2012

# Mathematical Modelling and Optimization of Flexible Job Shops Scheduling Problem

Vahid Roshanaei
*University of Windsor*

# Mathematical Modelling and Optimization of Flexible Job Shops Scheduling Problem

by

Vahid Roshanaei

A Thesis

Submitted to the Faculty of Graduate Studies

through Industrial & Manufacturing Systems Engineering

in Partial Fulfillment of the Requirements for

the Degree of Master of Science at the

University of Windsor

Windsor, Ontario, Canada

# Mathematical Modelling and Optimization of Flexible Job Shops Scheduling Problem

by

Vahid Roshanaei

APPROVED BY

---

Dr. Richard Caron

Department of Mathematics and Statistics

---

Dr. Fazle Baki

Odette School of Business

---

Dr. Hoda ElMaraghy, Co-supervisor

Department of Industrial & Manufacturing Systems Engineering

---

Dr. Ahmed Azab, Co-supervisor

Department of Industrial & Manufacturing Systems Engineering

---

Dr. Shervin Erfani, chair of thesis defense

May 17th 2012

# Author's Declaration of Originality/Previous Publications

This thesis includes two original papers that have been previously published in peer reviewed conferences as follows:

| Thesis Chapter | Publication title/full citation | Status |
|---|---|---|
| Chapter 3 | V. Roshanaei, H. ElMaraghy, A. Azab, 2012, Sequence-based MILP Modeling for Flexible Job Shop Scheduling. Proceedings of IIE Conference & Expo 2012, Orlando, Florida, USA, May 19-23, 2012. | Published |
| Chapter 3 | V. Roshanaei, H. ElMaraghy, A. Azab, 2012, Enhanced Mixed Integer Linear Programming for Flexible Job Shop Scheduling, Proceedings of the 4th CIRP Conference on Assembly Technologies & Systems (CATS), and 195-198. Ann Arbor, Michigan, USA, May 21-22, 2012 | Published |

I hereby certify that I have obtained a written permission from the copyright owner(s) to include the above published material(s) in my thesis. I certify that the above material describes work completed during my registration as graduate student at University of Windsor.

I declare that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such materials in my major paper and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my committee and the Graduate Studies office, and that this major paper has not been submitted for a higher degree to any other University or Institution.

# Abstract

The flexible job shop scheduling problem (F-JSSP) is mathematically formulated. One novel position-based and three sequence-based mixed integer linear programming models are developed. Since F-JSSPs are strongly NP-hard, MILPs fail to solve large-size instances within a reasonable timeframe. Thus, a meta-heuristic, a hybrid of artificial immune and simulated annealing (AISA), is developed for use with larger instances of the F-JSSP.

To prove the efficiency of developed MILPs and AISA, they are compared against state-of-the-art MILPs and meta-heuristics in literature. Comparative evaluations are conducted to test the quality and performance of the developed models and solution technique respectively. To this end, size complexities of the developed MILPs are investigated. The acquired results demonstrate that the proposed MILPs outperform the state-of-the-art MILP models in literature. Likewise, the proposed AISA outperforms all the previously developed meta-heuristics. The developed AISA has successfully been applied to a realistic case study from mould and die industry.

# Dedication

## To Akram

who optimally covers my heart with no absolute uncertainty

# Acknowledgments

I would like to express my deepest gratitude to my supervisors-Professor Hoda ElMaraghy and Dr. Ahmed Azab for their unmitigated and immense support and guidance during my graduate study and thesis preparation. It is quite gratifying to work under their supervision. I will eternally remember this life-changing experience to work with them.

Special thanks go to Professor Richard Caron and Dr. Fazle Baki for their careful reading of my thesis and provision of valuable comments on my thesis.

Thanks to my supervisors for the Research Assistantship (RA) they provided and to the Department of Industrial & Manufacturing Systems Engineering (IMSE) for the Graduate Assistantship I received during my graduate study. I specifically offer my outmost gratitude to Professor Wagiuh ElMaraghy-Head of IMSE department- for his professional guidance and unbiased comments.

I also thank my all course instructors who qualified me to accomplish this research. Their generosity in transferring their knowledge is massively appreciated.

My unconditional and undying appreciation rests with my best friend and mentor- Dr. Bahman Naderi from Amirkabir University of Technology in Iran who helped me write CPLEX codes of my mathematical models and meta-heuristic.

Last but not least, my unending appreciation goes out to Lana Saifalla- Editor, student in English Literature in the University of Windsor who substantially helped enhance the readability of 87 pages to their last bit. To her generous favor, I stay indebted forever.

To each of the above, I extend my deepest appreciation.

# Table of contents

# List of Tables

# List of Figures

# List of Abbreviations

BFI: best feasible integer

BIV: number of integer variables

CV: number of continuous variables

F-JSSP: flexible job shop scheduling problem

JSSP: job shop scheduling problem.

LB: lower bound

MILP: mixed integer linear programming

MINLP: mixed integer non-linear programming

NC: number of constraints

PF-JSSP: partially flexible job shop scheduling problem

SDST: Sequence-dependent set-up times

SIST: Sequence-independent set-up times

TF-JSSP: totally flexible job shop scheduling problem

AIA: Artificial immune algorithm

SA: Simulated Annealing

TS: Tabu Search

RPD: Relative percentage deviation

# List of nomenclatures for position-based MILP models

| | |
|---|---|
| $j$ | Subscript for parts where $1 \leq j \leq n$ |
| $i$ | Subscript for machines where $1 \leq i \leq m$ |
| $l$ | Subscript for operations of part $j$ where $1 \leq l \leq n_j$ |
| $e_{j,l,i}$ | Eligibility parameter that takes value 1 if machine $i$ is able to process operation $O_{j,l}$ and 0 otherwise. |
| $p_{j,l,i}$ | Processing time of operation $O_{j,l}$ on machine $i$ |
| $O_{j,l}$ | $l$ -th operation of job $j$ |
| $f_i$ | Subscript for processing positions of machine $i$ where $1 \leq f \leq f_i$ $(f_i = \sum_j \sum_l e_{j,l,i})$ |
| $M$ | A large positive number |
| $X_{j,l,i,f}$ | Binary decision variable taking value 1 if *l-th* operation of part $j$ is processed on the *f-th* position of machine $i$ |
| $S_{j,l}$ | Continuous decision variable for starting time of operation $O_{j,l}$ |
| $B_{i,f}$ | Continuous decision variable for beginning time of each processing position |

# List of nomenclatures for sequence-based MILP models

| | |
|---|---|
| $j, h$ | Indices for jobs where $1 \leq j, h \leq n$ |
| $i$ | Indices for machines where $1 \leq i \leq m$ |
| $l, z$ | Indices for operations of job $1 \leq l, z \leq n_j$ |
| $R_{j,l}$ | The set including machines eligible to process $O_{j,l}$ |
| $p_{j,l,i}$ | Processing time of $l$-th operation of job $j$ on machine $i$ |
| $O_{j,l}$ | $l$ -th operation of job $j$ |
| $e_{j,l,i}$ | Parameter that takes value 1 if machine $i$ can process $O_{j,l}$ and 0 otherwise. |
| $M$ | A large positive number |
| $X_{j,l,h,z}$ | Binary variable taking value 1 if $O_{j,l}$ is processed after $O_{h,z}$; and 0 otherwise. $j \epsilon \{1, 2, \dots, n-1\}, h > j$ |
| $X_{j,l,h,z,i}$ | Binary variable taking value 1 if $O_{j,l}$ is processed after $O_{h,z}$ on machine $i$; and 0 otherwise. $j \epsilon \{1, 2, \dots, n-1\}, h > j$ |
| $Y_{j,l,i}$ | Binary variable taking value 1 if $O_{j,l}$ is processed on machine $i$. |
| $C_{j,l}$ | Continuous variable for the completion time of $O_{j,l}$. |
| $S_{j,l}$ | Continuous variable for the starting time of $O_{j,l}$. |
| $C_{j,l,i}$ | Continuous variable for the completion time of $O_{j,l}$ on machine $i$ |
| $S_{j,l,i}$ | Continuous variable for the starting time of $O_{j,l}$ on machine $i$ |

# CHAPTER 1

## ENGINEERING PROBLEM

### 1.1 Background of production scheduling

Thirst for increased productivity in the modern business world has spurred manufacturing practitioners to seek every single opportunity for cost reduction and profit generation (Roshanaei et al. 2012[a]). Over the last six decades, effective production scheduling mechanisms have been recognized to be increasing productivity and machine utilization (Roshanaei et al. 2012[b]). The importance of scheduling as a logical enabler in manufacturing systems has increased recently due to the growing consumer demand for variety, reduced product life cycles, changing markets with global competition, and rapid development of new processes and technologies (Ho, Tay et al. 2007). These economic and market pressures stress the need for minimizing inventory while maintaining customer satisfaction of production and delivery; Thus, this requires efficient, effective and accurate scheduling. Algorithmic and scientific production scheduling came to existence once the first production scheduling heuristic technique was proposed by (Johnson 1954). Research in the area of production scheduling usually starts with single machine scheduling and is extended to scheduling of highly complex shop floors like changeable, reconfigurable, and flexible manufacturing systems abbreviated as (CMS), (RMS) and (FMS) respectively.

The scheduling problem studied in this thesis is a special case of FMS referred to as flexible job shop scheduling problem (F-JSSP) which is usually encountered in industries with high product variety and medium demand for each product. F-JSSP extends classical job shop scheduling problem (JSSP) by assuming that each machine is flexible and able to offer more than one particular capability. Therefore, in order to better understand the F-JSSP, introductory definitions of JSSP are required.

### 1.2 Introduction to job shop scheduling problem

In order to outline the problem studied in this thesis, certain definitions are required. Therefore, the main production configuration- Job Shop- is first addressed, and then clarified as to how a generalization of the main problem is created and solved. Hence, this thesis is commenced with the definition of Job Shop Scheduling Problem (JSSP) and its mechanics.

In the JSSP, there are *n* jobs *($J_1$, ....., $J_n$)* with varying sizes and each job needs to be scheduled on *m ($m_1$,....., $m_m$)* machines each job follows a predetermined operational route until all the operational requirements of all jobs are fulfilled. In the classical JSSP, the process plan of a part consists of the sequence of the machines the part must visit: there is an a priori assignment of operations to machines. Therefore, the process plan is fixed and no process plan flexibility is associated.

In order to explain how F-JSSP works, all sources of flexibilities in manufacturing systems are reviewed. Eventually, the related flexibility is addressed and the rest of details of F-JSSP are given.

## 1.3 Sources of flexibility in manufacturing systems:

Review of literature identifies at least ten types of manufacturing systems flexibilities (ElMaraghy 2005). They are as follows:

1. *Machine flexibility*: Various operations performed without set-up change,
2. *Material handling flexibility*: Number of used paths / total number of possible paths between all machines,
3. *Operational Flexibility*: Number of different processing plans available for part fabrication,
4. *Process Flexibility*: Set of part types that can be produced without major set-up changes, i.e. part-mix flexibility,
5. *Product Flexibility*: Ease (time and cost) of introducing products into an existing product mix. It contributes to agility,
6. *Routing Flexibility*: Number of feasible routes of all part types/Number of part types,
7. *Volume Flexibility*: The ability to vary production volume profitably within production capacity,
8. *Expansion Flexibility*: Ease (effort and cost) of augmenting capacity and/or capability, when needed, through physical changes to the system,
9. *Control Program Flexibility*: The ability of a system to run virtually uninterrupted (e.g. during the second and third shifts) due to the availability of intelligent machines and system control software,
10. *Production Flexibility*: Number of all part types that can be produced without adding major capital equipment.

This classification promotes better understanding of various types of flexibility although some of them are inter-related. Different sources of flexibilities of manufacturing systems

were addressed above. Among previously enumerated sources of flexibility in manufacturing systems, machine and routing flexibility are incorporated into classical JSSP. Taking into account the before-cited sources of flexibility, the considered problem is transformed from classical JSSP to F-JSSP.

## 1.4 Introduction to flexible job shop scheduling problem

Flexible Job shop scheduling problem (F-JSSP) is challenging due to expanding machine tools capabilities and increased products variety. Full utilization of added capabilities, versatilities and increased flexibilities in job shops makes scheduling these resources extremely challenging. The growing competition in international markets has generated demand for faster and more versatile machine tools, while preserving or improving the final product quality. The emergence of multi-purpose machineries provided manufacturers with competitive capabilities. However, managing changes in products and markets requires adaptation at two levels by developing physical enablers of change such as reconfigurable machines and systems and logical enablers including adaptable and re-configurable controls, process planning, production planning and scheduling (Wiendahl, ElMaraghy et al. 2007). Production planning and scheduling (PPS) map the production load of a factory to its capabilities and capacities in different time horizons and levels of detail. The two levels of PPS are coupled since planning sets the goals, as well as the resource and temporal constraints for scheduling (Vancza, Kis et al. 2004). Scheduling is responsible for unfolding a plan into detailed resource assignments and sequences. Scheduling the production in real industrial environment presents additional challenges of size, which is normally larger than the capabilities of most existing algorithms and typical benchmarks in literature.

## 1.5 Mechanics of F-JSSP

The manufacturing setting studied in this paper is a generalized variant of the classical job shop production systems known as F-JSSP. F-JSSP extends the job shop production systems by assuming that each machine is capable of offering more than one operation. According to (Kacem, Hammadi et al. 2002), flexibility in job shop, which refers to machine flexibility, may be partial or total - referred to as Partially Flexible Job Shop Scheduling Problem (PF-JSSP) and Totally Flexible Job Shop Scheduling Problem (TF-JSSP) respectively. PF-JSSP is a special case of F-JSSP wherein the preferences or feasibility of using some machines for certain operations arise. In PF-JSSP, there exist certain numbers of multi-purpose machines

distributed throughout the facility, the versatility and flexibility of which are not identical. This feature enables a certain part to be processed by at least one machine out of the available feasible machines. The routing flexibility of parts permits dynamic re-assignment of parts to other available machines in case of facing any dynamic event in the shop floor like machine breakdowns, order cancellation or arrival etc. Alternate routing is useful where capacity problems arise. In PF-JSSP, there are $m$ machines in the system and $n$ jobs to be processed. Each job $j$ requires $n_j$ precedence-constrained operations to be performed. Each operation $O_{j,l}$ can be processed on a number of non-identical or identical machines and the processing time differs based on the machine characteristics. This addresses the existence of multiple routings for some jobs. An alternate routing could be used if one machine tool is temporarily overloaded while another feasible one is available. Therefore, based on (Brandimarte 1993), in F-JSSP, two distinct decisions have to be made:

- *Assigning Operations to Machines*: in this stage, operations are assigned to their respective feasible machines. In TF-JPPS, any arbitrary assignment of operations to available machines is feasible as all the machines in the shop floor possess the same operational capabilities. But, in the PF-JSSP which represents the structure of our industrial problem as well, attention should be paid to the feasibility of machine assignments as machines do not possess identical tool-magazines.
- *Sequencing of operations*: once assignment decisions were made, sequencing decision is triggered. It goes without saying that sequencing decision is made for those operations whose processing route entails sharing the same machine, i.e., those operations that are not assigned to the same machine are not considered for sequencing with respect to each other on that particular machine. Therefore, sequencing decision is made for those operations sharing the same subset of machines. In other words, there is no justification to sequence operations while they have not been assigned to the same machine.

## 1.6 Assumptions for solving F-JSSP

The following assumptions are used for all mathematical formulations and meta-heuristics developed in this thesis:

- Optimal singular process plan is determined a priori for each part type. i.e., no process plan flexibility is considered.

- Certain operations can be processed on more than one machine, i.e., there exists routing flexibility.
- Jobs are independent and no priorities are assigned.
- Pre-emption or cancellation of jobs is not considered.
- Each machine can only process one job at a time.
- Each job can be processed by only one machine at a time.
- Processing times are deterministic and include set-up, operations, transportation and inspection (approval).
- All jobs are inspected a priori i.e., no defective part is considered.
- The orders volume is known a priori and jobs are simultaneously available at time zero because the purchase of raw material is done at the same time.
- Breakdowns are not considered.

## 1.7 Representation of F-JSSP using standard triplet

In order to facilitate the solution, the PF-JSSP is transformed to the TF-JSSP by adding "infinite processing times" referred to as big M to the incapable machines. Scheduling problems are usually represented by a standard triplet (α | β | γ). According to (Pinedo 2002), the TF-JSSP can be denoted by *FJc|Cmax* but the PF-JSSP is represented as follows: $FJc|Mi\ |C_{max}$. The first symbol indicates the type of shop which is F-JSSP, while the second symbol indicates machine eligibility issue. Finally the third symbol denotes the objective function which is make-span. The problem studied in this paper encompasses both TF-JSSP and PF-JSSP.

## 1.8 Graphical representation of TF-JSSP

In Figure1.1, as an example of TF-JSSP, the assignment of three jobs to three multi-purpose machines is depicted. As can be seen, all jobs have total routing flexibility to be assigned to *any* machine in the shop floor. This total flexibility is made possible if all machines have identical operational capabilities. In Figure 1.1, all machines have the same operational capabilities represented by different shapes.

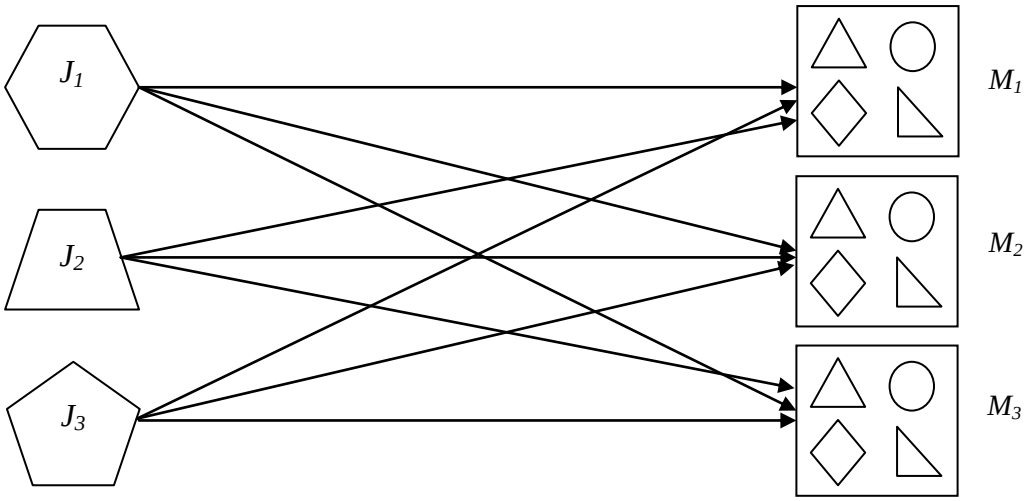


Figure 1.1, schematic representation of TF-JSSP

## 1.9 Graphical representation of PF-JSSP

In Figure 1.2, the assignment of three jobs to three multi-purpose machines is illustrated. As can be seen in Figure 1.2, functionalities and versatilities of different machines are not identical. This simply means that not all jobs have total routing flexibility to be processed by any available machine in the shop floor. The difference in operational capabilities among machines culminates in a phenomenon known as partial routing flexibility.

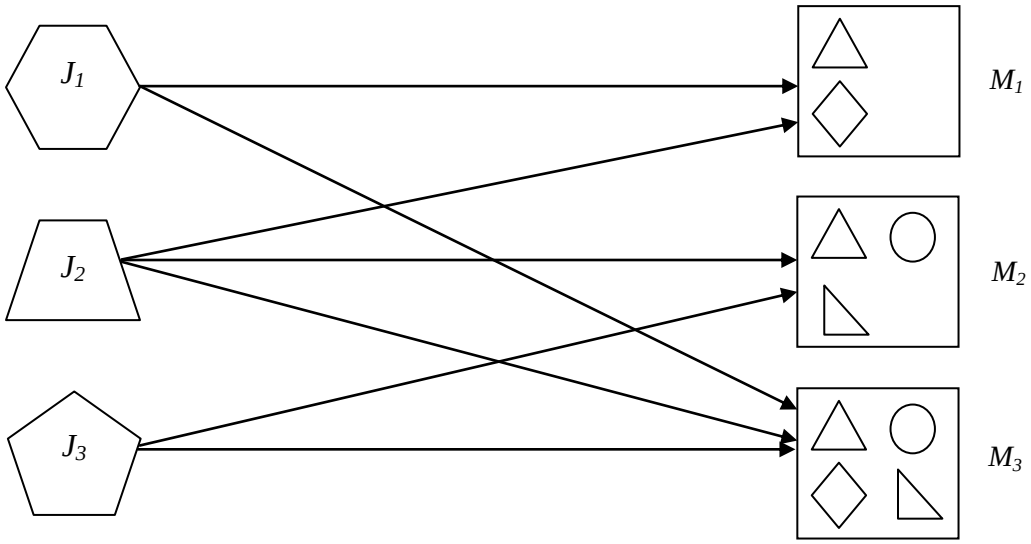


Figure 1.2, Schematic representation of PF-JSSP

## 1.10 Size complexity of F-JSSP

In TF-JSSP, there exist certain numbers of multi-purpose machines ($m$) in the shop floor. On any of these multi-purpose machines as many feasible schedules as the factorial of number

of jobs (n!) can be generated. This is equivalent to the complexity of the single machine scheduling. Since there is more than one machine in the shop floor, parts have to be considered for sequencing on other flexible machines as well. This issue begets tremendous complexity for F-JSSP and makes its complexity rise to (n!)m. The classical JSSP has proven to be strongly NP-hard by (Garey, Johnson et al. 1976). Therefore, the F-JSSP is also NP-hard in strong sense.

## 1.11 General approaches to F-JSSP

Due to the tremendous complexity of the F-JSSP, meta-heuristics are utilized. Two established frameworks for meta-heuristics have been proposed (Brandimarte 1993):

- ***Hierarchical*:** the assignments of operations to their respective machines are initially made and well after that sequencing procedure starts.
- ***Integrated*:** both assigning and sequencing decisions are made concurrently.

A hierarchical approach has been adopted in this work.

## 1.12 Proposed solution methodologies for the F-JSSP

As was mentioned in the abstract, this paper utilizes two widely known solution methodologies to fulfil the scheduling requirements of the industrial problem at hand:

- **Mathematical modelling:** four effective mathematical formulations are presented in form of Mixed Integer Linear Programming (MILP) for both TF-JSSP and PF-JSSP. The MILPs presented in this work follow both the *integrated and hierarchical* approaches to solve the problem at hand. The used approach by each MILP is explained in Chapter 3.
- **Hybrid meta-heuristic:** a hybrid meta-heuristic algorithm resulted from hybridization of Artificial Immune Algorithm (AIA) and Simulated Annealing (SA) referred to as AISA is presented.

## 1.13 Objective function:

Since the chief purpose of this paper is to solve the F-JSSP, several objective functions were examined. Among, all objective functions existent in literature, make-span ($C_{max}$) or maximum was chosen. The significance of this objective function in production scheduling problems is addressed in chapter 3 where mathematical models are explained.

# CHAPTER 2

## REVIEW OF PREVIOUS SOLUTION METHODOLOGIES:

This thesis is aimed at employing two classes of solution methodologies comprising mathematical programming and hybrid meta-heuristics for tackling F-JSSP. Therefore, the literature survey is divided into two subsections:

a) Review of mathematical models which have thus far been proposed for F-JSSP and other related manufacturing environments.
b) Review of original and hybrid meta-heuristics in literature which have been designed for F-JSSP.

