therefore, we want the job can flow into the bottleneck machine as soon as possible to minimize the idle time on M_B. It means that the jobs need to sequenced follow SPT rule from M_1 to M_B. For the machine following the M_B to last machine M_m, sequence the job follow the LPT rule to improve the performance. The main step of the method can be defined as follows:

Step.1: Calculate $SUM_i = \sum_{j=1}^{n} p_{j,i}$ where i=1…m, select the machine with maximum SUM_i as bottleneck machine.

Step.2: Generate the value of Torques (T). For M_1 to M_B, $T_{j,i} = (B - i + j) * p_{j,i}$. For the M_{B+1} to M_m, $T_{j,i} = (i - B + 1) * (-p_{j,i})$

Step.3: Sequence the job according to the ascending order of $SUM\_T_j = \sum_{j=1}^{n} \sum_{i=1}^{m} T_{j,i}$.

In order to highlight the contribution of the proposed heuristics, the performance was compared with LPT, SPT, and MIX heuristic which is proposed by Marcon and Dexter in 2006(Eric Marcon & Dexter, 2006). The numerical case study was carried on Taillard's benchmark. From the results, the SBLL heuristic obtained the smallest average deviation of 14.5% from the best-known solution of benchmark for makespan minimization objective. Furthermore, the author also compared the performance between SBLL and SBL, and proved that the heuristic can achieve a better performance when the lever concept was considered.

In the work of Ruiz (Ruiz & Maroto, 2005), they evaluated 25 existing heuristics until 2005 which includes the exact method, constructive heuristic and metaheuristic, for makespan objective. The author claimed that NEH heuristic is the best heuristic for both effectiveness and efficiency, when it is applied to solve Taillard's benchmark problem. Meanwhile, the frame of NEH heuristic has been applied in many existing heuristics for different objectives. In recent years, there are several researchers claimed that their heuristics can generate better performance than NEH heuristic. However, Kalczynski and Kamburowski proved that these claims cannot be justified and the NEH is still the best constructive heuristic for permutation flow shop scheduling problem (Kalczynski & Kamburowski, 2007).

In 2015 Gupta and Chauhan(A. Gupta & Chauhan, 2015) came up the weighting factors for each $p_{j,i}$, and then regroup the machines to generate several artificial 2-machines problems. To prove the performance of this heuristic, they compared the proposed heuristic with other existing algorithms. The author reported their heuristic is more effective than Palmer, CDS and RA which is presented by Dannenbring(Dannenbring, 1977) ,but could not better than NEH

For the solution construct phase, an objective function should be designed, and the partial sequence is evaluated by this objective function. Usually, the objective function can be the objective itself, such as the value of makespan. However, in the second phase of NEH heuristic, *k* partial sequences need to be evaluated when the *k*th job from initial sequence was inserted into *k* possible positions, the tie might be generated. Consider this situation, the tie-breaking strategies were deployed. In 2007, Kalczynski and Kamburowski compared NEHNM, which proposed by Nagano and Moccellin(Nagano & Moccellin, 2002), with NEH based on the Taillaid's benchmark. Furthermore, a new evaluation function (PA) was presented in their work and defined as $\mathrm{PA} = (\mathrm{No.\,of\,Wins} - \mathrm{No.\,of\,Losses}/N) * 100\%$. The No. of Wins and No. of Losses represent the number of solutions that better or worse than NEH, respectively, the total number of cases denote as N. the author reported that the average PA of NEHNM is -7.5%. It means that NEH heuristic could generate more best solution than NEHNM heuristic for Taillaid's benchmark.

Fernandez-Viagas et al(Fernandez-Viagas, Ruiz, & Framinan, 2017) reviewed and generated exhaustively comparison among the best heuristic and meta-heuristic from effective and efficient perspective. According to the results, only five heuristics can be determined as effective, which are NEHFF (proposed by Fernandez-Viagas and Framinan(Fernandez-Viagas & Framinan, 2014)), FBR (proposed by Rad et al.(Rad, Ruiz, & Boroojerdian, 2009)). For NEHFF heuristic, the tie-breaking strategy was applied in the job insertion phase. In FBR heuristic, the authors used the local search method in their heuristic. Therefore, FBR heuristic can be classified as a composite heuristic. Due to the most existing heuristic are developed from NEH heuristic, the author also admitted that the superiority of NEH heuristic.

2.2 Flowtime objective

The flowtime minimization (denote as $\min(\sum C_j)$) for permutation flow shop has been studied for several decades. Ho and Chang(Ho & Chang, 1991) (denote as HC) and obtained the best performance among CDS, Dannenbring, Gupta, Palmer, and Random sequence generation method.

Rajendran & Chauduri(Chandrasekharan Rajendran & Chaudhuri, 1992) (denote as RC) proposed several different effective heuristics for the flowtime minimization objective in 1992. In this work, three criteria were presented which are:

$$\sum_{i=2}^{m} \max\{C_{(\sigma a,i-1)} - C_{(\sigma,i)}, 0\}$$

$$\sum_{i=2}^{m} \text{abs}\{C_{(\sigma a,i-1)} - C_{(\sigma,i)}, 0\}$$

$$\sum_{i=2}^{m} \text{abs}\{C_{(\sigma a,i-1)} - C_{(\sigma,i)}\} + \sum_{i=1}^{m} C_{\sigma a,i}$$

where σ is the partial sequence and the job a append to partial sequence σ denote as σa.

Then, three algorithms were formed by using these three criteria. For the first algorithm, calculate the value of the first equation, which is listed above, for each unscheduled job, and the job with the minimum value should be appended to the partial sequence (σ). Similarly, replace the evaluation function by the second and third equation to form the other two algorithms. After carried out the numerical illustration, the author claimed that three proposed heuristics obtained best performance for flowtime objective among existing heuristics (such as the Ho and Chang's which we mentioned before.)

Because of the strength of NEH job insertion strategy, many researchers modified and revised this strategy to generate the heuristics for flowtime minimization objective on permutation flow shop scheduling problem. In 1993, Rajendran proposed a heuristic to minimize the total flowtime, named as Raj (Chandrasekharan Rajendran, 1993). In this heuristic, the jobs are sequenced according to the ascending order of T_j, where $T_j = \sum_{i=1}^{m}(m-i+1)p_{j,i}$, where $p_{j,i}$ is the processing time of job j on machine i. Then select the first job as the partial sequence, and insert the rest job one by one into all possible location of the partial sequence. From the computational results, the Raj heuristic can obtain better solutions than heuristics proposed by Ho and Chang(Ho & Chang, 1991) and Rajendran & Chauduri (Chandrasekharan Rajendran & Chaudhuri, 1992).

WY heuristic, proposed by Woo and Yim (Woo & Yim, 1998), also applied the insertion strategy of NEH heuristic. The difference of WY heuristic is that the initial sequence is not required which means the insertion phase should be applied to each unscheduled job. The procedures of WY are shown as follows:

Step.1: Set k=1, initialize the scheduled job sequence $S = \{\phi\}$, and unscheduled job set $U = \{\text{all jobs}\}$. Select each job from U and insert to the S as the first job. Calculate the flowtime for n different S, and pick one with the minimum value of flowtime. Delete the selected job from U.

Step.2 Set k=k+1, pick each job from U and insert to k possible position of S to form k partial sequence. Then, select the job with min(flowtime). Delete the selected from U.

Step.3 If k>n, stop, otherwise, return to step.2.

According to the experiment result, the performance of WY is the best among CDS, NEH and Raj on the mean flowtime objective.

In 2003, LF heuristic presented by Framinan(J. M. Framinan & Leisten, 2003) modified the insertion phase of NEH heuristic. The revised insertion strategy combined with forward inter-exchange method. We summarized this strategy and presented as follows (the following steps focus on the pair-wise exchange because it is the only difference between LF and NEH):

Step.1. Pick the best partial from k candidate sequences that generated by

NEH insertion method, denote as σ.

Step.2. If the size of σ is greater than 2 (i.e. $|\sigma| \geq 2$), exchange the position of job a and b ($1 \leq a \leq |\sigma|, a \leq b \leq |\sigma|$). Generate all possible sequences and select the one with minimum value of flowtime as the partial sequence.

Step.3. Return to NEH insertion phase, until $|\sigma| = n$.

The performance of LF heuristic is better than WY and RZ on flowtime minimization objective for both small (job number= {5,6,7,8,9}; machine number = {5,10,15,20} and large-scale (job number={10,20,30,40,50,60,70,80}, machine number={5,10,15,20}) test-bed. Furthermore, the author also combined the proposed scheme with IH-7 composite heuristic, which is a composite heuristic and proposed by Allahverdi, Aldowaisan in 2002(Allahverdi, 2002). However, because the computational complexity of LF heuristic is $O(n^4m)$ (J. M. Framinan & Leisten, 2003)the computation time for large-scale problem is very large.

In 2009, Laha and Sarin revised the pairwise interchange method of FL heuristic, and the new heuristic was denoted as FL-LS(Laha & Sarin, 2009). In this heuristic, the Step.2 of interchange method of LF, which we have presented above, was revised as follows. Each job in the *k*-job partial sequence that obtained from NEH insertion method inserted into (k-1) positions (i.e. insert *k*th into all position of current partial sequence except *k*th position). Therefore, (k-1) k-job sequences can be obtained and pick the one with minimum flowtime as the current partial sequence.

In the authors' work, they proved that the performance of FL has been significate improved if the new exchange method was deployed. The average relative percentage deviation, which is defined as $\mathrm{ARPD} = ((Heuristic - Best)/Best)$, was applied to compare LF and LF-LS heuristic. According to the results, ARPD of LF changes from 0.073% ~0.78% for small cases and 0.29%~1.37% for large cases. For LF-LS heuristic, the ARPD values change from 0.024%~0.56% and 0%~0.47% for small and large cases, respectively. In addition, according to the Pan's work in 2013(Pan & Ruiz, 2013), the author also claimed that the FL-LS heuristic can obtained the best performance among existing heuristics. However, both of LF and LF-LS heuristic, their computational complexity is increased to $\mathrm{O}(\mathrm{n}^4\mathrm{m})$, because of the application of interchange method.

Liu and Reeves presented LR heuristic in their work (Liu & Reeves, 2001). An index function was developed, which considered the effect of idle time and the expect completion time of unscheduled jobs. Assume that a k-jobs partial sequence S was generated, and a job J need to be selected from unscheduled jobs set U and append to S as (k+1)th job in S. Then, then the idle time between kth and k+1th job can be calculated by following equations:

$$\mathrm{IdleTime} = \sum_{\mathrm{i=2}}^{\mathrm{m}} \mathrm{w} * \max\{C_{k+1,i-1} - C_{k,i}\ ,0\}$$

$$w = \frac{\mathrm{m}}{i + k * (m - i)/(n - 2)}$$

To develop the expect flowtime part of index function, an artificial job A is created. The average processing time of all rest jobs in U is computed after the job J is appended to S with k jobs. Then, they used the average processing time as the processing time of the artificial job *A* and this artificial job will be appended to the S with (k+1) jobs. The average processing time is used as the processing time of job A. The specific calculation method of average processing time is defined by following equations:

$$\mathrm{p}_{\mathrm{k+2,i}} = \sum_{\mathrm{j=1}}^{|\mathrm{U}|} p_{j,i}/(n-k-1)$$

Then, the completion time ($\mathrm{C}_{\mathrm{j,i}}$) and expect flowtime (AT) can be calculated by following equations:

$$\mathrm{C}_{1,1} = \mathrm{p}_{1,1}$$

$$\mathrm{C}_{\mathrm{j},1} = \mathrm{C}_{\mathrm{j}-1,1} + p_{j,1} \text{ where } \mathrm{j} = 2 \ldots \mathrm{k}+2$$

$$\mathrm{C}_{1,\mathrm{i}} = \mathrm{C}_{1,\mathrm{i}-1} + \mathrm{p}_{1,\mathrm{i}} \; where\ i = 2 \ldots m$$

$$\mathrm{C}_{\mathrm{j,i}} = \max\{C_{j-1,i}, C_{j,i-1}\} + p_{j,i}$$

$$\mathrm{AT} = \mathrm{C}_{\mathrm{k+1,m}} + \mathrm{C}_{\mathrm{k+2,m}}$$

After artificial flowtime and idle time are generated, the index function can be formulated as below:

$$f = (n-k-2) * IdleTime + AT$$

The specific procedure of LR heuristic is as follows:

Step.1 Set $N = \{\text{all jobs}\}$ and $S = \phi$, select each job from *N* and insert to the first position of S, then calculate the value of index function. Sort all jobs according to the ascending order of f and save this sequence as *U*

Step.2 Use the first x sorted jobs as the first job in *S*, pick each of rest jobs in *U* and append to *S*, select the job with minimum index function value as the last job in S and delete from *U*.

