PARALLEL-MACHINE SCHEDULING WITH LOAD-BALANCING AND SEQUENCE-DEPENDENT SETUPS

A Thesis by

Karthikeyan Senniappan

Bachelor's of Mechanical Engineering, Bharathiyar University, 2001

Submitted to the Department of Industrial and Manufacturing Engineering
and the faculty of the Graduate School of
Wichita State University
in partial fulfilment of
the requirements for the degree of
Master of Science

December 2006

PARALLEL-MACHINE SCHEDULING WITH LOAD-BALANCING AND SEQUENCE-DEPENDENT SETUPS

I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science with a major in Industrial and Manufacturing Engineering.

_____
Mehmet Bayram Yildirim, Committee Chair

We have read this thesis and recommend its acceptance:

_____
Krishna K. Krishnan, Committee member

_____
Coskun Cetinkaya, Committee member

ACKNOWLEDGMENTS

I am immensely happy for having a wonderful graduate tenure at Wichita State University. I am greatly indebted and thankful to my advisor Dr. Bayram Yildirim for giving me the opportunity to explore my talents within and for being supportive. His immense encouragement and ideas contributed greatly to the successful completion of my thesis. I will be grateful in my entire life for the personal care he gave me.

Two other professors Dr. Krishna K. Krishnan and Dr. Coskun Cetinkaya provided valuable suggestions that improved the subject of my thesis to a great extent.

My parents were the true drivers of my courage and enthusiasm. I thank them. I also thank my graduate colleagues, friends, and roommates for having supported me during my difficult times. Last, but not least, the almighty was responsible for my success and courage, and I am deeply beholden

ABSTRACT

In many practical manufacturing environments, setups consume a significant amount of industrial resources. Therefore, reducing setups in a non-identical parallel machine environment will significantly enhance a company's performance level. In this thesis, the problem of minimizing total completion time with load balancing and sequence-dependent setups in a non-identical parallel machine environment was studied. A mathematical model for minimizing total completion time with a workload-balancing constraint is presented. Since this problem is an NP-hard problem, some simple heuristics and a genetic algorithm were developed for efficient scheduling of resources. Both were tested on random data.

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Overview

Scheduling, a tool to optimize use of any type resources such as machines, workers, etc., consists of systematic planning and prioritizing tasks in order to meet certain requirements, constraints and objectives. In today's competitive manufacturing world, scheduling industrial resources like machines, manpower and facilities are considered as critical to increase efficiency and capacity utilization of resources [47]. One of the problems that is considered in the scheduling is to schedule jobs, machines and manpower in order to minimize make span [27] and to balance the workload [38]. Minimizing make span can be defined as minimizing the total completion time (including setups) of all the jobs and workload balancing can be defined as distributing the available workload among the available machines equally [38]. By efficiently balancing the workload in a manufacturing system, bottlenecks can be eliminated, throughput is maximized, work in process, finished goods inventory and operating costs are lowered [38].

The motivation of this thesis comes from situations in both service and manufacturing industries: For example, In an intensive care unit, nurses should have a balanced workload in order to provide a quality care and possibly allocate additional workload. In this case, setups might be related to hygienic requirements for different patients. If a nurse visits her patients in a particular order, then there can be significant savings in setups. Also, consider two workers in a machine shop. The first worker can operate a drill, but the more experienced worker can operate a CNC machine at the same time. The goal of the production manager is to assign jobs to each worker in such a way that their workloads are close, i.e., both workers scheduling the drilling operation can be done by any of the workers, if a job that needs to be processed on the CNC machine arrives, the second worker has to be assigned. The order of jobs on CNC machine and drill might affect the amount of setup required to complete the incoming orders.

Another example is a rolled aluminium sheet metal manufacturing process whereby ingots or slabs of pure aluminum are melted in a furnace. Then, additive elements such as magnesium, vanadium, etc., which determine the alloy of aluminum, are added to the furnace in specific amounts.

The melted metal in the furnace flows through a shaper and a rolling mill. The entire process takes place on casting lines. In this type of process, continuous production is required to avoid reheating a furnace after it cools down, which is long and costly. After the casting operation, to obtain the desired width and thickness, the metal is fed in the form of a coil through a series of cold rolling mills, which successively reduces the metal thickness and recoils it after each rolling pass. The type of additives and width and thickness of the rolled aluminium sheets determine the characteristics of an order (i.e., job). Each casting line is capable of processing aluminum sheets up to a certain width but the lines that can process wider sheets can also process narrow sheets. Thus, each line has a certain capability. The objective in casting-line scheduling is to minimize the total sequence-dependent setup incurred between orders and balance the workload between parallel casting lines to accommodate potential orders.

In this thesis, the goal is to formulate a mathematical model to solve the load-balancing problem in a parallel machine environment with sequence-dependent setups and to minimize the sum of total completion time objective. Several heuristics, including a Genetic Algorithm(GA) are proposed to solve this problem.

This chapter is organized as follows: Section 1.2 provides a detailed overview of scheduling. Section 1.3 discusses in detail about scheduling with sequence-dependent setup time and also gives a detailed description of the problem taken into consideration. Section 1.4 identifies the research objectives. Section 1.5 gives the organization of the thesis and section 1.6 provides a summary.

1.2 Background in Scheduling

A scheduling problem can be represented by a triplet $\alpha/\beta/\gamma$. The α field contains a single entry and explains the machine environment. The β field contains processing characteristics and constraints. The γ field contains the objectives to be minimized. Let n and m be the number of jobs and machines, respectively. Let j and i be a job and machine, respectively. Let (i, j) be the operation of job j on machine i. The following definitions are provided before giving details on the $\alpha/\beta/\gamma$ notation:

Processing time (P_{ij}): Amount of time a task of job j needs to be completed on machine i. The subscript i will be omitted in two cases: (a) If the processing time of job j is independent of machine i, and (b) If only one machine can process the job j.

Release date (r_j): Also known as ready date, or the earliest time at which a job can be processed.

Due date (d_j): The completion time of a job.