### 2.1 Review of mathematical programming formulations:

Mathematical modelling as a solution technique for production scheduling problems founded its application after the seminal work of (Wagner 1959). Ever since, researchers have employed wide variety of objective functions, assumptions, and solution techniques in their studies to formulate and solve production scheduling problems. Ozguven, Yavuz et al. (2012) in their lately complied literature survey, have garnered all the relevant mathematical models for F-JSSP. Below, the updated version of literature survey compiled by (Ozguven, Yavuz et al. 2012) is given. The gap between the first MILP for JSSP and that for F-JSSP is because flexible manufacturing systems (FMS) came to existence in early 90s. Seven years after the advent of multi-purpose machines, the first MILP was proposed by (Jiyin and MacCarthy 1997) for FMSs. They proposed a mixed-integer linear programming model (MILP) for FMS scheduling to minimize mean completion times of operations. The model took into consideration practical constraints on storage and transportation. The complexity of the proposed MILP was also discussed. Two MILP-based heuristic procedures were additionally developed. In their work, the development of iterative global heuristics based on mathematical programming formulations was advocated for a wide class of FMS scheduling problems. Brandimarte (1999) dealt with the machine loading problem in job shop scheduling environment with process and routing flexibility. Two different heuristics were proposed: one based on surrogate duality theory, and one based on a genetic descent algorithm. Kim and Egbelu (1999) addressed the problem of scheduling multiple jobs with each having multiple process plans in a job shop environment. The problem was formulated

and tackled using a mathematical approach. Because of the high computational effort required to solve the model using an exact solution procedure, they developed two algorithms and computed their optimality gap by comparing them with an optimum-finding technique. The results of the test problems showed that one of the algorithms, referred to as the pre-processing algorithm, found the optimal solution in all problem cases. The second algorithm, referred to as the iterative algorithm, was also effective in finding good solutions; however, its overall performance was lower than that of the first algorithm. Low and Wu (2001) studied an operation scheduling problem with the objective of minimizing total tardiness in an FMS with set-up time consideration. The considered problem was first formulated as a binary integer programming model, and was then solved optimally. Subsequently, a heuristic was proposed to solve the problem in an acceptable running time. Computational experiments showed that the proposed SA-based heuristic performed well with respect to solution accuracy and efficiency. Thomalla (2001) proposed a Mixed Integer Non-Linear Programming (MINLP) for scheduling jobs in a just-in-time environment. The non-pre-emptive case where each job consisted of a distinct number of operations to be processed in a specified order was considered. The objective function was to minimize the sum of the weighted quadratic tardiness of the jobs. A fast near-optimal algorithm with guaranteed bounds for the distance to the optimum by using Lagrangian relaxation was proposed and also shown that only one relaxation sufficed to solve the problem. Choi and Choi (2002) proposed an MILP formulation to solve the F-JSSP with alternative operations and sequence-dependent set-up time. They also presented an effective greedy local search which was computationally effective. Make-span is optimized as their objective function. Gomes, Barbosa-Povoa et al. (2005) presented two new MILP models for F-JSSP. The model considered groups of parallel homogeneous machines, limited intermediate buffers and negligible set-up effects. The model allowed re-circulation to take place. As their objective functions, they minimized costs associated with just-in-time due-dates and in-process inventories. Gao, Gen et al. (2006) studied the F-JSSP with machine availability constraints. A novel MINLP was proposed to optimize make-span in F-JSSP. They introduced a hybrid GA to solve the F-JSSP with non-fixed availability constraints (F-JSSP-nfa). Imanipour and Zegordi (2006) addressed the F-JSSP with two types of flexibility in process and sequence planning. He also formulated the F-JSSP with sequence-dependent set-up times as an MINLP. The objective function was the make-span optimization. To solve the model, an effective approach based on Tabu Search (TS) was developed. The effectiveness of the proposed

algorithm was shown via numerical experiments. Low, Yukling et al. (2006) proposed a multi-objective framework For solving FMS scheduling problems with consideration of three performance measures, namely minimum mean job flow time, mean job tardiness, and minimum mean machine idle time. In addition, hybrid heuristics which were combinations of two common local search methods, SA and TS, were also developed for solving the addressed FMS scheduling problems. The feasibility and adaptability of the proposed heuristics were demonstrated through experimental results. Fattahi, Mehrabad et al. (2007) developed the first position-based MILP model based on Wagner's definition of integer variables for F-JSSP. They also proposed six heuristic approaches for F-JSSP. Mathematical model was used to achieve optimal solutions for small-size instances of the problem. Since F-JSSP is an NP-hard problem, two heuristic approaches including integrated and hierarchical ones were developed to solve the real-size problems. Six different hybrid searching structures were presented. Numerical experiments were used to evaluate the performance of the developed algorithms. Saidi-Mehrabad and Fattahi (2007) presented a TS algorithm for F-JSSP to minimize the make-span. They presented a model for the F-JSSP considering alternative operation sequences and sequence-dependent set-ups. The purpose of their paper was to minimize the make-span and find the best sequence of operations and the best choice of machine alternatives simultaneously. Then the computational results were presented and results showed that the proposed algorithm could produce optimal solutions in a reasonable computational time for small-to medium size instances of the problem. Moreover, it could be applied easily in real factory conditions and for large-size problems. Fattahi, Jolai et al. (2009) considered an F-JSSP with a new approach (overlapping in operations). Since this problem is recognized as NP-hard class, a hierarchical approach based on SA was developed to solve large problem instances. Moreover, an MILP method was presented. To validate the proposed SA algorithm, the results were compared with the optimal solution obtained with the traditional optimization technique (Branch and Bound method). The computational results validated the efficiency of the proposed algorithm. Also the computational results show that the consideration of overlapping operations can improve the make-span and machines utilization measures. Ozguven, Ozbakir et al. (2010) dealt with two NP-hard optimization problems: 1) F-JSSP that encompassed routing and sequencing sub-problems and 2) the F-JSSP with process plan flexibility (FJSP-PPFs). The latter additionally included the process plan selection sub-problem. The study was carried out in two steps. In the first step, an MILP model based on Manne's definition of binary

variables was developed for F-JSSPs and compared to an alternative model in the literature (Fattahi, Mehrabad et al. 2007) in terms of computational efficiency and solution effectiveness. In the second step, another MILP model, a modification of the MILP-1, for the FJSP-PPFs was presented along with its computational results on hypothetically generated test problems. Moradi, Fatemi Ghomi et al. (2011) investigated integrated F-JSSP with preventive maintenance (PM) activities under the multi-objective optimization approaches. They attempted to simultaneously optimize two objectives: 1) the minimization of the make-span for the production part, and 2) the minimization of the system unavailability for the maintenance part. Ozguven, Yavuz et al. (2012) studied an advanced form of the F-JSSP which also covered process plan flexibility and separable/ non-separable sequence-dependent set-up times in addition to routing flexibility. Two MILP models were formulated. In the first model (Model A) the sequence-dependent set-up times were non-separable. In the second one (Model B) they were separable. Model B was constructed based on Model A with a minor modification. Finally, computational results were obtained on test problems. The updated tabular version of the literature survey carried out by Ozguven, Yavuz et al. (2012) is reproduced in Table 2.1.

Table2.1. Review of mathematical models for F-JSSP

| Reference | Math models | Problem addressed | Objectives |
|---|---|---|---|
| (Jiyin and MacCarthy 1997) | MILP | F-JSSP with SDST*, transportation times and limited buffers | Mean completion time, Make-span and Maximum tardiness |
| (Brandimarte 1993) | Bi-Criterion MILP | F-JSSP with process plan and routing flexibility | Machine load Cost |
| (Kim and Egbelu 1999) | MILP | JSSP with alternative process plans | Make-span |
| (Tamaki, Ono et al. 2001) | MILP | F-JSSP with SIST** | Make-span, Total tardiness |
| (Low and Wu 2001) | MILP | F-JSSP with SIST | Total tardiness |
| (Thomalla 2001) | MINLP | F-JSSP | Weighted quadratic tardiness |
| (Choi and Choi 2002) | MILP | F-JSSP with SDST | Make-span |
| (Lee, Jeong et al. 2002; Low, Yukling et al. 2006)) | MINLP | F-JSSP with due dates and outsourcing | Make-span |
| (Gomes, Barbosa-Povoa et al. 2005) | Two MILPs | F-JSSP with and without recirculation | Costs related to just in time due dates, in-process inventories and orders not fully completed |
| (Gao, Gen et al. 2006) | MINLP | F-JSSP with preventive maintenance tasks | Make-span, Maximal workload at any machine, Total workload overall machines |
| (Low, Yukling et al. 2006) | MILP | F-JSSP with SIST | Mean flow time, Mean job tardiness, Mean machine idle time |
| (Fattahi, Mehrabad et al. 2007) | MILP | F-JSSP | Make-span |
| (Ozguven, Ozbakir et al. 2010) | MILP | F-JSSP | Make-span |
| (Saidi-Mehrabad and Fattahi 2007) | MILP | F-JSSP with SDST | Make-span |
| (Fattahi, Jolai et al. 2009) | MILP | F-JSSP with overlapping in operations | Make-span |
| (Ozguven, Yavuz et al. 2012) | MIGP | F-JSSP with process plan and routing flexibility with SDST &SIST | Make-span & balancing the workloads of the machines |
| (Moradi, Fatemi Ghomi et al. 2011) | Bi-criterion MILP | F-JSSP with preventive maintenance task | Make-span & minimization of system unavailability |

Among proposed the MILPs for different variants of FMS, only the MILPs proposed by Fattahi, Mehrabad et al. (2007) and (Ozguven, Ozbakir et al. 2010) are relevant to the subject of this thesis.

Regretfully, there are certain technical inaccuracies associated with the MILP proposed by (Fattahi, Mehrabad et al. 2007) which are addressed by the proposed MILP-2 model in section 3.4.

### MILP model proposed by (Fattahi, Mehrabad et al. 2007) (MILP-1):

### Parameters and decision variables

| | |
|---|---|
| $j$ | Subscript for parts where $1 \leq j \leq n$ |
| $i$ | Subscript for machines where $1 \leq i \leq m$ |
| $l$ | Subscript for operations of part $j$ where $1 \leq l \leq n_j$ |
| $e_{j,l,i}$ | Eligibility parameter that takes value 1 if machine $i$ is able to process operation $O_{j,l}$ and 0 otherwise. |
| $f_i$ | Subscript for processing positions of machine $i$ where $1 \leq f \leq f_i$ ($f_i = \sum_j \sum_l e_{j,l,i}$) |
| $M$ | A large positive number |
| $X_{j,l,i,f}$ | Binary decision variable taking value 1 if *l-th* operation of part *j* is processed on the *f-th* position of machine *i* |
| $Y_{j,l,i}$ | Binary decision variable taking value 1 If machine i is selected to process operation ($O_{j,l}$) |
| $Ps_{j,l}$ | Processing time of operation $O_{j,l}$ after selecting a machine for processing it |
| $S_{j,l}$ | Continuous decision variable for starting time of operation $O_{j,l}$ |
| $B_{i,f}$ | Continuous decision variable for beginning time of each processing position |
| $f_i$ | A decision variable deciding number of assigned operations to machine *i* |

*Min* $C_{max}$

$$\sum_i \sum_f X_{j,l,i,f} = 1 \qquad \forall_{j,l} \qquad (2\text{-}1)$$

$$\sum_i Y_{j,l,i} = 1 \qquad \forall_{j,l} \qquad (2\text{-}2)$$

$$\sum_f X_{j,l,i,f} = Y_{j,l,i} \qquad \forall_{j,l,i} \qquad (2\text{-}3)$$

$$Y_{j,l,i} \leq e_{j,l,i} \qquad \forall_{j,l,i} \qquad (2\text{-}4)$$

$$\sum_i Y_{j,l,i}.\ P_{j,l,i} = P_{sj,l} \qquad \forall_{j,l} \qquad (2\text{-}5)$$

$$S_{j,l+1} \geq S_{j,l} + P_{sj,l} \qquad \forall_{j,l-1} \qquad (2\text{-}6)$$

$$B_{i,f+1} \geq B_{i,f} + P_{sj,l}.X_{j,l,i,f} \qquad \forall_{j,l,i,f-1} \qquad (2\text{-}7)$$

$$B_{i,f} \leq S_{j,l} + M\left(1 - X_{j,l,i,f}\right) \qquad \forall_{j,l,i,f} \qquad (2\text{-}8)$$

$$B_{i,f} \geq S_{j,l} - M\left(1 - X_{j,l,i,f}\right) \qquad \forall_{j,l,i,f} \qquad (2\text{-}9)$$

$$C_{max} \geq S_{j,n_j} + P_{sj,n_j} \qquad \forall_j \qquad (2\text{-}10)$$

$$S_{j,l} \geq 0, P_{sj,l} \geq 0\ ,\ B_{i,f} \geq 0 \qquad (2\text{-}11)$$

$$X_{j,l,i,f}\ , Y_{j,l,i} \in \{0,1\} \qquad (2\text{-}12)$$

### Descriptions of the model:

Constraint sets (2-1) and (2-2) force each operation can be performed only on one machine and at one priority. The values that $X_{j,l,i,f}$ take determine assignments of operations to machines and sequence assigned operations on all machines. Constraint sets (2-3) and (2-4) ensure feasible assignments of operations to machines. Constraint set (2-5) determines the processing time of operation $O_{j,l}$ by selected machine. Constraint set (2-6) enforces each job to follow a specified operation sequence. Constraint set (2-7) forces each machine to process one operation at a time. Constraint sets (2-8 and 2-9) force each operation $O_{j,l}$ can be start after its assigned machine is idle and previous operation $O_{j,l-1}$ is completed. Constraint set (8) assigns the operations to a machine and sequence assigned operations on all machines. Constraint set (2-10) determines the make-span. Constraint sets (2-11, 2-12) show the natures of decision variables used in the MILP-1.

### Technical issues:

In abovementioned model, there are three technical errors which are addressed in the following subsections.

### $f_i$ (Number of assigned operations to machine $i$)

The first error is that the authors have claimed that $f_i$ is a decision variable the value of which is determined by their proposed MILP model. Their claim is invalid since $f_i$ has been used as a parameter inside their mathematical model meaning that the value for $f_i$ is a priori known. Number of operations assigned to each machine, which is equivalent to the processing positions in that machine is calculated as such ($f_i = \sum_j \sum_l e_{j,l,i}$ ). Therefore, this value is known a priori. This invalid claim is present in almost all equations.

### Constraint sets (2-5) and (2-7)

The second error is even more technically critical from the standpoint of linear programming modelling principles. Constraint sets (2-5) and (2-7) in the MILP-1 are directly related to each other. If attention is paid to constraint set (5), it is realized that $Ps_{j,l}$ is obtained from the following equation $\sum_i Y_{j,l,i}.\ P_{j,l,i} = P_{sj,l}$ . Likewise, $Ps_{j,l}$ is used in constraint set (2-7) as the processing time of operation $O_{j,l}$ once machine $i$ is selected as eligible machine to process it. Therefore, if the right hand side of the equation in constraint set (2-5) which calculates the $Ps_{j,l}$ , is substituted in constraint set (2-7), the subsequent formula is resulted ($T_{i,f} + \sum_i Y_{j,l,i}.\ P_{s\ j,l,i}.X_{j,l,i,f} \leq T_{i,f+1}$). In the combined equation, it is

easily noticed that $Y_{j,l,i}$ is being multiplied by $X_{j,l,i,f}$. From the combined equation, it is simply understood that two binary decision variables are multiplied to one another which turns constraint set (2-7) into non-linear equation. Apart from non-linearity that violates the basic principles of MILP models, this issue does not communicate any practical interpretation. In other words, even if non-linearity could be solved by different optimization software, it would not express any meaning in the context of scheduling.

**Missing constraints**

In F-JSSP, for each available machine in the shop floor as many as $f_i = \sum_j \sum_l e_{j,l,i}$), processing positions are considered. Therefore, if $f_i$is multiplied by the number of available machines in the shop floor, the following number $(\sum f_i)$ is obtained. This number $(\sum f_i)$ which is equivalent to the number of all available processing positions of all machines in the shop floor, far exceeds the number of available operations for all jobs which is shown by $\sum n_j$. Therefore, some processing positions remain unallocated. This imbalance in F-JSSP has to be represented by a constraints set. Otherwise, when the output from the MILP model is obtained, a situation happens that the numbers of binary decision variables taking value 1 exceeds the total number of operations $(\sum n_j)$ and become equal to the total number of processing positions $(\sum f_i)$. In this case, an operation is allocated to more than one processing position so that all the processing positions are filled by existing numbers of operations. This issue also renders the developed MILP infeasible.

Therefore, based on foregoing explanations, the mathematical representation of the MILP-1 is wrong. In chapter 3, existing issues in the MILP-1 is remedied and more efficient mathematical models are proposed.

## 2.2 Review of meta-heuristics for flexible job shop scheduling

Brucker and Schlie (1990) are pioneers in presenting a polynomial algorithm for two jobs in the generalization of the classical job-shop scheduling problem in which a set of machines are associated with each operation of a job. Brandimarte (1993) proposed a hierarchical algorithm for the F-JSSP based on TS. He employed a hierarchical strategy to decompose the F-JSSP into two levels of decisions: routing and scheduling which was obtained by assigning each operation of each job to one among the equivalent machines. Both sub-problems are tackled by TS. Hurink, Jurisch et al. (1994) applied new application of TS technique to F-JSSP

and show that their newly devised TS could yield excellent results for benchmark problems. Dauzere-Peres and Paulli (1997) considered an F-JSSP to minimize the make-span as the objective function. He introduced an extended version of the disjunctive graph model that was able to take into account the fact that operations had to be assigned to machines. This enabled them to present an integrated approach, by defining a neighbourhood structure for the problem where there is no distinction between re-assigning or re-sequencing an operation. Finally, a TS procedure was proposed and the computational results proved its effectiveness. Hussain and Joshi (1998) analyzed JSSP with alternate routing. To attack this problem, a two-pass genetic algorithm was used. The first pass picks the alternatives using a genetic algorithm; the second pass provides the order and start time of jobs on the selected alternatives by solving a non-linear program. A non-linear constraint reduces the dimensional complexity of the best known formulation for a job shop problem, and is used in the second pass of the algorithm. Preliminary results of this algorithm were encouraging and the algorithm solved small test problems to optimality. Dauzere-Peres, Roux et al. (1998) tackled a practical problem in which an operation could have more than one predecessor and/or more than one successor on the routing. To minimize the make-span, they made use of hierarchical approach to both assign operations to resources and sequence operations on the resources. A disjunctive graph representation of this problem is offered and a connected neighbourhood structure was proposed. They demonstrated the effectiveness of their algorithm by applying it to benchmarks. Brucker and Neyer (1998) studied the make-span optimization for a multi-mode JSSP (MMJSSP). For the MMJSSP, a novel TS algorithm was presented and its effectiveness was established by certain numerical experiments. Chen, Ihlow et al. (1999) presented a new genetic algorithm (GA) to solve the F-JSSP with make-span criterion. They encoded their problem by chromosomes consisting of two parts: a) the first part defines the routing policy and b) the second part the sequence of the operations on each machine. New genetic operators were used to the reproduction process of the algorithm. Numerical experiments showed that GA was able to find high-quality schedules. Mastrolilli and Gambardella (2000) optimized the make-span for F-JSSP. A local search technique along with two neighbourhood functions (Nopt1, Nopt2) was presented. Their main contribution was the reduction of the set of possible neighbours to a subset for which it always contained the neighbour with the lowest make-span. Eventually, an efficient approach (TS) to compute such a subset of feasible neighbours was given. They proved that their proposed procedure could outperform previous approaches.

Kacem, Hammadi et al. (2002) proposed a Pareto approach based on the hybridization of fuzzy logic (FL) and evolutionary algorithms (EAs) to solve the F-JSSP. This hybrid approach exploits the knowledge representation capabilities of FL and the adaptive capabilities of EAs. The objectives considered were to minimize the make-span, the total workload of machines, and the workload of the most loaded machine. Many examples were presented to illustrate some theoretical considerations and to show the efficiency of the suggested methodology. Kacem (2003) presented two new integrated approaches to solve jointly the assignment and JSSP (with total and partial flexibility). The first one is the approach by localization (AL). It made it possible to solve the problem of resource allocation and built an ideal assignment model (assignments schemata). The second one was an evolutionary approach controlled by the assignment model. They applied advanced genetic manipulations in order to enhance the solution quality. They also explained some of the practical and theoretical considerations in the construction of a more robust encoding that enabled them to solve the F-JSSP by applying the GAs. Two examples were presented to show the efficiency of the two suggested methodologies. Abdallah, Elmaraghy et al. (2002) addressed the deadlock-free scheduling problem in Flexible Manufacturing Systems. An efficient deadlock-free scheduling algorithm was developed, using timed Petri nets, for a class of FMSs called Systems of Sequential Systems with Shared Resources. The algorithm generates a partial reachability graph to find the optimal or near-optimal deadlock-free schedule. The objective is to minimize the mean flow time.

Ho and Tay (2004) offered an efficient methodology called GENACE for solving the F-JSSP with recirculation. They showed how CDRs could be used to solve the F-JSSP with recirculation to provide a bootstrapping mechanism to initialize GENACE. Then, they adapted a cultural evolutionary architecture to maintain knowledge of schemata and resource allocations learned over each generation. Experimental results showed that GENACE could obtain better upper bounds when compared to results of other algorithms in the literature. Scrich, Armentano et al. (2004) presented two heuristics based on TS for F-JSSP to minimize total tardiness. Xia and Wu (2005) developed a hybrid algorithm resulted from the synthesis of Particle Swarm Optimization (PSO) and S SA for the multi-objective F-JSSP. The results obtained from the computational study established that the proposed algorithm was a viable and effective approach for the multi-objective F-JSSP, especially for large-scale problems. The considered objectives were to minimize make-span, the total workload of machines, and the workload of the critical machine. Imanipour and Zegordi

(2006) addressed the minimization of Total Weighted Earliness/Tardiness (TWET) of jobs in an F-JSSP. They proposed an MILP formulation for their F-JSSP with a TWET criterion. Due to the NP-hardness of the F-JSSP, the MILP formulation could not solve the problem in large-size instances. Therefore, they proposed an algorithm based on TS. The proposed algorithm employed TS to find the best routing of each job. Then they made use of backward procedure to find the best operations sequencing. The numerical experiments showed the effectiveness of the suggested algorithm to solve F-JSSP in a reasonable CPU time. Gao, Gen et al. (2006) considered the F-JSSP with three objectives: min make-span, min maximal machine workload and min total workload. They developed a new genetic algorithm hybridized with an innovative local search procedure (bottleneck shifting) for the problem. The GA used two representation methods to depict solution candidates of the F-JSSP. Advanced crossover and mutation operators were proposed to adapt to the special chromosome structures and the characteristics of the problem. The bottleneck shifting worked over two kinds of effective neighbourhood using interchange of operation sequences, and assignment of new machines for operations on the critical path. In order to strengthen search ability, the neighbourhood structure was adjusted dynamically in the local search procedure. The performance of the proposed method was tested by numerical experiments on a large number of representative problems. Ho, Tay et al. (2007) proposed architecture for learning and evolving F-JSSP schedules called LEarnable Genetic Architecture (LEGA). LEGA provided an effective integration between evolution and learning within a random search process. Unlike the canonical evolution algorithm, where random elitist selection and mutational genetics were assumed; through LEGA, the diversity and quality of offspring were influenced by the knowledge extracted from previous generation by its schemata learning module. In addition, the architecture specified a population generator module that generated the initial population of schedules and also trained the schemata learning module. A large range of benchmark data taken from literature and some generated benchmark were used to analyze the efficacy of LEGA. Saidi-Mehrabad and Fattahi (2007) presented a TS algorithm that could solve the F-JSSP to minimize the make-span. They presented a mathematical model for F-JSSP. They also presented a TS for the same problem. Randomly generated test problems were used to evaluate the performance of the proposed algorithm. Results of the algorithm were compared with the optimal solutions using a mathematical model solved by the traditional optimization technique (the branch and bound method). Computational results indicated

that the proposed algorithm could produce optimal solutions in a short computational time for small and medium sized problems. Moreover, it could be applied easily in real factory conditions and for large size problems. Rossi and Dini (2007) proposed an ant colony optimisation-based (ACO) for solving flexible manufacturing system (FMS) scheduling in a job-shop environment with routing flexibility, sequence-dependent set-up, and transportation time. In particular, the optimisation problem for a real environment, including parallel machines and operation lag times, was approached by means of an effective pheromone trail coding and tailored ant colony operators for improving solution quality. The method used to tune the system parameters was also described. The algorithm was tested by using standard benchmarks and problems, specifically designed for a typical FMS layout. The effectiveness of the proposed system was verified in comparison with alternative approaches. A mathematical model and heuristic approaches were proposed by (Fattahi, Mehrabad et al. 2007) for F-JSSP. Mathematical model was used to achieve optimal solution for small-size problems. Since F-JSSP is an NP-hard problem, two heuristic approaches comprised of integrated and hierarchical approaches were developed to solve the real-size problems. Six different hybrid searching structures were presented. Numerical experiments were used to evaluate the performance of the developed algorithms. Finally, it was concluded that, the hierarchical algorithms could outperform integrated algorithms. Pezzella, Morganti et al. (2008) designed a GA for the F-JSSP. The GA integrated different strategies for generating the initial population, selecting the individuals for reproduction and reproducing new individuals. Computational result showed that the integration of more strategies in a genetic framework could lead to better results as opposed to other GAs. Gao, Sun et al. (2008) addressed the F-JSSP with three objectives: min make-span, min maximal machine workload and min total workload. A hybrid genetic algorithm (HGA) was developed for the problem. The GA used two vectors to represent solutions. Advanced crossover and mutation operators were used to adapt to the special chromosome structure and the characteristics of the problem. In order to strengthen the search ability, individuals of GA were first improved by a variable neighborhood descent (VND), which involved two local search procedures: local search of moving one operation and local search of moving two operations. An extensive computational study on benchmark problems showed the high-performance of their approach. Liu, Abraham et al. (2009) formulated the scheduling problem for the multi-objective F-JSSP and attempted to solve the problem using a Multi Particle Swarm Optimization (MPSO) approach. MPSO consisted of multi-swarms of

particles, which searched for the operation order update and machine selection. All the swarms search the optima synergistically and maintain the balance between diversity of particles and search space. They theoretically proved that the multi-swarm synergetic optimization algorithm could converge with a probability of one towards the global optima. The results indicated that the proposed algorithm was an efficient approach for the multi-objective F-JSSP, especially for large-scale problems. Girish and Jawahar (2009) proposed a particle swarm optimization (PSO) based heuristic for solving the F-JSSP for minimum make-span criterion. They, by means of comparative evaluations proved the effectiveness of the proposed PSO for solving F-JSSP instances. A PSO algorithm and a TS algorithm were combined to solve the multi-objective F-JSSP with several conflicting and incommensurable objectives by (Zhang, Shao et al. 2009). Through reasonably hybridizing the two optimization algorithms, an effective hybrid approach for the multi-objective FJSP was proposed. The computational results proved that the proposed hybrid algorithm was an efficient and effective approach to solve the multi-objective F-JSSP, especially for the problems on a large-scale. Bagheri, Zandieh et al. (2010) addressed the F-JSSP to minimize make-span. To tackle this problem, an artificial immune algorithm (AIA) based on integrated approach was proposed. The algorithm used several strategies for generating the initial population and selecting the individuals for reproduction. Different mutation operators were also utilized for reproducing new individuals. They, finally, showed the effectiveness of the proposed method via numerical experiments by applying it to benchmark problems. Consequently, the computational results validated the quality of the proposed approach. Wang, Gao et al. (2010) considered a special case of F-JSSP in which each machine is subject to preventive maintenance during the planning period and the starting times of maintenance activities were either flexible in a time window or fixed beforehand. Moreover, two cases of maintenance resource constraint were considered: sufficient maintenance resource available or only one maintenance resource available. To deal with this variant of F-JSSP with maintenance activities, a filtered beam search (FBS) based heuristic algorithm was proposed. With a modified branching scheme, the machine availability constraint and maintenance resource constraint were easily incorporated into the proposed algorithm. Simulation experiments were conducted on some representative problems. The results demonstrated that the proposed FBS-based heuristic algorithm was a viable and effective approach for the F-JSSP with maintenance activities. De Giovanni and Pezzella (2010) proposed an improved GA for make-span optimization in the distributed F-JSSP (DF-JSSP).