Step.3 Repeat the second step, until there is only one job left in *U*. Append the last job to *S* directly.

In their work, the author showed that LR heuristic outperformed existing heuristics, such as Ho (Ho & Chang, 1991) and WY (Woo & Yim, 1998). From the literature, the LR(1) is the best constructive heuristic to minimize flowtime with the computational complexity of $O(n^3m)$. When the parameter x equal to 1, the proposed heuristic can obtain a better performance than Ho, WY, and RZ.

2.3 Multi-objective

In several works, the multi-objective optimization problem is solved by minimizing one objective subject to some conditions. For example, Gupta proposed a heuristic to minimize the flowtime in a 2-machine flow shop environment with minimum makespan. In their work, the exact solution method was developed from Johnson's algorithm for 2-machine flow shop. Furthermore, they also presented several heuristics based on the dominance rule to generate the approximate solutions to the flow shop scheduling problem.

In 1994, Rajendran and Chaudhuri proposed a heuristic (denote as RC) based on *two* job selection schemes which are developed based on the lower bound of completion time. These two relations can be defined as follows:

$$\mathrm{LB}(\sigma \mathrm{a}, \mathrm{i}) = \mathrm{C}_{\sigma,1} + \sum_{i=1}^{m} p_{j,i}$$

- Relation.1:

The partial sequence σa is preferred to sequence σb, when:

$$\mathrm{C}_{\sigma,1} + \sum_{i=1}^{m} p_{a,i} \leq C_{\sigma,1} + \sum_{i=1}^{m} p_{b,i}$$

- Relation 2:

For the second relation, a weighting factor was allocated to the processing time on each machine.

$$\mathrm{C}_{\sigma,1} + \sum_{i=1}^{m} (m - i + 1) * p_{a,i} \leq C_{\sigma,1} + \sum_{i=1}^{m} (m - i + 1) * p_{b,i}$$

Moreover, the whole procedures of RC heuristic are shown as bellows:

Step.1. Applying the NEH heuristic to generate the initial sequence, denote as *S*.

Step.2 Compute the values of following 2 indicators:

$$\mathrm{D}_{\mathrm{k}} = \sum_{i=1}^{m} p_{k,i} - \sum_{i=1}^{m} p_{k+1,i}$$

$$\mathrm{D'}_{\mathrm{k}} = \sum_{i=1}^{m} (m - i + 1) * p_{k,i} - \sum_{i=1}^{m} (m - i + 1) * p_{k+1,i}$$

Step.3 Pick the jobs with $D_k \geq 0$ to create a set *L* if there is no job can be selected, then stop.

Step.4 Sort the job according the descending order of the value of D_k and tie can be broken by assign the job with larger D'_k first.

Step.5 Select the first job *k* in the *L* and interchange the corresponding job *k* and (k+1) in S, denote the new sequence as S'. Compute the relative increment of makespan and flowtime of S' by following equations:

$$\mathrm{R}_{\mathrm{S}'} = \frac{Cmax' - \min(Cmax, Cmax')}{min(Cmax, Cmax')} + \frac{\sum C' - \min(\sum C, \sum C')}{\min(\sum C, \sum C')}$$

$$\mathrm{R}_{\mathrm{S}} = \frac{Cmax\ - \min(Cmax, Cmax')}{min(Cmax, Cmax')} + \frac{\sum C\ - \min(\sum C, \sum C')}{\min(\sum C, \sum C')}$$

If $\mathrm{R}_{\mathrm{S}'} < R_S$, then save the S' as S, and delete job k from set L

Step.6 Return to Step.5 until the set L is empty

In 2004, Ravindran proposed three heuristics (denote as HAMC1, HAMC2, HAMC3) to solve the makespan and flowtime minimization problem(Ravindran, Selvakumar, Sivaraman, & Haq, 2004). In these three heuristics, they solved the problem by RC firstly, and used the solution as the initial sequence. Then, interchange the position of job j and job i, where $1 \le \mathrm{i} \le \mathrm{n} - 1$ and $\mathrm{i} + 1 \le \mathrm{j} \le \mathrm{n}$. The new sequence is evaluated by using same evaluation scheme as RC, which are $\mathrm{R}_{\mathrm{S}'}$ and R_{S}. The sequence with the minimum value of R_S is saved as current sequence. Repeat the iteration for a fixed number (denote as x) which generally varies from 10 to 20. For HAMC1 heuristic, the author selected the sequence with minimum makespan from the x sequences obtained from each iteration. For HAMC2, select the sequence with the minimum flowtime value from these x sequences. For HAMC3, select the sequence generated from the last iteration as the result.

Framinan et.al developed a multi-objective heuristic in their work to minimize the makespan and flowtime, and the NEH insertion method was applied(Jose M Framinan, Leisten, & Ruiz-Usano, 2002). However, in this heuristic, a function $Y = w * (C_{max}) * (n/2) + (1 - w) * \sum C_j$ was developed, and the partial sequence with minimum Y is selected as current partial sequence. They compared the proposed heuristic with other existing heuristics, such as WY and R94(C Rajendran, 1994) and R95(Chandrasekharan Rajendran, 1995), which we have already mentioned above. The results show that the performance of the heuristic is better than others. However, in their work, Ho heuristic(Ho, 1995) can obtain better solution when the value of *w* is equal to 0, which means that we only focus on the flowtime objective. When *w* is equal to 1, which means that we only focus on the makespan, the performance of proposed heuristic is worse than NEH.

Furthermore, a lot of evolutionary algorithms were developed to solve the flow shop scheduling problem. For example, Varadharajan and Rajendran(Varadharajan & Rajendran, 2005) applied the simulated annealing(SA) algorithm to minimize flowtime and makespan. Sayadi et al (Sayadi, Ramezanian, & Ghaffari-Nasab, 2010) combined the firefly metaheuristic and local search method to solve the makespan minimization problem in permutation flow shop. However, several existing evolution algorithms and meta-heuristics applied constructive heuristics to generate the initial solution (population solution). For example, Framinan and Leisten(Jose M. Framinan & Leisten, 2007) proposed the multi-objective iterated greedy search with makespan and flowtime criteria. In this heuristic, they used the NEH and FL heuristic to obtain initial sequences. In 2015, Li proposed a multi-objective local search algorithm for flow shop scheduling problem (X. Li & Li, 2015)by applying NEH heuristic to generate the initial solution. In addition, the random sequence is also used as the initial sequence in several meta-heuristics. For example, Lei and Guo proposed a parallel neighborhood search method for flow shop scheduling(Lei & Guo, 2015). In their work, the initial solution was randomly generated. Moreover, in 2014, Marichelvam et al(Marichelvam, Prabaharan, & Yang, 2014) proposed a discrete firefly algorithm for makespan and mean flowtime minimization. They also generated initial population solution randomly. In 2017, Framinan compared existing meta-heuristics and claimed that the IG, which proposed by Ruiz and Stuzle in 2007(Ruiz & Stützle, 2007), can be identified as the most effective meta-heuristic(Fernandez-Viagas et al., 2017).

However, as we mentioned before, the computation time of meta-heuristic is much longer than constructive. Furthermore, based on Baskar’s idea that the research progress on constructive heuristic also can refine the meta-heuristic(Baskar, 2016). Therefore, in our work, we will focus on the development of constructive heuristic based on the analysis of the properties of permutation flow shop.

Chapter Three: Methodology

The proposed heuristic (denote as the CFD) consists of two main stages: (1). the initial sequence is generated according to the value of the deviations from the lower bound and upper bound. In the second stage, we applied the insertion technique to improve the solution quality. In this chapter, the initial sequence generation method is presented, which also includes the lower and upper bound calculation method. Moreover, the processes of CFD heuristic are discussed in this chapter.

3.1 Problem description

Due to the inconsistent of the makespan minimization and flowtime minimization objective. In this research, the objective is trying to balance the trade-off between the makespan and flowtime minimization objectives. In other words, we seek to find an optimal sequence with the minimum value of trade-off.

For a permutation flow shop scheduling problem (PFSP), there are some general assumptions and conditions are listed as follows:

- All jobs have to be available at t=0.
- No setup time for the machine.
- The job sequence cannot be changed during the manufacturing process.
- The intermediate storage between any two machines is unlimited.
- Preemption is not allowed.
- The processing time of each job on each machine is deterministic.

- Each machine can process only one job at same time.

To demonstrate the procedure of heuristics and definitions of permutation flow shop scheduling problems, we generated a Gantt chart (Figure 3.1) to explain the calculation method of completion time of each job on each machine, and introduced following terms that will be applied in this thesis:

n:	The number of jobs
m:	The number of machines
$p_{j,i}$:	Processing time of job *j* on machine *i*
$C_{j,i}$:	Completion time of the *j*th on machine *i*
$IT_{j,i}$:	Idle time of the jth on machine *i*
$LB_{j,i}$:	Lower bound of completion time of the *j*th job on machine i.
$UB_{j,i}$:	Upper bound of completion time of the *j*th job on machine i.
Cmax:	The makespan (i.e. $C_{n,m}$)
ΣC_j:	The flowtime of the sequence.

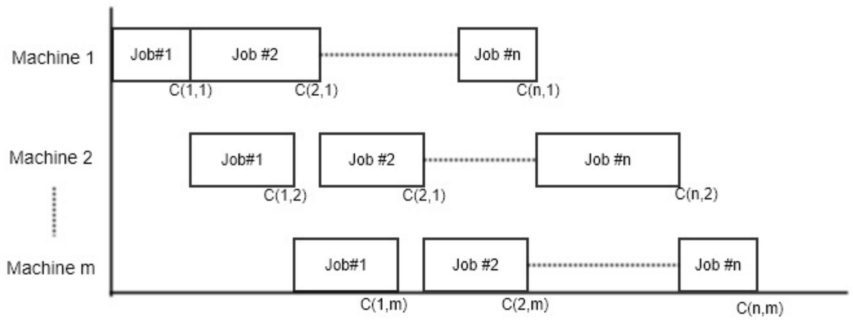


Figure 3.1 Gantt chart for a permutation flow shop

Because all jobs have to be available at *t*=0, there is no idle time on the first machine. Furthermore, from Figure 3.1, we can see that the completion time ($C_{j,i}$) of job j on machine i can be obtained by following equations:

$$C_{1,1} = p_{j,i} \tag{1}$$

$$C_{j,1} = C_{j-1,1} + p_{j,1} \tag{2}$$

$$C_{1,i} = C_{1,i-1} + p_{1,i} \tag{3}$$

$$C_{j,i} = max\{C(j-1,i), C(j,i-1)\} + p_{j,i} \tag{4}$$

$$\mathrm{Cmax} = \mathrm{C_{n,m}} \tag{5}$$

$$\Sigma \mathrm{C}_j = \sum_{j=1}^{n} C_{j,m} \tag{6}$$

3.2 Lower and upper bound of completion time

In this section, we introduced the lower bound and upper bound of completion time ($C_{j,i}$) generation method. According to the main concept of our proposed heuristic, we need to compute the bound for each job on each machine (i.e. the $LB_{j,i}, UB_{j,i}$). In order to obtain the theoretical lower and upper bound, we applied the sequence independent method. It means that we do not need to follow the same order on each machine.

The sequence-independent lower and upper bounds for machine *i* are calculated based on the minimum and maximum idle time on machine *i* respectively. The minimum idle time (*minIT*) on machine *i* can be obtained by a fast flow from machine *i*-1 and a slow flow out of machine *i*. Moreover, the maximum idle time (*maxIT*) on machine *i* are generated by a slow flow from machine *i*-1 and a fast flow out of machine *i*. Therefore, the calculation method of minimum and maximum idle time is introduced as follows:

$$minIT_{j,1} = 0 \tag{7}$$

$$minIT_{j,i} = \ max\{C_{j,i-1} - C_{j-1,i}, 0\} \tag{8}$$

$$minIT_{j,i} = max\{C_{j-1,i-1} + minIT_{j,i-1} + p_{j,i-1}^{SPT} - C_{j-2,i} - minIT_{j-1,i} - p_{j,i-1}^{LPT}, 0\} \tag{9}$$

$$maxIT_{j,i} = 0 \tag{10}$$

$$maxIT_{j,i} = max\{C_{j,i-1} - C_{j-1,i}, 0\} \tag{11}$$

$$maxIT_{j,i} = max\{C_{j-1,i-1} + maxIT_{j,i-1} + p_{j,i-1}^{LPT} - C_{j-2,i} - max\, IT_{j-1,i} - p_{j,i-1}^{SPT}, 0\} \tag{12}$$

where $LB_{0,i} = LB_{j,0} = 0$ and $UB_{0,i} = UB_{j,0} = 0$. The $p_{j,i}^{LPT}$ and $p_{j,i}^{SPT}$ are the processing time of jth job on machine *i* that follow the decreasing and increasing order of processing time of all jobs on machine *i*.