Weight (w_j): The priority factor with respect to other jobs in the system. For instance, it represents the inventory or holding cost of retaining a job in the system.

Some of the possible machine environments (i.e., the α field) are as follows:

Single Machine (1): The simplest of all machine environments and also a special case of all other complicated systems.

Identical Machines in Parallel (P_m): A number of identical m machines available in parallel and jobs can be processed in any one of them. In some cases, jobs may have dependency (a succeeding job in the sequence may not be processed until the preceding job finishes its operation). If the notation M_j is in beta field, then it denotes that job j can be processed only on the specific subset called M_j.

Machine in Parallel with Different Speeds (Q_m): Where there are m machines in parallel with different speeds. The speed of the machine is represented by v_i. The time p_{ij} that job j takes on machine i will be equal to $\frac{p_j}{v_i}$ (if job j has been processed on machine i). This kind of environment is known as *uniform machines*. If all machines have the identical speed, then v_i is equal to one.

Non-Identical Machines in Parallel (R_m): A generalization of the previous case involving m different machines in parallel. Machine i can process job j at speed v_{ij}. Then the time p_{ij} that job j takes on machine i will be $\frac{p_j}{v_{ij}}$ (if job j has been processed on machine i). If the speed of the machine is not dependent on job j then v_{ij} is equal to v_i for all i and j, and the environment is the same as the previous case. In this thesis, a problem that is a special case of R_m is analyzed. This problem is a *semi-related* machine case, which we will be described in detail in Chapter 3, section 3.2.

The β field may include multiple entries that consist of restrictions and constraints. The β field in this thesis represents load-balancing constraints and sequence-dependent setups. Some of the related parameters in the β field are as follows:

sequence-Dependent Setup Times (S_{jk})**:** The setup time between jobs j and k. If the value of j is zero, then the variable becomes S_{0k}, which denotes that job k is first in the sequence and there is no setup time, and if the value of k is zero, then job j is last in the sequence. If the setup time depends on the machine, then the subscript i is included in the notation, i.e., S_{ijk} . No S_{jk} in the β field indicates that the setup time is zero or sequence-independent.

Preemptions (*prmp*): Indicates that a job can be interrupted at any time during its processing in a machine. The operator can schedule another job based on priorities. The preempted job can be scheduled on that machine or any other machine (if it is a parallel machine) at any point in time. After the job has been scheduled, it takes only the remaining time required to complete its operation on that particular machine. The word *prmp* will be added in the β field based on its requirements. In this thesis, preemption is not allowed.

Precedence Constraints (*prec*)**:** Is applicable to both single-and parallel-machine environments. It refers to one or more jobs required to be completed before a particular job starts its processing. There are many kinds of precedence constraints. If each job has a maximum of one predecessor and successor, then it is called *chains*. If each job has the maximum of one successor, then it is called *intree*. If each job has the maximum of one predecessor, then it is called *outtree*. In this research work, there is no precedence constraint in the β field.

Machine eligibility restrictions (M_j)**:** Indicated in the β field, there are m machines in parallel. M_j indicates that only a certain number of the m machines are capable of processing job j. If there is no M_j in the β field, then the job can be processed on any of the available machines. This research work involves machine eligibility restrictions in the β field.

The γ field contains the objective functions to be minimized. Some of the related parameters in the γ field are as follows:

Make span(C_{max})**:** The time taken to complete all jobs in a system. Usually, a minimum make span indicates a high utilization of machines.

Total weighted completion time $\sum(w_jC_j)$**:** A measure by which the sum of holding or inventory costs can be calculated. The sum of completion time and total weighted completion time are often referred to as $flow\ time$ and $weighted\ flow\ time$, respectively.

This thesis concentrates on the non-identical parallel machine environment with sequence-dependent setups, and the nature of the problem is completely different from the existing ones.

1.3 Problem Statement

The goal of workload balancing is to distribute jobs/tasks to resources in such a way that relative imbalance is minimized, i.e., utilization of resources is approximately equal. Rajakumar et al. [38] tested the effectiveness of Random (RN), Shortest Processing Time (SPT) and Longest Processing Time (LPT) dispatching rules on workload balancing in a parallel machine setting. They showed that the longest processing time rule shows better performance for the higher number of jobs and/or machines. Also, Becker et al.[3] provided a survey of balancing on assembly lines. Another application can be found in workload balancing in printed circuit board assembly [15].

In parallel machine scheduling problems, the machines are either identical (related) [7, 20, 21, 32, 38] or non-identical (unrelated) [2, 39, 51, 55, 37]. In this setting, the machines (resources) are non-identical but related. As in the case of aluminium casting lines, some resources can process all jobs while others are less capable.

1.4 Research Objectives

In this thesis, the goal is to find the following:

- Minimum total completion time with a load-balancing constraint.
- Optimal allocation of jobs to machines.
- Percentage of imbalance with respect to average workload.

1.5 Organization of Thesis

This thesis is organized as follows: Chapter 2 gives a detailed overview of scheduling literatures involving setups. It discusses various solution methodologies like Linear Programming, Exact Algorithm, Heuristics, Single machine and Parallel machine's Genetic Algorithm. Chapter 3 presents the problem description, the mathematical model and the proposed methodology. Chapter 4 explains the experimental design and the case study to validate the proposed methodology to solve the problem. Finally, Chapter 5 provides conclusions and future research directions.

1.6 Summary

This chapter provides an overview of basic concepts and the importance of scheduling in manufacturing industries. It explains various kinds of machine shops in practice. In addition, it provides an introduction to the problem. Finally, this chapter concludes with the research objectives and organization of the thesis.

CHAPTER 2
LITERATURE REVIEW

2.1 Introduction

The goal of this chapter is to provide information about single and parallel machine scheduling environments, scheduling with setups and load-balancing, a comprehensive review of the literatures related to solution methodologies like Mathematical modeling, Exact algorithms, Heuristics and Genetic Algorithm (GA).