With respect to the solution representation for non-distributed JSSP, gene encoding is extended to include information on job-to-FMU assignment, and a greedy decoding procedure exploited flexibility and determined the job routings. Besides traditional crossover and mutation operators, a new local search based operator was used to improve available solutions by refining the most promising individuals of each generation. The proposed approach was compared with other algorithms for distributed scheduling and evaluated with satisfactory results on a large set of distributed-and-flexible scheduling problems derived from classical job-shop scheduling benchmarks. Ben Hmida, Haouari et al. (2010) suggested a variant of the climbing discrepancy search approach for solving F-JSSP. They also presented various neighbourhood structures related to assignment and sequencing problems. They reported the results of extensive computational experiments carried out on well-known benchmarks for F-JSSP. The results demonstrated that the proposed approach could outperform the best-known algorithms for the F-JSSP on some types of benchmarks. A parallel variable neighbourhood search (PVNS) is utilized by Yazdani, Amiri et al. (2010) to tackle the F-JSSP to minimize make-span. They use the concept of multiple independent searches (parallelization procedure) in their algorithm to increase the exploration in the search space. The proposed PVNS used various neighborhood structures which carried out the responsibility of making changes in assignment and sequencing of operations for generating neighbouring solutions. The results obtained from the computational study revealed that the proposed algorithm is a viable and effective approach for the F-JSSP. Adibi, Zandieh et al. (2010) proposed a VNS algorithm for dynamic F-JSSP. A trained artificial neural network (ANN) was utilized to update parameters of VNS at any rescheduling points. Additionally, a multi-objective performance measure is applied consisting of make-span and tardiness. The proposed method is compared with some common dispatching rules that have been widely used in the literature for dynamic F-JSSP. Results illustrated the remarkable effectiveness and efficiency of the proposed method in a variety of shop floor conditions. Bozejko, Uchronski et al. (2010) proposed a parallel approach to F-JSSP. Two double-level parallel meta-heuristic algorithms based on the new method of the neighbourhood determination were introduced by them. The proposed algorithms possess two primary modules: the machine selection module refers to executed sequentially, and the operation scheduling module executed in parallel. In order to prove the efficiency of their algorithm, they conducted a computational experiment using Graphics Processing Units (GPU). Moslehi and Mahnam (2011) presented a new approach based on a

hybridization of the PSO and local search algorithm to solve the multi-objective F-JSSP. The obtained results for their algorithm indicated that the proposed algorithm could satisfactorily handle the multi-objective F-JSSP and compete well with similar approaches. Zhang, Gao et al. (2011) proposed an effective GA for solving the F-JSSP to minimize make-span. In the proposed algorithm, Global Selection (GS) and Local Selection (LS) were employed to generate high-quality initial population in the initialization stage. An improved chromosome representation was introduced to conveniently represent a solution of the F-JSSP, and different strategies for crossover and mutation operator were adopted. Various benchmark data taken from literature were tested. Computational results proved the proposed GA effective and efficient for solving the considered problem. Al-Hinai and ElMekkawy (2011) developed a hybridized GA architecture for the F-JSSP. The efficiency of their GA was augmented by integrating it with an initial population generation algorithm and a local search method. The usefulness of the proposed methodology was illustrated with the aid of an extensive computational study on 184 benchmark problems with the objective of minimizing the make-span. Obtained results highlight the ability of the proposed algorithm to first obtain optimal or near-optimal solutions, and second to outperform or produce comparable results with those obtained by other best-known approaches in literature. Bagheri and Zandieh (2011) considered the F-JSSP with sequence-dependent set-up times to minimize make-span and mean tardiness. A VNS algorithm based on integrated approach was proposed for F-JSSP. In the presented optimization method, the external loop controlled the stop condition of algorithm and the internal loop executed the search process. To search the solution space, the internal loop used two main search engines, i.e. shake and local search procedures. In addition, neighborhood structures related to the sequencing problem and the assignment problem are employed to generate neighbouring solutions. Consequently, computational results and comparisons validated the quality of the proposed approach. Gutierrez and Garcia-Magarino (2011) presented heuristic methods resulted from the hybridization of genetic algorithms with repair heuristics. The proposed solution was tested in order to analyze its level of constraint satisfaction and its make-span, which were two of the main parameters considered. They discussed this experimentation showing the improvements over existing methods.

## 2.3. Gap analysis and contribution to the field

The literature of F-JSSP contains several proposals of mathematical models with different objective functions, assumptions etc. In some papers, new mathematical models for a new problem are proposed whereas in many others previously proposed mathematical models are optimized. In this thesis, efforts are directed towards proposing and enhancing two famous MILP models in literature to minimize make-span for F-JSSP. Attempts are made to reduce the number of binary integers and continuous variables and also number of constraints to improve the MILPs. The improvements of the proposed MILPs cause the newly developed MILPs to be more computationally efficient and capable of solving larger-size instances of the problem. As can be seen, numerous applications of meta-heuristic algorithms have also been proposed for wide variety of F-JSSPs. Several benchmarks for F-JSSP have also been proposed for measuring the effectiveness of the proposed meta-heuristics. Fattahi, Mehrabad et al. (2007) generated a benchmark consisting of 20 small-to-medium size instances of F-JSSP. Ten instances of small problems are shown by (SFJS1 to SFJS10) and the other ten medium-size instances are shown by (MFJS1 to MFJS10). This benchmark is known as F-data in literature. It is believed this benchmark is the best benchmark in literature for measuring the effectiveness of meta-heuristics. The support for the previous statement is the fact that the optimal and near-optimal solutions of this benchmark have already been calculated using different MILPs. Therefore, the optimality gap of the meta-heuristic proposed in this thesis can be measured using this benchmark. Once the proposed meta-heuristic was applied to the F-data set and its optimality gaps were measured, it was compared vis-à-vis the six meta-heuristics of (Fattahi, Mehrabad et al. 2007) and the artificial immune algorithm of (Bagheri, Zandieh et al. 2010).

In light of the foregoing literature survey and identified gaps in F-JSSP, the research reported in this thesis proposes four new mathematical models and a new hybrid meta-heuristic algorithm capable of producing new optimal and more effective feasible solutions for the F-JSSP represented by the F-data.

# CHAPTER 3

# PROPOSED SOLUTION METHODOLOGIES

## 3.1 Mathematical models

Mathematical models for production scheduling problems came to existence in the late 50s. At that period of time, there were few solution techniques to solve different types of mathematical models. In spite of recognizing few solution techniques to solve mathematical models, there were no viable computing technologies to solve them. The imbalance between mathematical models and computing technologies discouraged many practitioners to deploy mathematical models as their primary tool through which optimal solutions could be obtained. . This issue continued to affect practitioners and academics, as relevant solution techniques and computational technologies could not solve large-size instances of the problem. Obviously, mathematical programming would be deemed the best tool for generating optimal solutions if computational technology could keep up. This notorious issue did not discourage academicians from developing mathematical models as part of their solution techniques with the hope of reaching the point where solving the mathematical models to optimality is possible. But for many researchers, this was just an unachievable delusion. So in view of many researchers, this issue could not be resolved until expected super-computers with huge processors were introduced in the market. But, for few other researchers, mathematical models were conceived possibly solvable to optimality for mid-size problems taking advantage of current technology. They thus initiated rethinking their modeling approaches so that the mathematical models' decision variables, and constraints to be significantly reduced.

## 3.2 Classification of production scheduling modelling paradigms:

Mathematical modelling as a solution technique for production scheduling problems founded its application after the seminal work of Wagner (1959). Soon after Wagner proposed his first model, (Bowman 1959) propounded an equivalent mathematical model for the same problem with a different modelling paradigm. Shortly after the presentation of the second mathematical model by Bowman (1959), Manne (1960) presented a completely new mathematical modelling paradigm. The production scheduling modelling paradigms

proposed by Wagner, Bowman and Manne are generally referred to as position-based, time-based, and sequence-based respectively. These variants of modelling paradigms have separately been extended for regular job shop scheduling problems. No other modelling paradigms have been suggested after the foregoing groundbreaking paradigms. Brief explanation of each of these modelling paradigms is provided below.

$X_{j,i,f} = 1$ *if job j is scheduled on the f-th processing position on machine i; 0, otherwise.*

$X_{j,i,t} = 1$, *if job i is processed by machine i during time-unit t; 0 otherwise.*

$X_{j,h,i} = 1$, *if job j succeeds job h (not necessarily immediately) on machine i; 0 otherwise.*

The MILPs proposed in this thesis adopt the modelling paradigm of Wagner and Manne commonly referred to as position-based and sequence-based respectively. These two modeling paradigms have proven to generate fewer numbers of decision variables, and constraints.

## 3.3 Guideline for formulating production scheduling problems

It has been witnessed that some researchers do not follow a proper guideline to formulate their problems (Fattahi, Mehrabad et al. 2007). The following constraints of the F-JSSP are used. If following constraints are properly defined, the whole model is constructed easier.

- *Constraints for assigning and sequencing of operations to available processing positions*
- *Constraints for machine capability utilization*
- *Machine eligibility constraints*
- *Technical / logical precedence constraints among operations of a part*
- *Machine non-interference constraints*
- *Constraints for relating processing positions to operations*
- *Constraints for capturing the value of objective function*
- *Constraints demonstrating the nature of decision variables*

In this thesis, four enhanced MILPs for F-JSSP are proposed. One of them is constructed based on Wagner's definition of binary integer variables, and the other three are formulated based on Manne's definition of binary integer variables. In Table 3.1, the differences between MILPs are articulated and the contribution of each MILP is elaborated.

Table3.1, differences between proposed mathematical models

| Name | Modeling Paradigm | Novelty |
|---|---|---|
| MILP-1 | Position-based | MILP-1 is the only position-based MILP in literature (Fattahi, Mehrabad et al. 2007). |
| MILP-2 | Position-based | Proposed MILP-2 enhances MILP-1 in all aspects. MILP-2 uses only one type of binary decision variable as opposed to MILP-1 which uses two different types of binary decision variables. This in turn reduces the computational space (RAM on a computer) required to solve larger instances of the problem and hence allow for better solutions to be arrived at. |
| MILP-3 | Sequence-based | MILP-3 is the only sequence-based MILP in literature. (Ozguven, Ozbakir et al. 2010) |
| MILP-4 | Sequence-based | Proposed MILP-4 enhances MILP-3 by reducing its number of decision variables and constraints. It also proposes new binary decision variables which are hierarchical. This in turn reduces the computational space (RAM on a computer) required to solve larger instances of the problem and hence allow for better solutions to be arrived at. |
| MILP-5 | Sequence-based | Proposed MILP-5 enhances MILP-3 in terms of number of continuous decision variables and proposes new binary variable for sequencing and assignment problem. |
| MILP-6 | Sequence-based | Proposed MILP-6 outperforms all other MILPs. |

Analogous to the classification suggested for meta-heuristics in section 1.12, mathematical models are also classified as either integrated or hierarchical. An MILP is considered hierarchical if separate distinct binary variables are used to represent the sequencing and routing (assignment) of operations; it is integrated, otherwise.

## 3.4 Enhanced position-based MILP (MILP-2):

The MILP-2 is an enhanced version of the MILP-1 proposed by (Fattahi, Mehrabad et al. 2007). The MILP-2 utilizes only one type of binary decision variable to handle routing and sequencing sub-problems as opposed to the MILP-1 which uses two distinct binary variables.

### Parameters and decision variables

$j$ Subscript for jobs where $1 \leq j \leq n$

$i$ Subscript for machines where $1 \leq i \leq m$

$l$ Subscript for operations on job $j$ where $0 \leq l \leq n_j$

$O_{j,l}$ $l$ -th operation of job $j$

$e_{j,l,i}$ Parameter that takes value 1 if machine $i$ is able to process $O_{j,l}$ and 0 otherwise.

$p_{j,l,i}$ Processing time of operation $O_{j,l}$ on machine $i$

$f_i$ Subscript for processing positions of machine $i$ where $1 \leq f \leq f_i$ ($f_i = \sum_j \sum_l e_{j,l,i}$)

$M$ A large positive number

$X_{j,l,i,f}$ Binary decision variable taking value 1 if is $O_{j,l}$ is processed on the *f-th* position of machine $i$

$Y_{j,l,i}$ Binary decision variable taking value 1 If machine $i$ is selected to process operation ($O_{j,l}$)

$C_{j,l}$ Continuous variable for the completion time of $O_{j,l}$.

$S_{j,l}$ Continuous decision variable for starting time of operation $O_{j,l}$

$B_{i,f}$ Continuous decision variable for beginning time of each processing position

Min $C_{max}$

$$\sum_{i=1}^{m} \sum_{f=1}^{f_i} X_{j,l,i,f} = 1 \qquad \forall_{j,l} \qquad (3\text{-}1)$$

$$\sum_{j=1}^{n} \sum_{l=1}^{n_j} X_{j,l,i,f} \leq 1 \qquad \forall_{i,f} \qquad (3\text{-}2)$$

$$\sum_{f=1}^{f_i} X_{j,l,i,f} \leq e_{j,l,i} \qquad \forall_{j,l,i} \qquad (3\text{-}3)$$

$$S_{j,l+1} \geq S_{j,l} + \sum_{i=1}^{m} \sum_{f=1}^{f_i} X_{j,l,i,f} \cdot p_{j,l,i} \qquad \forall_{j,l<n_j} \qquad (3\text{-}4)$$

$$B_{i,f+1} \geq B_{i,f} + \sum_{j=1}^{n} \sum_{l=1}^{n_j} X_{j,l,i,f} \cdot p_{j,l,i} \qquad \forall_{i,f<f_i} \qquad (3\text{-}5)$$

$$B_{i,f} \leq S_{j,l} + M\left(1 - X_{j,l,i,f}\right) \qquad \forall_{j,l}, \forall_{i,f} \qquad (3\text{-}6)$$

$$B_{i,f} \geq S_{j,l} - M\left(1 - X_{j,l,i,f}\right) \qquad \forall_{j,l}, \forall_{i,f} \qquad (3\text{-}7)$$

$$C_{max} \geq S_{j,n_j} + \sum_{i=1}^{m} \sum_{f=1}^{f_i} X_{j,n_j,i,f} \cdot p_{j,n_j,i} \qquad \forall_{j} \qquad (3\text{-}8)$$

$$S_{j,l}, B_{i,f} \geq 0 \qquad \forall_{j,l}, \forall_{i,f} \qquad (3\text{-}9)$$

$$X_{j,l,i,f} \in \{0,1\} \qquad \forall_{j,l,i,f} \qquad (3\text{-}10)$$

The purpose of the proposed MILP is to find optimal or feasible schedules for F-JSSP. To this end, an appropriate objective function has to be employed to suitably fulfill the manufacturing strategy. Various objective functions were explored and eventually maximum completion times of operations, commonly referred to as make-span ($C_{max}$) was selected as optimization criterion. The rationale behind utilizing make-span as optimization criterion is that make-span incorporates machine utilization through reduction of idle time on all machines and ensures even workload distribution among work centres (Pan 1997). Constraint set (3-1) ensures that each operation is assigned to one and only one position of all available machines. Constraint set (3-2) ensures the fact that some capabilities of all machines may not be fully utilized. Constraint set (3-3) ensures that each operation is processed on the eligible machines that have been determined a priori. Therefore, the first three sets of constraints ensure schedule feasibility. Constraint set (3-4) preserves the

precedence relationships between the starting times of operations of a job. Constraint set (3-5) express the fact that each order-position of a machine can be occupied only if the preceding positions have fulfilled the processing requirements ($P_{j,l,i}$) of other operations. Constraint sets (3-6) and (3-7) ensure that an operation can occupy one position of a machine when both the operation and the machine are available. These two constraints are Either-Or constraints. For this model, $X_{j,l,i,f}$ takes value 1 if *l*-th operation of job "*j*" is scheduled on machine *f*-th slot/order of machine "*i*". If $X_{j,l,i,f}$ takes value 0, both constraint sets (3-6) and (3-7) become redundant since $B_{i,f}$ would be less than infinity and greater than negative infinity-naturally. On the other hand, if it does have a value, $X_{j,l,i,f} = 1$, then $B_{i,f}$ is less than or equal $S_{j,l}$ and at the same time greater than or equal same. Hence, that leaves us with only one possibility ($S_{j,l} = B_{i,f}$); i.e., beginning time of slot *f* on machine *i* is equal to starting time of operation *l* of job *j*. Constraint set (3-8) takes care of calculating the last completed operation for each job which is the maximum completion time of operations on all available machines. Constraint set (3-9) show that the continuous variables representing starting times of operations and positions are invariably positive. Constraint set (3-10) demonstrates the binary nature of the decision variables. In the proposed MILP, certain technical and mathematical representation errors in the MILP-1 are being addressed and resolved; see section 2.1 for more account and critique of the model. As a result, the binary decision variable ($Y_{j,l,i}$) is removed from the MILP-1, which substantially increases the computational efficiency and solution effectiveness of the proposed MILP. Getting rid of this binary decision variable reduces the number of binary decision variables of the proposed MILP by ($nm^2$) as compared to the MILP-1.

## 3.5. Sequence-based MILPs

In this subsection, all the mathematical models which have been formulated based on Manne's definition of binary integer variables are presented.

### 3.5.1. Enhanced Ozguven's hierarchical MILP (MILP-4)

The MILP-4 is an enhanced version of the MILP-3 proposed by (Ozguven, Ozbakir et al. 2010) in that instead of using two continuous decision variables ($S_{j,l,i}$ and $C_{j,l,i}$) only $S_{j,l,i}$ has been utilized. Moreover, instead of using only one integrated binary variable, two different binary variables: one for assigning, and the other for sequencing has been used. This issue

causes the number of continuous decision variables to be significantly reduced. All of the constraints in the MILP-3 have been restructured so that the comprehensibility of the proposed MILP enhances.

$MinC_{max}$

$$\sum_{i=1}^{m} Y_{j,l,i} = 1 \qquad \forall_{j,l} \qquad (3\text{-}11)$$

$$S_{j,l,i} \leq M\left(Y_{j,l,i}\right) \qquad \forall_{j,l,i} \qquad (3\text{-}12)$$

$$\sum_{i=1}^{m} S_{j,l+1,i} \geq \sum_{i=1}^{m} S_{j,l,i} + \sum_{i=1}^{m} Y_{j,l,i} \cdot p_{j,l,i} \qquad \forall_{j,l<n_j} \qquad (3\text{-}13)$$

$$S_{j,l,i} \geq S_{h,z,i} + p_{h,z,i} - M\left(3 - X_{j,l,h,z} - Y_{j,l,i} - Y_{h,z,i}\right) \qquad \forall_{j<n,l;h>j,z;i\epsilon\{R_{j,l}\cap R_{h,z}\}} \qquad (3\text{-}14)$$

$$S_{h,z,i} \geq S_{j,l,i} + p_{j,l,i} - M\left(X_{j,l,h,z} + 2 - Y_{j,l,i} - Y_{h,z,i}\right) \qquad \forall_{j<n,l;h>j,z;i\epsilon\{R_{j,l}\cap R_{h,z}\}} \qquad (3\text{-}15)$$

$$C_{max} \geq \sum_{i=1}^{m} S_{j,n_j,i} + \sum_{i=1}^{m} Y_{j,n_j,i} \cdot p_{j,n_j,i} \qquad \forall_{j} \qquad (3\text{-}16)$$

$$S_{j,l,i} \geq 0 \qquad (3\text{-}17)$$

$$X_{j,l,h,z,i}, Y_{j,l,i} \in \{0,1\} \qquad (3\text{-}18)$$

Constraint set (3-11) make sure that operation $O_{j,l}$ is assigned to only one machine. If operation $O_{j,l}$ is not assigned to machine *i*, the constraint set (3-12) set the starting times of it on machine *i* equal to zero. Otherwise, the constraint set (3-13) guarantee that the difference between the starting times of operation ($O_{j,l+1}$) and the starting times of its previous operation ($O_{j,l}$) is equal to at the least to the processing time on machine *i* ($p_{j,l,i}$). Constraint sets (3-14) and (3-15) simultaneously take care of the requirement that operation $O_{j,l}$ and operation $O_{h,z}$ cannot be sequenced at the same time on any machine in the shop floor.

Constraint set (3-16) determines the make-span. Constraint sets (3-17) and (3-18) demonstrate the nature of the decision variables.