Based on the analysis that we mentioned above, the $LB_{j,i}$ and $UB_{j,i}$ can be defined as follows:

$$LB_{j,i} = C_{j-1,i} + minIT_{j,i} + p_{j,i}^{SPT} \tag{13}$$

$$UB_{j,i} = C_{j-1,i} + maxIT_{j,i} + p_{j,i}^{LPT} \tag{14}$$

There is no idle time on machine 1, therefore, the $LB_{j,1}$ and $UB_{j,1}$ can be computed by:

$$LB_{j,1} = LB_{j-1,1} + p_{j,1}^{SPT} \tag{15}$$

$$UB_{j,1} = UB_{j-1,1} + p_{j,1}^{LPT} \tag{16}$$

To explain the lower and upper bound calculation method, a 5-jobs, 3-machines problem are given in Table 3.2.1 as an example:

Table 3.2.1 Processing time of 5-jobs, 3-machines instance

$P_{j,i}$	M1	M2	M3
J1	70	82	44
J2	4	69	38
J3	28	32	76
J4	5	95	79
J5	10	4	19

According to the methods that we mentioned before, to generate the *minIT*(j,i), the jobs are sorted by following SPT on first machine 1 which is {2-4-5-3-1}, and follow the LPT rule on machine 2 which is {4-1-2-3-5}. Using equation (1) to (4), we can obtain the completion time on the first and second machine as follows: $C_{j,1} = [4,9,19,47,117]$; $C_{j,2} = [99,181,250,282,286]$, then minimum idle time on machine 2 is $MinIT_{j,2} = [4,0,0,0,0]$ (according to equation (6) to (9)). Therefore, the lower bound on machine 2 is $LB_{j,2} = [8,40,109,191,286]$. To obtain the lower bound on machine 3, we sequence the jobs on machine 1 and 2 following LPT rule and applied SPT rule on machine 3, then we can obtain the $LB_{j,3} = [27,65,109,185,306]$. Similarly, the upper bound of each position on each machine are calculated by equation (10) to (16), and the results are listed below (Table 3.2.2 and Table 3.2.3):

Table 3.2.2 The value of lower bound ($LB_{j,i}$)

$LB_{j,i}$	M1	M2	M3
position 1	4	8	27
position 2	9	40	65
position 3	19	109	109
position 4	47	191	185
position 5	117	286	305

Table 3.2.3 The value of upper bound ($UB_{j,i}$)

$UB_{j,i}$	M1	M2	M3
position 1	70	165	244
position 2	98	271	407
position 3	108	340	482
position 4	113	372	520
position 5	117	376	539

3.3 Two coupled deviations

In our work, the proposed heuristic aims to balance the trade-off between makespan and flowtime minimization. From the existing literature, it easy to see that the LPT rule is good for improving the performance of heuristic on makespan objective. Inspired by this idea, we minimize the deviation from upper bound for makespan objective, because it less likely generates idle time on machine *i*. For the flowtime minimization, the SPT rule can obtain good results. Therefore, we minimize the deviation from lower bound, which can generate small idle times on machine *i*, depending on the value of completion time on previous machines.

From the analysis, we designed two different deviation calculation method for makespan and flowtime, respectively: (1) For makespan objective, we minimize the deviation from upper bound, because it less likely generate idle time on previous machines; (2) For the flowtime minimization objective, we minimize the deviation from lower bound, which can generate small idle times on previous machines, depending on the value of completion time on previous machines. Furthermore, for flowtime minimization objective, we did not only consider the impact of scheduled jobs but also consider the impact of unscheduled jobs on our objectives. Therefore, we denote the scheduled job set as $\{S\}$ and unscheduled job set as $\{U\}$.

To calculate the deviation from upper and lower bound of completion time, we insert each job $J_{[j]}$ from unscheduled job set $\{U\}$ to the current location k (i.e. the kth position of current sequence). Therefore, the deviations can be defined as follows:

a) Deviation from upper bound:

$$\Delta CM_j = \sum_{i=1}^{m} |C_{k,i} - UB_{k,i}| \tag{17}$$

where j changes from 1 to (n-k+1), $C_{k,i}$ is completion time of kth job of current sequence on the ith machine. The $UB_{k,i}$ is the upper bound of completion time of kth job on ith machine

b) Deviation from lower bound

For the flowtime minimization objective, we are not only considered the impact of the current job in kth position, but we also consider the effect of the unscheduled jobs. The deviation of current job for flowtime minimization objective (denoted as ΔCT_j^H) can be defined as follows:

$$\Delta CT_j^H = \sum_{i=1}^{m} (m - i + 1) * \left(\left|C_{k,i} - LB_{k,i}\right|\right) \tag{18}$$

where j is the jobs from unscheduled job set $\{U\}$and changes from 1 to (n-k+1). The $C_{k,i}$ is the completion of the kth job on machine i, and the $LB_{k,i}$ is the lower bound of completion time of the kth job on the ith machine.

In order to evaluate the effect of unscheduled jobs (denoted as ΔCT_j^T), we use average processing time of all unscheduled jobs on machine i as the processing time of (*n*-*k*+1) unscheduled jobs (except the job $J_{[j]}$) on machine *i*. The average processing time can be obtained by following equation: $AveP_i = \sum_{j=1}^{n-k+1} p_{j,i}/(n-k+1)$. After the processing time of these (*n*-*k*+1) unscheduled jobs are obtained, we append these jobs to current *k*-jobs sequence and calculate the completion time from *k+1*th job to *n*th job. Then, the deviation from lower bound for unscheduled can be computed by following equations:

$$\Delta CT_j^T = \sum_{i=1}^{m} \left[(m-i+1) * \sum_{q=k+1}^{n} \left| C_{q,i} - LB_{q,i} \right| \right] \tag{19}$$

Moreover, based on the deviation of current job and unscheduled jobs, we can generate the deviation from lower bound for flowtime minimization objective:

$$\Delta CT_j = \Delta CT_j^H + \Delta CT_j^T \tag{20}$$

From the equations for flowtime objective, there is a weighting factor was assigned to the deviation part and the following example illustration can explain why we select (m-i+1) as the weight in our heuristic. According to the definition of completion time $C_{j,i} = max\{C_{j,i}, C_{j,i-1}\} + p_{j,i}$, the processing time of a job might be added for several times when we calculate the flowtime value. For example, $C_{1,1} = p_{1,1}$, $C_{1,2} = p_{1,1} + p_{1,2}$. Therefore, the deviations generated on early machines have greater effects than those generated on later machines. The weight factor (*m*-*i*+1) shows the decreasing effects as the machine number increases.

3.4 Development of CFD heuristic

3.4.1 Initial sequence generation

As the CFD heuristic aims to balance the trade-off between makespan minimization and flowtime minimization objectives, the preference relation between two objectives are considered in the initial sequence generation method. In the proposed heuristic we allocate a *weighting factor* α on ΔCM_j and (1-α) on ΔCT_j, to describe the different preference on the deviation. For the initial sequence, to determine whether a job is scheduled in current position or not, we proposed an evaluation scheme denote as *total deviation* (*TD_j*), and the definition is shown as follows:

$$TD_j = \alpha \cdot \Delta CM_j + (1-\alpha) \cdot \Delta CT_j \text{ where j} \in \{U\} \tag{21}$$

The job $J_{[j]}$ with the minimum *total deviation* (i.e. min(*TD*)) will be appended to the current sequence.

In order to explain the sequencing method specifically, the steps of initial sequence generation are shown as follows:

Step 1: Set location index *k*=1. Set $S = \emptyset$ and $U = \{J_1, J_2, \dots, J_n\}$.

Step 2: Select the *j*th job, denote as $J_{[j]}$ in *U* (*j*=1,…,*n*-*k*+1), and insert into the *k*th position of *S*. Then we calculate the average processing time ($AveP_i$) on each machine of the jobs in *U* except the $J_{[j]}$. We generated (*n*-*k*) artificial jobs with $AveP_i$ as the processing time of each artificial job on each machine. These artificial jobs are temporarily appended to *S*. from (*k*+1)th to *n*th in *S*.

Step 3: Computed the completion times (*C_{ji}*) of $\{S\}$ by applying the equation (1) to (3). Then, the current and future deviations for each objective can be generated by following equations (17) to (20):

$$\Delta CM_j = \sum_{i=1}^{m} |C_{k,i} - UB_{k,i}| \text{ where j} \in \{\text{U}\}$$

$$\Delta CT_j^H = \sum_{i=1}^{m} (m - i + 1) * (|C_{k,i} - LB_{k,i}|)$$

$$\Delta CT_j^T = \sum_{\mathrm{i}=1}^{\mathrm{m}} \left[(\mathrm{m} - \mathrm{i} + 1) * \sum_{\mathrm{j}=\mathrm{k}+1}^{|\mathrm{T}|} |\mathrm{C}_{\mathrm{j,i}} - \mathrm{LB}_{\mathrm{j,i}}| \right]$$

$$\Delta CT_j = \Delta CT_j^H + \Delta CT_j^T$$

Then the total deviation can be obtained by equation (21):

$$TD_j = \alpha \cdot \Delta CM_j + (1 - \alpha) \cdot \Delta CT_j$$

Where the α is the preference factor for *DevCmax* and *DevSUMC* which is obtained from decision makers. Then the job $J_{[j]}$ with the minimum value of total deviation (TD_j) will be selected and inserted to the *k*th location of *S*.

Step 4: Remove the select job $J_{[j]}$ from the *U*. If $k<n$-1, set $k=k$+1 and go to step 2. If $k=n$-1, append the remaining job in *U* to *S*, and save the *S* as initial sequence $\{\pi\}$

3.4.2 CFD heuristic

We also applied the insertion technique in the second phase of our heuristic to improve the performance after obtaining the initial sequence. As the CFD heuristic is designed for the trade-off balancing objective, we also introduced the preference factor α into our insertion phase and developed a new evaluation scheme based on the relative deviation increment value (*RIV*) from lower bound of makespan and flowtime for the current partial sequence. The lower bound for a partial sequence can be computed by applying the equations (7) to (16).

In addition, according to the calculation methods of makespan and flowtime, we can see that the scale of these two objectives are not same, the value of flowtime is significantly larger than makespan. Therefore, we normalized the deviation for both makespan and flowtime to reduce the impact of their different scales and defined the RIV as follows:

$$RIV = \alpha \cdot \left(\frac{C_{k,m} - LB_{k,m}}{UB_{k,m} - LB_{k,m}}\right) + (1-\alpha) \cdot \left(\frac{\sum_{j=1}^{k} C_{j,m} - \sum_{j=1}^{k} LB_{j,m}}{\sum_{j=1}^{k} UB_{j,m} - \sum_{j=1}^{k} LB_{j,m}}\right) \qquad (22)$$

where $C_{k,m}$ and $\sum_{j=1}^{k} C_{j,m}$ are makespan and flowtime for the k-jobs partial sequence $\{\phi\}$. The candidate partial sequence with the minimum *RIV* is selected as current partial sequence $\{\phi\}$ from all candidate sequences.

To illustrate the strategy of job insertion phase more specific, the steps of job insertion phase are shown as below in details:

Step 1: Generate the initial sequence (π) using the initial sequence generation method from section 3.2.

Step 2: Set k=2. Select the first two jobs from π to create a new k-jobs partial sequence $\{\phi\}$. Then exchange the position of these two jobs, and calculate the value of *RIV* in the following equations for two candidate partial sequences:

$$RIV = \alpha \cdot \left(\frac{C^{\{\phi\}} - LB_{k,m}}{UB_{k,m} - LB_{k,m}}\right) + (1-\alpha) \cdot \left(\frac{\Sigma C^{\{\phi\}} - \sum_{j=1}^{k} LB_{j,m}}{\sum_{j=1}^{k} UB_{j,m} - \sum_{j=1}^{k} LB_{j,m}}\right)$$

Step 3: Set k=k+1, choose the kth job from initial sequence and insert to all k possible locations of $\{\phi\}$. Calculate the *RIV* value for k candidate sequences. Update the $\{\phi\}$ by the candidate sequence with minimum *RIV*.

Step 4: If k<n, go to Step 3, otherwise output the current partial sequence $\{\phi\}$ as the final solution.

From the analysis that we present above, we can see that the computational complexity of our CFD heuristic is determined by the insertion phase in Step 3 (i.e. the job insertion phase). Hence, the CFD heuristic has the same computational complexity as NEH and LR heuristics, which is $O(n^3m)$.