The organization of this chapter is as follows: Section 2.2 provides an overview of single and parallel machine scheduling. Section 2.3 discusses about various literatures in scheduling with setups. Section 2.4, provides definition of load-balancing and the review literatures related to it. Section 2.5 provides a comprehensive overview of literatures dealing with various solution methodologies like Mathematical Modeling, Exact Algorithms, Heuristics and Genetic Algorithms (GA). Finally, the chapter concludes with a summary.

2.2 Relevant Works in Machine Scheduling

2.2.1 Single Machine Scheduling

In a single-machine problem, jobs are required to be processed. Each job might have its own processing and setup times and each job must be sequenced in such a way as to optimize performance measures. Some of the objectives that can be considered in a single-objective environment are minimizing total completion time, tardiness and load imbalance. In order to determine the best sequence, a number of sequences with different job combinations along with the objective function value can be developed then the best one is selected. This is defined as exhaustive enumeration. The maximum number of sequences that can be explored is $N!$ (N being the number of jobs) combinations. The problem cannot be solved efficiently when the size of the problem increases and no algorithm exists to solve this problem in polynomial time. Because this is an NP-hard problem, the exhaustive enumeration technique is restricted especially when N is large [47].

2.2.2 Parallel Machine Scheduling

A number of machines are available in parallel and the unscheduled jobs can be assigned to the machines in such a way as to optimize some performance measures. The three kinds of parallel machine environments as explained in Chapter 1 section 1.2, are as follows.

- Identical machines in parallel (P_m)
- Machines in parallel with different speeds (Q_m)
- Non-identical machines in parallel (R_m)

Even in parallel machine environments, the single-machine model plays an important role. In most parallel machine environments, bottlenecks are determined by only one machine. Hence, efficient scheduling strategies are necessary to reduce the bottleneck line. Moreover, this model also has its application in decomposition models. If the machine environment is complex, then it is possible to find solutions by reducing it to a number of single-machine environments [36].

The nature of the problem in this thesis is unique from existing cases. There are different kinds of machines - some can perform all jobs and the rest can perform only certain jobs. In addition, there are sequence-dependent setup times between jobs. Minimizing the sequence-dependent setup time plays a vital role in this thesis. The objective is to minimize the total processing time on all machines with load-balancing constraints. Therefore, an efficient strategy must be developed to solve this problem. For smaller problems (i.e., less than 10 jobs), an exhaustive enumeration technique or a branch-bound technique may be a good approach. But larger size problems may not be solved within the polynomial time [47], due to greater number of variables and calculations involved.

2.3 Scheduling with Setups

Setups during production involve work and time to prepare the machines, processing, obtaining and returning tools, cleaning up the tools and workplaces, etc. The setup time for these activities is often considered negligible and hence ignored. However, in some cases, setups must be considered separately from the processing time [25].

There are two kinds of setups:

1. Sequence independent setups.

2. sequence-dependent setups.

In the first case, the setup time of a succeeding job does not depend on the setup time of the preceding job, while in the second case, the setup time of a succeeding job depends mainly on the preceding job. Scheduling with sequence-dependent setups has received the attention of many researchers in the past [19]. Scheduling jobs with sequence-dependent jobs is a well-studied problem. In their survey paper, Allahverdi et al. [1] present state of the art machine scheduling problems with setups. Moreover, they classified the literatures according to the shop environments like single machine, parallel machines, flow shops and job shops and also have discussed in detail the literatures involving setups in identical and non-identical machine environments for the objective of minimizing make span with the conventional methodologies. They classified the problem with setups as shown in figure 2.1.

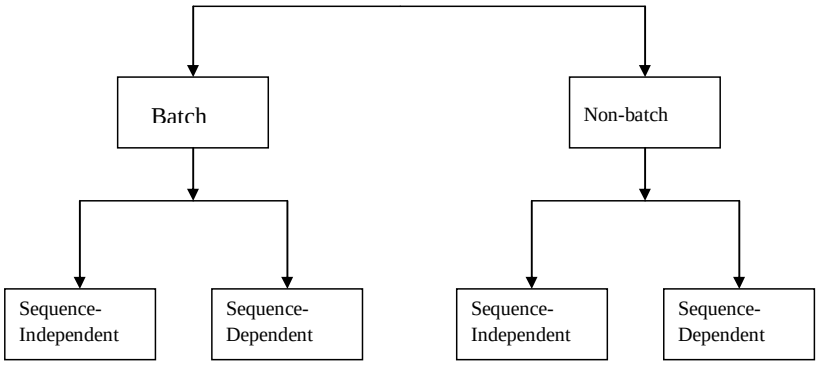

Figure 2.1: Scheduling problems with setups.

Due to the nature of complexity in terms of formulation and computational time, Stowers and Palekar [46] simplified the problem by considering the processing time and the setup time separately. Some researchers combined the setup time with processing time [25]. As an initial step in solving problems with sequence-dependent setups, single machine problems are solved by various methods: dynamic programming [8], branch-and-bound [6] and heuristics [36]. Dietrich and Escudero [6] formulated a 0-1 integer programming problem for non-identical parallel machines with a number of job types involving sequence-dependent setups in a single-stage manufacturing environment. They adopted the branch-and-bound method to solve the integer programming. The main drawback of

the branch-and-bound method and complete enumeration/exact solution methods is that when the problem size becomes larger, the number of variables also become larger, thereby increasing the complexity in formulation and computational time. Hence, they are applicable only for small-sized problems. Similarly, Hu and Caramanis [17] devised an optimal feedback for scheduling setups in a single machine environment with three part types. Srivatsan and Greshwin [44] extended the idea of a single machine to a multiple machines environment in determining setup rates. Ovacik and Uzsoy [34] proposed Rolling horizon heuristics which decomposes the scheduling problem into many sub-problems and then optimally solves by a branch-and-bound algorithm. They decomposed the identical parallel machine problem into a number of sub-problems and used Ovacik and Uzsoy's [33] method to solve the resulting single machine sub-problems. Zhou and Egbelu [57] developed a heuristic for a single machine problem with multi-stage, multi-types and sequence-dependent setups. Cao and Bedworth [4] solved a multi-item serial machine problem with sequence-dependent setups by johnson's rule-based heuristics. The genetic algorithms which were first developed by Holland [16] have also been applied in scheduling parallel machines with setups. For example, Van Hop and Nagarur [52] applied genetic algorithms to group similar boards, balance machines, and reduce setups in printed circuit board production.