### 3.5.2. Novel hierarchical sequence-based MILP (MILP-5).

$MinC_{max}$

$$\sum_{i=1}^{m} Y_{j,l,i} = 1 \qquad \forall_{j,l} \qquad (3\text{-}11)$$

$$Y_{j,l,i} \leq e_{j,l,i} \qquad \forall_{j,l,i} \qquad (3\text{-}19)$$

$$C_{j,l} \geq C_{j,l-1} + \sum_{i=1}^{m} Y_{j,l,i} \cdot p_{j,l,i} \qquad \forall_{j,l} \qquad (3\text{-}20)$$

$$C_{j,l} \geq C_{hz} + p_{j,l,i} - M \cdot \left(3 - X_{j,l,h,z} - Y_{j,l,i} - Y_{h,z,i}\right) \qquad \forall_{j<n,l;h>j,z;i} \qquad (3\text{-}21)$$

$$C_{h,z} \geq C_{j,l} + p_{h,z,i} - M \cdot (X_{j,l,h,z} + 2 - Y_{j,l,i} - Y_{h,z,i}) \qquad \forall_{j<n,l;h>j,z;i} \qquad (3\text{-}22)$$

$$C_{max} \geq C_{j,n_j} \qquad \forall_{j} \qquad (3\text{-}23)$$

$$C_{j,l} \geq 0 \qquad \forall_{j,l} \qquad (3\text{-}24)$$

$X_{j,l,h,z}, Y_{j,l,i} \in \{0, 1\}$ (3-25)

where $C_{j,0} = 0$ (3-26)

Constraint set (3-11) ensure that all operations are investigated for processing on all available machines and eventually are imperatively assigned to one of them. Constraint set (3-19) ensures the feasibility of the machine assignments which are investigated for any of the operations in constraint set (3-11). Constraint set (3-20) represents the logical/natural precedence constraint among the operations of a job. It simply means that so long as the previous operation of a job has not been completed on any of the machines in the shop floor the succeeding operation is not processed. Constraint sets (3-21) and (3-22) referred to as Either-Or constraints simultaneously ensure the following: 1) an operation cannot be at the same time both the predecessor and the successor of another operation, and 2) satisfaction of non-interference constraints (precedence constraint among operations of different jobs); i.e., for operations of different jobs that are eligible to be processed on the same machine. In other words, if two operations of two different jobs do not share the same subset of machines, consideration of the operational precedence between them is not done and both constraints become redundant. Hence, two operations $O_{j,l}$ and $O_{h,z}$ can only be sequenced when both their binary integer variables assignment $Y_{j,l,i}$ and $Y_{h,z,i}$ take value of 1; otherwise, they bear no relationship with one another on machine $i$. Once machine $i$ was established as eligible machine to process $O_{j,l}$ and $O_{h,z}$ ($Y_{j,l,i} = 1 \; and \; Y_{h,z,i} = 1$), the precedence relationship between these two operations should be decided by the sequencing binary decision variable which is $X_{j,l,h,z}$. In any circumstance other than stated above, both constraint sets (3-21) and (3-22) become redundant constraints. If $O_{j,l}$ is processed after $O_{h,z}$ on machine $i$, the sequencing binary decision variable takes value 1 and therefore constraint set (3-30) become active. Otherwise, this constraint show that $C_{h,z}$ is just greater than a large negative number, which is naturally true. The sequencing binary variable ($X_{j,l,h,z}$) takes value 0 if $O_{j,l}$ is not processed after $O_{h,z}$. Since two operations in sequencing decision have no more than two states with respect to each other (predecessor or successor), if $O_{j,l}$ does not succeed $O_{h,z}$, it has to precede it, in which case constraint set (3-21) become active and constraint set (3-22) become evident inequality equations. Constraint set (3-23) keeps track of make-span and compute it. Constraint set (3-24) show the non-negativity nature of the MILP'S continuous variables. Constraint set (3-25) demonstrates the binary nature of decision variables.

### 3.5.3. New reduced hierarchical sequence-based MILP (MILP-6):

The MILP-6 is an enhanced and reduced version of the MILP-5. Changes made in the MILP-6 are explained right after the proposed MILP model.

$MinC_{max}$

$\sum_{i\epsilon R_{j,l}} Y_{j,l,i} = 1$ $\forall_{j,l}$ (3-27)

$C_{j,l} \geq C_{j,l-1} + \sum_{i\epsilon R_{j,l}} Y_{j,l,i} \cdot p_{j,l,i}$ $\forall_{j,l}$ (3-28)

$C_{j,l} \geq C_{hz} + p_{j,l,i} - M \cdot (3 - X_{j,l,h,z} - Y_{j,l,i} - Y_{h,z,i})$ $\forall_{j<n,l;h>j,z;i\epsilon\{R_{j,l}\cap R_{h,z}\}}$ (3-29)

$C_{h,z} \geq C_{j,l} + p_{h,z,i} - M \cdot (X_{j,l,h,z} + 2 - Y_{j,l,i} - Y_{h,z,i})$ $\forall_{j<n,l;h>j,z;i\epsilon\{R_{j,l}\cap R_{h,z}\}}$ (3-30)

$C_{max} \geq C_{j,n_j}$ $\forall_j$ (3-23)

$C_{j,l} \geq 0$ (3-24)

$X_{j,l,h,z}, Y_{j,l,i} \in \{0,1\}$ (3-25)

where $C_{j,0} = 0$ (3-26)

The MILP-6 is an enhanced and reduced version of the MILP-5 which does not have constraint sets (3-19) in the MILP-5. For scheduling of operations on different machines, the capabilities of machines are recognized and listed and based on that operations are routed towards machines capable of processing them. In TF-JSSP, all machines have the capability of processing all operations; therefore, machine feasibility is not an applicable issue. But in PF-JSSP the issue of feasible assignments of operations to machines arises. Therefore, second constraint sets in the MILP-5 are required. In the MILP-5, the first and second constraints sets are interrelated. In constraints set (3-11) in MILP-5, all operations are investigated on all machines assuming that all the operations can be processed on all machines. But in the second constraint sets (3-19) in MILP-5 ($Y_{j,l,i} \leq e_{j,l,i}$), ineligible machines are recognized and excluded from assignment considerations. If attention is paid to the first constraint sets in the MILP-5, it is realized that the value of the assignment variable ($Y_{j,l,i}$) depends on the value of $e_{j,l,i}$ which its value is known beforehand. In case of TF-JSSP, the value of parameter $e_{j,l,i}$ is always 1 for all operations; meaning that all operations are feasibly executable on all machines. But in case of PF-JSSP, some machines are unable to process certain operations; therefore, the value of $e_{j,l,i}$ is zero for that operation. In the MILP-5, all machines are considered for processing all operations and hence the number of generated assignment variables for all operations is equivalent to the number of machines. In the MILP-6, this issue is treated differently. By eliminating the

second constraints in the MILP-5, the MILP-6 becomes more efficient. Let's assume there exist three jobs with five machines in the shop floor as in the Table 3.2.

Table 3.2, operation-machine eligibility

| | | $M_1$ | $M_2$ | $M_3$ | $M_4$ | $M_5$ |
|---|---|---|---|---|---|---|
| $J_1$ | $O_{11}$ | 1 | 1 | 1 | 1 | 0 |
| | $O_{12}$ | 0 | 1 | 0 | 1 | 1 |
| | $O_{13}$ | 0 | 1 | 1 | 1 | 1 |
| $J_2$ | $O_{21}$ | 0 | 0 | 0 | 1 | 1 |
| | $O_{22}$ | 1 | 1 | 1 | 0 | 0 |
| | $O_{23}$ | 1 | 0 | 0 | 0 | 0 |
| $J_3$ | $O_{31}$ | 1 | 0 | 1 | 0 | 1 |
| | $O_{32}$ | 0 | 1 | 1 | 1 | 1 |
| | $O_{33}$ | 1 | 0 | 0 | 0 | 1 |

Based on the machine eligibility provided in Table 3.2 constraints sets (3-11) and (3-19) have been expanded Operation $O_{21}$ and its assignment to different machines are given as an example.

$$Y_{2,1,1} + Y_{2,1,2} + Y_{2,1,3+}Y_{2,1,4} + Y_{2,1,5} = 1$$

$$Y_{2,1,1} \leq 1$$

$$Y_{2,1,2} \leq 0$$

$$Y_{2,1,3} \leq 0$$

$$Y_{2,1,4} \leq 0$$

$$Y_{2,1,5} \leq 1$$

As can be seen, just for one operation ($O_{21}$), six constraints and five decision variables are generated while in the MILP-6 the following constraint and decision variables are generated:

$$Y_{2,1,4} + Y_{2,1,5} = 1$$

The reason behind this reduction is that for each operation, a set of eligible machines has been defined ($R_{j,l}$). As can be seen, for assigning $O_{21}$ to existing machines, only one constraint and two decision variables are generated. Thus, elimination of constraints set (3-19) from

the MILP-5 leads to the MILP-6 and it generates far fewer number of constraints and decision variables. This issue is even more sensed in constraints (3-29) and (3-30) in the MILP-6. Having recognized that one operation is not eligible to be processed on certain machine; that operation is not considered for sequencing decision as well. The structure of constraint sets (3-21) and (3-22) in the MILP-5 causes many redundant constraints to be generated. For example, based on Table 3.2, operations $O_{11}$ can be processed on machine 1 ($Y_{111}$=1) while $O_{21}$ cannot be processed on that machine ($Y_{211}$=0). So based on the foregoing example, the following two redundant constraints are generated:

$$C_{1,1} \geq C_{2,1} - M \rightarrow C_{1,1} \geq -M$$

$$C_{2,1} \geq C_{1,1} - M \rightarrow C_{2,1} \geq -M$$

In sequence-based modeling paradigm, all operations are examined for sequencing with respect to each other. This issue is quite natural in TF-JSSP but not in PF-JSSP. In PF-JSSP, some operations may not be processed on the same machine. The prerequisite for operations sequencing in PF-JSSP is that operations should be assigned to the same machine. If and only if one of the operations is not assigned to the same machine, those Either-Or constraints become redundant constraints. The following redundant constraints are generated only when $O_{11}$ is considered for sequencing, with respect to other operations on machine number one.

$$C_{1,1} \geq C_{2,1} - M \rightarrow C_{1,1} \geq -M$$

$$C_{2,1} \geq C_{1,1} - M \rightarrow C_{2,1} \geq -M$$

$$C_{1,1} \geq C_{2,2} - M \rightarrow C_{1,1} \geq -M$$

$$C_{2,2} \geq C_{1,1} - M \rightarrow C_{2,2} \geq -M$$

$$C_{1,1} \geq C_{2,3} - M \rightarrow C_{1,1} \geq -M$$

$$C_{2,3} \geq C_{1,1} - M \rightarrow C_{2,3} \geq -M$$

$$C_{1,1} \geq C_{3,1} - M \rightarrow C_{1,1} \geq -M$$

$$C_{3,1} \geq C_{1,1} - M \rightarrow C_{3,1} \geq -M$$

$$C_{1,1} \geq C_{3,2} - M \rightarrow C_{1,1} \geq -M$$

$C_{3,2} \geq C_{1,1} - M \rightarrow\ C_{3,2} \geq -M$

$C_{1,1} \geq C_{3,3} - M \rightarrow\ C_{1,1} \geq -M$

$C_{3,3} \geq C_{1,1} - M \rightarrow\ C_{3,3} \geq -M$

In order to prevent these redundant constraints from being generated, certain restrictions have to be exerted on the machine assignments of operations. This restriction should ensure that operations sequencing should be merely done among those operations being assigned to the same machine. To this end, the following restrictions ($i\epsilon\{R_{j,l} \cap R_{h,z}\}$) have been applied to constraint sets (3-29) and (3-30) in the MILP-6. This restriction ensures that operations sequencing is done only when any two operations are processed on the same machine. Applying the foregoing restriction and elimination of the second constraint set in the MILP-5 causes the number of binary variables for assigning and sequencing and as a consequence the number of constraints to be significantly decreased in the MILP-6.

In chapter 4, the performances of all the MILPs are investigated and reported. In section, the procedure of developing the proposed meta-heuristic is explained.

## 3.6. Proposed Solution Methodology (A hybrid meta-heuristic)

The proposed meta-heuristic is a hybrid of the Artificial Immune Algorithm (AIA) and Simulated Annealing (SA) and is referred to as AISA.

### 3.6.1. Brief introduction to Artificial Immune Algorithm

AIAs are adaptive computational systems that simulate the behaviour of the immune system of natural living organisms where the body recognizes foreign substances known as antigens and generates a set of antibodies to exterminate them. AIAs, by virtue of *affinity values (inverse of make-span values)*, discern the *antibodies (feasible schedules)* that demonstrate more potential in exterminating *antigens (minimization of F-JSSP)* so as to further proliferate their respective variations in next generations of antibodies. Therefore, the effectiveness of each antibody is measured by its affinity value. Hence, in order for the AISA to be better comprehended, a one-by-one correspondence has to be established to relate the elements of the F-JSSP and to those of the AISA. The correspondence is as follows:

- Antigens: antigen is the F-JSSP that is to be solved.

- Antibodies: antibodies are feasible schedules.
- Affinity values: in our case are the inverses of objective function values. The higher the affinity value, the lower is the make-span.

Cloning selection algorithm (CSA) which is specific type of AIAs has proven to be superior to other AIAs according to (Zandieh, Fatemi Ghomi et al. 2006). CSA is thus used in this thesis. AIAs based on CSA possess two primary operators:

- *Cloning selection*
- *Affinity maturation*

In the former, those schedules that effectively minimize the F-JSSP are proliferated by cloning. Affinity maturation encompasses two basic phases: *hyper-mutation* and *receptor editing*. In the hyper-mutation phase, inferior schedules undergo higher rate of mutation as opposed to superior ones. Receptor editing manages the hyper-mutation procedure. Then, the population evolves by a set of operators until some stopping criterion is met. Complete description of AIAs for production scheduling problems can be found in (Zandieh, Fatemi Ghomi et al. 2006). *Unlike the original AIAs where only inferior schedules undergo hyper-mutation, in the proposed AISA, all schedules undergo hyper-mutation by applying a fast and effective SA.* The proposed hybrid AISA algorithm searches the problem space populated with encoded feasible schedules.

### 3.6.2 Antibody (feasible schedule) representation

F-JSSP is actually a combination of machine assignment and operations sequencing. A hierarchical approach with a novel encoding scheme consisting of two strings is utilized. In the sequencing string, all operations of a job are listed with the same symbol (number) and interpreted according to its already technically precedence-constrained operations set, and then any permutation of these numbers is an operation sequence. Each job $j$ emerges $n_j$ times to represent its $n_j$ ordered operations. By scanning the permutation from left to right, the $k$-th occurrence of a job number indicates the $k$-th operation in its processing route. The following example is provided to demonstrate how a feasible schedule consisting of sequencing string and assigning string is constructed for F-JSSP. Table 3.3 shows the

processing times of operations on each machine, plus their respective feasible machines to execute them.

Table 3.3, processing times of operations and operation-machine assignment

| *Processing times* | $M_1$ | $M_2$ | $M_3$ | *Assignment* | $M_1$ | $M_2$ | $M_3$ |
|---|---|---|---|---|---|---|---|
| $O_{11}$ | *30* | ∞ | *18* | $O_{11}$ | *1* | *0* | *1* |
| $O_{12}$ | ∞ | *40* | *40* | $O_{12}$ | *0* | *1* | *1* |
| $O_{13}$ | *34* | *40* | ∞ | $O_{13}$ | *1* | *1* | *0* |
| $O_{21}$ | ∞ | *60* | *65* | $O_{21}$ | *0* | *1* | *1* |
| $O_{22}$ | *66* | ∞ | *60* | $O_{22}$ | *1* | *0* | *1* |
| $O_{31}$ | ∞ | ∞ | *40* | $O_{31}$ | *0* | *0* | *1* |
| $O_{32}$ | *20* | *32* | ∞ | $O_{32}$ | *1* | *1* | *0* |
| $O_{33}$ | ∞ | *30* | *30* | $O_{33}$ | *0* | *1* | *1* |
| $O_{41}$ | ∞ | *62* | *40* | $O_{41}$ | *0* | *1* | *1* |
| $O_{42}$ | *50* | *60* | ∞ | $O_{42}$ | *1* | *1* | *0* |

Considering above Table 3.3, an example of encoded sequencing is provided. Let's assume the initial random sequence is the following string {1,2,1,3,3,4,1,4,3,2}. Decoded sequence is as follows: $\{O11, O21, O12, O31, O32, O41, O13, O42, O33, O22\}$. Consider the following three-row representation of the initial feasible sequence. Ten random numbers between [0, 1] equivalent to the number of operations are generated and each random number is assigned to one of the operations. After random numbers were generated, they are sorted in a non-decreasing order. Therefore, the sorted string is the initial feasible sequence. This operation is done by an operator called SHIFT (single-point) operator.

Table 3.4, initially generated solution

| Rand no | 0.06 | 0.95 | 0.12 | 0.89 | 0.76 | 0.32 | 0.99 | 0.23 | 0.47 | 0.51 |
|---|---|---|---|---|---|---|---|---|---|---|
| Operations | 1 | 3 | 2 | 4 | 1 | 3 | 2 | 1 | 3 | 4 |

Table 3.5, sorted initial solution

| Rand no | 0.06 | 0.12 | 0.23 | 0.32 | 0.47 | ↓ *0.51* | 0.76 | 0.89 | 0.95 | 0.99 |
|---|---|---|---|---|---|---|---|---|---|---|
| Operations | 1 | 2 | 1 | 3 | 3 | 4 | 1 | 4 | 3 | 2 |

In order to generate new sequence out of the sorted initial solution (Table 3.5), one of the ten random numbers is randomly selected and the value of which is randomly regenerated between [0, 1]. As an example, the random number 0.51 is randomly regenerated as 0.19 and the whole random numbers are sorted in a non-decreasing order again. As you can see,

the first operation of part number fouris relocated from cell 6 to cell 3. Below, the newly constructed sequence is shown.

Table 3.6, newly generated sequence via initial solution

| Rand no | 0.06 | 0.12 | *0.19* | 0.23 | 0.32 | 0.47 | 0.76 | 0.89 | 0.95 | 0.99 |
|---|---|---|---|---|---|---|---|---|---|---|
| Operations | 1 | 2 | 4 | 1 | 3 | 3 | 1 | 4 | 3 | 2 |
| | $O_{11}$ | $O_{21}$ | $O_{41}$ | $O_{12}$ | $O_{31}$ | $O_{32}$ | $O_{13}$ | $O_{42}$ | $O_{33}$ | $O_{22}$ |

So far, the sequencing step was explained. No machine has yet been assigned to each operation. Therefore, the second string shows the machine selected from the eligible machines to process each operation. Based on Table 3.6, a subset of machines with maximum number of machines per operation is defined. Therefore, for each row in Table 3.6, a number showing the maximum number of eligible machines is defined. Then, within the defined range, a random integer number representing the selected machine is generated. In the second string, based on the possible assignments of operations to machines, operation $O_{11}$ is processed by its first eligible machine, $M_1$, operation $O_{21}$ by machine two which is the first eligible machine for it, the operation $O_{41}$ is executed on machine three which is the second eligible machine for it etc. Table 3.7, completely demonstrates how a feasible schedule (antibody) consisting of two strings are generated.

Table 3.7, representation of an antigen (a feasible schedule)

| Rand no | 1 | 2 | 4 | 3 | 1 | 4 | 2 | 3 | 1 | 3 | Affinity= |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Operations | $O_{11}$ | $O_{21}$ | $O_{41}$ | $O_{31}$ | $O_{12}$ | $O_{42}$ | $O_{22}$ | $O_{32}$ | $O_{13}$ | $O_{33}$ | 1/170 |
| Machines | 1 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 2 | 1 | Make-pan= |
| Process time | 30 | 60 | 40 | 40 | 40 | 50 | 60 | 20 | 40 | 30 | 170 |

In Figure 3.1, above feasible schedule is unfolded to its equivalent Gantt chart. As you can see the objective function value of this feasible schedule is 170. Any regeneration of random numbers for two strings generates a new feasible schedule with a new objective function value. Each of these feasible schedules represents an antigen in the initial population of the AISA.

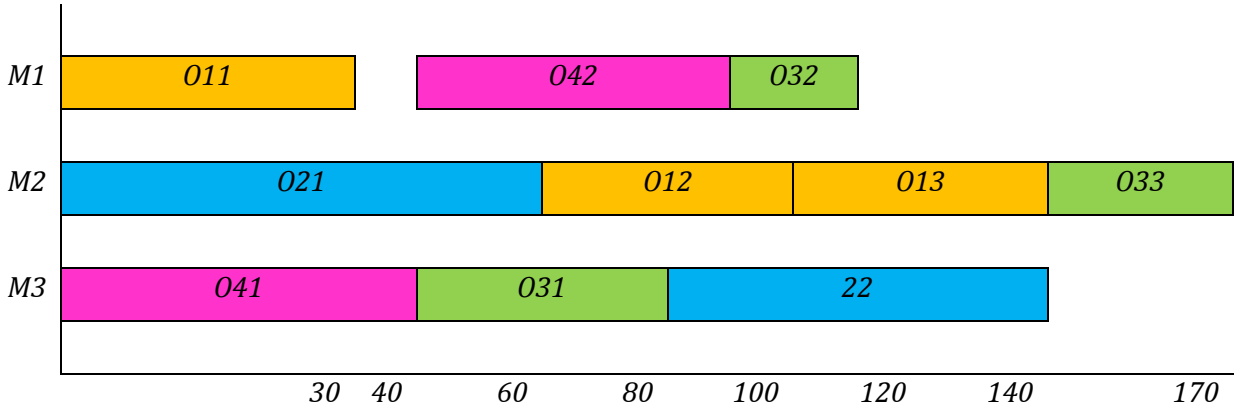


Figure3.1, equivalent Gantt chart of the decoded feasible solution.

The initialization is used for random generation of schedules (pop-size) from the feasible region. Once schedules have been made recognizable to the AISA, an *affinity* value is assigned to each schedule according to their performance - The higher the affinity is, the more desirable the schedule becomes. Then, the schedules evolve by an effective SA until the stopping criterion is met. A typical iteration of AIA *generation* proceeds as follows: according to an author-defined *affinity function*, the number of the schedule clones proliferated by each schedule generator is calculated and added to a *mutating pool*. A *selection* mechanism chooses the schedules in current *mutating pool* where schedules with lower make-span values (higher affinity values) have more chance of being selected.

### 3.6.3. Cloning selection procedure:

Schedules with lower make-span values have higher affinity values and are considered to be superior for solving PF-JSSP. But since the objective function is the minimization of make-span, better schedules are those resulting in lower values of make-span. Therefore, the following function is defined to convert the objective function value of each schedule into its affinity value:

$$Affinity\ (t) = 1 / Make\text{-}span\ (t) \qquad (3\text{-}33)$$

Hence, higher *affinity* values lead to lower objective function values. The probability of cloning each schedule to transfer into the *mutating pool* is directly proportional to its affinity value. The *mutating pool* has a fixed number of schedules (population size). The best schedule in a generation is copied to the next mutation pool (elite strategy). A *selection* mechanism based on binary tournament is used (*popsize*-1). The population of schedules evolve by a set of SA operators until some stopping criterion (*n.m.0.2 second*) is met. The

selected clones are hyper-mutated and generate new schedules (*offsprings)*. The new population is evaluated and the whole process is repeated.

### 3.6.4. Affinity maturing procedure via SA

All schedule clones in the pool undergo an operator called *hyper-mutation* which makes a random change in the schedule clones using a very fast SA. In order for any meta-heuristic algorithm to avoid getting trapped in local optima, it should possess two built-in capabilities:

- Exploration or diversification: alludes to the fact that a meta-heuristic is capable of visiting diverse promising regions in the search space by its operators. AIAs are very well-known for this capability.
- Exploitation or intensification: alludes to the capability of meta-heuristics to completely exploit prospective solutions in newly found regions by diversification operators. SAs are well-established techniques for this task.

*Therefore, the combination of the flexible and effective population-based algorithm to search for the optimal solution and the convergent characteristics of SA provides the rationale for developing the proposed AISA strategy to schedule PF-JSSP while minimizing make-span.* Figure 3.2, shows the pseudo code of the proposed AISA. The strategy to hybridize AIA with SA is as such. After cloning schedules with lower make-span values, the SA is applied to *all clones* in the *mutating pool*. For both sequencing and assigning phases, SHIFT procedure which according (Roshanaei, Seyyed Esfehani et al. 2010) has proven to be superior to other schedule generators is applied. First, based on the sequencing string, a new sequence of operations is constructed and then again SHIFT procedure is used for operation assignment; ten random machine assignments are generated for each sequence and the best one is selected. The newly generated schedule ($s$) from the incumbent solution ($x$) is accepted if the following equation holds ($\Delta C = make\text{-}span(s) - make\text{-}span(x) \leq 0$, otherwise, schedule $s$ undergoes another probabilistic acceptance criterion which is $exp - (\Delta C / t_i)$. Parameter $t$, called the temperature, controls the acceptance rule. At each temperature ($t_i$), the SA algorithm constructs 15 schedules under the exponential annealing scheme ($t_i = \alpha . t_{i-1}$ where $\alpha \in (0, 1)$ is temperature decrement rate) and stops search explorations if no improvement is made after seven consecutive temperatures.