3.4.3 A Numerical example for CFD heuristic

To explain the procedure of CFD heuristic in details, we use the same instance that presented in Section 3.2 (Table 3.2.1) and set the $\alpha = 0$. The processing time $p_{j,i}$ are shown as follows:

$P_{j,i}$	M1	M2	M3
J1	70	82	44
J2	4	69	38
J3	28	32	76
J4	5	95	79
J5	10	4	19

Step.1 Set $U = \{J1, J2, J3, J4, J5\}$ and $S = \{\phi\}$

Step.2 Calculated the lower bound ($LB_{j,i}$) and upper bound ($UB_{j,i}$) by applying the generation method which is mentioned in section 4.5.1. The matrix of $LB_{j,i}$ is

$LB_{j,\mathrm{i}}$	M1	M2	M3
position 1	4	8	27
position 2	9	40	65
position 3	19	109	109
position 4	47	191	185
position 5	117	286	305

Moreover, the matrix of $UB_{j,i}$ is shown as follows:

$UB_{j,i}$	M1	M2	M3
position 1	70	165	244
position 2	98	271	407
position 3	108	340	482
position 4	113	372	520
position 5	117	376	539

Step.3 Computed the total deviation (*TD*) for each job and selected the job with the minimum value of *TD* to append to *S*. In this example: $TD_j = \{2909.5; 1316.5; 1585; 1640.5; 1229.5\}$. Then, the job 5 is picked to append to *S* as the first job of scheduled job

set, and delete from the *U*. Repeat *step.3* and select the job with minimum value of *TD* in each iteration to append to *S*, until $U = \{\phi\}$.For this case, the *TD* values and unscheduled job set $\{U\}$ for each iteration are listed as below:

Iteration #1	*U*={*J1*,*J2*,*J3*,*J4*,*J5*} $TD_j = \{2909.5; \mathbf{1316.5}; 1585; 1640.5; 1229.5\}$. We select the *J2* to append to *S* and delete from *U*
Iteration #2	U= {*J1* ,*J3*,*J4*,*J5*} $TD_j = \{2047, \mathbf{781}, 991,984\}$. We select the *J3* to append to *S* and delete from *U*
Iteration #3	*U* = {*J1*,*J4*,*J5*} $TD_j = \{863, \mathbf{432}, 653\}$ We select the J4 to append to *S* and delete from *U*
Iteration #4	$TD_j = \{341, \mathbf{185}\}$ We select the *J5* to append to *S* and delete from *U*

Therefore, we can obtain an initial sequence as {5-2-3-4-1} for this case.

Step.4 Set k=2; Select first two jobs from *S*, and generate two possible candidate sequences {5-2}, {2-5} and their RIV value are 154 and 241.So we select the minimum one, which is {5-2} as current sequence.

Step.5 Set k=k+1. Pick the k^{th} job from S and insert to k possible position in current sequence. For k=3, generate k candidate partial sequences which are: {3-5-2}, {5-3-2}, {5-2-3}. The *RIV* values are [484, 363, 351]. Then we choose the one with min(*RIV*), which is {5-2-3}, as current partial sequence.

For k=4, we insert the job 4 to the current partial sequence and obtain{4-5-2-3},{5-4-2-3},{5-2-4-3},{5-2-3-4}. The corresponding RIV are [925,725,744,640], the {5-2-3-4} is picked as current sequence. For last iteration, we insert the Job 1 and generate five candidate sequences which are: {1-5-2-3-4},{5-1-2-3-4}, {5-2-1-3-4}, {5-2-3-1-4} and {5-2-3-4-1} with RIV values as [1444,1290,1022,963,976]. Therefore, the {5-2-3-1-4} is selected as current partial sequence.

Step.6 If $k \leq n$, return to Step.5, else stop and save current sequence as the solution. For this example, the final solution is {5-2-3-1-4}

Chapter Four: Case Study

In the computational experiment, we compared our CFD heuristic with NEH and LR heuristics on makespan (α=1) minimization, flowtime (α=0) minimization, and trade-off (α=0.5) minimization objectives based on random small-scale problem and Taillard's benchmark. Besides, we use the statistical process control to verify our CFD heuristic is better than the other two in terms of sustainable stableness.

4.1 Evaluation scheme

We test our CFD heuristic on both small-scale and large-scale instances. The processing times for small-scale instances are randomly generated following the uniform distribution in [1, 99]. For small-scale instances, the number of jobs is 5, 6, 7, 8, 9, 10, and the number of machines is [3, 5, 7, 9, 11, 13, 15, 17, 19]. Thus, there are 54 combinations. For each combination, 100 cases are randomly generated. Totally, we have 5400 instances for small-scale. For large-scale instances, the Taillard's benchmarks are used to test the performances of heuristics for flow shop scheduling, consisting of 120 instances in 12 combinations, where the number of jobs is 20, 50, 100, 200 or 500, and the number of machines is 5, 10 or 20. In each combination, there are 10 instances.

In order to evaluate the effectiveness of our CFD heuristic on makespan, flowtime and trade-off value minimization objectives. We applied three criteria to evaluate the performances of CFD heuristic for permutation flow shop scheduling problem:

a) For the makespan minimization objective (i.e. *Fm|prmu|C_{max}*), we used the average relative percent deviation (ARPD) to evaluate the effectiveness. The calculation method of ARPD is defined as follows:

- Average relative percent deviation (*ARPD*) for makespan:

$$\mathrm{ARPD}_{\mathrm{Cmax}} = \frac{1}{N} * \left(\sum_{i=1}^{N} \left(\frac{Cmax_i - MinCmax_i}{MinCmax_i}\right)\right) * 100 \qquad (23)$$

where the N is the total case number and the *$Cmax_i$* is the makespan of the *i*th case. For the small cases, the *$MinCmax_i$* is the optimal solution which is generated by enumeration method. For large-scale cases, the *$MinCmax_i$* is the best-known solution of Taillard's benchmark.

b) Similarly, to evaluate the effectiveness of CFD heuristic on flowtime minimization objective (i.e. *Fm|prmu|ΣC_j*), the *ARPD* vale is applied and can be calculated by the following equations:

- Average relative percent deviation (*ARPD*) for flowtime:

$$ARPD_{\Sigma C} = \frac{1}{N} * \left(\sum_{i=1}^{N} \left(\frac{SUMC_i - MinSUMC_i}{MinSUMC_i}\right)\right) * 100 \qquad (24)$$

where the *N* is the cases number and the *$SUMC_i$* is the flowtime time for the *i*th case. The value of *$MinSUMC_i$* is the optimal solution which is generated by enumeration method for small cases. For large-scale cases, *$MinSUMC_i$* is the best-known solution for the *i*th case.

c) As we said that the goal of our CFD heuristic is trade-off balancing. In order to describe the trade-off between minimization of makespan and flowtime, we defined the trade-off as the following equation:

$$TO_i = \beta \times \left(\frac{Cmax_i - MinCmax_i}{MinCmax_i}\right) + (1-\beta) \times \left(\frac{SUMC_i - MinSUMC_i}{MinSUMC_i}\right) \quad (25)$$

where the $Cmax_i$ and $SUMC_i$ are makespan and flowtime for i^{th} instance. For small cases, $MinCmax_i$ and $MinSUMC_i$ are optimal solutions obtained by enumeration method. For Taillard's benchmark, $MinCmax_i$ and $\mathrm{MinSUMC_i}$ are the best-known solutions for the i^{th} instance of Taillard's benchmark. *N* is the number of instances for each combination. It means that *N* is 100 for small cases, but 10 for large-scale instances. β is the preference factor to evaluate the *trade-off* value, changing from 0 to 1 with the step of 0.1.

4.2 Case study results

4.2.1 Small-scale cases

The case study results for small-scale cases are shown in Table.4.2.1, Table.4.2.2 and Table 4.2.3. we can see that our CFD heuristic can achieve the best performance on flowtime minimization and trade-off minimization objective. For the makespan minimization, the performance of CFD heuristic is very close to the NEH.

From the Table 4.2.1, we can see that our proposed CFD heuristic has smallest ARPD of 1.27% among three heuristics on makespan objective, while the NEH and LR are 1.28% and 11.14%, respectively. Moreover, our heuristic obtained smallest max(ARPD) of 12.09%, and the largest number of optimal solutions of 2171 (40.20%).

According to the results in Table 4.2.2, the CFD$(\alpha = 0)$ heuristic generated minimum ARPD of 0.90% on flowtime minimization objective. The NEH and LR have

ARPDs of 6.57% and 1.39%, respectively. Furthermore, our CFD heuristic achieves the minimum max(ARPD) and number of best solutions of 34.17%

Table 4.2.1 Average relative percent deviations(ARPD) of makespan for small-scale cases.

ARPD of Cmax	CFD(α=1)	NEH	LR
ARPD	0.0127	0.0128	0.1114
Min(ARPD)	0	0	0
Max(ARPD)	0.1209	0.1240	0.4255
# of Best Solutions	2171	2088	137
% of Best Solutions	40.20%	38.67%	2.54%

Table 4.2.2 Average relative percent deviations(ARPD) of flowtime for small-scale cases.

ARPD of SUMC	CFD(α=0)	NEH	LR
ARPD	0.0090	0.0657	0.0139
Min(ARPD)	0.0000	0.0000	0.0000
Max(ARPD)	0.0954	0.3666	0.1105
# of Best Solutions	1845	110	1255
% of Best Solutions	34.17%	2.04%	23.24%

To justify the performance of our heuristic on trade-off balancing objective, we set the $\alpha = [0; 0.5; 1]$, and applying the $TO_i = \beta \times \left(\frac{Cmax_i - MinC_i}{MaxC_i - MinC_i}\right) + (1-\beta) \times \left(\frac{SUMC_i - MinSUMC_i}{MaxSUMC_i - MinSUMC_i}\right)$ to evaluate the performance. The results of experiment are presented in Table 4.2.3.

From the Table 4.2.3 and Figure 4.2.1, the CFD heuristic obtained the best performance, which is 0.0313, and for LR and NEH are 0.0392 and 0.0627. Furthermore, when $\alpha = 0$, our heuristic dominates LR heuristic on trade-off minimization objective. For CFD$(\alpha = 0.5)$, our heuristic can dominate other two heuristics when β changes from 0.2 to 0.6.

Table 4.2.3 Trade-off (TO) value for different heuristics.

TO	CFD(α=0)	CFD(α=0.5)	CFD(α=1)	NEH	LR
0	0.0090	0.0312	0.0659	0.0657	0.0139
0.1	0.0187	0.0312	0.0606	0.0604	0.0237
0.2	0.0284	0.0312	0.0553	0.0551	0.0334
0.3	0.0381	0.0312	0.0500	0.0498	0.0432
0.4	0.0478	0.0313	0.0446	0.0445	0.0529
0.5	0.0576	0.0313	0.0393	0.0392	0.0627
0.6	0.0673	0.0313	0.0340	0.0339	0.0724
0.7	0.0770	0.0313	0.0286	0.0286	0.0822
0.8	0.0867	0.0314	0.0233	0.0233	0.0919
0.9	0.0964	0.0314	0.0180	0.0181	0.1017
1	0.1061	0.0314	0.0127	0.0128	0.1114
Ave	0.0576	0.0313	0.0393	0.0392	0.0627
st.Dev	0.0307	0.0001	0.0168	0.0167	0.0308