Methodologies like simulated annealing, tabu search, and neural network have also been applied to find solutions to sequence- dependent setup problems. Even though they have been proved effective, all methods have their own limitations. To converge to the optimal solution in simulated annealing requires exponential time. Limited theoretical knowledge is available in designing search process for a tabu search. Although neural network give a solution by massively parallel computation, it requires extensive training and time.

2.4 load-balancing

Load-balancing can be defined as distributing the available workload among the available machines as equally as possible and in such a way that the workload imbalance performance in the system can be minimized [38].

Rajakumar et al. [38] solved the identical parallel machines without setups for the objective of load-balancing. They proposed three heuristics: Shortest Processing Time, Longest Processing

Time and Random. Finally, they concluded that the Longest Processing Time was the best strategy. Yu et al. [55] solved an unrelated parallel machines scheduling problem in a Printed Wiring Board (PWB) manufacturing line for a set of performance measures that included load-balancing on machines. They used Integer Programming, Lagrangian Relaxation, and heuristics to solve the problem. They reported that the Lagrangian Relaxation technique found the best solution for the objective of balancing the workload. Sun et al. [48] proposed a genetic algorithm for optimizing two objectives - minimizing the number of pick ups and balancing the workload between two gantries in a Printed Circuit Board (PCB) manufacturing industry. They designed a gene encoding method for interference between feeders of varying widths and used greedy heuristics for the work-cycle formation and pick-up- sequencing decisions. Setup time reduction and load-balancing in a Group Technology (GT) flow-line work cell was researched by Gung and Steudel [12]. They developed a heuristic model and then combined it with a load-balancing model in order to find the setup reduction scheme for various levels of batch size reduction. Their model was based on the Theory of Constraints to balance the workload at each station. Also, Tiwari and Vidyarthi [50] proposed GA-based heuristics to solve the Flexible Manufacturing System (FMS) problems. Their main objective was to minimize system imbalance while satisfying the technological constraints of machining time and tool slots. The proposed GA-based heuristics determined part-type sequence and part-machine allocation to obtain the optimal solution. In addition, researchers like Modi and Shanker [30], Stecke [45], Sawik [42] and Lee and Ueng [22] developed mathematical models to minimize the system imbalance. This is discussed in detail in the next section.

2.5 Solution Methodologies

2.5.1 Mathematical Modeling

In the last ten years, many researchers have formulated linear, non-linear and integer programming for minimizing the total completion time, transportation time and part-movement minimization together with balancing the workload among workstations. Many of the available literatures have focused on these kinds of problems in Flexible Manufacturing Systems (FMS).

Loading problems in (FMS) can be defined as the allocation of machines and tools to produce different parts [30]. Modi and Shanker [30] cited that many researchers have focused on contents

and scope of loading problems, their inter-relationship and the hierarchies of various decisions in FMS. For example, Stecke [45] addressed several loading criteria like balancing the processing time, workload per machine, and minimizing the part movements from machine to machine. Shanker et al. [43] also addressed similar kinds of problems. As an extension of these problems, Modi and Shanker [30] solved the loading problem with the objective of part-movement minimization and balancing the workload among machines. They used the branch-and-bound procedure and 0-1 integer programming methods.

Linear Programming (LP) technique was applied to a Flexible Assembly Line (FAL). As an initial step in minimizing both the part movement and workload imbalance, Modi and Shanker [30] solved a single objective with an additional constraint of allowable workload on each machine. They formulated a quadratic programming problem and the quadratic terms were linearized by Balas, Glover and Woolsey methods. They also used hyper LINDO to solve the 0-1 linear programming. Having this as a base, Sawik [42] devised a bi-criterion integer programming problem with two objectives of minimizing total transportation time and balancing workload among stations. Lee and Ueng [22] developed a Linear Programming model for their vehicle routing problem and discovered that it performed well only for smaller-sized problems. For larger-sized problems, they developed heuristics, which performed better than LP.

2.5.2 Exact Algorithms

Chen and Powell [5] proposed a column generation-based branch-and-bound exact solution algorithm for a set of identical machines with sequence-dependent setups. Also, Rios and Bard [40] developed a branch-and-bound algorithm for a permutation flow shop with sequence-dependent setups to minimize completion time. Lee et al. [23] designed a branch-and-bound algorithm for a bi-criterion single machine scheduling problem with learning consideration. Their objective was to minimize the linear combination of total completion time and maximum tardiness.

2.5.3 Heuristics

Due to the limitations involved in exhaustive enumeration, branch- and-bound and mathematical modeling for solving large sized problems, researchers started developing heuristics. Even though

these heuristics did not guarantee optimal solutions, they gave feasible solutions within a reasonable computational time [47], [42], [51]. Pinedo [36] classified the parallel machines scheduling in the increasing order of difficulty as follows:

1. Identical machines
2. Uniform machines
3. Unrelated machines

Yu et al. [55] stated that unrelated parallel machine scheduling problems are the most difficult to solve with respect to complexity in hierarchy of deterministic scheduling. Monma and Potts [32] stated that "Models that consider setup times are largely ignored in the scheduling literature." Monma and Potts [31] studied the computational complexity of several algorithms with batch setups. As an extension of their earlier work, Monma and Potts [32] proposed several heuristics for identical parallel machine scheduling with batch setups, which are capable of generating approximate solutions within polynomial time. Also, Suresh and Chaudhuri [49] solved unrelated parallel machine scheduling by a local search method.