**Initialize the population (N) randomly and set parameters**
**Evaluate affinity value of each schedule**
**Use ranking selection method for cloning selection phase**
Apply ***SA*** algorithm to all clones for affinity maturing procedure
**Procedure** *SA*
Initialization (the one clone in the *mutating pool*)
counter = 0
**while** *counter* <= n **do**
**for** *i* = m **do**
Generate a new neighbor from current solution (shift operator)
Acceptance criterion
Update the best solution so far found
**endfor**
**if** *the best solution is improved in this temperature* **do**
counter = 0
**else**
counter = counter + 1
**endif**
Temperature reduction through exponential cooling scheme
**Endwhile**
**If stopping** criterion is met, return the best schedule

Figure 3.2, pseudo code of AISA

In this chapter, the proposed solution methodologies were completely elaborated. Four mathematical models based on position-and-sequence based modeling paradigms were proposed and developed for F-JSSP. Along with the MILPs which are usually used to solve small-to-medium size instances of the problem, a new meta-heuristic (AISA) which is a hybrid of artificial immune algorithm and SA was suggested and implemented. The AISA has been designed to both solve larger instances of F-JSSP and solve the real case study in this thesis. In chapter four, performance appraisals of the MILPs and the proposed meta-heuristic are done by applying them to standard benchmark from literature. The results obtained from the MILPs are compared with best-performing the MILPs in literature. The same procedure is applied to the AISA.

# CHAPTER 4

# COMPARATIVE EVALUATIONS AND DISCUSSIONS

In this chapter, numerical and computational analyses are conducted based on the optimal, feasible, and lower bound solutions of the enhanced MILPs. Finally, efficiency of each solution methodology is quantified.

## 4.1. Numerical size complexity measurement of MILPs:

This section numerically measures the size complexity of each MILP. Extensive computational study is carried out on a wide range of randomly generated problem instances known as F-data generated by (Fattahi, Mehrabad et al. 2007). Several benchmarks for F-JSSP have also been proposed for measuring the effectiveness of the proposed meta-heuristics. Fattahi, Mehrabad et al. (2007) generated a benchmark consisting of 20 small-to-medium size instances of F-JSSP. 10 instances of small problems are shown by (SFJS1 to SFJS10) and the other 10 instances of medium instances are shown by (MFJS1 to MFJS10). Bases for comparing the MILPs are as follows:

1. *Number of Binary integer variables (BIVs)*
2. *Number of Continuous variables (CVs)*
3. *Numbers of constraints (NCs)*

Recently, new measures for appraising the performances of the MILPs have been introduced (Stafford Jr, Tseng et al. 2005). Sometimes, certain MILP possesses fewer numbers of BIVs while having higher numbers of constraints and other MILPs may have higher number of BIVs but fewer numbers of constraints. The aforesaid three performance measures could hardly serve as a credible basis to adjudicate whether certain MILP surpasses other MILPs. Hence, three other performance measures are used as follows:

4. **Size complexity:** The maximum size of the problem that solutions techniques applied to one MILP can solve to optimality or feasibility. The larger the size of the problem, the better is the MILP.
5. **Computational time:** The speed with which a certain instance of the problem is solved. This measure is referred to as computational or run time. The lower the

computational time, the better is the MILP. This measure depends on the computer used and cannot be used to compare other researchers work who use different computers and capabilities.

6. **Quality of schedules generated by each MILP:** This measure is computed through calculating objective function values. Since the objective functions of scheduling problems are principally of minimization nature, the lower this value, the better is the MILP.

## 4.2. Measuring numerical size complexity of position-based MILPs:

As was pointed out, overall performances of position-based MILPs are rigorously investigated on afore-cited six performance measures. Relative Performance Deviation (RPD) is used as a key comparative performance measure. RPD measures the deviation of each MILP with respect to the best performing MILP on all performance measures.

$$RPD = \frac{MILPsolution - MILPbest}{MILPbest} * 100\% \qquad (3\text{-}43)$$

These performance measures consist of BIVs, CVs and NCs, computational times, maximum size complexity and quality of generated schedules. At the end of analysis, specific tables have been designed to rigorously appraise the performances of MILPs on one of particular performance measures indicated above. Subsequently, the best-performing mathematical model is recognized, and so is the best modeling paradigm. Once the best-performing MILP was identified, the values of best-performing MILP are used as bases for measuring the optimality gaps of the proposed meta-heuristic in different instances of the problem. Finally, by showing some examples, the practical significance of the obtained results are emphasized.

Table 4.1, comparisons of MILP-1 and MILP-2 on schedules quality and computational time

| Instance | Size (j,l,i) | MILP-1 | | MILP-2 | |
|---|---|---|---|---|---|
| | | CPU (s) | $C_{max}$ | CPU (s) | $C_{max}$ |
| SFJS1 | 2.2.2 | **0.0** | **66*** | **0.0** | **66*** |
| SFJS2 | 2.2.2 | **0.0** | **107*** | **0.0** | **107*** |
| SFJS3 | 3.2.2 | 7 | **221*** | **0.20** | **221*** |
| SFJS4 | 3.2.2 | 11 | **355*** | **0.41** | **355*** |
| SFJS5 | 3.2.2 | 87 | **119*** | **7.43** | **119*** |
| SFJS6 | 3.3.2 | 129 | **320*** | **0.33** | **320*** |
| SFJS7 | 3.3.5 | 135 | **397*** | **0.06** | **397*** |
| SFJS8 | 3.3.4 | 116 | **253*** | **4.71** | **253*** |
| SFJS9 | 3.3.3 | 319 | **210*** | **0.08** | **210*** |
| SFJS10 | 4.3.5 | 510 | **516*** | **1.87** | **516*** |
| MFJS1 | 5.3.6 | 3600 | 470 BFI | **60** | **468 BFI** |
| MFJS2 | 5.3.7 | 3600 | 484 BFI | **60** | **446 BFI** |
| MFJS3 | 6.3.7 | 3600 | 564 BFI | **60** | **466 BFI** |
| MFJS4 | 7.3.7 | 3600 | 684 BFI | **60** | **565 BFI** |
| MFJS5 | 7.3.7 | 3600 | 696 BFI | **60** | **514 BFI** |
| MFJS6 | 8.3.7 | 3600 | 786 BFI | **60** | **616 BFI** |
| MFJS7 | 8.4.7 | 3600 | 619 LB | **3600** | 764 LB |
| MFJS8 | 9.4.8 | 3600 | 619 LB | **3600** | 764 LB |
| MFJS9 | 11.4.8 | 3600 | 764 LB | **3600** | 764 LB |
| MFJS10 | 12.4.8 | 3600 | 944 LB | **3600** | 944 LB |
| Total | | 37314 | | **14775** | |

*: optimal solution, BFI: Best Feasible Integer, LB: Lower bound

The solutions obtained from CPLEX software shows that the MILP-1 is capable of producing feasible integer solutions up to MFJS1 (5 jobs on 6 machines) while the MILP-2 is capable of producing feasible integers up to MFJS6 (8 jobs on 7 machines). The quality of feasible integer solution generated by the MILP-1 is inferior to the MILP-2 MFJS1. The best feasible integer obtained by the MILP-2 (468) dominates the best feasible integer produced by MILP-1 (470). Therefore, MILP-2 generates six new enhanced feasible integer solutions for F-data. The MILP-1 is capable of solving F-JSSP to $(5!)^6$ while the MILP-1 is capable of solving it to $(8!)^7$. The MILP-1 used 37,314 seconds and the MILP-2 used 14775 seconds to solve 20 instances of F-data. The MILP-1 used 152.5% more computational time that the MILP-1.

Table 4.2, comparative evaluation of MILP-1 and MILP-2 based on their constituents

| *Instance no* | *Size (j,l,i)* | *BIV* | | *CV* | | *NCs* | |
|---|---|---|---|---|---|---|---|
| | | *MILP-1* | *MILP-2* | *MILP-1* | *MILP-2* | *MILP-1* | *MILP-2* |
| *SFJS1* | *2.2.2* | *40* | ***32*** | *26* | ***13*** | *134* | ***106*** |
| *SFJS2* | *2.2.2* | *32* | ***24*** | *24* | ***11*** | *108* | ***84*** |
| *SFJS3* | *3.2.2* | *72* | ***60*** | *36* | ***17*** | *234* | ***178*** |
| *SFJS4* | *3.2.2* | *84* | ***60*** | *38* | ***17*** | *236* | ***178*** |
| *SFJS5* | *3.2.2* | *84* | ***72*** | *38* | ***19*** | *272* | ***208*** |
| *SFJS6* | *3.3.2* | *189* | ***135*** | *50* | ***25*** | *497* | ***366*** |
| *SFJS7* | *3.3.5* | *225* | ***162*** | *55* | ***28*** | *598* | ***445*** |
| *SFJS8* | *3.3.4* | *216* | ***162*** | *55* | ***28*** | *589* | ***437*** |
| *SFJS9* | *3.3.3* | *243* | ***162*** | *56* | ***28*** | *584* | ***429*** |
| *SFJS10* | *4.3.5* | *300* | ***240*** | *66* | ***33*** | *862* | ***631*** |
| *MFJS1* | *5.3.6* | *720* | ***495*** | *99* | ***49*** | *1829* | ***1218*** |
| *MFJS2* | *5.3.7* | *840* | ***585*** | *106* | ***55*** | *1986* | ***1430*** |
| *MFJS3* | *6.3.7* | *1260* | ***846*** | *131* | ***66*** | *2819* | ***2006*** |
| *MFJS4* | *7.3.7* | *1617* | ***1176*** | *149* | ***78*** | *3789* | ***2723*** |
| *MFJS5* | *7.3.7* | *1617* | ***1113*** | *149* | ***75*** | *3726* | ***2588*** |
| *MFJS6* | *8.3.7* | *2184* | ***1512*** | *174* | ***88*** | *4766* | ***3446*** |
| *MFJS7* | *8.4.7* | *3584* | ***2496*** | *219* | ***111*** | *7883* | ***5539*** |
| *MFJS8* | *9.4.8* | *4896* | ***3096*** | *256* | ***123*** | *9778* | ***6838*** |
| *MFJS9* | *11.4.8* | *7040* | ***4532*** | *308* | ***148*** | *14190* | ***9849*** |
| *MFJS10* | *12.4.8* | *8832* | ***5376*** | *346* | ***161*** | *16784* | ***11608*** |
| *Total* | | *34075* | ***22336*** | *2381* | ***1173*** | *71664* | ***50307*** |
| *RPD* | | ***52.5%*** | ***0%*** | *103%* | ***0%*** | *42.5%* | ***0%*** |

The MILP-1 produced **34075** BIVs for 20 instances of F-data while the MILP-2 produced **22336** BIVs. The MILP-1 produced **2381** CVs on F-data whereas the MILP-2 generated only **1173** CVs. The MILP-1 generated **71664** NCs while the MILP-2 produced **50307** NCs. Above-cited figures simply demonstrate the outperformance of the MILP-2 with respect to the MILP-1. The MILP-1 produced RPDs of 52.5%, 103% and 42.5% for BIVs, CVs, and NCs respectively.

## 4.3. Measuring numerical size complexity of sequence-based MILPs:

In this section, detailed performances of the MILPs are quantified and reported. The sequence-based MILPs are appraised versus standard performance measures.

Table 4.3, comparisons of MILP-3 and MILP-4 on quality of schedules and CPU time

| Instance no | Size (j,l,i) | MILP-3 | | MILP-4 | |
|---|---|---|---|---|---|
| | | CPU (s) | $C_{max}$ | CPU (s) | $C_{max}$ |
| SFJS1 | 2.2.2 | **0.02** | **66*** | **0.02** | **66*** |
| SFJS2 | 2.2.2 | **0.00** | **107*** | 0.02 | **107*** |
| SFJS3 | 3.2.2 | **0.02** | **221*** | **0.01** | **221*** |
| SFJS4 | 3.2.2 | **0.00** | **355*** | 0.01 | **355*** |
| SFJS5 | 3.2.2 | 0.06 | **119*** | **0.05** | **119*** |
| SFJS6 | 3.3.2 | **0.03** | **320*** | **0.03** | **320*** |
| SFJS7 | 3.3.5 | **0.02** | **397*** | **0.02** | **397*** |
| SFJS8 | 3.3.4 | **0.02** | **253*** | 0.06 | **253*** |
| SFJS9 | 3.3.3 | **0.03** | **210*** | **0.03** | **210*** |
| SFJS10 | 4.3.5 | **0.02** | **516*** | 0.03 | **516*** |
| MFJS1 | 5.3.6 | **0.44** | **468*** | 0.73 | **468*** |
| MFJS2 | 5.3.7 | 6.49 | **446*** | **1.46** | **446*** |
| MFJS3 | 6.3.7 | **4.14** | **466*** | **1.03** | **466*** |
| MFJS4 | 7.3.7 | 1779 | 564 | **245** | **554*** |
| MFJS5 | 7.3.7 | 50.98 | **514*** | **13.12** | **514*** |
| MFJS6 | 8.3.7 | 3600 | $635^{BFI}$ | 3600 | **608*** |
| MFJS7 | 8.4.7 | 3600 | $935^{BFI}$ | **60** | $\mathbf{881}^{BFI}$ |
| MFJS8 | 9.4.8 | 3600 | $905^{BFI}$ | **180** | $\mathbf{894}^{BFI}$ |
| MFJS9 | 11.4.8 | 3600 | $1192^{BFI}$ | **300** | $\mathbf{1192}^{BFI}$ |
| MFJS10 | 12.4.8 | 3600 | $1276^{BFI}$ | **3600** | $\mathbf{1276}^{BFI}$ |
| Total | | 19841 | | **8001** | |

*: optimal solutions, BFI: best feasible integer, LB: Lower Bound

The MILP-4 which possesses fewer numbers of BIVs, CVs and NCs with respect to the MILP-3, consumes 8001 second on 20 instances of F-data as opposed to the MILP-3 with 19841 seconds. The MILP-4 consumes 140% less computational time (RPD) as compared to the MILP-3.

Table 4.4, comparative evaluation of MILP-3 and MILP-4 based on their constituents

| *no* | *Size (j,l,i)* | *BIV* | | *CV* | | *NCs* | |
|---|---|---|---|---|---|---|---|
| | | *MILP-3* | *MILP-4* | *MILP-3* | *MILP-4* | *MILP-3* | *MILP-4* |
| *SFJS1* | *2.2.2* | 16 | **12** | 19 | **9** | 42 | **40** |
| *SFJS2* | *2.2.2* | 10 | **9** | 15 | **7** | **30** | **28** |
| *SFJS3* | *3.2.2* | 26 | **21** | 24 | **11** | **67** | **64** |
| *SFJS4* | *3.2.2* | 26 | **22** | 24 | **11** | **67** | **64** |
| *SFJS5* | *3.2.2* | 36 | **24** | 28 | **13** | 87 | **84** |
| *SFJS6* | *3.3.2* | 39 | **34** | 34 | **16** | **99** | **96** |
| *SFJS7* | *3.3.5* | 36 | **35** | 40 | **19** | **93** | 94 |
| *SFJS8* | *3.3.4* | 45 | **40** | 40 | **19** | **111** | **108** |
| *SFJS9* | *3.3.3* | 55 | **45** | 40 | **19** | **131** | **128** |
| *SFJS10* | *4.3.5* | 48 | **46** | 45 | **21** | **124** | **120** |
| *MFJS1* | *5.3.6* | 103 | **84** | 72 | **34** | **241** | **236** |
| *MFJS2* | *5.3.7* | 128 | **101** | 84 | **40** | **291** | **294** |
| *MFJS3* | *6.3.7* | 190 | **141** | 103 | **48** | **422** | **408** |
| *MFJS4* | *7.3.7* | 250 | **184** | 120 | **57** | 549 | **542** |
| *MFJS5* | *7.3.7* | 243 | **184** | 118 | **54** | **535** | **502** |
| *MFJS6* | *8.3.7* | 307 | **240** | 133 | **64** | **670** | 682 |
| *MFJS7* | *8.4.7* | 475 | **364** | 165 | **79** | **1022** | **1014** |
| *MFJS8* | *9.4.8* | 519 | **408** | 182 | **86** | **1119** | **1088** |
| *MFJS9* | *11.4.8* | 751 | **604** | 218 | **104** | **1601** | **1598** |
| *MFJS10* | *12.4.8* | 899 | **719** | 237 | **113** | **1906** | **1900** |
| *Total* | | 4202 | 3317 | 1741 | 824 | 9207 | 9090 |
| *RPD* | | **26.6%** | **0%** | **111%** | **0%** | **1.28%** | **0%** |

Table 4.4 demonstrates the evaluation between the MILP-3 and the MILP-4 based on their constituents. The MILP-4 produced 26.6%, 111% and 1.12% fewer numbers of BIVs, CVs and NCs respectively versus the MILP-3. The superiority of the MILP-4 is inferred from the figures obtained in Table 4.4. The MILP-4 uses hierarchical approach to assign and sequence operations on different machines while the MILP-3 uses integrated approach. Utilization of hierarchical approach to F-JSSP is deemed to be the main reason for the perceived improvement.

Table 4.5, comparisons of MILP-3 and MILP-5 on quality of schedules and CPU time

| Instance no | Size (j,l,i) | MILP-3 | | MILP-5 | |
|---|---|---|---|---|---|
| | | CPU (s) | $C_{max}$ | CPU (s) | $C_{max}$ |
| SFJS1 | 2.2.2 | **0.02** | **66*** | **0.02** | **66*** |
| SFJS2 | 2.2.2 | **0.00** | **107*** | **0.00** | **107*** |
| SFJS3 | 3.2.2 | **0.02** | **221*** | **0.02** | **221*** |
| SFJS4 | 3.2.2 | **0.00** | **355*** | 0.01 | **355*** |
| SFJS5 | 3.2.2 | 0.06 | **119*** | **0.03** | **119*** |
| SFJS6 | 3.3.2 | **0.03** | **320*** | **0.03** | **320*** |
| SFJS7 | 3.3.5 | **0.02** | **397*** | **0.02** | **397*** |
| SFJS8 | 3.3.4 | **0.02** | **253*** | **0.02** | **253*** |
| SFJS9 | 3.3.3 | **0.03** | **210*** | **0.03** | **210*** |
| SFJS10 | 4.3.5 | **0.02** | **516*** | 0.06 | **516*** |
| MFJS1 | 5.3.6 | **0.44** | **468*** | 0.47 | **468*** |
| MFJS2 | 5.3.7 | 6.49 | **446*** | **1.33** | **446*** |
| MFJS3 | 6.3.7 | **4.14** | **466*** | 4.49 | **466*** |
| MFJS4 | 7.3.7 | 1779 | $564^{BFI}$ | **960** | **554*** |
| MFJS5 | 7.3.7 | 50.98 | **514*** | **6.80** | **514*** |
| MFJS6 | 8.3.7 | 3600 | $635^{BFI}$ | **680.6** | **608*** |
| MFJS7 | 8.4.7 | 3600 | $935^{BFI}$ | **1800** | $\mathbf{881^{BFI}}$ |
| MFJS8 | 9.4.8 | 3600 | $905^{BFI}$ | **1800** | $\mathbf{895^{BFI}}$ |
| MFJS9 | 11.4.8 | 3600 | $\mathbf{1192^{BFI}}$ | 1800 | $\mathbf{1192^{BFI}}$ |
| MFJS10 | 12.4.8 | 3600 | $\mathbf{1276^{BFI}}$ | 3600 | $\mathbf{1276^{BFI}}$ |
| Total | | 19841 | | **10653.93** | |

*: optimal solutions, BFI: best feasible integer, LB: Lower Bound

Table 4.5 reports computational time and objective function values of the MILP-3 and the MILP-5 on all instances of F-data. As can be seen, solutions obtained by the MILP-3 have been outperformed by the MILP-5 on four instances of F-data. The MILP-5 produces two optimal solutions for MFJS4 and MFJS6 while the MILP-3 just obtains feasible integer solutions for these two instances of the problem. Both the MILP-3 and MILP-5 produce feasible integer solutions for MFJS7 and MFJS8. The qualities of schedules generated by the MILP-5 outdo those of the MILP-3 on aforementioned two instances of the problem. Therefore, in terms of quality of schedules (objective function values), the MILP-5 outperforms the MILP-3.

Table 4.6, comparative evaluation of MILP-3 and MILP-5 based on their constituents

| *no* | *Size (j,l,i)* | *BIV* | | *CV* | | *NCs* | |
|---|---|---|---|---|---|---|---|
| | | *MILP-3* | *MILP-5* | *MILP-3* | *MILP-5* | *MILP-3* | *MILP-5* |
| *SFJS1* | *2.2.2* | 16 | **12** | 19 | **7** | 42 | **40** |
| *SFJS2* | *2.2.2* | **10** | 12 | 15 | **7** | **30** | 40 |
| *SFJS3* | *3.2.2* | 26 | **24** | 24 | **10** | **67** | 84 |
| *SFJS4* | *3.2.2* | 26 | **24** | 24 | **10** | **67** | 84 |
| *SFJS5* | *3.2.2* | 36 | **24** | 28 | **10** | 87 | **84** |
| *SFJS6* | *3.3.2* | **39** | 54 | 34 | **13** | **99** | 222 |
| *SFJS7* | *3.3.5* | **36** | 72 | 40 | **13** | **93** | 348 |
| *SFJS8* | *3.3.4* | **45** | 63 | 40 | **13** | **111** | 285 |
| *SFJS9* | *3.3.3* | 55 | **54** | 40 | **13** | **131** | 222 |
| *SFJS10* | *4.3.5* | **48** | 114 | 45 | **17** | **124** | 644 |
| *MFJS1* | *5.3.6* | **103** | 180 | 72 | **21** | **241** | 1225 |
| *MFJS2* | *5.3.7* | **128** | 195 | 84 | **21** | **291** | 1420 |
| *MFJS3* | *6.3.7* | **190** | 261 | 103 | **25** | **422** | 2082 |
| *MFJS4* | *7.3.7* | 250 | 336 | 120 | **26** | 549 | 2870 |
| *MFJS5* | *7.3.7* | **243** | 336 | 118 | **29** | **535** | 2870 |
| *MFJS6* | *8.3.7* | **307** | 420 | 133 | **33** | **670** | 3748 |
| *MFJS7* | *8.4.7* | **475** | 672 | 165 | **41** | **1022** | 6608 |
| *MFJS8* | *9.4.8* | **519** | 864 | 182 | **46** | **1119** | 9630 |
| *MFJS9* | *11.4.8* | **751** | 1232 | 218 | **56** | **1601** | 14586 |
| *MFJS10* | *12.4.8* | **899** | 1440 | 237 | **61** | **1906** | 17448 |
| *Total* | | **4202** | **6389** | **1741** | **472** | **9207** | **64540** |
| *RPD* | | **0%** | **52%** | 269% | **0%** | **0%** | 600% |

Table 4.6 captures the number of BIVs, CVs and NCs that the MILP-3 and the MILP-5 produce. The MILP-5 is outperformed by the MILP-3 in terms of number of BIVs. The MILP-5 bears 52% higher number of BIVs versus the MILP-3. The MIL-5 outperforms the MILP-3 in terms of CVs by producing 269% fewer numbers of CVs. The MILP-3 dominates the MILP-5 in terms of NCs. The MILP-3 has much fewer numbers of NCs which causes the computational time of the MILP to reduce considerably. As can be seen, each of the MILPs has their own strengths and weaknesses. The ultimate performance measure for determining the best-performing MILP is the quality of generated schedules. As was mentioned, MILP-5 dominated MILP-3 in this regard.