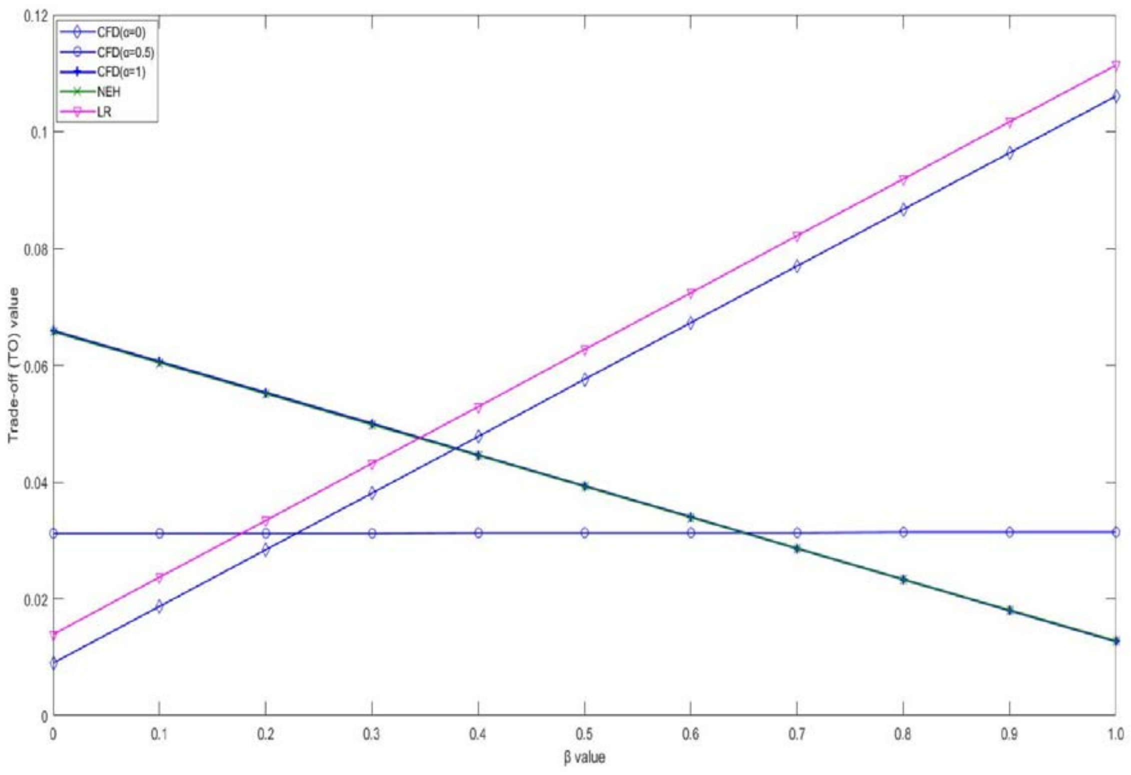


Figure 4.2.1 Trade-off value for small-scale cases

4.2.2 Large-scale cases (Taillard's benchmark)

For large-scale instances, we evaluate the heuristic performance by ARPD and maximum deviation from the best-known solutions. As shown in the Table 4.2.4, for the makespan objective, our proposed heuristic generated best ARPD of 3.32%, which is better than NEH of 3.33% and LR of 12.51%. For flowtime minimization objective, CFD also obtains the best performance among three heuristics. According to the results that shown in the Table 4.2.4, the ARPD of CFD on flowtime is 2.1%, and the deviation of NEH and LR are 10.09% and 2.23%. Moreover, for the first case in Taillard's benchmark (i.e. Ta001), CFD heuristic obtained a solution which is better than current best-known solutions of Taillard's benchmark on flowtime minimization objective.

Table 4.2.4 Single objective optimization results for Taillard's benchmark.

ARPD	CFD		LR		NEH	
	C_{max}	ΣC_j	C_{max}	ΣC_j	C_{max}	ΣC_j
20*5	0.0273	0.0150	0.1185	0.0154	0.0330	0.1007
20*10	0.0474	0.0217	0.1734	0.0266	0.0460	0.0850
20*20	0.0369	0.0167	0.1651	0.0303	0.0373	0.0679
50*5	0.0085	0.0252	0.0838	0.0140	0.0073	0.1526
50*10	0.0518	0.0246	0.1697	0.0338	0.0507	0.1047
50*20	0.0686	0.0232	0.1923	0.0268	0.0665	0.0770
100*5	0.0042	0.0248	0.0401	0.0098	0.0053	0.1225
100*10	0.0220	0.0262	0.0905	0.0205	0.0221	0.1126
100*20	0.0533	0.0160	0.1926	0.0314	0.0534	0.0776
200*10	0.0134	0.0189	0.0598	0.0165	0.0126	0.1140
200*20	0.0426	0.0212	0.1349	0.0279	0.0444	0.0943
500*20	0.0228	0.0183	0.0799	0.0151	0.0207	0.1021
Ave	0.0332	0.0210	0.1251	0.0223	0.0333	0.1009

In Table 4.2.5, the performance on trade-off balancing of each heuristic is shown, the CFD ($\alpha = 0.5$), which means we allocated same preference on makespan and flowtime objective, achieves the smallest value of trade-off (TO) of 0.0479 and minimum standard deviation of 0.0032, while the NEH and LR obtained 0.0671 and 0.0737 on trade-off (TO) objective. In addition, the CFD ($\alpha = 0$) and CFD($\alpha = 1$), which aimed to minimize flowtime and makespan respectively, also generate better solutions than LR and NEH.

Table 4.2.5 Trade-off value of different heuristics on Taillard's benchmark.

TO	CFD(α=0)	CFD(α=0.5)	CFD(α=1)	NEH	LR
0	0.0210	0.0428	0.0987	0.1009	0.0223
0.1	0.0305	0.0439	0.0922	0.0942	0.0326
0.2	0.0399	0.0449	0.0856	0.0874	0.0429
0.3	0.0494	0.0459	0.0791	0.0806	0.0532
0.4	0.0589	0.0469	0.0725	0.0739	0.0634
0.5	0.0683	0.0479	0.0660	0.0671	0.0737
0.6	0.0778	0.0489	0.0594	0.0603	0.0840
0.7	0.0872	0.0499	0.0529	0.0536	0.0942
0.8	0.0967	0.0509	0.0463	0.0468	0.1045
0.9	0.1062	0.0520	0.0398	0.0400	0.1148
1	0.1156	0.0530	0.0332	0.0333	0.1251
Ave	0.0683	0.0479	0.0660	0.0671	0.0737
st.Dev	0.0299	0.0032	0.0207	0.0214	0.0325

Figure 4.2.2 plots the trend of trade-off value based on the different β value. From this figure, we can see that the NEH and LR are dominated by CFD($\alpha = 0.5$) when β changes from 0.3 to 0.7. The CFD($\alpha = 1$) and CFD($\alpha = 0$) can dominate the NEH and LR respectively for all β values. Therefore, when CFD($\alpha = 0$), CFD($\alpha = 0.5$)

and CFD$(\alpha = 1)$ are applied in different range of β, we can generate the solutions that dominate other two heuristics.

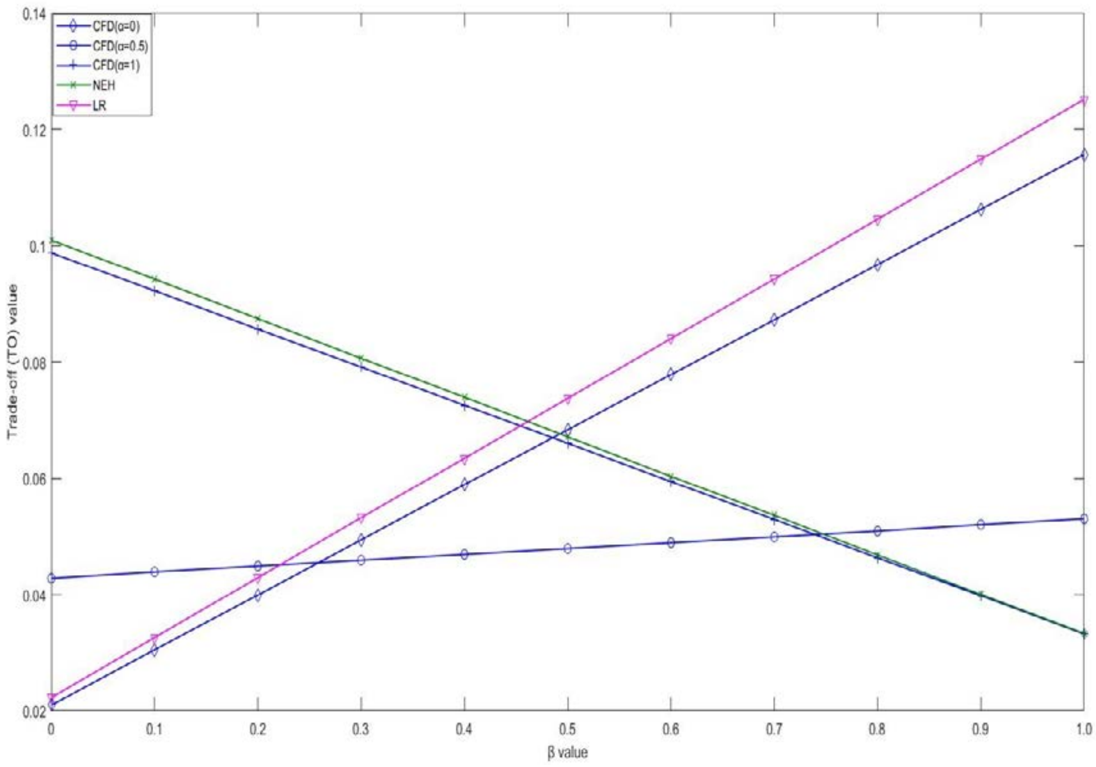
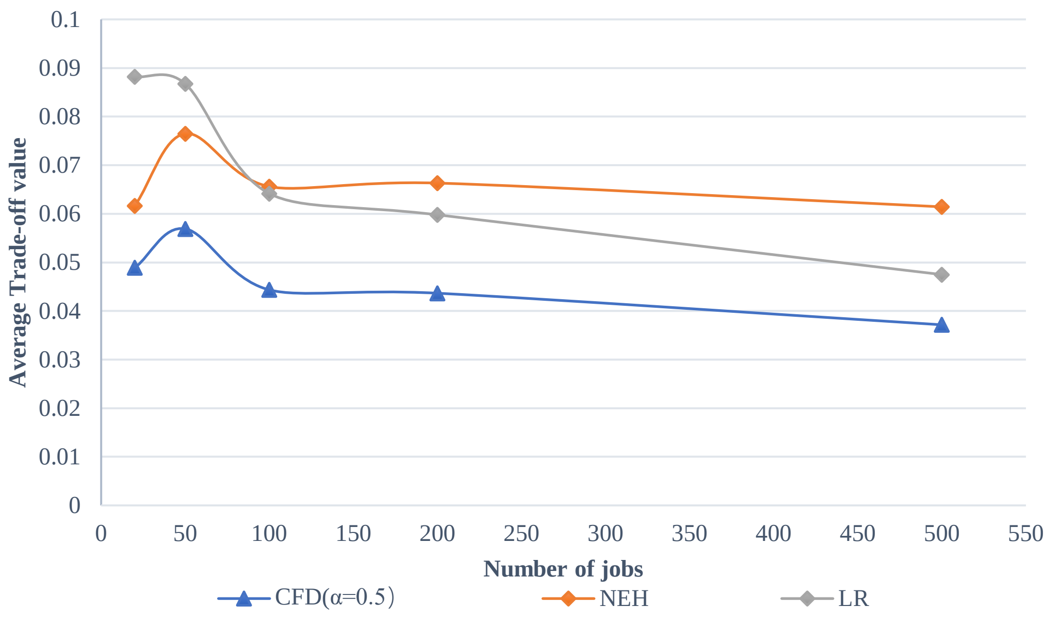
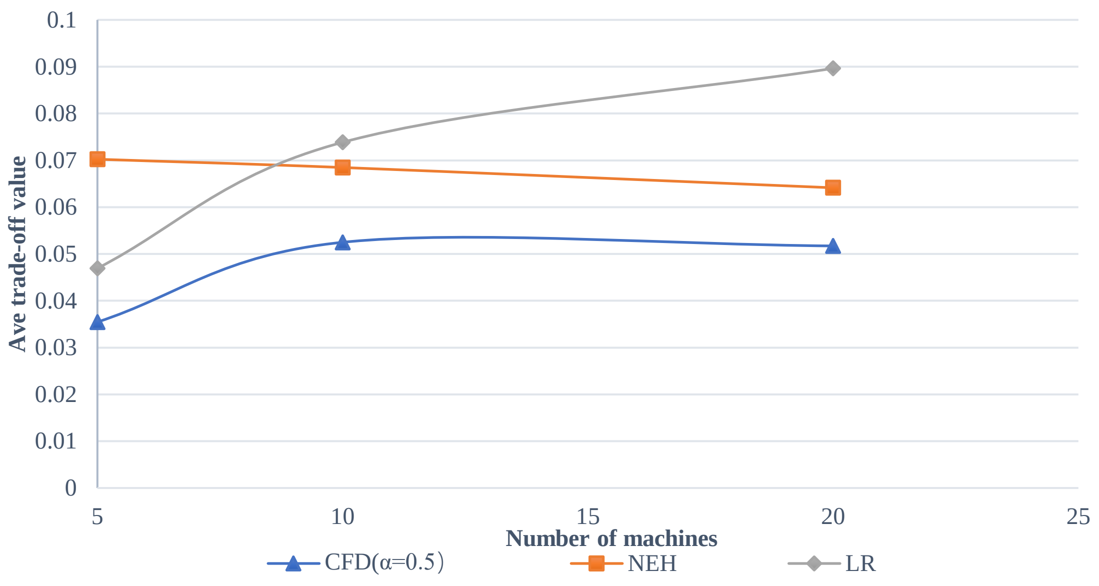


Figure 4.2.2 Trade-off value for Taillard's benchmark.