Randhawa and Kuo [39] evaluated various scheduling heuristics for non-identical parallel processors for the objective of minimizing mean flow time and tardiness. They also exploited the use of hybrid heuristics. Lee and Pinedo [21] devised a three stage-heuristics for identical parallel machines with sequence-dependent setups. Their objective was to minimize the sum of total weighted tardiness. In the first phase, a pre-processing procedure was developed. The second and final phase dealt with dispatching rules and simulated annealing method, respectively. Scheduling non-identical parallel CNC machines was focused on by Turkcan et al. [51]. They proposed a two-stage algorithm for two objectives, such as minimizing the manufacturing cost and total weighted tardiness. The generated algorithm gave different solutions as per the relative importance of objectives. Gunther et al. [13] proposed a heuristic solution procedure for PCB manufacturing for minimizing the number of changeovers in an assembly line.

Aranaout and Rabadi [2] solved the problem of batch scheduling in an unrelated parallel machine scheduling with sequence-dependent setups for the objective of minimizing weighted mean completion time. They grouped similar jobs together to be available at time zero. The processing

time of a particular job depends on the machine in which it has to be assigned and also depends on the batch. The uniqueness of their problem is that their processing times and setups times were stochastic. They classified the problem as NP-hard problems therefore, they developed heuristics such as weighted shortest processing time first, Weng's algorithm, Pick Minimum Weighted Processing (PMWP) Time. In addition to this, they conducted several experimental runs and compared the performance of their heuristics with the existing ones. Finally they concluded that their proposed PMWP algorithm outperformed the other algorithms.

Rabadi et al. [37] solved the problem of unrelated parallel machine scheduling for the objective of minimizing make span. The processing time and setups were deterministic in nature. The setups were machine-dependent and also sequence-dependent. They also considered that the jobs were non-preemptive and all jobs were available at time zero. They classified the problem as an NP-hard problem. Hence, they found the optimal solution for small-sized problems only, and for large-sized problems, they proposed a new meta-heuristics called Meta-Raps. They reported that their Meta-Raps heuristics found all optimal solutions for small problems and obtained better results for large problems than the existing heuristics.

Kurz and Askin [20] concentrated on identical parallel machine scheduling with sequence-dependent setups for the goal of minimizing make span. The jobs were characterized by sequence-dependent setups and possibly by non-zero ready times. They developed an integer programming model and also several heuristics such as the Slicing heuristic, Multiple multi-fit heuristic, Multiple insertion heuristic and Genetic Algorithm. In addition to this, they conducted several experiments to validate their methodology. Finally, they reported that the modified insertion heuristic performed better.

Even though some authors have focused on minimizing make span in non-identical parallel machines, they did not consider the load-balancing factor. Moreover, in the existing literatures the term "non-identical" refers to machines that are capable of performing all jobs with different processing times. The nature of the problem being dealt with in this research is entirely different from the existing ones.

2.5.4 Genetic Algorithm (GA)

Heuristics are capable of producing only feasible solutions but not optimal or close-to-optimal solutions. Hence, researchers have begun using the latest methodologies such as Genetic Algorithm, Neural Network, Simulated Annealing, Tabu Search and Target analysis to obtain solutions close to optimal. Glover and Greenberg [9] reviewed these five methods and concluded that their results are promising in most cases. This research makes use of a GA to solve the problem.

The development of GA dates back to 1960 by Holland and colleagues at the University of Michigan [10]. It is considered one of the most dynamic and efficient tools used in the study of Artificial Intelligence (AI). GA is based on the concept of natural selection and the survival of fittest proposed by Charles Darwin. In GA, initial and feasible solutions are generated in the form of chromosomes. A chromosome represents the nature of the solution being generated. The initial solutions are termed the initial population (search space) and the GA explores the best solution from the search space. The objective function of the problem can be called the "Fitness Value" of a particular chromosome. The quality of the chromosomes is evaluated by using the fitness value and only those chromosomes having better fitness values than the previous generation are allowed to survive in the next generation. The new generations (offsprings) are created by reproduction operations called crossover and mutation. A GA possess special characters that are unique from the other search tools. Since the GA is not a rigid procedure, it can be modified and designed according to the individual problem without the loss of generality. Not only GA stops with only one solution (as like other tools) but also has the capability of exploring different set of solutions within a reasonable computational time. The following are the some of differences between the GA and other methods:

1. A GA explores a different set of solutions in the form of chromosomes.

2. A GA explores the search space to the required extent in finding a wide variety of solutions.

3. For generating offsprings, a GA uses the fitness value in order to determine survivals.

4. A GA applies probabilistic transition rules.

Holland states that the GA uses a stochastic global search method in which the principle of biological evolution has been embedded [10]. Basically, the GA creates and works on a set of initial

populations. The theory of "survival of fittest" has been applied to the initial population to generate increasingly better offsprings which in turn create the near-optimal solutions. The better offsprings are created by means of reproduction operations called crossover and mutation. The new offsprings survive based on their fitness values (Objective functions). The five basic components of a GA are as follows:

1. The chromosome that represent the solution of a problem.
2. Initial population.
3. Fitness value.
4. Reproduction operators such as crossover and mutation.
5. Crossover and mutation probabilities.

Genetic Algorithm Terminology

- **Gene:** The constituents in the chromosome.
- **Chromosome:** A collection of genes.
- **Population:** The compilation of different generations.
- **Alleles:** The different combinations of a gene.
- **Locus:** The location of the gene in the chromosome.
- **Genome:** The entire set of chromosomes.
- **Genotype:** A particular set of chromosomes in the genome.
- **Phenotype:** Another name for solution space, validation for survival takes place.
- **Reproduction:** The process by which two parents join together to form a child (offspring):helps to create newer generations with better fitness values.
- **Crossover:** One of the reproduction operators that decides the locus where two parents can cross each other. Usually involves a very high probability (0.95) to occur. Results in

hybridization of chromosomes to produce better chromosomes, thereby increasing the variety of solutions.

- **Mutation:** One of the reproduction operators that carried after the crossover operation by exchanging the locus in a chromosome. Normally, involves a very low probability (0.05) to occur. Most commonly caused by random exchange, inverting and swapping. Prevents the solution from being struck at local optimum.

- **Elitism:** Copying the best chromosomes of one generation to the subsequent generations without any modifications.

General Procedure of a GA

According to Goldberg [10], the following are steps involved in a GA:

- **Coding:** Refers to the representation of a chromosome (or string) particular to a problem. Generally, for scheduling problems it represents the sequence of jobs or machines to be scheduled.