Table 4.7, comparisons of MILP-3 and MILP-6 on quality of schedules and CPU times

| Instance no | Size (j,l,i) | MILP-3 | | MILP-6 | |
|---|---|---|---|---|---|
| | | CPU (s) | $C_{max}$ | CPU (s) | $C_{max}$ |
| SFJS1 | 2.2.2 | **0.02** | **66*** | 0.02 | **66*** |
| SFJS2 | 2.2.2 | **0.00** | **107*** | 0.02 | **107*** |
| SFJS3 | 3.2.2 | **0.02** | **221*** | 0.03 | **221*** |
| SFJS4 | 3.2.2 | **0.00** | **355*** | 0.03 | **355*** |
| SFJS5 | 3.2.2 | 0.06 | **119*** | 0.03 | **119*** |
| SFJS6 | 3.3.2 | **0.03** | **320*** | 0.03 | **320*** |
| SFJS7 | 3.3.5 | **0.02** | **397*** | 0.02 | **397*** |
| SFJS8 | 3.3.4 | **0.02** | **253*** | 0.03 | **253*** |
| SFJS9 | 3.3.3 | **0.03** | **210*** | 0.05 | **210*** |
| SFJS10 | 4.3.5 | **0.02** | **516*** | 0.05 | **516*** |
| MFJS1 | 5.3.6 | **0.44** | **468*** | 0.39 | **468*** |
| MFJS2 | 5.3.7 | 6.49 | **446*** | 0.34 | **446*** |
| MFJS3 | 6.3.7 | **4.14** | **466*** | 0.64 | **466*** |
| MFJS4 | 7.3.7 | 1779 | $564^{BFI}$ | 1578 | **554*** |
| MFJS5 | 7.3.7 | 50.98 | **514*** | 2.83 | **514*** |
| MFJS6 | 8.3.7 | 3600 | $635^{BFI}$ | 600.1 | **608*** |
| MFJS7 | 8.4.7 | 3600 | $935^{BFI}$ | 600 | $\mathbf{881}^{\mathbf{IF}}$ |
| MFJS8 | 9.4.8 | 3600 | $\mathbf{905}^{\mathbf{BFI}}$ | 600 | $\mathbf{895}^{\mathbf{IF}}$ |
| MFJS9 | 11.4.8 | 3600 | $\mathbf{1192}^{BFI}$ | 600 | $\mathbf{1135}^{\mathbf{BFI}}$ |
| MFJS10 | 12.4.8 | 3600 | $\mathbf{1276}^{\mathbf{BFI}}$ | 600 | $\mathbf{1276}^{\mathbf{BFI}}$ |
| Total | | 19841 | | **4582.61** | |

Table 4.7 reports computational time and objective function values of the MILP-3 versus the MILP-6 on all instances of F-data. As can be seen, solutions obtained by the MILP-3 have been outperformed by the MILP-5 on six instances of F-data. The MILP-6 produces two optimal solutions for MFJS4 and MFJS6 while the MILP-3 just obtains feasible integer solutions for these two instances of the problem. Both the MILP-3 and MILP-6 produce feasible integer solutions for MFJS7 and MFJS8. The qualities of schedules generated by the MILP-6 outdo those of the MILP-3 on aforementioned two instances of the problem. Moreover, the MILP-6 obtains an enhanced feasible integer solution for MFJS9 (1135) as opposed to feasible integer value of the MILP-3 (1192). Therefore, in terms of quality of schedules (objective function values), the MILP-6 outperforms the MILP-3.

Table 4.8, comparative evaluation of MILP-3 and MILP-6 based on their constituents

| *No* | *Size (j,l,i)* | *BIV* | | *CV* | | *NCs* | |
|---|---|---|---|---|---|---|---|
| | | *MILP-3* | *MILP-6* | *MILP-3* | *MILP-6* | *MILP-3* | *MILP-6* |
| *SFJS1* | *2.2.2* | 16 | 12 | 19 | **7** | 42 | **32** |
| *SFJS2* | *2.2.2* | 10 | 9 | 15 | **7** | 30 | **24** |
| *SFJS3* | *3.2.2* | 26 | 21 | 24 | **10** | 67 | **56** |
| *SFJS4* | *3.2.2* | 26 | 22 | 24 | **10** | 67 | **56** |
| *SFJS5* | *3.2.2* | 36 | 24 | 28 | **10** | 87 | **72** |
| *SFJS6* | *3.3.2* | 39 | 34 | 34 | **13** | 99 | **81** |
| *SFJS7* | *3.3.5* | 36 | 35 | 40 | **13** | 93 | **73** |
| *SFJS8* | *3.3.4* | 45 | 40 | 40 | **13** | 111 | **87** |
| *SFJS9* | *3.3.3* | 55 | 45 | 40 | **13** | 131 | **107** |
| *SFJS10* | *4.3.5* | 48 | 46 | 45 | **17** | 124 | **100** |
| *MFJS1* | *5.3.6* | 103 | 84 | 72 | **21** | 241 | **195** |
| *MFJS2* | *5.3.7* | 128 | 101 | 84 | **21** | 291 | **241** |
| *MFJS3* | *6.3.7* | 190 | 141 | 103 | **25** | 422 | **344** |
| *MFJS4* | *7.3.7* | 250 | 184 | 120 | **29** | 549 | **465** |
| *MFJS5* | *7.3.7* | 243 | 184 | 118 | **29** | 535 | **431** |
| *MFJS6* | *8.3.7* | 307 | 240 | 133 | **33** | 670 | **596** |
| *MFJS7* | *8.4.7* | 475 | 364 | 165 | **41** | 1022 | **906** |
| *MFJS8* | *9.4.8* | 519 | 408 | 182 | **46** | 1119 | **972** |
| *MFJS9* | *11.4.8* | 751 | 600 | 218 | **56** | 1601 | **1108** |
| *MFJS10* | *12.4.8* | 899 | 719 | 237 | **61** | 1906 | **1222** |
| *Total* | | **4202** | **3313** | **1741** | **475** | **9207** | **7168** |
| *RPD* | | **26.83%** | **0%** | **266%** | **0%** | **28.5%** | **0%** |

Table 4.8 captures the numbers of BIVs, CVs and NCs that the MILP-3 and the MILP-6 produce. The MILP-6 outperforms the MILP-3 in all aspects. The MILP-6 produces 26.83% fewer numbers of BIVs versus the MILP-3. Also, the MILP-6 surpasses the MILP-3 in terms of number of CVs. The MILP-6 possesses 266% fewer CVs than the MILP-3. The MILP-6 bears

28.5% fewer number of NCs versus the MILP-3. The MILP-6 outperforms the MILP-3 in terms of CVs by producing 266% fewer numbers of CVs. As can be seen, the MILP-6 completely outperforms the MILP-3 in all performance measures. The ultimate performance measure for determining the best-performing MILP is the quality of generated schedules. As was mentioned, the MILP-6 dominated the MILP-3 in this regard. So far, comparative evaluations were made among the MILP-1 and the MILP-2 belonging to position-based modeling paradigm and the MILP-3, MILP-4, MILP-5 and MILP-6 belonging to sequence-based modeling paradigm. In the next section, all the position- and sequence-based MILPs are compared with each other. As a result, the best MILP and also the best modeling paradigm are recognized.

## 4.4 Comparative evaluations between position-based and sequence-based MILPs

In this section, performances of all the MILPs on the six performance measures are investigated, namely, number of BIVs, number of CVs, number of NCs, computational time, size dimensionality, and quality of generated solutions. As a result, conclusive comments can be made regarding the performances of the MILPs. Once analyses have been conducted, the best-performing MILP is determined, and so is the best modeling paradigm. In this section, detailed accounts of each of the six performance measures are given respectively.

For BIVs, it can be seen in Table 4.9, the MILP-6, MILP-4, MILP-3, MILP-5, MILP-2 and MILP-1 produce 3313, 3317, 4202, 6389, 22336, and 34075 numbers of binary integer variables respectively. The efficiency of each MILP depends substantially on the number of binary integer variables it produces. The branch and cut tree may be as large as $2^n$ nodes, where $n$ is the number of binary integer variables. A problem containing only 30 binary variables could produce a tree having over 1 billion nodes. From previous example, it can be concluded that BIVs have huge impact on the efficiency of MILPs. The lower this number the more efficient is the MILP. As can be seen, among proposed MILPs, the sequence-based MILPs outperform position-based MILPs. The best-performing sequence-based MILP which is MILP-6 produces 3313 BIVs on all instances of F-data as opposed to the best-performing position-based MILP (MILP2) with 22336 BIVs. From computing standpoint, this difference is astronomical. Therefore, sequence-based MILPs outperform position-based MILPs with

respect to number of BIVs. Productions scheduling modelling paradigms are differentiated based on their definition of binary variables. Obtained figures in Table 4.9 reveal the fact that production scheduling formulations following sequence-based paradigm, structurally outperform position-based formulations as they produce fewer number of binary integer variables.

Table 4.9, comparisons of all MILPs based on their number of binary integer variables

| *Instance no* | *Binary Integer Variable (BIVs)* | | | | | |
|---|---|---|---|---|---|---|
| | *MILP-1* | *MILP-2* | *MILP-3* | *MILP-4* | *MILP-5* | *MILP-6* |
| *SFJS1* | *40* | *32* | 16 | **12** | **12** | **12** |
| *SFJS2* | *32* | *24* | 10 | **9** | 12 | **9** |
| *SFJS3* | *72* | *60* | 26 | **21** | 24 | **21** |
| *SFJS4* | *84* | *60* | 26 | **22** | 24 | **22** |
| *SFJS5* | *84* | *72* | 36 | **24** | **24** | **24** |
| *SFJS6* | *189* | *135* | 39 | **34** | 54 | **34** |
| *SFJS7* | *225* | *162* | 36 | **35** | 72 | **35** |
| *SFJS8* | *216* | *162* | 45 | **40** | 63 | **40** |
| *SFJS9* | *243* | *162* | 55 | **45** | 54 | **45** |
| *SFJS10* | *300* | *240* | 48 | **46** | 114 | **46** |
| *MFJS1* | *720* | *495* | 103 | **84** | 180 | **84** |
| *MFJS2* | *840* | *585* | 128 | **101** | 195 | **101** |
| *MFJS3* | *1260* | *846* | 190 | **141** | 261 | **141** |
| *MFJS4* | *1617* | *1176* | 250 | **184** | 336 | **184** |
| *MFJS5* | *1617* | *1113* | 243 | **184** | 336 | **184** |
| *MFJS6* | *2184* | *1512* | 307 | **240** | 420 | **240** |
| *MFJS7* | *3584* | *2496* | 475 | **364** | 672 | **364** |
| *MFJS8* | *4896* | *3096* | 519 | **408** | 864 | **408** |
| *MFJS9* | *7040* | *4532* | 751 | 604 | 1232 | **600** |
| *MFJS10* | *8832* | *5376* | 899 | **719** | 1440 | **719** |
| *Total* | 34075 | 22336 | 4202 | 3317 | 6389 | **3313** |

As can be seen, the first four best-performing MILPs belong to the sequence-based modeling paradigm. This evaluation shows the strengths of the sequence-based MILPs in terms of having fewer numbers of BIVs. For number of CVs, Table 4.10 reports numbers of CVs that each MILP produces on each instance of the F-data. The MILP-6, MILP-5, MILP-4, MILP-2,

MILP-3 and MILP-1 produces 472, 472, 824, 1173, 1741 and 2381 continuous decision variables respectively.

Table 4.10, comparative evaluation of MILPs based on their number continuous variables

| *Instance no* | *Continuous Variables (CVs)* | | | | | |
|---|---|---|---|---|---|---|
| | *MILP-1* | *MILP-2* | *MILP-3* | *MILP-4* | *MILP-5* | *MILP-6* |
| *SFJS1* | 26 | 13 | 19 | 9 | **7** | **7** |
| *SFJS2* | 24 | 11 | 15 | **7** | **7** | **7** |
| *SFJS3* | 36 | 17 | 24 | 11 | **10** | **10** |
| *SFJS4* | 38 | 17 | 24 | 11 | **10** | **10** |
| *SFJS5* | 38 | 19 | 28 | 13 | **10** | **10** |
| *SFJS6* | 50 | 25 | 34 | 16 | **13** | **13** |
| *SFJS7* | 55 | 28 | 40 | 19 | **13** | **13** |
| *SFJS8* | 55 | 28 | 40 | 19 | **13** | **13** |
| *SFJS9* | 56 | 28 | 40 | 19 | **13** | **13** |
| *SFJS10* | 66 | 33 | 45 | 21 | **17** | **17** |
| *MFJS1* | 99 | 49 | 72 | 34 | **21** | **21** |
| *MFJS2* | 106 | 55 | 84 | 40 | **21** | **21** |
| *MFJS3* | 131 | 66 | 103 | 48 | **25** | **25** |
| *MFJS4* | 149 | 78 | 120 | 57 | 26 | 26 |
| *MFJS5* | 149 | 75 | 118 | 54 | **29** | **29** |
| *MFJS6* | 174 | 88 | 133 | 64 | **33** | **33** |
| *MFJS7* | 219 | 111 | 165 | 79 | **41** | **41** |
| *MFJS8* | 256 | 123 | 182 | 86 | **46** | **46** |
| *MFJS9* | 308 | 148 | 218 | 104 | **56** | **56** |
| *MFJS10* | 346 | 161 | 237 | 113 | **61** | **61** |
| *Total* | 2381 | 1173 | 1741 | 824 | **472** | **472** |

Due to the similar structure of the MILP-5 and MILP-6, they produce the same number of CVs and they are the most efficient MILPs in this respect. The MILP-4 also dominates the MILP-3 which is the best-forming MILP in literature. The MILP-2 which belongs to position-based modeling paradigm produces fewer numbers of CVs as opposed to the MILP-3 which belong to sequence-based modeling paradigm. All in all, the sequence-based modeling paradigm is superior to position-based modeling paradigm with respect to generated CVs as well.

Table 4.11 reports the number of NCs that each MILP produces. The MILP-6 is the most efficient MILP among others with number of constraints of 8044. The worst-performing MILP is the MILP-1 with 71664.

Table 4.11, comparative evaluation of MILPs based on their number of constraints

| *Instance no* | *Number of Constraints* | | | | | |
|---|---|---|---|---|---|---|
| | *MILP-1* | *MILP-2* | *MILP-3* | *MILP-4* | *MILP-5* | *MILP-6* |
| *SFJS1* | 134 | 106 | 42 | 40 | 40 | **32** |
| *SFJS2* | 108 | 84 | 30 | 28 | 40 | **24** |
| *SFJS3* | 234 | 178 | 67 | 64 | 84 | **56** |
| *SFJS4* | 236 | 178 | 67 | 64 | 84 | **56** |
| *SFJS5* | 272 | 208 | 87 | 84 | 84 | **72** |
| *SFJS6* | 497 | 366 | 99 | 96 | 222 | **81** |
| *SFJS7* | 598 | 445 | 93 | 94 | 348 | **73** |
| *SFJS8* | 589 | 437 | 111 | 108 | 285 | **87** |
| *SFJS9* | 584 | 429 | 131 | 128 | 222 | **107** |
| *SFJS10* | 862 | 631 | 124 | 120 | 644 | **100** |
| *MFJS1* | 1829 | 1218 | 241 | 236 | 1225 | **195** |
| *MFJS2* | 1986 | 1430 | 291 | 294 | 1420 | **241** |
| *MFJS3* | 2819 | 2006 | 422 | 408 | 2082 | **344** |
| *MFJS4* | 3789 | 2723 | 549 | 542 | 2870 | **465** |
| *MFJS5* | 3726 | 2588 | 535 | 502 | 2870 | **431** |
| *MFJS6* | 4766 | 3446 | 670 | 682 | 3748 | **596** |
| *MFJS7* | 7883 | 5539 | 1022 | 1014 | 6608 | **906** |
| *MFJS8* | 9778 | 6838 | 1119 | 1088 | 9630 | **972** |
| *MFJS9* | 14190 | 9849 | 1601 | 1598 | 14586 | **1458** |
| *MFJS10* | 16784 | 11608 | 1906 | 1900 | 17448 | **1748** |
| *Total* | 71664 | 50307 | 9207 | 9090 | 64540 | **8044** |

The MILP-4 is ranked third followed by the MILP-3, MILP-2 MILP-5 and MILP-1 with 9090, 9207, 50307, 64540, and 71664 numbers of constraints respectively. The best-performing MILP which is the MILP-6 belongs to sequence-based modeling paradigm. The MILP-6 produces the least number of constraints on all instances of the problem. Having done an analogy between MILPs with respect to their number of constraints, it was realized that sequence-based MILPs in most cases are more efficient than position-based MILPs. After rigorously analyzing the number of constituents of the proposed MILPs, merits, and

demerits of each of them were recognized and quantified. Excluding the MILP-6 and MILP-4 which absolutely surpassed all the other MILPs in all aspects, other MILPs showed inconsistencies in their performances. Some of them produced fewer numbers of continuous variables but higher numbers of constraints and vice versa. Thus, no compelling comment can be made on the superiority of the modeling paradigm up to this point. However, performances of some of the MILPs are to large extent predictable. Aforesaid performance measures paved the ground favorable to estimate the performances of MILPs. But since no definitive and assertive judgments could be offered from foregoing performance measures, the combined effect of having different numbers of decision variables and constraints is investigated on their computational efficiency and quality of generated schedules.

Ultimate measures for appraising the MILPs performances are the maximum size of the problem they can solve and the quality of generated schedules which are measured by their objective function value. Table 4.12 reports the computational time for each MILP on all instances of F-data.

Table 4.12, comparative evaluation of MILPs based on their computational time

| *Instance no* | *Computational time* | | | | | |
|---|---|---|---|---|---|---|
| | *MILP-1* | *MILP-2* | *MILP-3* | *MILP-4* | *MILP-5* | *MILP-6* |
| *SFJS1* | **0.0** | **0.0** | 0.02 | 0.02 | 0.02 | 0.02 |
| *SFJS2* | **0.0** | **0.0** | **0.00** | 0.02 | **0.00** | 0.02 |
| *SFJS3* | 7 | 0.20 | 0.02 | **0.01** | 0.02 | 0.03 |
| *SFJS4* | 11 | 0.41 | **0.00** | 0.01 | 0.01 | 0.03 |
| *SFJS5* | 87 | 7.43 | 0.06 | 0.05 | 0.03 | 0.03 |
| *SFJS6* | 129 | 0.33 | **0.03** | **0.03** | **0.03** | **0.03** |
| *SFJS7* | 135 | 0.06 | **0.02** | **0.02** | **0.02** | **0.02** |
| *SFJS8* | 116 | 4.71 | **0.02** | 0.06 | **0.02** | 0.03 |
| *SFJS9* | 319 | 0.08 | **0.03** | **0.03** | **0.03** | 0.05 |
| *SFJS10* | 510 | 1.87 | **0.02** | 0.03 | 0.06 | 0.05 |
| *MFJS1* | 3600 | 60 | 0.44 | 0.73 | 0.47 | **0.39** |
| *MFJS2* | 3600 | 60 | 6.49 | 1.46 | 1.33 | **0.34** |
| *MFJS3* | 3600 | 60 | 4.14 | 1.03 | 4.49 | **0.64** |
| *MFJS4* | 3600 | **60** | 1779 | **245** | **960** | 1578 |
| *MFJS5* | 3600 | 60 | 50.98 | **13.12** | 6.80 | **2.83** |
| *MFJS6* | 3600 | **60** | 3600 | 3600 | **680.6** | 600.1 |
| *MFJS7* | 3600 | 3600 | 3600 | **60** | **1800** | 600 |
| *MFJS8* | 3600 | 3600 | 3600 | **180** | 1800 | 600 |
| *MFJS9* | 3600 | 3600 | 3600 | **300** | 1800 | **600** |
| *MFJS10* | 3600 | 3600 | 3600 | **3600** | 3600 | **600** |
| *Total* | 37314 | 14775 | 19841 | **8001** | **10653.93** | **4582.61** |

Computational time used on F-dataset by each MILP is reported in Table 4.12. The most computationally efficient MILP is the MILP-6 followed by the MILP-4, MILP-5, MILP-2, MILP-

3 and MILP-1 by CPU time of 4582.61, 8001, 1065.93, 14775, 19841 and 37314. The amount of time each MILP has spent is in proportion to its constituents. As a rule of thumb, those MILPs bearing fewer numbers of BIVs should consume less computational time than those bearing higher numbers of BIVs. Table 4.13 shows the strengths of MILPs on schedule generations.

Table 4.13, comparative evaluation of MILPs based on their quality of generated schedules

| *Instance no* | *Quality of generated schedules* | | | | | |
|---|---|---|---|---|---|---|
| | *MILP-1* | *MILP-2* | *MILP-3* | *MILP-4* | *MILP-5* | *MILP-6* |
| *SFJS1* | **66*** | **66*** | **66*** | **66*** | **66*** | **66*** |
| *SFJS2* | **107*** | **107*** | **107*** | **107*** | **107*** | **107*** |
| *SFJS3* | **221*** | **221*** | **221*** | **221*** | **221*** | **221*** |
| *SFJS4* | **355*** | **355*** | **355*** | **355*** | **355*** | **355*** |
| *SFJS5* | **119*** | **119*** | **119*** | **119*** | **119*** | **119*** |
| *SFJS6* | **320*** | **320*** | **320*** | **320*** | **320*** | **320*** |
| *SFJS7* | **397*** | **397*** | **397*** | **397*** | **397*** | **397*** |
| *SFJS8* | **253*** | **253*** | **253*** | **253*** | **253*** | **253*** |
| *SFJS9* | **210*** | **210*** | **210*** | **210*** | **210*** | **210*** |
| *SFJS10* | **516*** | **516*** | **516*** | **516*** | **516*** | **516*** |
| *MFJS1* | 470$^{BFI}$ | **468*** | **468*** | **468*** | **468*** | **468*** |
| *MFJS2* | 396 $^{LB}$ | **446*** | **446*** | **446*** | **446*** | **446*** |
| *MFJS3* | 396$^{LB}$ | **466*** | **466*** | **466*** | **466*** | **466*** |
| *MFJS4* | 496 $^{LB}$ | 565* | 564* | **554 *** | **554 *** | **554 *** |
| *MFJS5* | 414$^{LB}$ | **514*** | **514*** | **514*** | **514*** | **514*** |
| *MFJS6* | 469 $^{LB}$ | 616$^{BFI}$ | 635$^{BFI}$ | **608*** | **608*** | **608*** |
| *MFJS7* | 619 $^{LB}$ | 764$^{LB}$ | 935$^{BFI}$ | **881** | **881**$^{BFI}$ | **881**$^{BFI}$ |
| *MFJS8* | 619 $^{LB}$ | 764$^{LB}$ | 905$^{BFI}$ | **894** | **895**$^{BFI}$ | **894**$^{BFI}$ |
| *MFJS9* | 764 $^{LB}$ | 764$^{LB}$ | 1192$^{BFI}$ | 1192$^{BFI}$ | 1192$^{BFI}$ | **1135**$^{BFI}$ |
| *MFJS10* | **944** $^{LB}$ | **944**$^{LB}$ | **1276** $^{BFI}$ | **1276** $^{BFI}$ | **1276** $^{BFI}$ | **1276** $^{BFI}$ |

Finally, MILPs are evaluated based on their quality of solutions. Based on Table 4.13, the MILP-6 surpasses all the MILPs in terms of quality of generated schedules. The MILP-1 obtains ten optimal solutions in the first ten instances of F-data and it produces one feasible solution for MFJS1. The MILP-2 obtains ten optimal solutions up to SFJS10 and it additionally produces six new feasible integer solutions up to MFJS6. The MILP-3 produced

14 optimal solutions out of the existing 20 instances of F-data. Both MILP-5 and MILP-6 obtained 16 optimal solutions on F-data. The objective function value of both the MILP-5 and the MILP-6 are the same on the first 18 instances of the problem up to MFJSP-8 but the MILP-6 obtains one better feasible solution on MFJS9. Therefore, in terms of quality of schedules, the MILP-6 outperforms the rest of MILPs.

## 4.5. Performance evaluation of the proposed meta-heuristic (AISA)

In this section, the AISA is compared vis-a-vis seven best-performing meta-heuristics in literature. Fattahi, Mehrabad et al. (2007) proposed six integrated and hierarchical meta-heuristics and applied them to F-data and showed that their proposed meta-heuristics perform well for the F-JSSP. Bagheri, Zandieh et al. (2010) proposed an effective AIA for F-JSSP and proved the efficiency of their algorithm by applying it to F-data. Therefore, in this thesis, the AISA is compared against seven best-performing meta-heuristics applied to F-data. The rationale behind selecting F-data for our comparison purposes lies in the fact that F-data has been designed for small-to-medium size instances of the problem. Since all the proposed mathematical models in literature have been applied to this dataset, optimum, feasible, and lower bound values of each of these instances are known a priori. Therefore, this dataset is a valuable dataset to measure the optimality gap of the proposed meta-heuristics. Having developed new mathematical models in this thesis, several new optimal and feasible solutions were obtained for F-data. Table 4.14 measures optimality gaps of AISA on F-data.