In addition, as the Figure 4.2.3(a) and (b) plots the changing of the average value of trade-off against the job numbers and machine numbers. It shows that CFD$(\alpha = 0.5)$ achieves the best performance for all job numbers and machine numbers. Furthermore, the LR can obtain better performance than NEH when the job number is increased. However, from the machine number perspective, the NEH generated better performance than LR heuristic on trade-off minimization objective.



(a) The average trade-off by number of jobs.



(b) The average trade-off by number of machines.

Figure 4.2.3 The average trade-off value by job number and machine number

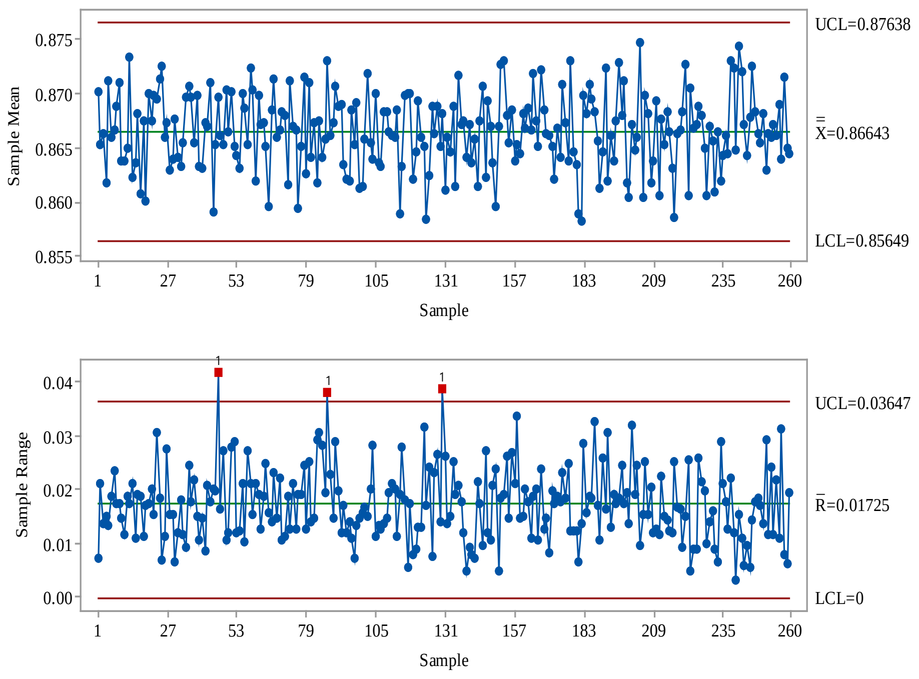
4.3 Statistical Process Control (SPC)

As a quality control method, the statistical process control, which is developed by is wide applied in industry to control and monitor the production process. Usually, there are two charts are used in SPC method named X-bar (mean)chart and R (range)-chart to evaluate the quality. In the X-bar and R chart, there are three important indicators: upper control limit, lower control limit, and central line. A point (or case) can be seen as out of control, if this point (or case) is above the upper control limit or below the lower control limit.

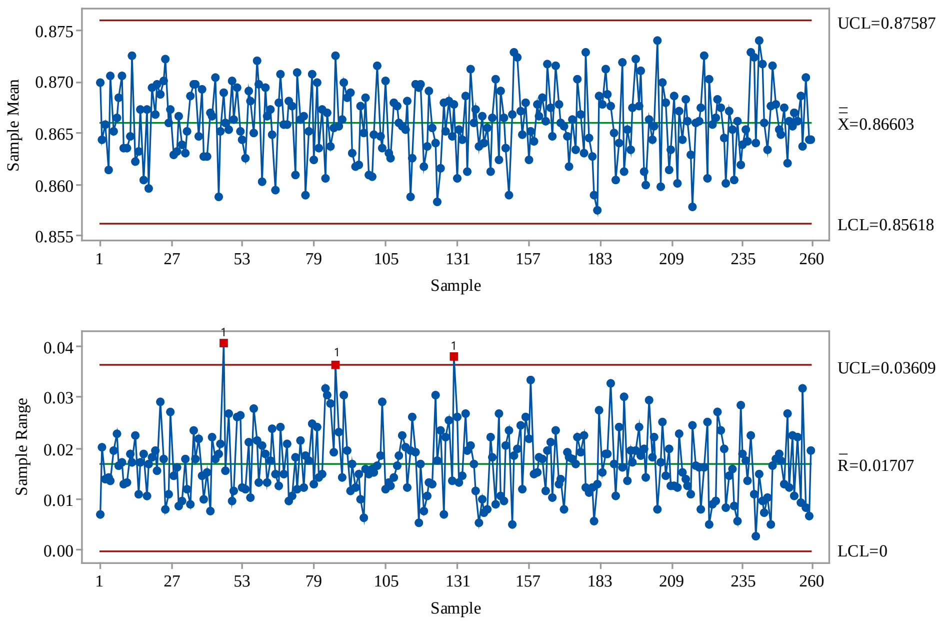
To validate our CFD (α=0.5) heuristic for operating room (OR) scheduling across the perioperative process, we carry out case studies on historical OR data from University of Kentucky HealthCare, which consists of around 30,000 cases in a year from 2013 to 2014. Excluding the data from the weekend and holidays, we have 26,000 cases in 260 days for a year.

In this case study, utilization of the perioperative process and patient flowtime across the perioperative process is used to evaluate performances of OR scheduling methods. The value of patient flowtime and utilization are calculated by applying CFD(α=0.5) on UK healthcare database. Then the results are compared with results of *first come first serve* scheduling method, which is used by UK Healthcare currently. As we mentioned before, the utilization of the whole perioperative process is related to the makespan. Moreover, in the UK Healthcare case study, we also compare the performance of CFD(α=0.5) with UK Healthcare based on patient flowtime. The patient flowtime equals to the total completion time of all patient divide by total patient number.

After we generate the data of utilization and patient flowtime by using CFD(α=0.5) and UK healthcare scheduling method, the statistic process control (SPC) technique is applied to generate the X-bar charts and R-charts, and shown in Figure 4.3.1