- **Generation of chromosome:** A large number of randomly generated different sequences (chromosomes) or codings, each having a calculated fitness value.

- **Generation of initial population:** Out of the chromosomes created select the n best chromosomes based on the fitness value and it represents the initial population. The value of n is the population size and left to the decision of researcher.

- **Roulette wheel selection:** The process, repeated n times, by which the parents are selected for reproduction whereby spinning a roulette wheel, a random number is generated and the chromosome corresponding to the generated random number is selected for reproduction.

- **Crossover:** Follows the selection of chromosomes for reproduction and is considered one of the reproduction operators whereby a portion of a chromosome is replaced with a portion of another chromosome, resulting in a new chromosome, or child or offspring, which may or may not posses better characteristics than the parent due to hybridization of the two chromosomes. The fitness value for child is calculated and compared to the parent; if it is better than the

parent, then it will be considered for survival in the next generation. Usually, the probability of crossover to happen is set as 0.95. Methods to perform the crossover include (a) Singlepoint crossover (b) Two-point crossover and (c) Uniform crossover.

Single-point crossover: The generation of a random number between 0 and l-1 (l is the length of chromosome) to determine the locus of the chromosome for the crossover operation, whereby the genes in the first parent (chromosome) until the point of locus will be retained and the remaining portion will be created from the second parent. Resulting in a new chromosome called a child or offspring and is repeated until the required number of offsprings are created.

Two-point crossover: The generation of two random numbers between 0 and l-1 (l is the length of the chromosome), whereby the genes of the first chromosome will be retained until the point of the first random number and then the genes after the second random number in the second chromosome are selected, leaving the remaining portion of the first chromosome to be copied from the second parent to produce a new offspring.

Uniform Crossover: The direct exchange of genes from one point to another point after the locus.

- **Mutation:** The process of swapping or interchanging genes in a chromosome is performed after the crossover. but not considered as important as crossover and helps to maintain diversity. The probability of occurrence is 0.05. A number change will lead to a remarkable change in search direction [24].

- **Elitism:** The process of creating a new generation with better chromosomes by replacing chromosomes in the old generation. Rudolph [1994] proved that this process will be of great help to converge to the optimum.

- **Termination:** The final step in the GA that decides the number of iterations (number of generations) to be carried out before getting the solution close to optimal.

Shortpseudo Code for a Genetic Algorithm

1. Generate an initial population with size n.

2. Perform crossover and mutation operations to generate offsprings.

3. Evaluate the fitness value of generated offsprings and give only those offspring that have better fitness values a chance to survive in the next generation.

4. Repeat steps 2 and 3 until the required number of generations is obtained.

5. Finally, select the best chromosome.

Single-Machine Genetic Algorithm

Miller et al. [29] proposed a GA for single machine, single-stage scheduling problem with sequence-dependent setups in a fixed planning horizon. Their objective was to minimize the sum of setup cost, inventory cost and backlog cost. Their algorithm performed up to 50 percent better than the Just-In-Time heuristic and 30 percent better than the complete batching heuristic. Maimon and Braha [28] developed a GA for scheduling a Printed Circuit Board (PCB) on a single machine. The developed GA minimized the total number of component changes (setups) for better sequencing and was extended to the applications of Group Technology (GT). Wang et al. [53] formulated a GA to minimize the flow time of grouped jobs on a single machine. Their research showed that their algorithm was also applicable to schedule a large number of jobs in a practical production scheduling. Similarly, Liu and Tang [26] proposed a modified GA to schedule jobs on a single machine. They modified the GA with two new steps and proved that their algorithm performed better than both the simple GA and special-purpose heuristics. Rubin and Ragatz [41] also proposed a GA for a single machine with n jobs associated with sequence-dependent setups with the objective of minimizing total tardiness. They also compared their work with a branch-and-bound algorithm.

Parallel Machine Genetic Algorithm

The methodologies for single-machine and parallel-machine scheduling are different. Single-machine scheduling is interested only in the sequence of jobs to be scheduled, but in parallel machine scheduling, jobs must be assigned to machines and the scheduled jobs must be sequenced for each and every machine. Moreover, heuristics solutions used in a single-machine environment are not feasible in a parallel-machine environment [7]. According to Fowler et al. [7], initially GAs were applied to solve optimization problems and were not applied to scheduling problems until the last decade. They developed a hybridized GA to solve identical parallel-machine scheduling

with sequence-dependent setups with the objective of minimizing total completion time and total weighted tardiness. Departing from traditional approaches in which a GA finds the solution directly, they used the GA to assign jobs to machines and then employed dispatching rules to schedule individual machines. Finally, their results concluded that their hybridized genetic algorithm performed better than the Setup Avoidance and Apparent Tardiness Cost Setup strategies.

Herrmann et al. [14] proposed a two-space genetic algorithm to solve minimax optimization problems. They basically used a robust discrete optimization technique to deal with uncertainty in the decision-making process and solved parallel-machine scheduling problems with uncertain processing times for the objective of minimizing worst-case make span.

For Flexible Manufacturing Shops with flexible routing and flexible machines, Zhao and Wu [56] proposed a Genetic Algorithm to solve the job-sequencing problem. They introduced new concepts like virtual and real operations. The main advantage of their GA approach was that they relaxed certain routing constraints of some of the parts to be produced, thereby introducing flexibility to machine selection, routing selection and processing-sequence selection.

Jahangirian and Conroy [18] addressed the dynamic nature of scheduling problems with the help of a GA. Their framework basically consisted of a two modules: (a) Intelligent simulation and (b) incremental learning. They used a GA for their learning module and also proposed crossover and three kinds of mutations to obtain the best solution.

2.6 Summary

This chapter explains in detail about single-machine and parallel-machine scheduling with setups. It reviews the literatures related to scheduling with setups and load-balancing. Next, it discuses, in detail, the literatures involving various solution methodologies like Mathematical modeling, Exact algorithms, Heuristics and GA. In the GA subsection, it provides a comprehensive review of terminology, procedures and applications to single and parallel machines.