Table 4.14, comparison between AISA and solutions of the MILP-6

| Instance no | Size (j,l,i) | MILP-6 | | AISA | |
|---|---|---|---|---|---|
| | | CPU (s) | $C_{max}$ | CPU (s) | $C_{max}$ |
| SFJS1 | 2.2.2 | 0.02 | **66*** | **0.8** | **66*** |
| SFJS2 | 2.2.2 | 0.02 | **107*** | **0.8** | **107*** |
| SFJS3 | 3.2.2 | 0.03 | **221*** | **1.2** | **221*** |
| SFJS4 | 3.2.2 | 0.03 | **355*** | 1.2 | **355*** |
| SFJS5 | 3.2.2 | **0.03** | **119*** | **1.2** | **119*** |
| SFJS6 | 3.3.2 | **0.03** | **320*** | **1.2** | **320*** |
| SFJS7 | 3.3.5 | **0.02** | **397*** | **3** | **397*** |
| SFJS8 | 3.3.4 | 0.03 | **253*** | **2.4** | **253*** |
| SFJS9 | 3.3.3 | 0.05 | **210*** | **1.8** | **210*** |
| SFJS10 | 4.3.5 | 0.05 | **516*** | 4 | **516*** |
| MFJS1 | 5.3.6 | **0.39** | **468*** | 6 | **468*** |
| MFJS2 | 5.3.7 | **0.34** | **446*** | **7** | **446*** |
| MFJS3 | 6.3.7 | **0.64** | **466*** | 8.4 | **466*** |
| MFJS4 | 7.3.7 | 1634 | **554*** | **9.8** | **554*** |
| MFJS5 | 7.3.7 | **2.83** | **514*** | **9.8** | **514*** |
| MFJS6 | 8.3.7 | 1275 | **608*** | **11.2** | **608*** |
| MFJS7 | 8.4.7 | **60** | 881[BFI] | **11.2** | **879** [BFI] |
| MFJS8 | 9.4.8 | **60** | 895 [BFI] | **14.4** | **894** [BFI] |
| MFJS9 | 11.4.8 | **120** | 1135 [BFI] | **17.6** | **1088** [BFI] |
| MFJS10 | 12.4.8 | **600** | 1276 [BFI] | **19.2** | **1196** [BFI] |

Therefore, in order to demonstrate the excellence of the AISA and before the AISA is compared against best-performing meta-heuristics in literature, it is compared against the solutions obtained from the best-performing mathematical model. As can be seen in Table 4.14, the objective function values obtained by the MILP-6 are used as a basis for evaluating the performance of the AISA. The MILP-6 obtains optimal solutions for the first 16 instances of the problem (up to MFJS6), so does the AISA. In the last four instances of the problem (MFJS7 to MFJS10), the AISA produces better integer feasible solutions as opposed to the MILP-6. Therefore, the optimality gap of the AISA is zero on the first 16 instances of the problem. Since the AISA obtains better results even in the last four instances of the problem, it produces zero RPD. Therefore, AISA obtains best solutions on all instances of F-data.

From this comparison, the absolute superiority of the AISA is inferred. The optimality gap of the AISA on the F-data is 0%.

Now that excellence of the AISA was corroborated, it is compared against best-performing meta-heuristics in literature.

In this section, the AISA is compared vis-a-vis seven best-performing meta-heuristics in literature. Fattahi, Mehrabad et al. (2007) proposed six integrated and hierarchical meta-heuristics and applied them to F-data and showed that their proposed meta-heuristics perform well for the F-JSSP. Bagheri, Zandieh et al. (2010) proposed an effective AIA for F-JSSP and proved the efficiency of their algorithm by applying it to F-data. Table 4.15 and Table 4.16 have been designed to show this comparison. The customary procedure for evaluating the performance of any newly proposed meta-heuristic is to compare it with the best results of other existing meta-heuristics in literature on the same benchmark. Following the previous procedure does help evaluate the strength of the proposed meta-heuristic but it never provides any deviation value from optimal solution (optimality gap). In this thesis, thanks to the optimal solutions of MILPs on the first 16 instances of the problem, the optimality gap of the AISA can be measured. Table 4.15 measures optimality gap of all the meta-heuristics in literature and also the proposed meta-heuristic (AISA) on the first 16 instances of F-data. Inasmuch as optimal solutions for the last four instances of F-data do not exist, the RPDs of meta-heuristics are calculated by comparing them with the results of the best-performing meta-heuristic.

Table 4.15, comparison with the state-of-the-art integrated approaches on F-data.

| *Instance* | *MILP-6* | | *AISA* | | *AIA* | | *ISA* | | *ITS* | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $C_{max}$ | *RPD* | $C_{max}$ | *RPD* | $C_{max}$ | *RPD* | $C_{max}$ | *RPD* | $C_{max}$ | *RPD* |
| SFJS1 | **66*** | 0 | **66*** | 0 | **66*** | 0 | **66*** | **0** | **66*** | **0** |
| SFJS2 | **107*** | 0 | **107*** | 0 | **107*** | 0 | **107*** | **0** | **107*** | **0** |
| SFJS3 | **221*** | 0 | **221*** | 0 | **221*** | 0 | **221*** | **0** | **221*** | **0** |
| SFJS4 | **355*** | 0 | **355*** | 0 | **355*** | 0 | **355*** | **0** | 390 | 9.86 |
| SFJS5 | **119*** | 0 | **119*** | 0 | **119*** | 0 | **119*** | **0** | 137 | 15 |
| SFJS6 | **320*** | 0 | **320*** | 0 | **320*** | 0 | **320*** | **0** | **320*** | **0** |
| SFJS7 | **397*** | 0 | 397* | 0 | **397*** | 0 | **397** | **0** | **397** | **0** |
| SFJS8 | **253*** | 0 | **253*** | 0 | **253*** | 0 | **253*** | **0** | **253*** | **0** |
| SFJS9 | **210*** | 0 | **210*** | 0 | **210*** | 0 | **215*** | **0** | 215 BFI | 0 |
| SFJS10 | **516*** | 0 | **516*** | 0 | **516*** | 0 | **516*** | **0** | 617 BFI | 19.57 |
| MFJS1 | **468*** | **0** | **468*** | **0** | **468*** | **0** | 488 BFI | 4.22 | 548 BFI | 17.1 |
| MFJS2 | **446*** | **0** | **446*** | **0** | 448 BFI | **0.44** | 478 BFI | 6.69 | 457 BFI | 2.46 |
| MFJS3 | **466*** | **0** | **466*** | **0** | 468 BFI | **0.43** | 599 BFI | 0.43 | 606 BFI | 30 |
| MFJS4 | **554 *** | **0** | **554*** | **0** | **554*** | **0** | 703 BFI | 26.8 | 870 BFI | 57 |
| MFJS5 | **514*** | **0** | **514*** | **0** | 527 BFI | **2.5** | 674 BFI | 31.1 | 729 BFI | 41.8 |
| MFJS6 | **608*** | **0** | **608*** | **0** | 635 BFI | **4.4** | 856 BFI | 40.7 | 816 BFI | 34.2 |
| MFJS7 | 881 BFI | 0.22 | **879** BFI | **0** | **879** BFI | **0.22** | 1066 BFI | 21.2 | 1048 BFI | 19.2 |
| MFJS8 | 895 BFI | 1.24 | 894 BFI | 1.13 | **884** BFI | **0** | 1328 BFI | 50.22 | 1220 BFI | 38 |
| MFJS9 | 1135 BFI | 4.31 | **1088** BFI | **0** | **1088** BFI | **0** | 1148 BFI | 5.5 | 1124 BFI | 3.3 |
| MFJS10 | **1276** BFI | 6.7 | 1196 BFI | **0** | 1267 BFI | **6.7** | 1546 BFI | 29.3 | 1737 BFI | 45.2 |
| Total RPD | | 12.47 | | **1.13** | | 14.7 | | 187 | | 356.9 |
| Average RPD | | 0.6235 | | 0.0565 | | 0.735 | | 9.36 | | 17.84 |

Based on Table 4.15, the MILP-6 provides optimal solutions on F-data up to MFJS6 and so does the AISA. It means that the optimality gap of AISA is zero on the first 16 instances of the problem. This fact shows that the AISA obtains optimal solutions on instances where the MILP-6 obtains optimal solutions. In the last four instances of F-data, both the MILP-6 and the AISA obtain feasible integer solutions. Interestingly enough, feasible integer solutions acquired by AISA dominate those obtained by the MILP-6. Therefore, what can be concluded is the fact that even in presence of optimal and near-optimal solutions of the best-performing MILP (MILP-6), the AISA works better than the MILP-6. Proposed AISA dominates other best-performing meta-heuristics in literature as well. Based on values reported in Table 4.15, AISA with average RPD of 0.0565 is the best-performing solution technique followed by MILP-6, AIA, ISA and ITS with average RPDs of 0.6235, 0.735, 9.36 and 17.845 respectively.

Among best-performing meta-heuristics in literature, the AIA proposed by (Bagheri et al. 2010) is the best competitor for the AISA. After the extensive evaluations of the MILP-6 and other best-performing meta-heuristics, it can be asserted that the AISA is the best solution methodology ever presented for the F-JSSP.

Table 4.16, comparison with the state-of-the-art hierarchical approaches on Fdata

| *Instance* | *MILP* | *AISA* | | *HSA/SA* | | *HSA/TS* | | *HTS/TS* | | *HTS/SA* | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | $C_{max}$ | $C_{max}$ | *RPD* | $C_{max}$ | *RPD* | $C_{max}$ | *RPD* | $C_{max}$ | *RPD* | $C_{max}$ | *RPD* |
| SFJS1 | **66*** | **66*** | **0** | **66*** | **0*** | **66*** | **0*** | **66*** | **0*** | **66*** | **0*** |
| SFJS2 | **107*** | **107*** | **0** | **107*** | **0*** | **107*** | **0*** | **107*** | **0*** | **107*** | **0*** |
| SFJS3 | **221*** | **221*** | **0** | **221*** | **0*** | **221*** | **0*** | **221*** | **0*** | **221*** | **0*** |
| SFJS4 | **355*** | **355*** | **0** | **355*** | **0*** | **355*** | **0*** | **355*** | **0*** | **355*** | **0*** |
| SFJS5 | **119*** | **119*** | **0** | **119*** | **0*** | **119*** | **0*** | **119*** | **0*** | **119*** | **0*** |
| SFJS6 | **320*** | **320*** | **0** | **320*** | **0*** | **320*** | **0*** | **320*** | **0*** | **320*** | **0*** |
| SFJS7 | **397*** | **397*** | **0** | **397*** | **0*** | **397*** | **0*** | **397*** | **0*** | **397*** | **0*** |
| SFJS8 | **253*** | **253*** | **0** | **253*** | **0*** | **253*** | **0*** | **253*** | **0*** | 256[BFI] | **0*** |
| SFJS9 | **210*** | **210*** | **0** | **210*** | **0*** | **210*** | **0*** | **210*** | **0*** | **210*** | **0*** |
| SFJS10 | **516*** | **516*** | **0** | **516*** | **0*** | **516*** | **0*** | **516*** | **0*** | 519[BFI] | 0.58 |
| MFJS1 | **468*** | **468*** | **0** | 479 [BFI] | 2.35 | 491[BFI] | 2.5 | 469 [BFI] | 0.1 | 469[BFI] | 0.21 |
| MFJS2 | **446*** | **446*** | **0** | 495 [BFI] | 10.76 | 482[BFI] | 8.1 | 482[BFI] | 8.1 | 468[BFI] | 4.93 |
| MFJS3 | **466*** | **466*** | **0** | 553 [BFI] | 18.67 | 538[BFI] | 15.5 | 533[BFI] | 14.38 | 538[BFI] | 15.4 |
| MFJS4 | **554 *** | **554*** | **0** | 656 [BFI] | 18.41 | 650[BFI] | 17.3 | 634[BFI] | 14.4 | 618[BFI] | 11.55 |
| MFJS5 | **514*** | **514*** | **0** | 650 [BFI] | 26.45 | 662 [BFI] | 28.8 | 625[BFI] | 21.6 | 625[BFI] | 21.6 |
| MFJS6 | **608*** | **608*** | **0** | 762 [BFI] | 25.32 | 785[BFI] | 29.1 | 717[BFI] | 17.92 | 730[BFI] | 20 |
| MFJS7 | 88[BFI] | **879**[BFI] | **0** | 1020[BFI] | 16 | 108[BFI] | 22.9 | 964[BFI] | 9.67 | 947[BFI] | 7.7 |
| MFJS8 | 895[BFI] | **894**[BFI] | **0** | 1030[BFI] | 15.21 | 1122[BFI] | 25.5 | 970[BFI] | 8.5 | 922[BFI] | 3.1 |
| MFJS9 | 1135[BFI] | **1088**[BFI] | **0** | 1180[BFI] | 8.45 | 1243[BFI] | 14.24 | 1105[BFI] | 1.56 | 1105[BFI] | 1.56 |
| MFJS10 | 1276[BFI] | **1196**[BFI] | **0** | 1538[BFI] | 28.6 | 1615[BFI] | 35 | 1404[BFI] | 17.4 | 1384[BFI] | 15.7 |
| Total RPD | | | **0** | | 170.22 | | 199 | | 113.63 | | 102.33 |
| Average RPD | | | **0** | | 8.51 | | 9.95 | | 5.68 | | 5.11 |

As can be seen, the AISA with average RPD of *zero* is the best-performing algorithm. The HTS/SA, HTS/TS, HSA/SA and HSA/TS with average RPDs of 5.11, 5.68, 8.51 and 9.95 are

ranked second, third, fourth and fifth respectively. Based on Tables 4.15 and 4.16, it can be concluded that hierarchical approaches proposed by (Fattahi et al. 2007) outperforms their integrated approaches. The same conclusion can be made regarding the performances of the AISA with the AIA proposed by (Bagheri et al. 2010) which follows the structure of integrated approaches.

## 4.6. Significance of improved results:

It might be deemed that the improved results are just some improved figures with no practical significance. Previous statement is by no means true. In order to highlight the practical importance of newly obtained results, some extreme examples from F-data is supplied. The optimal solution for SFJS10 is 516 hours while (Fattahi, Mehrabad et al. 2007) algorithm (ITS) has obtained the value of 617 hours. The best feasible integer solution for MFJS10 is 1196 hours while the ITS has obtained the value of 1737 hours. Considering eight hours per working shift, the proposed AISA saves 103 and 541 hours on SFJS10 and MFJS10 instances of the problem which are equivalent to 13 and 68 working shifts respectively. Taking into consideration the number of workers per shift and all the related direct and indirect costs associated with it, these savings play huge role in the productivity of any machine shop.

# CHAPTER 5

# CASE STUDY

Despite intensely growing competition in academic society for developing state-of-the-art optimization methodologies for wide variety of industrial applications including production scheduling environments, most industries are barely familiar with these techniques and they use very basic techniques incorporated in commercial software packages to meet their short-term and long-term scheduling needs. In this thesis, the developed scheduling optimization methodologies are applied to an industrial problem for demonstration and testing. Below, detailed activities of a molding job shop are described. The company supplies many automotive companies in North America. In this thesis, the used case study is limited to the scheduling of mould making machines on the shop floor while mould assembly operations after completing the machining operations are not considered.

The mould and die manufacturing company makes different moulds used to create products such as tail lenses or head lamp reflectors for automotive manufacturers. They design and build sand production thermo-plastic, thermoset, multi-color and multi-material injection moulds using ejectable thermo plastic press, and specialize in moulds for automobiles head and tail lights parts.

The data used in the case study include the following inputs:

### Initial Input (static):

1. Machine type groups and names and number of machines in each group.
2. Capabilities for each machine in a group including typical/primary usage, alternate usage etc.
3. Work shifts: normal number of shifts/per day, number of hours per shift. This could be changed due to use of overtime or unmanned hours.
4. Mould hierarchy (mould components and their instances and codes for the composite mould).
5. Normal processing times for each operation
6. Current machine loading status (which machine is working on which part/operation).
7. Approximate set-up and dismantling time: Constant or fraction of each operation time which is included in processing times.

As an example, a mould (Job) has two main parts: the core side and the cavity side (male/female). A typical mould consists of: a) *Cavity* b) *Core c) Slides d) Retractor and e) Clamp Plates*. The machining processes consist of the following operations: 1) *roughing,* 2) *semi-finishing,* 3) *stress relief,* 4) *finishing,* 5) *boring,* 6) *gun drilling,* 7) *carbon,* 8) *electro discharge machining (EDM), and finally 9) mould assembly*. The scheduling of machining operations using 14 CNC machines is considered. All cavities and cores of a typical mould require all the above-mentioned operations. Mould slides need only four operations: roughing, finishing, carbon and EDM. Retractors require two operations: roughing and finishing and clamp plates need only a boring mill operation. Operations naming scheme is exemplified as follows: $O_{11}$ denotes the first operation of cavity#1 which is roughing. $O_{42}$ represents the second operation of part number four in retractor#1 which is finishing. Therefore, the first index in $O_{j,l}$, $j$ indicates the part number and the second index, $l$, denotes the required operation.

Table 5.1, Operational coding and requirement of different parts

| No | No | Part names | Required operations |
|---|---|---|---|
| Mould # 1 | Part1 | Cavity1 | $O_{11}$, $O_{12,}$ $O_{13,}$ $O_{14,}$ $O_{15,}$ $O_{16,}$ $O_{17,}$ $O_{18}$ |
| | Part2 | Core1 | $O_{21}$, $O_{22,}$ $O_{23,}$ $O_{24,}$ $O_{25,}$ $O_{26,}$ $O_{27,}$ $O_{28}$ |
| | Part3 | Slide1 | $O_{31}$, $O_{32,}$ $O_{33,}$ $O_{34}$ |
| | Part4 | Retractor1 | $O_{41}$, $O_{42}$ |
| | Part5 | Clamp-Plates1 | $O_{51}$ |
| Mould # 2 | Part6 | Cavity2 | $O_{61}$, $O_{62,}$ $O_{63,}$ $O_{64,}$ $O_{65,}$ $O_{66,}$ $O_{67,}$ $O_{68}$ |
| | Part7 | Core2 | $O_{71}$, $O_{72,}$ $O_{73,}$ $O_{74,}$ $O_{75,}$ $O_{76,}$ $O_{77,}$ $O_{78}$ |
| | Part8 | Slide2 | $O_{81}$, $O_{82,}$ $O_{83,}$ $O_{84}$ |
| | Part9 | Retractor2 | $O_{91}$, $O_{92}$ |
| | Part10 | Clamp-Plates2 | $O_{101}$ |
| Mould # 3 | Part11 | Cavity3 | $O_{111}$, $O_{112,}$ $O_{113,}$ $O_{114,}$ $O_{115,}$ $O_{116,}$ $O_{117,}$ $O_{118}$ |
| | Part12 | Core3 | $O_{121}$, $O_{122,}$ $O_{123,}$ $O_{124,}$ $O_{125,}$ $O_{126,}$ $O_{127,}$ $O_{128}$ |
| | Part13 | Slide3 | $O_{131}$, $O_{132,}$ $O_{133,}$ $O_{134}$ |
| | Part14 | Retractor3 | $O_{141}$, $O_{142}$ |
| | Part15 | Clamp-Plates3 | $O_{151}$ |
| Mould # 4 | Part16 | Cavity4 | $O_{161}$, $O_{162,}$ $O_{163,}$ $O_{164,}$ $O_{165,}$ $O_{166,}$ $O_{167,}$ $O_{168}$ |
| | Part17 | Core4 | $O_{171}$, $O_{172,}$ $O_{173,}$ $O_{1744,}$ $O_{175,}$ $O_{176,}$ $O_{177,}$ $O_{178}$ |
| | Part18 | Slide4 | $O_{181}$, $O_{182,}$ $O_{183,}$ $O_{184}$ |
| | Part19 | Retractor4 | $O_{191}$, $O_{192}$ |
| | Part20 | Clamp-Plates4 | $O_{201}$ |

The company currently uses the following CNC machines for machining the moulds as shown in Table 5.1. Table 5.1, illustrates the operational requirements per part. Table 5.2 shows the names and capabilities of each machine.

Table 5.2, List of machine capabilities

| Operations | Names of eligible machines |
|---|---|
| Roughing | $M_1$(AWEA)-$M_2$(Johnford)-$M_3$(Dynamic)-$M_4$ (Eumach) |
| Stress relief | $M_5$ (Outsourced) |
| Semi-finishing | $M_4$(Eumach)-$M_3$(Dynamic) |
| Finishing | $M_6$(Exceeder 1)-$M_7$ (Exceeder 2) |
| Boring Mill | $M_8$(Kuraki), $M_9$(Parpas), $M_{10}$(Takumi) |
| Gun-drilling | $M_{11}$(Outsourced) |
| Carbon | $M_3$( Dynamic)-$M_{12}$(Datic) |
| E.D.M | $M_{13}$(Techno)-$M_{14}$(NX8) |

Table 5.2 demonstrates machine capabilities. Based on established policy in the mould and die company, certain priorities have been assigned to CNC machines to ensure the best quality and even distribution of workloads between machines. Table 5.3, shows the coefficients assigned to each machine. Table 5.2 reveals the fact that machines are flexible but their flexibilities are not identical. In front of each operation, a number of capable machines has been cited. Therefore, this case study follows the principles of PF-JSSP as each machine performs certain operations and therefore different parts do not have total routing freedom. This means that for each part, certain preferred machines have been designated. The algorithm that is used (AISA) to solve this case study has to take care of machine assignment flexibility and should choose the best available machine.

Table 5.3, priority coefficient assigned to machines in shop floor

| Operations | Names of eligible machines |
|---|---|
| Roughing | (1) $M_1$, (1.1) $M_2$, (1.2) $M_3$, (1.3) $M_4$ |
| Stress relief | (1) $M_5$ |
| Semi-finishing | (1) $M_4$), (1.1) $M_3$ |
| Finishing | (1) $M_6$, (1)$M_7$ |
| Boring Mill | (1) $M_8$, (1.1) $M_9$, (1.2) $M_{10}$ |
| Gun-drilling | (1) $M_{11}$ |
| Carbon | (1) $M_3$, (1.1) $M_{12}$ |
| E.D.M | (1) $M_{13}$, (1.1) $M_{14}$ |

Coefficients shown in Table 5.3 are multiplied by the processing times of operations on each machine. The higher the coefficient assigned to a machine, the less desirable it gets for processing certain operations. These coefficients are adjusting factors for balancing machine workloads. The typical picture of the mould is demonstrated in Figure 5.1.

Figure 5.1. Typical complete mould consisting of core and cavity (moulded part is shown in blue) http://www.hongsenmould.com/industrial_goods_molds.html

Therefore, considering above capabilities of machines, the proper assigning and sequencing of four moulds (jobs) consisting of 20 parts with 92 operations on 14 CNC machines is an application used to verify the scheduling algorithms developed in this thesis.