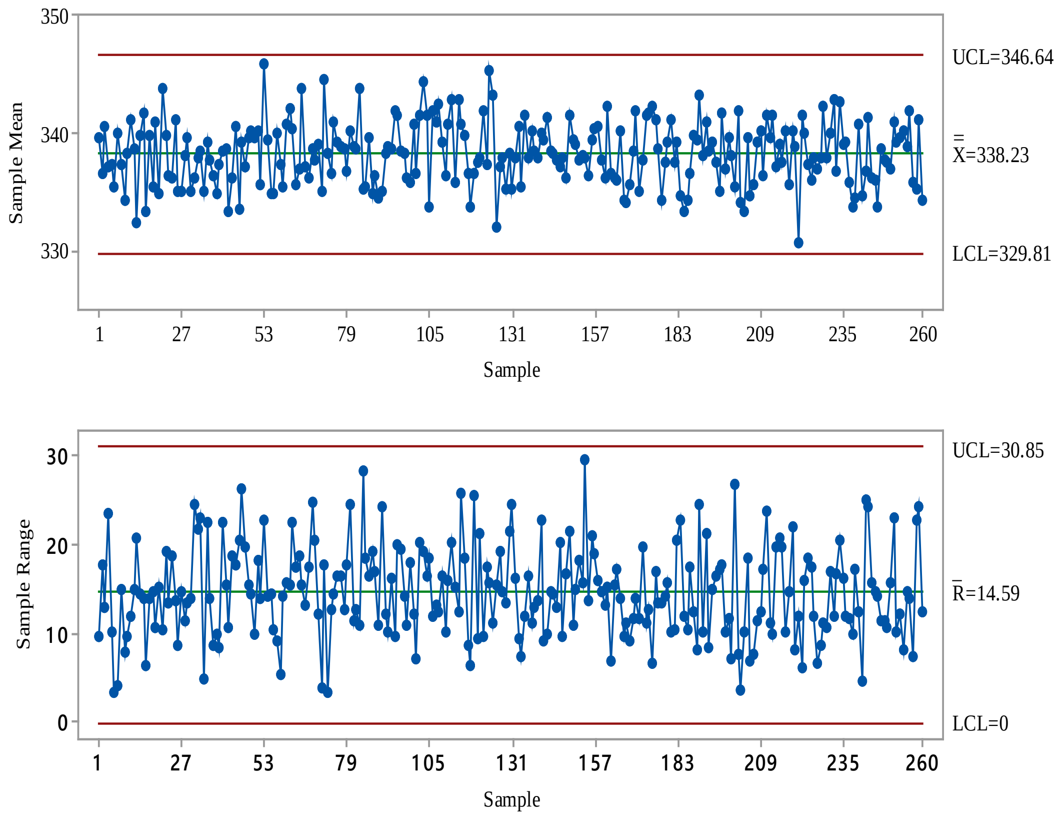


(a) Utilization of CFD heuristic

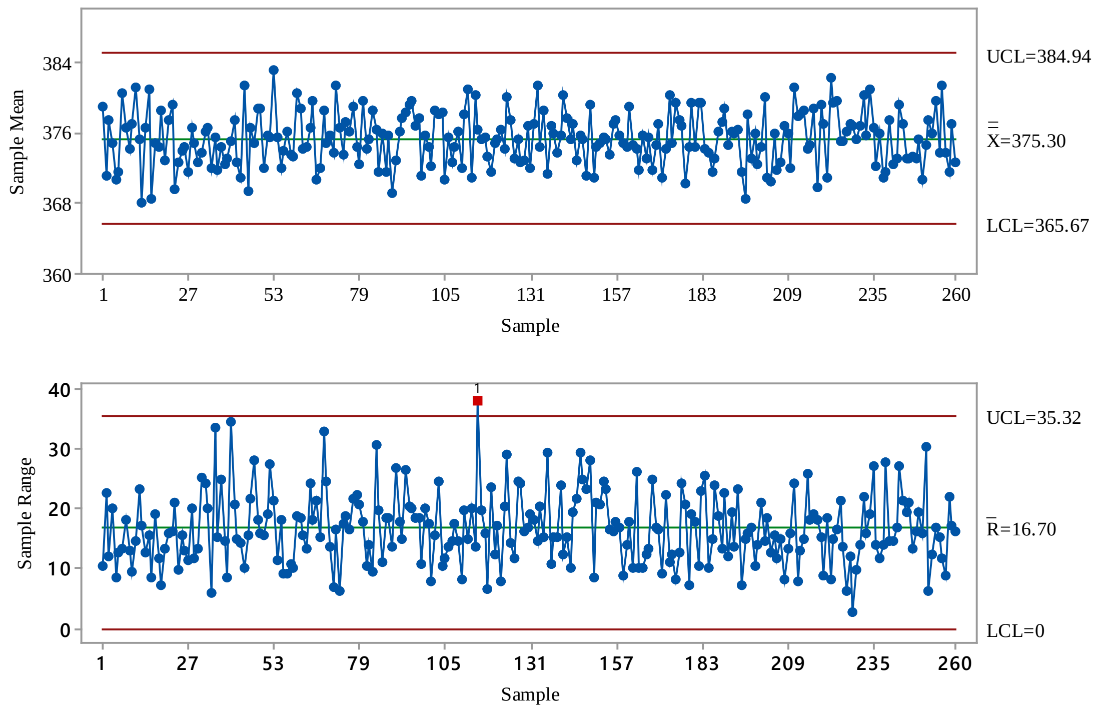


(c) utilization of UKHC

Figure 4.3.1 X-bar and R chart for utilization



(a) Patient flowtime generated by CFD heuristic



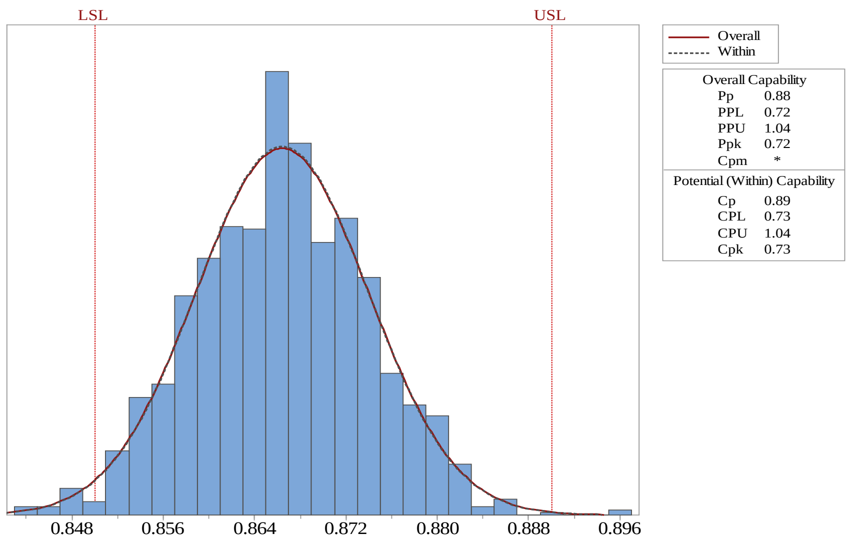
(b) Patient flowtime generated by UKHC

Figure 4.3.2 X-bar and R chart for patient flowtime

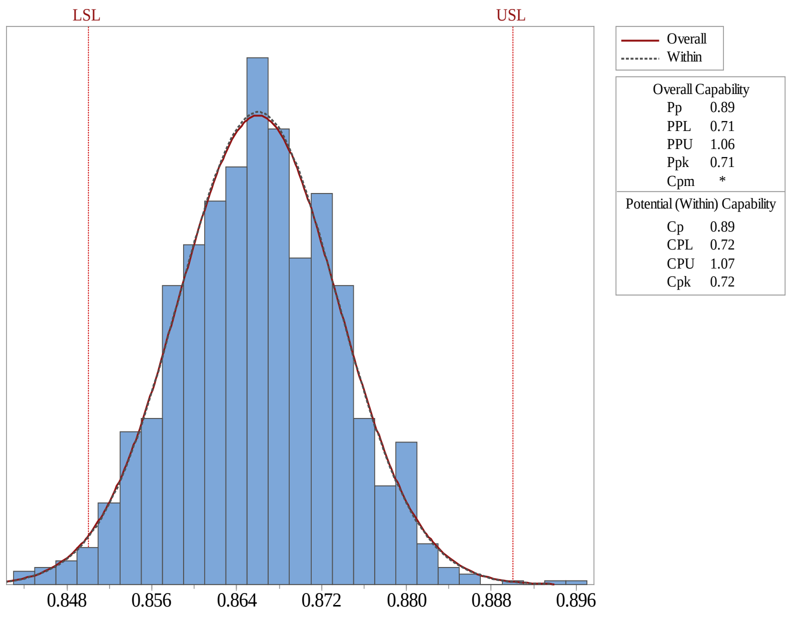
From the Figure 4.3.1 (a) and (b), the average utilization value (i.e. the value of $\overline{\overline{X}}$) that generated by CFD is slightly larger than the UKHC. Moreover, the upper control limit and lower control limit of CFD are both slightly larger than UKHC. However, for both CFD and UKHC, the out of control points are generated. In Figure 4.3.2 (a) and (b), we plot the X-bar and R chart for the patient flow criteria. It shows that our heuristic achieves the lower average patient flowtime and the range of upper control limit and lower control limit is narrower than UKHC. Based on the X-bar chart, the we can see that our CFD heuristic can obtain a patient flow of 338.23, which is smaller than UKHC of 375.30. It means the improvement of ((375.30-338.23))⁄338.23= 0.1096 = 10.96%. From the R-chart, our CFD heuristic does not generate any out of control points. However, there is a out of control point for UKHC.

In addition, we generate the process capabilities for both CFD and UK Healthcare and the results are shown in Figure 4.4.3. In this case study, the process capabilities indicator c_p and c_{pk} can be calculated by following equations: $c_p = \frac{USL-LSL}{6\sigma}$ and $c_{pk} = \min(\frac{USL-\mu}{3\sigma}, \frac{\mu-\text{LSL}}{3\sigma})$, where the μ is the average value of output (i.e. utilization and patient flowtime) and σ is the standard deviation. Moreover, the USL and LSL are upper and lower specification limit.

According to the definition of c_p and c_{pk}, the output of a process is more under control, if the value of c_p is small, and a small c_{pk} value means that the output of a process is more concentrate to the μ. Therefore, given the results in Figure 4.3.3 (a), we can see that the c_p of CFD heuristic is equal to the UK Healthcare on utilization optimization objective, but the c_{pk} of CFD heuristic is smaller than UK Healthcare. It means that the output (i.e. utilization) of CFD heuristic is more concentrate to μ.

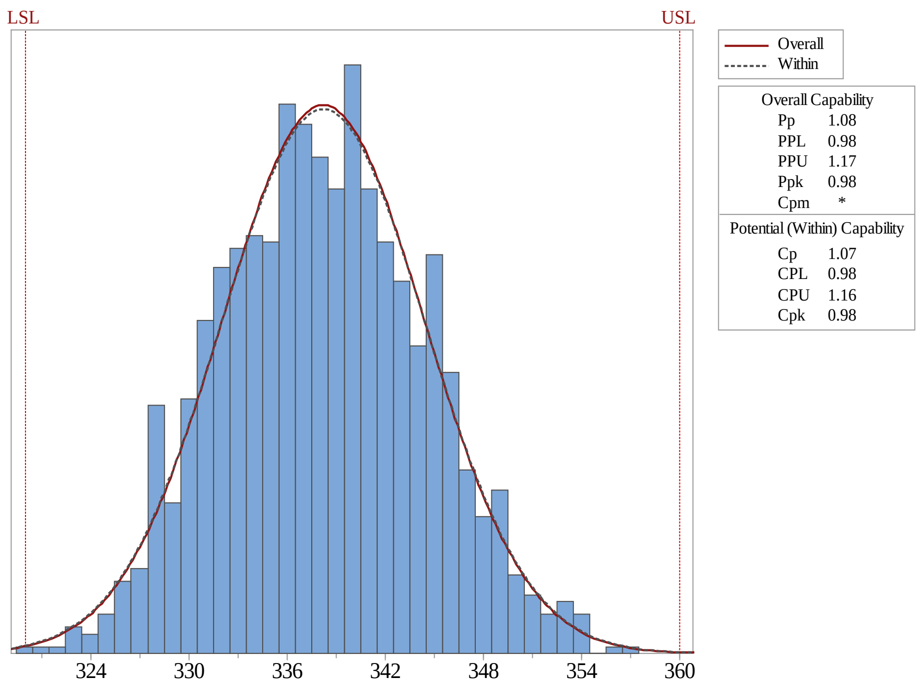


(a) Process capability of CFD for utilization optimization objective

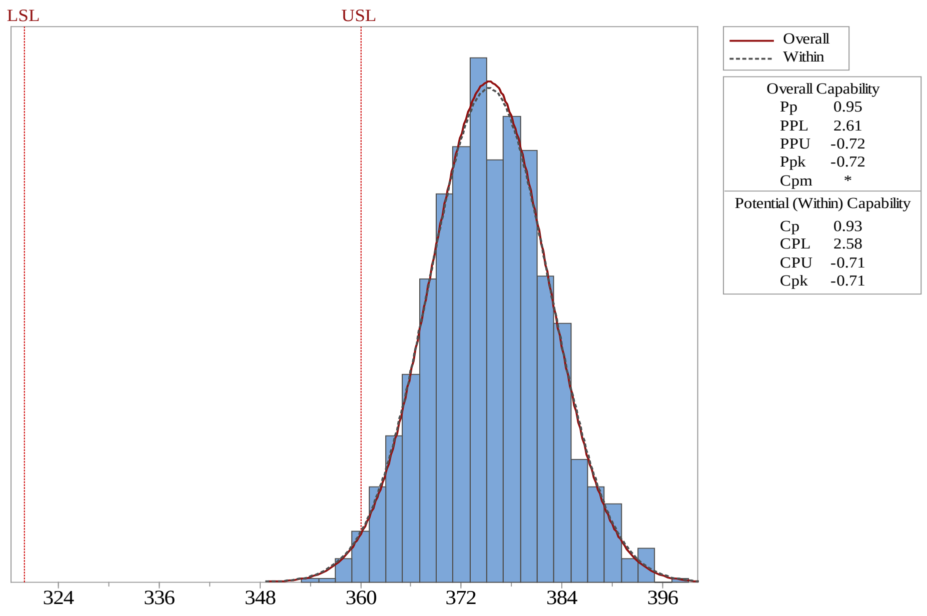


(b) Process capability of UKHC for utilization optimization objective.

Figure 4.3.3 Process capability for utilization optimization objective.



(a) Process capability of CFD for patient flowtime optimization objective



(b) Process capability of UKHC for patient flowtime optimization objective

Figure 4.3.4 Process capability for patient flowtime optimization objective.

According to the results of patient flowtime that shown in Figure 4.4.3(b), the c_p of CFD heuristic is 1.07, which is greater than the UK Healthcare of 0.93. Based on the figure, it easy to identify that the patient flowtime that generated by UK Healthcare is not under control. Furthermore, the c_{pk} value of our CFD heuristic is 0.98, and it is also larger than UK Healthcare on patient flowtime optimization objective.

Therefore, from the X-bar and R charts, our CFD heuristic can generate better performance for utilization and patient flow time optimization objectives. Furthermore, given the value of process capabilities index, we can see that the performance of CFD heuristic is more under control than UKHC in a long-term period.

Chapter Five: Conclusion and Future work

5.1 Conclusion

As a classical scheduling problem, the flow shop scheduling has been researched for many years. There are hundreds of heuristics and algorithms were developed in the last few decades, such as the Raj, LR, NEH heuristics and a lot of evolutionary algorithms. These heuristics and algorithms generated good solutions for makespan minimization or flowtime minimization objective. However, one weakness of these heuristics and algorithms is that they only optimizing single objective instead of multi-objective optimization. For example, NEH is designed to minimize the makespan, and LR is a heuristic which is designed to minimize the flowtime objective. In real-life problems, the schedulers and decision makers might need to consider multi-objective optimization. As we mentioned in the thesis, although there are various criteria can be applied to generate a solid performance evaluation for a flow shop scheduling problem, the makespan and flowtime are the fundamental criteria and others can be related to these two criteria. Furthermore, it has been proved that these two objectives are not consistent. Therefore, there are trade-offs between makespan minimization and flowtime minimization objectives. According to the Pinnedo's idea, the makespan and flowtime are related to the utilization and work-in-process. Usually, the decision makers might want to improve the utilization and reduce the level of work-in-process, because the production cost is impacted by the utilization and the work-in-process affects the inventory cost. Therefore, balancing trade-offs between production cost and holding cost is critical for production and operations management

Based on this perspective, the new heuristic is proposed to balance the trade-off between the makespan and flowtime (i.e. utilization and work-in-process). In this thesis, we proposed a current and future deviation (CFD) deviation heuristic to balance the trade-off between makespan and flowtime minimizations. In the CFD heuristic, we first generate the lower and upper bounds of completion time. Then, we proposed an initial sequence generation method based on the deviations from lower or upper bounds. To further improve the solutions, we developed a new normalized evaluation scheme which named as relative deviation increment value (RIV) and applied this scheme in the job insertion (i.e. the second phase of CFD heuristic) to improve the initial sequence.

In the current literature, the NEH heuristic is the best constructive heuristic to minimize makespan, and the LR heuristic is the best to minimize flowtime. In this thesis, the comparison of the CFD heuristic with NEH and LR and the computational experiments are carried on random small cases and Taillard's benchmark database. From the results of the case studies, our heuristic generates the best performance among three heuristics (i.e. CFD, NEH, LR) on makespan minimization, flowtime minimization and trade-off balancing objective. For small-scale cases Our proposed $\mathrm{CFD}\,(\alpha = 1)$ and $\mathrm{CFD}\,(\alpha = 0)$ heuristics can obtain minimum average relative percentage deviation (ARPD) of makespan and flowtime of 1.27% and 0.9% respectively. Furthermore, we also carried the case study on the Taillard's benchmark dataset (large-scale cases). Our proposed heuristic also provides the best performance on both makespan and flowtime minimization objectives with 3.32% and 2.10% correspondingly. For the trade-off balancing objective, the minimum trade-off values are provided by $\mathrm{CFD}(\alpha = 0.5)$ with 0.313 and 0.479 for small and large test-bed respectively.

In order to justify the effectiveness of CFD heuristic on real-life flow shop scheduling problem, we applied our heuristic to solve the operation room scheduling problems. In the case study, we model the perioperative period as a three-machine flow shop scheduling problem. The processing time data of these three stages of perioperative is obtained from UK healthcare and applied in our case study. We used the statistic process control (SPC) method to evaluate the performance of CFD in a long-term period. From the x-bar R chart, the CFD heuristic achieves higher utilization level of 0.8664 and lower patient flow value of 338.23 than the value of the original method that applied by UKHC. In addition, the range of upper and lower control limit is much narrower than UKHC. It means that our CFD heuristic has a more stable performance on utilization and patient flow objective for long-term scheduling.