CHAPTER 3
METHODOLOGY

3.1 Introduction

Reducing the total processing time in a manufacturing environment brings economic benefits. The sequence-dependent setup time can be an especially significant source of waste in a production-planning environment. Liu and Chang [27] reported that setups will consume 20 percent of existing machine capacity in many manufacturing environments if not taken care of properly. Hence reducing the total setup time by reducing the sequence-dependent setup time might increase the efficiency of a plant significantly. In addition, workload-balancing will help to reduce bottlenecks, maximize throughput, and minimize inventory and operating costs [38]. The problems of assigning nurses, CNC machine operators and jobs to casting lines in an aluminum processing plant (as explained in Chapter 1) and the findings of researchers like Liu and Chang [27], Rajakumar et al. [38] was the motivation behind developing an efficient methodology to reduce setups and balance the workload simultaneously. This chapter aims at giving a detailed description and mathematical model of the problem. It also discusses the complexity involved and the development of heuristics and a GA to solve the problem.

This chapter is organized as follows: In section 3.2 a detailed description of the research problem and the mathematical formulation are presented. Section 3.3 explains how the heuristics are used to solve semi-related parallel-machine problems. Section 3.4 explains the Genetic Algorithm used to solve SPM-LBSDS on semi-related parallel-machines. Section 3.5 provides a summary of the chapter.

3.2 Problem Description

Let K be the set of parallel machines and $|K|$ denote the cardinality of set K, i.e., the number of machines. Similarly, J is the set of jobs. K_j is the set of machines on which job j can be processed and J_k is the set of jobs that can be processed on machine k. Let p_{ik} be the processing time of job i on machine k. If job i precedes job j on machine k, then there is a sequence-dependent setup time of S_{ijk} between jobs i and j. The setups follow the triangle inequality. Let α be the maximum

allowable level of imbalance on the production line. Without loss of generality, it is assumed that if $k < k'$ then $J_k \subseteq J_{k'}$, i.e., jobs which can be processed on lower indexed machines can also be processed on higher indexed machines. Although the machines are non-identical, they are related. It can be assumed that they have different capabilities. We define this case as *semi-related machines*.

Jobs are available at time zero and each job has processing time and sequence-dependent setup time between consecutive jobs. No preemption is allowed between the jobs. Each job is processed on one machine and exactly once. The objective that we consider is minimizing the sum of the total completion times while balancing the load on all the non-identical parallel machines. The total completion time on machine k is the sum of processing time and sequence-dependent setup time.

Below, we present the mathematical program (SPM-LBSDS) to schedule jobs on semi-related parallel machines to minimize sum of completion times on all machines with load-balancing constraints and sequence dependent setups. The objective function minimizes the total completion time, C_{total}, (i.e., sum of processing time and setup time) needed to complete all of the available jobs on all of the processing lines

$$\min \quad C_{total} \tag{3.1}$$

where

$$C_{total} \;=\; \sum_{k \in K} C_k. \tag{3.2}$$

C_k is the total processing and setup time on line k. Mathematically,

$$C_k = \sum_{i=1}^{n} y_{ik} p_{ik} + \sum_{i \in J_k} \sum_{j \in_{J_k}} x_{ijk} s_{ijk} \quad \forall k \in K \tag{3.3}$$

where

$$y_{ik} = \begin{cases} 1 & \text{if job } i \text{ is assigned to machine } k \\ 0 & \text{Otherwise} \end{cases}$$

and

$$x_{ijk} = \begin{cases} 1 & \text{if job } i \text{ is the immediate predecessor of job } j \text{ on machine } k \\ 0 & \text{Otherwise.} \end{cases}$$

$$C_k \leq \frac{1}{|K|} C_{total}(1+\alpha) \quad \forall k \in K \tag{3.4}$$

$$C_k \geq \frac{1}{|K|} C_{total}(1-\alpha) \quad \forall k \in K \tag{3.5}$$

Constraint (3.4) and (3.5) are workload-balancing constraints for each machine. A perfect balancing is not usually possible. Hence, a certain percentage of tolerance α above and below the average work load are permitted.

$$\sum_{k \in N_k} y_{ik} = 1 \qquad i = 1 \cdots n \tag{3.6}$$

Constraint (3.6) ensures that each job should be assigned to a processing line.

$$x_{ijk} \leq y_{ik} \qquad \forall i \quad j \in K_i \quad k \in K \tag{3.7}$$

Constraint (3.7) guarantees that a job cannot precede another job on machine k unless it has been assigned to machine k.

$$\sum_{i \in J_k} x_{ijk} = 1 \qquad \forall k \in K \quad j \in J_k \tag{3.8}$$

$$\sum_{j \in J_k} x_{ijk} = 1 \qquad \forall k \in K \quad i \in J_k \tag{3.9}$$

Constraints (3.8) and (3.9) give a more detailed assignment of jobs to lines.

$$\sum_{i \in S_k} \sum_{j \in S_k \ j \neq i} x_{ijk} \leq \mid S_k \mid -1 \qquad S_k \subseteq J_k \tag{3.10}$$

Constraint (3.10) represents sub-tour elimination constraints which ensures that a job cannot be the immediate predecessor or successor of two or more different jobs at the same time.

$$
\begin{aligned}
& \min && C_{total} && \\
C_{total} &= \sum_{k \in K} C_k && && (1)\\
C_k &= \sum_{i=1}^{n} y_{ik} p_{ik} + \sum_{i \in J_k} \sum_{j \in J_k} x_{ijk} s_{ijk} && \forall k \in K && (2)\\
C_k &\leq \frac{1}{m} C_{total}(1+\alpha) && \forall k \in K && (3)\\
C_k &\geq \frac{1}{m} C_{total}(1-\alpha) && \forall k \in K && (4)\\
\sum_{k \in K_i} y_{ik} &= 1 && i = 1..n && (5)\\
x_{ijk} &\leq y_{ik} && \forall i, j \in K_i, k \in K && (6)\\
\sum_{i \in J_k} x_{ijk} &= 1 && \forall k \in K, j \in J_k && (7)\\
\sum_{j \in J_k} x_{ijk} &= 1 && \forall k \in K, i \in J_k && (8)\\
\sum_{i \in S_k} \sum_{j \in S_k, j \neq i} x_{ijk} &\leq |S_k| - 1 && S_k \subseteq J_k && (9)
\end{aligned}
$$

Complexity of the problem Finding the solution for $1|S_{ij}|C_{max}$ is NP-Complete. Hence, $P_m||C_{max}$ and $R_m|S_{ijk}|C_{max}$ problem are NP-complete [20]. When $m = 1$, SPM-LBSDS reduces to $1|S_{ij}|C_{max}$. As a result, SPM-LBSDS is NP-complete as well [37].