Table 5.4. Decoded solution of AISA with starting and completing time of each operation

| | |
|---|---|
| $M_1$ | $O_{11}$(0-60), $O_{121}$(60-132), $O_{161}$(132-187), $O_{31}$(187-198), $O_{171}$(198-265), $O_{111}$(265-325), $O_{141}$(325-339), $O_{91}$(336-371) |
| $M_2$ | $O_{21}$(0-79), $O_{71}$(79-153), $O_{131}$(153-169), $O_{61}$(169-231) |
| $M_3$ | $O_{16}$(261-345), $O_{25}$(345-429), $O_{76}$(84-513), $O_{176}$(513-549), $O_{166}$(549-585), $O_{126}$(585-669), $O_{66}$(669-753), $O_{116}$(753-837) |
| $M_4$ | $O_{13}$(145-169), $O_{23}$(169-193), $O_{73}$(233-281), $O_{163}$(281-317), $O_{173}$(335-371), $O_{123}$(371-461), $O_{113}$(461-497), $O_{63}$(497-545) |
| $M_5$ | $O_{12}$(60-145), $O_{191}$(145-168), $O_{162}$(187-263), $O_{172}$(265-335), $O_{41}$(-351), $O_{122}$(351-425) |
| $M_6$ | $O_{81}$(0-30), $O_{181}$(30-53), $O_{22}$(79-159), $O_{72}$(159-233), $O_{62}$(233-318), $O_{112}$(325-419), $O_{183}$(419-439), $O_{83}$(-464), $O_{177}$(549-575), $O_{167}$(585-617), $O_{17}$(617-649), $O_{67}$(753-774), $O_{117}$(837-858) |
| $M_7$ | $O_{132}$(169-189), $O_{182}$(189-210), $O_{32}$(210-245), $O_{164}$(317-371), $O_{174}$(371-431), $O_{92}$(-467), $O_{192}$(-505), $O_{64}$(545-597) |
| $M_8$ | $O_{82}$(30-50), $O_{14}$(169-221), $O_{24}$(221-291), $O_{74}$(291-351), $O_{42}$(351-391), $O_{124}$(461-521), $O_{114}$(521-573), $O_{142}$(573-589) |
| $M_9$ | $O_{51}$(0-36), $O_{15}$(221-261), $O_{115}$(573-613), $O_{65}$(597-642), |
| $M_{10}$ | $O_{101}$(0-45), $O_{165}$(371-425), $O_{151}$(425-470), $O_{125}$(521-580) |
| $M_{11}$ | $O_{75}$(351-416), $O_{175}$(431-504), $O_{201}$(504-554), $O_{26}$ (554-638), |
| $M_{12}$ | $O_{133}$(189-201), $O_{33}$(245-270), $O_{77}$(513-553), $O_{27}$(638-685), $O_{127}$(685-715), |
| $M_{13}$ | $O_{178}$(575-621), $O_{84}$(621-641), $O_{18}$(649-675), $O_{34}$(675-694), $O_{68}$(774-800), $O_{118}$(858-881) |
| $M_{14}$ | $O_{134}$(201-236), $O_{78}$(553-613), $O_{184}$(613-652), $O_{128}$(715-769), $O_{168}$(769-795), $O_{28}$(795-855) |

Figure 5.2, decoded solution of the case study in format of a Gantt chart

Figure 5.2 depicts the decoded solution of the case study in format of a Gantt chart. The case study has 20 jobs (j=20) and 14 CNC (m=14). This is considered a large size instance of F-JSSP. In Figure 5.2, there are certain numbers and symbols. In following, they are completely explained. In each cell, the name of each operation has been included. Below each cell, there exist two numbers. The first number is the processing time of that operation on that machine and the second number is the completion time of that operation. For example, $O_{13}$ on $M_4$ has processing time of 24 and completion time is 169 which is the summation of processing times of preceding operations of $O_{13}$ ($O_{11}$=60 on $M_1$, $O_{12}$=85 on $M_5$ and $O_{13}$=24 on $M_4$).

Having done rigorous assessment of the performance of the proposed AISA through comparative analyses, the developed AISA is applied to solve real case study of PF-JSSP with 20 parts (92 operations) on 14 flexible machining centers.

The AISA utilized runtime of 56 seconds. This value obtained by multiplying number of parts and number of machines with 0.2 seconds (n=20*m=14*0.2 seconds) to solve the case study. As the results show, the developed AISA is a fast algorithm capable of finding near/sub-optimal solutions for any large size problem. In order to solve the case study, data regarding both the processing times of all parts on different machines and also their eligible machines were collected. The near-optimum make-span of 881 hours was obtained using AISA. This time is equivalent to 110 working shifts. The decoded solution of AISA for the

case study is depicted in Figure 5.2. Each part is shown with a unique color so that the processing route of each part on different machines can be tracked. In the Gantt chart, each operation of a part has two numbers; the first is operation processing time, and the second is the summation of processing times a part has received so far. The company used to manufacture these 92 operations on its current resources around 960 hours which was equivalent to 120 working shifts (two shifts per day including weekdays and weekends). The difference between the time typically spent by the company and the time stipulated by applying the AISA is 79 hours which is equivalent to nearly ten working shifts saving. Considering this production time saving, if the company decides to apply the AISA, it can reduce the mould manufacturing time by 8.3%. Considering the wages of workers, inventory cost, overhead costs etc, and this machining time reduction would potentially result in a significant difference in time, cost, and operational efficiency.

In this section, an industrial scale case study was solved by the state-of-the-art meta-heuristic (AISA). The obtained results were unfolded into an equivalent Gantt chart. To solve the case study, information regarding machine capabilities, operational requirements per part, processing time of each operation of different machines, and preference of machine usage were gathered and analyzed. By comparing the results obtained by AISA with those of the company, 79 hours were saved which is equivalent to ten working shifts.

# CHAPTER 6

# CONCLUSIONS AND FUTURE STUDIES

## 6.1 Conclusions

The problem of flexible job shop scheduling problem (F-JSSP) was researched in this thesis. Two different types of F-JSSP known as partially and totally flexible job shops were studied and solved. Different modeling paradigms, namely position-based and sequence-based were utilized to fully demonstrate operations of the shop floor. Proposed mathematical models contained both continuous and binary integer variables. Therefore, they are categorized as mixed integer linear programming (MILP) models. Rigorous comparative evaluations among proposed MILPs and those existing in literature corroborated the computational efficiency and solution effectiveness of proposed MILPs. The proposed position-based MILP outperformed the MILP proposed by (Fattahi et al. 2007) in all intended performance measures. The proposed MILP used substantially fewer binary integer and continuous variables and also number of constraints for standard benchmark of (Fattahi et al. 2007). The proposed position-based MILP consumed much less computational time and generated enhanced feasible solutions for MFJS-1 to MFJS-6. Three other sequence-based MILP models were also proposed. All three proposed sequence-based MILPs dominated the best-performing MILP proposed by (Ozguven, Ozbakir et al. 2010) in literature in terms of solution effectiveness.

Since MILPs are unable to solve industrial scale problems, an enhanced meta-heuristic which is a hybrid of AIA and SA (AISA) was proposed and applied to solve flexible job shop scheduling problems. In order to ensure that the AISA is effective and suitable for the problem at hand, it was compared against seven of best-performing meta-heuristics in literature. The obtained results manifestly proved the efficiency and effectiveness of the AISA.

Having done extensive analyses on AISA, it was applied to an industrial case study. The obtained result of AISA on industrial dataset was unfolded into a Gantt chart.

To recapitulate, in this thesis, two classes of solution methodologies were developed: mathematical models and meta-heuristics. Superiority of the developed methodologies can be attributed to the hierarchical structures of them. In the hierarchical approach, infeasible solutions are avoided and the available computational time is utilized to completely explore the search space.

## 6.2 Future studies

The following research topics can be pursued to bridge current gaps in literature.

I. The industrial problem solved in this thesis encompassed two parts: machining and assembly operations. Scheduling of machining operations was considered in this study while scheduling of assembly operations was not addressed in our solution methodologies. A mathematical model and a new meta-heuristic can be proposed to solve the mixed shop consisting of PF-JSSP and Flow-shop scheduling problem.

II. Dynamic version of the proposed mathematical model can be examined to include addition or removal of machines, order cancellations or arrivals, machine breakdown etc during the scheduling horizon.

III. Distributed version of the proposed mathematical model can be offered to include the scheduling of more than one production facility according to real-life problems.

IV. The previous two problems can be combined as a very realistic scheduling problem.

V. Transportation times among machines in the shop floor and production facilities can be included in above models.

VI. Since the proposed models proved their efficiency, it is recommended that they be extended to multi-objective cases by considering additional objectives such as make-span and maximum tardiness, minimization of maximal machine workload etc.

VII. Conducting comparative evaluations of production scheduling modelling paradigms for F-JSSP (position-based, time-based, and sequence-based).

VIII. Integrating process plan flexibility into F-JSSP by using the proposed MILPs

# References


Abdallah, I. B., H. A. Elmaraghy and T. Elmekkawy (2002). "Deadlock-free scheduling in flexible manufacturing systems using Petri nets." International Journal of Production Research **40**(12): 2733-2756.

Adibi, M. A., M. Zandieh and M. Amiri (2010). "Multi-objective scheduling of dynamic job shop using variable neighborhood search." Expert Systems with Applications **37**(1): 282-287.

Al-Hinai, N. and T. Y. ElMekkawy (2011). "An efficient hybridized genetic algorithm architecture for the flexible job shop scheduling problem." Flexible Services and Manufacturing Journal **23**(1): 64-85.

Bagheri, A. and M. Zandieh (2011). "Bi-criteria flexible job-shop scheduling with sequence-dependent setup times-Variable neighborhood search approach."

Bagheri, A., M. Zandieh, I. Mahdavi and M. Yazdani (2010). "An artificial immune algorithm for the flexible job-shop scheduling problem." Future Generation Computer Systems **26**(4): 533-541.

Ben Hmida, A., M. Haouari, M. J. Huguet and P. Lopez (2010). "Discrepancy search for the flexible job shop scheduling problem." Computers & Operations Research **37**(12): 2192-2201.

Bowman, E. H. (1959). "Schedule-sequencing problem." Operations Research **7**(5): 612-614.

Bozejko, W., M. Uchronski and M. Wodecki (2010). "Parallel hybrid metaheuristics for the flexible job shop problem." Computers and Industrial Engineering **59**(2): 323-333.

Brandimarte, P. (1993). "Routing and scheduling in a flexible job shop by tabu search." Annals of Operations Research **41**(1-4): 157-183.

Brandimarte, P. (1999). "Exploiting process plan flexibility in production scheduling: a multi-objective approach." European Journal of Operational Research **114**(1): 59-71.

Brucker, P. and J. Neyer (1998). "Tabu-search for the multi-mode job-shop problem." OR Spektrum **20**(1): 21-28.

Brucker, P. and R. Schlie (1990). "Job-shop scheduling with multi-purpose machines." Computing **45**(4): 369-375.

Chen, H., J. Ihlow and C. Lehmann (1999). A genetic algorithm for flexible job-shop scheduling. Proceedings of International Conference on Robotics and Automation, 10-15 May 1999, Piscataway, NJ, USA, IEEE.

Choi, I.-C. and D.-S. Choi (2002). "A local search algorithm for jobshop scheduling problems with alternative operations and sequence-dependent setups." Computers and Industrial Engineering **42**(1): 43-58.

Dauzere-Peres, S. and J. Paulli (1997). "An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search." Annals of Operations Research **70**: 281-306.

Dauzere-Peres, S., W. Roux and J. B. Lasserre (1998). Multi-resource shop scheduling with resource flexibility. Project Management and Scheduling: Fifth International Workshop, 11-13 April 1996, Netherlands, Elsevier.

De Giovanni, L. and F. Pezzella (2010). "An Improved Genetic Algorithm for the Distributed and Flexible Job-shop Scheduling problem." European Journal of Operational Research **200**(2): 395-408.

ElMaraghy, H. A. (2005). Flexible and reconfigurable manufacturing systems paradigms. Reconfigurable Manufacturing Systems, Kluwer Academic Publishers.

Fattahi, P., F. Jolai and J. Arkat (2009). "Flexible job shop scheduling with overlapping in operations." Applied Mathematical Modelling **33**(7): 3076-3087.

Fattahi, P., M. S. Mehrabad and F. Jolai (2007). "Mathematical modeling and heuristic approaches to flexible job shop scheduling problems." Journal of Intelligent Manufacturing **18**(3): 331-342.

Gao, J., M. Gen and L. Sun (2006). "Scheduling jobs and maintenances in flexible job shop with a hybrid genetic algorithm." Journal of Intelligent Manufacturing **17**(4): 493-507.

Gao, J., L. Sun and M. Gen (2008). "A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems." Computers and Operations Research **35**(9): 2892-2907.

Garey, M. R., D. S. Johnson and R. Sethi (1976). "The complexity of flowshop and jobshop scheduling." Mathematics of Operations Research **1**(2): 117-129.

Girish, B. S. and N. Jawahar (2009). A particle swarm optimization algorithm for flexible job shop scheduling problem. 2009 IEEE International Conference on Automation Science and Engineering (CASE 2009), 22-25 Aug. 2009, Piscataway, NJ, USA, IEEE.

Gomes, M. C., A. P. Barbosa-Povoa and A. Q. Novais (2005). "Optimal scheduling for flexible job shop operation." International Journal of Production Research **43**(11): 2323-2353.

Gutierrez, C. and I. Garcia-Magarino (2011). "Modular design of a hybrid genetic algorithm for a flexible job-shop scheduling problem." Knowledge-Based Systems **24**(1): 102-112.

Ho, N. B. and J. C. Tay (2004). GENACE: an efficient cultural algorithm for solving the flexible job-shop problem. Proceedings of the 2004 Congress on Evolutionary Computation, 19-23 June 2004, Piscataway, NJ, USA, IEEE.

Ho, N. B., J. C. Tay and E. M. K. Lai (2007). "An effective architecture for learning and evolving flexible job-shop schedules." European Journal of Operational Research **179**(2): 316-333.

Hurink, J., B. Jurisch and M. Thole (1994). "Tabu search for the job-shop scheduling problem with multi-purpose machines." OR Spektrum **15**(4): 205-215.

Hussain, M. F. and S. B. Joshi (1998). A genetic algorithm for job shop scheduling problems with alternate routing. SMC '98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics, 11-14 Oct. 1998, New York, NY, USA, IEEE.

Imanipour, N. and S. H. Zegordi (2006). "A heuristic approach based on Tabu Search for early/tardy flexible job shop problems." Scientia Iranica **13**(1): 1-13.

Johnson, S.M. (1954). "Optimal two and three stage production schedules and setup times included." Nav. Res. Logist. Quart.

Jiyin, L. and B. L. MacCarthy (1997). "A global MILP model for FMS scheduling." European Journal of Operational Research **100**(3): 441-453.

Kacem, I. (2003). Genetic algorithm for the flexible job-shop scheduling problem. SMC '03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics, 5-8 Oct. 2003, Piscataway, NJ, USA, IEEE.

Kacem, I., S. Hammadi and P. Borne (2002). "Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems." IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews) **32**(1): 1-13.

Kim, K. H. and P. J. Egbelu (1999). "Scheduling in a production environment with multiple process plans per job." International Journal of Production Research **37**(12): 2725-2753.

Lee, Y. H., C. S. Jeong and C. Moon (2002). "Advanced planning and scheduling with outsourcing in manufacturing supply chain." Computers and Industrial Engineering **43**(1-2): 351-374.

Liu, H., A. Abraham and Z. Wang (2009). "A multi-swarm approach to multi-objective flexible job-shop scheduling problems." Fundamenta Informaticae **95**(4): 465-489.

Low, C. and T.-H. Wu (2001). "Mathematical modelling and heuristic approaches to operation scheduling problems in an FMS environment." International Journal of Production Research **39**(4): 689-708.

Low, C., Y. Yukling and W. Tai-Hsi (2006). "Modelling and heuristics of FMS scheduling with multiple objectives." Computers & Operations Research **33**(3): 674-694.

Manne, A. S. (1960). "On job-shop scheduling problem." Operations Research **8**(2): 219-223.

Mastrolilli, M. and L. M. Gambardella (2000). "Effective neighbourhood functions for the flexible job shop problem." Journal of Scheduling **3**(1): 3-20.

Moradi, E., S. M. T. Fatemi Ghomi and M. Zandieh (2011). "Bi-objective optimization research on integrated fixed time interval preventive maintenance and production for scheduling flexible job-shop problem." Expert Systems with Applications **38**(6): 7169-7178.

Moslehi, G. and M. Mahnam (2011). "A Pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search." International Journal of Production Economics **129**(1): 14-22.

Ozguven, C., L. Ozbakir and Y. Yavuz (2010). "Mathematical models for job-shop scheduling problems with routing and process plan flexibility." Applied Mathematical Modelling **34**(6): 1539-1548.

Ozguven, C., Y. Yavuz and L. Ozbakir (2012). "Mixed integer goal programming models for the flexible job-shop scheduling problems with separable and non-separable sequence dependent setup times." Applied Mathematical Modelling **36**(2): 846-858.

Pan, C.-H. (1997). "Study of integer programming formulations for scheduling problems." International Journal of Systems Science **28**(1): 33-41.

Pinedo, M. (2002). "Scheduling: Theory, Algorithms and Systems", Prentice-Hall, Englewood Cliffs, NJ.

Pezzella, F., G. Morganti and G. Ciaschetti (2008). "A genetic algorithm for the flexible job-shop scheduling problem." Computers and Operations Research **35**(10): 3202-3212.

Roshanaei, V., M. M. Seyyed Esfehani and M. Zandieh (2010). "Integrating non-preemptive open shops scheduling with sequence-dependent setup times using advanced metaheuristics." Expert Systems with Applications **37**(1): 259-266.

Roshanaei, V., H. ElMaraghy, A. Azab (2012-a). "Enhanced mixed integer linear programing for flexible job shop". Proceedings of the 4th Conference on Assembly Technologies and Systems, 195-198, May 20-22, 2012, Ann Arbor, Michigan, USA.

Roshanaei, V., H. ElMaraghy, A. Azab (2012-b). "Sequence-based MILP Modeling for Flexible Job Shop Scheduling". Proceedings of IIE Conference & Expo 2012, May 19-23, Orlando, Florida, USA, May 19-23, 2012.

Rossi, A. and G. Dini (2007). "Flexible job-shop scheduling with routing flexibility and separable setup times using ant colony optimisation method." Robotics and Computer-Integrated Manufacturing **23**(5): 503-516.

Saidi-Mehrabad, M. and P. Fattahi (2007). "Flexible job shop scheduling with tabu search algorithms." International Journal of Advanced Manufacturing Technology **32**(5-6): 563-570.

Scrich, C. R., V. A. Armentano and M. Laguna (2004). "Tardiness minimization in a flexible job shop: a tabu search approach." Journal of Intelligent Manufacturing **15**(1): 103-115.

Stafford Jr, E. F., F. T. Tseng and J. N. D. Gupta (2005). "Comparative evaluation of MILP flowshop models." Journal of the Operational Research Society **56**(1): 88-101.

Tamaki, H., T. Ono, H. Murao and S. Kitamura (2001). Modeling and genetic solution of a class of flexible job shop scheduling problems. 8th International Conference on Emerging

Technologies and Factory Automation (ETFA 2001), October 15, 2001 - October 18, 2001, Antibes-Juan les pins, France, Institute of Electrical and Electronics Engineers Inc.

Thomalla, C. S. (2001). "Job shop scheduling with alternative process plans." International Journal of Production Economics **74**(1-3): 125-134.

Vancza, J., T. Kis and A. Kovacs (2004). "Aggregation - The key to integrating production planning and scheduling." CIRP Annals - Manufacturing Technology **53**(1): 377-380.

Wagner, H.M (1959). "An integer linear-programming model for machine scheduling." Nav. Res. Logist. Quart. (6) 131-140.

Wang, X., L. Gao, C. Zhang and X. Shao (2010). "A multi-objective genetic algorithm based on immune and entropy principle for flexible job-shop scheduling problem." International Journal of Advanced Manufacturing Technology **51**(5-8): 757-767.

Wiendahl, H. P., H. A. ElMaraghy, P. Nyhuis, M. F. Zah, H. H. Wiendahl, N. Duffie and M. Brieke (2007). "Changeable Manufacturing - Classification, Design and Operation." CIRP Annals - Manufacturing Technology **56**(2): 783-809.

Xia, W. and Z. Wu (2005). "An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems." Computers and Industrial Engineering **48**(2): 409-425.

Yazdani, M., M. Amiri and M. Zandieh (2010). "Flexible job-shop scheduling with parallel variable neighborhood search algorithm." Expert Systems with Applications **37**(1): 678-687.

Zandieh, M., S. M. T. Fatemi Ghomi and S. M. Moattar Husseini (2006). "An immune algorithm approach to hybrid flow shops scheduling with sequence-dependent setup times." Applied Mathematics and Computation (New York) **180**(1): 111-127.

Zhang, G., L. Gao and Y. Shi (2011). "An effective genetic algorithm for the flexible job-shop scheduling problem." Expert Systems with Applications **38**(4): 3563-3573.

Zhang, G., X. Shao, P. Li and L. Gao (2009). "An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem." Computers and Industrial Engineering **56**(4): 1309-1318.

# Vita Auctoris

## VAHID ROSHANAEI

| | |
|---|---|
| **RESEARCH INTERETS** | Mathematical Modeling & Programming<br>Stochastic & Deterministic Solution Methodologies<br>Simulation & Optimization of Dynamic settings<br>Process Modeling & Performance Evaluation |
| **APPLICATION AREAS** | Manufacturing Systems, Supply Chain Networks, Stock Markets and Inventory Control |
| **EDUCATION** | **University of Windsor**, Windsor, Canada<br>M.A.S.c. II, Industrial & Manufacturing Systems Engineering (May 17th 2012)<br>GPA: 12 out of 13 (A)<br>Thesis title: *Mathematical Modeling & Optimization of Flexible Job Shops Scheduling*<br><br>**Amirkabir University of Technology (Tehran Polytechnic),**Tehran, Iran<br>M.A.S.c. I, Industrial Engineering (Systems Management& Productivity)-Nov 2008<br>GPA: 18.11 out of 20 (A)<br>Thesis title: *Open Shop Scheduling Optimization Using Advanced AI techniques*<br><br>**(Azad) University of Tehran**, Tehran, Iran<br>B.A. Industrial Management -Sep 2004<br>GPA:15.73 out 20 (B+)<br>Capstone Project title: *Aggregate Production Planning* (A) |
| **MAJOR ACADEMIC ACHIEVEMENTS** | **International awards:**<br>Granted Excellency Award From University of Windsor<br>$4380, 2 years, Canada, 2010-2012<br>Granted International Scholarship From University of Windsor<br>$6000, 2 years, Canada, 2010-2012<br>**National Awards:**<br>Recognized With National Exceptional Talent Award From Ministry of Science, Research & Technology, Iran, 2007<br>Ranked 10th among 6000 participants in national graduate matriculation exam for Industrial Engineering Major, 2006, Iran |
| **TEACHING EXPERIENCE** | **Teaching Assistant**<br>**University of Windsor,** Windsor, Canada<br>• IMSE 522: Supply Chain Management & Logistics |

Winter 2012 (Graduate Course)
- IMSE 312: Operations Research (I)
  Fall 2011 (Undergraduate Course)
- IMSE 413: Production Analysis & Logistics
  Fall 2011 (Undergraduate Course)
- IMSE 222: Experimental Data Analysis
  Winter 2011 (Undergraduate Course)
- IMSE 413: Production Analysis & Logistics
  Fall 2010 (Undergraduate Course)

**Amirkabir University of Technology**, Tehran, Iran
- IE 2582823: Production Control (I)
  Fall 2008 (Graduate Course)
- IE 2591473: Multivariate Analysis
  Fall 2007 (Graduate Course)

**(Azad) University of Tehran**, Tehran, Iran
- IM 2752: Work Measurement & Time Study
  Fall 2002 (Undergraduate Course)

**WORKING EXPERIENCE**

**Bank Mellat,** Tehran, Iran
**Department**: Information Technology, Sep 2006- Sep 2009
**Position**: Senior Project Manager

**Department**: Processes Optimization & Re-Engineering, Sep 2003- Sep 2006
**Position**: Systems Engineering Specialist

**Department**: Credit Allocation & Economic Analyses, Sep 2000- Sep 2003
**Position**: Business Process Analyst

**MAJOR GRADUATE COURSES**

Engineering Design Methodologies ($A^+$), Supply Chain Management (A), Manufacturing Systems Simulation (A), Leadership and Organization ($A^+$), Production Control ($A^+$), Manufacturing Systems Paradigm ($A^-$), Strategic Management ($A^+$), Quality Assurance (A), Multivariate Analysis (A), Advanced Engineering Economics ($A^+$), Financial Engineering ($A^-$)

**SOFTWARE EXPERTISE**

C++, CPLEX 12.1, GAMS, LINGO

**PROFESSIONAL CERTIFICATES**

Project Management Professional (PMP), Hamilton, Canada, 2010

**PROFESSIONAL MEMBERSHIP**

Project Management Institute (PMI)
Institute for Operations Research & Management Science (INFORMS)
Institute of Industrial Engineering (IIE)