5.2 Limitation and Future work

As the main concept of proposed heuristic is based on the deviation from the lower bound and upper bound, the accuracy of bounds is very important during the development of heuristic. Currently, the lower and upper bounds are fixed in our CFD heuristic. It means that the lower and upper bound will not be updated when a job is appended to the scheduled job set. However, the accuracy of the bounds can be improved, if we recalculate the bounds after a job is appended to the current sequence.

What’ more, the adaptive CFD heuristic can be proposed to solve the stochastic problems while the processing times are not deterministic. Another future work is to integrate the CFD heuristic into the operating room (OR) schedule with other constraints, such as surgery type(Abedini, Ye, & Li, 2016) and priority blocking(Abedini, Li, & Ye, 2017) and allocation of OR block times(Aringhieri, Landa, Soriano, Tànfani, & Testi, 2015).

References

Abedini, A., Li, W., & Ye, H. (2017). An optimization model for operating room scheduling to reduce blocking across the perioperative process. *Procedia Manufacturing, 10*, 60-70.

Abedini, A., Ye, H., & Li, W. (2016). Operating Room Planning under Surgery Type and Priority Constraints. *Procedia Manufacturing, 5*, 15-25.

Abouei Ardakan, M., Hakimian, A., & Rezvan, M. T. (2013). A branch-and-bound algorithm for minimising the number of tardy jobs in a two-machine flow-shop problem with release dates. *International Journal of Computer Integrated Manufacturing, 27*(6), 519-528. doi:10.1080/0951192x.2013.820349

Allahverdi, A., Tariq Aldowaisan. (2002). New heuristics to minimize total completion time in m-machine flowshops. *International Journal of Production Economics, 77*(1), 71-83.

Aringhieri, R., Landa, P., Soriano, P., Tànfani, E., & Testi, A. (2015). A two level metaheuristic for the operating room scheduling and assignment problem. *Computers & Operations Research, 54*, 21-34.

Baskar, A. (2016). Revisiting the NEH algorithm- the power of job insertion technique for optimizing the makespan in permutation flow shop scheduling. *International Journal of Industrial Engineering Computations*, 353-366. doi:10.5267/j.ijiec.2015.9.001

Campbell, H. G., Dudek, R. A., & Smith, M. L. (1970). A heuristic algorithm for the n job, m machine sequencing problem. *Management science, 16*(10), B-630-B-637.

Cardoen, B., Demeulemeester, E., & Beliën, J. (2010). Operating room planning and scheduling: A literature review. *European journal of operational research, 201*(3), 921-932.

Chakraborty, U. K., and Dipak Laha. (2007). An improved heuristic for permutation flowshop scheduling. *International Journal of Information and communication technology, 1*(1), 89-97.

Dannenbring, D. G. (1977). An evaluation of flow shop sequencing heuristics. *Management science, 23*(11), 1174-1182.

Denton, B., Viapiano, J., & Vogl, A. (2007). Optimization of surgery sequencing and scheduling decisions under uncertainty. *Health care management science, 10*(1), 13-24.

Fernandez-Viagas, V., & Framinan, J. M. (2014). On insertion tie-breaking rules in heuristics for the permutation flowshop scheduling problem. *Computers & Operations Research, 45,* 60-67. doi:10.1016/j.cor.2013.12.012

Fernandez-Viagas, V., Ruiz, R., & Framinan, J. M. (2017). A new vision of approximate methods for the permutation flowshop to minimise makespan: State-of-the-art and computational evaluation. *European journal of operational research, 257*(3), 707-721. doi:10.1016/j.ejor.2016.09.055

Framinan, J. M., & Leisten, R. (2003). An efficient constructive heuristic for flowtime minimisation in permutation flow shops. *Omega, 31*(4), 311-317. doi:10.1016/s0305-0483(03)00047-1

Framinan, J. M., & Leisten, R. (2007). A multi-objective iterated greedy search for flowshop scheduling with makespan and flowtime criteria. *OR Spectrum, 30*(4), 787-804. doi:10.1007/s00291-007-0098-z

Framinan, J. M., Leisten, R., & Rajendran, C. (2003). Different initial sequences for the heuristic of Nawaz, Enscore and Ham to minimize makespan, idletime or flowtime in the static permutation flowshop sequencing problem. *International Journal of Production Research, 41*(1), 121-148. doi:10.1080/00207540210161650

Framinan, J. M., Leisten, R., & Ruiz-Usano, R. (2002). Efficient heuristics for flowshop sequencing with the objectives of makespan and flowtime minimisation. *European journal of operational research, 141*(3), 559-569.

Garey, M. R., Johnson, D. S., & Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of operations research, 1*(2), 117-129.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of discrete mathematics, 5*, 287-326.

Gupta, A., & Chauhan, S. R. (2015). A heuristic algorithm for scheduling in a flow shop environment to minimize makespa. *International Journal of Industrial Engineering Computations, 6*(2), 173-184. doi:10.5267/j.ijiec.2014.12.002

Gupta, J. N. (1971). A functional heuristic algorithm for the flowshop scheduling problem. *Journal of the Operational Research Society, 22*(1), 39-47.

Ho, J. C. (1995). Flowshop sequencing with mean flowtime objective. *European journal of operational research, 81*(3), 571-578.

Ho, J. C., & Chang, Y.-L. (1991). A new heuristic for the n-job, M-machine flow-shop problem. *European journal of operational research, 52*(2), 194-202.

Johnson, S. M. (1954). Optimal two-and three-stage production schedules with setup times included. *Naval Research Logistics (NRL), 1*(1), 61-68.

Kalczynski, P. J., & Kamburowski, J. (2007). On the NEH heuristic for minimizing the makespan in permutation flow shops. *Omega, 35*(1), 53-60.

King, J., & Spachis, A. (1980). Heuristics for flow-shop scheduling. *International Journal of Production Research, 18*(3), 345-357.

Laha, D., & Sarin, S. (2009). A heuristic to minimize total flow time in permutation flow shop☆. *Omega, 37*(3), 734-739. doi:10.1016/j.omega.2008.05.002

Lei, D., & Guo, X. (2015). A parallel neighborhood search for order acceptance and scheduling in flow shop environment. *International Journal of Production Economics, 165*, 12-18. doi:10.1016/j.ijpe.2015.03.013

Li, W., Freiheit, T., & Miao, E. (2016). A lever concept integrated with simple rules for flow shop scheduling. *International Journal of Production Research, 55*(11), 3110-3125. doi:10.1080/00207543.2016.1246762

Li, W., Mitchell, V. L., & Nault, B. R. (2014). *Inconsistent Objectives in Operating*

Room Scheduling. Paper presented at the IIE Annual Conference. Proceedings.

Li, X., & Li, M. (2015). Multiobjective Local Search Algorithm-Based Decomposition for Multiobjective Permutation Flow Shop Scheduling Problem. *IEEE Transactions on Engineering Management, 62*(4), 544-557. doi:10.1109/tem.2015.2453264

Liu, J., & Reeves, C. R. (2001). Constructive and composite heuristic solutions to the P//∑ Ci scheduling problem. *European journal of operational research, 132*(2), 439-452.

Marcon, E., & Dexter, F. (2006). Impact of surgical sequencing on post anesthesia care unit staffing. *Health care management science, 9*(1), 87-98.

Marcon, E., & Dexter, F. (2007). An observational study of surgeons' sequencing of cases and its impact on postanesthesia care unit and holding area staffing requirements at hospitals. *Anesth Analg, 105*(1), 119-126. doi:10.1213/01.ane.0000266495.79050.b0

Marichelvam, M. K., Prabaharan, T., & Yang, X. S. (2014). A Discrete Firefly Algorithm for the Multi-Objective Hybrid Flowshop Scheduling Problems. *IEEE Transactions on Evolutionary Computation, 18*(2), 301-305. doi:10.1109/tevc.2013.2240304

Meskens, N., Duvivier, D., & Hanset, A. (2013). Multi-objective operating room scheduling considering desiderata of the surgical team. *Decision Support Systems, 55*(2), 650-659. doi:10.1016/j.dss.2012.10.019

Nagano, M., & Moccellin, J. (2002). A high quality solution constructive heuristic for flow shop sequencing. *Journal of the Operational Research Society, 53*(12), 1374-1379.

Nawaz, M., Enscore, E. E., & Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega, 11*(1), 91-95.

Palmer, D. (1965). Sequencing jobs through a multi-stage process in the minimum total time—a quick method of obtaining a near optimum. *Journal of the Operational Research Society, 16*(1), 101-107.

Pan, Q.-K., & Ruiz, R. (2013). A comprehensive review and evaluation of permutation flowshop heuristics to minimize flowtime. *Computers & Operations Research, 40*(1), 117-128. doi:10.1016/j.cor.2012.05.018

Pinedo, M. (2012). *Scheduling*: Springer.

Rad, S. F., Ruiz, R., & Boroojerdian, N. (2009). New high performing heuristics for minimizing makespan in permutation flowshops. *Omega, 37*(2), 331-345. doi:10.1016/j.omega.2007.02.002

Rajendran, C. (1993). Heuristic algorithm for scheduling in a flowshop to minimize total flowtime. *International Journal of Production Economics, 29*(1), 65-73.

Rajendran, C. (1994). A heuristic for scheduling in flowshop and flowline-based manufacturing cell with multi-criteria. *THE INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH, 32*(11), 2541-2558.

Rajendran, C. (1995). Heuristics for scheduling in flowshop with multiple objectives.

European journal of operational research, 82(3), 540-555.

Rajendran, C., & Chaudhuri, D. (1992). An efficient heuristic approach to the scheduling of jobs in a flowshop. *European journal of operational research, 61*(3), 318-325.

Rand, G. K. (1977). Machine scheduling problems: Classification, complexity and computations: AHG RINNOOY KAN, Martinus Nijhoff, The Hague, 1976, ix+ 180 pages: North-Holland.

Ravindran, D., Selvakumar, S. J., Sivaraman, R., & Haq, A. N. (2004). Flow shop scheduling with multiple objective of minimizing makespan and total flow time. *The International Journal of Advanced Manufacturing Technology, 25*(9-10), 1007-1012. doi:10.1007/s00170-003-1926-1

Ruiz, R., & Maroto, C. (2005). A comprehensive review and evaluation of permutation flowshop heuristics. *European journal of operational research, 165*(2), 479-494. doi:10.1016/j.ejor.2004.04.017

Ruiz, R., & Stützle, T. (2007). A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European journal of operational research, 177*(3), 2033-2049.

Ruiz, R., & Vázquez-Rodríguez, J. A. (2010). The hybrid flow shop scheduling problem. *European journal of operational research, 205*(1), 1-18.

Sarin, S., and M. Lefoka. (1992). Scheduling heuristic for the n-jobm-machine flow shop. *Omega, 21*(2), 229-234.

Sayadi, M. K., Ramezanian, R., & Ghaffari-Nasab, N. (2010). A discrete firefly meta-heuristic with local search for makespan minimization in permutation flow shop scheduling problems. *International Journal of Industrial Engineering Computations, 1*(1), 1-10. doi:10.5267/j.ijiec.2010.01.001

Singhal, E., Singh, S., & Dayma, A. (2012). An Improved Heuristic for Permutation Flow Shop Scheduling. *NEH ALGORITHM). International Journal of Computational Engineering Research, 2*(6), 95-100.

Taillard, E. (1990). Some efficient heuristic methods for the flow shop scheduling. *European journal of operational research, 47,* 65-74.

Varadharajan, T., & Rajendran, C. (2005). A multi-objective simulated-annealing algorithm for scheduling in flowshops to minimize the makespan and total flowtime of jobs. *European journal of operational research*, *167*(3), 772-795.

Woo, H.-S., & Yim, D.-S. (1998). A heuristic algorithm for mean flowtime objective in flowshop scheduling. *Computers & Operations Research, 25*(3), 175-182.

Vita

Name: Feidi Dang
Education:

University of Kentucky	Lexington, Kentucky
Master of Science	Aug,2015~Now
Department of Mechanical Engineering	
Xian University of Science and Technology	Xi'an, China
Bachelor of Engineering	Aug,2010—May,2014
Department of Mechanical Engineering	

Research interests:
Optimization, Operation research, Flowshop scheduling.

Publication:

Under review:

Dang, F.D., Li, W., Ye, H.H. An efficient constructive heuristic to balance trade-offs between makespan and flowtime in permutation flow shop scheduling. *North American Manufacturing Research Conference.*

Preparation:

Li, W., Dang, F.D., Abedini, A., Ye, H.H. Balancing trade-offs between maximum and total completion times in permutation flow shop scheduling (Working paper, in preparation for submission to OMEGA-International Journal of Management Science)