3.3 Heuristics to Solve Semi-Related Parallel Machines

There are several heuristics available to schedule identical and non-identical parallel machines. The most commonly used heuristics are Shortest Processing Time (SPT), Longest Processing Time (LPT), Largest Weight (LW) and Setup Avoidance (SA). Kurz and Askin [20] developed heuristics like (i) Slicing Heuristics (ii) Multiple MULTI-FIT heuristic and (iii) Multiple insertion heuristic for the objective of minimizing make span in identical parallel machines and they finally concluded that multiple insertion heuristic performed better. Also Rabadi et al [37] developed Meta-RaPS (Meta-heuristics for Randomized Priority Search) for the objective of minimizing make span which includes priority rule for jobs. In connection with this, Fowler et al [7] used dispatching rules like First In First Out (FIFO) and Earliest Due Date (EDD) for scheduling identical machines with sequence-dependent setups. But this research problem deals with minimizing make span with

load-balancing constraints. It is well known fact in Operations Research field that a particular methodology will not necessarily satisfy other objectives. Hence, an efficient strategy has to be developed to address these two aspects (Minimizing make span and load-balancing) simultaneously.

The following is the outline of the procedure for load-balancing on semi-related machines when there are sequence-dependent setups:

Step 1: INPUT the problem data

Step 2: SAVE the assignment order of the jobs in a LIST

Step 3: SELECT a job from the list according to the ORDER

Step 4: ASSIGN the job to a machine with respect to an ASSIGNMENT RULE

Step 5: REMOVE the job from the LIST

Step 6: CALCULATE the IMBALANCE. If the LIST= $\emptyset$, STOP. Check the feasibility. Otherwise go to step 4.

Using this procedure, based on how the LIST is formed (i.e., the assignment order is determined) and which ASSIGNMENT RULE is utilized, several dispatching rules/heuristics can be proposed. The assignment orders/LISTs that have been considered are

- Random (RN): The order of jobs is created randomly.
- Longest Processing Time (LPT): Jobs are ordered in non-increasing processing time.
- Longest Processing Time with Most Restricted Assignment First (LPT-MRAF): the LPT-MRAF list is created as follows: the jobs are first grouped with respect to the number of machines, $|K_j|$, which they can be processed in non-decreasing order (i.e., the grouping is as follows: jobs that can be processed on one machine, then on two machines,..., and finally on $|K|$ machines). In each group, the job order is obtained using the longest processing time.

In the fourth step, using the assignment rule, jobs selected from the list are assigned to machines to be processed. The assignment rules which we have considered are:

- Setup Avoidance (SA): Whenever jobs and machines are available, the SA rule searches for the machine/job combination that causes the least setup time. Fowler et al. [7] noted that SA rule is a commonly used rule by scheduling practitioners for problems with sequence-dependent setups when the objective is to minimize the make span.

- Cumulative Processing Time (CPT): CPT assigns the job to the machine with the least cumulative workload

- Hybrid Cumulative Processing Time and Setup Avoidance (CPT-SA): At any iteration, if the imbalance is within α then SA rule is used. Otherwise, CPT is applied.

In the above procedure, in the sixth step, the imbalance for machine k is calculated for any heuristic as follows:

$$\alpha_k = |1 - \frac{C_k}{\bar{C}}| \text{ where } \bar{C} = \frac{1}{|K|} C_{total}.$$

This definition of imbalance is different then the one proposed by Rajakumar et. al [38] who used the maximum completion time instead of the average completion time in their calculations.

Based on the above assignment "orders" and "rules," the following heuristics are proposed:

- LPT-SA

- RN-CPT

- LPT-CPT

- LPT-MRAF-CPT

- LPT-MRAF-CPT-SA

In the heuristics listed above, the first part is the assignment order and the last part is the assignment rule. For example, LPT-SA is the heuristic, which orders the job in the longest processing time order and then assign to the lines using the setup avoidance heuristic. Similarly, in LPT-MRAF-CPT-SA, jobs are sorted with respect to the Longest Processing Time with Most Restricted Assignment First and then the Hybrid Cumulative Processing Time and Setup Avoidance heuristic is utilized to assign the jobs to machines.

To clarify the functioning of heuristics we will present a step by step procedure of heuristic LPT-MRAF-CPT-SA:

- STEP 0: Initially consider all the available jobs as unscheduled jobs.
- STEP 1: Arrange the jobs in the unscheduled list according to LPT (MRAF) rule.
- STEP 2: Consider the next available job in the list.
- STEP 3: Calculate the Cumulative Processing Time (CPT) on all the processing lines
- STEP 4: Calculate the total CPT of all the lines.
- STEP 5: Calculate the average workload of all the machines.
- STEP 6: Check if the load on a particular machine is within the tolerance limit with respect to the average workload.
- STEP 7: If it is within the limit then follow steps from 8 to 11 otherwise follow steps from 12 to 14.
- STEP 8: If more than one machine can process the job, check the machine requires least setup with respect to previously assigned job.
- STEP 9: If more than one machine requires the same setup, then assign the job to machine which has least workload otherwise assign the job to the machine requires least setup.
- STEP 10: Update the workload on that particular machine with sum of sequence-dependent setup time and processing time of the job.
- STEP 11: Check if all the jobs have been assigned, If yes, stop the process otherwise go to step 3.
- STEP 12: If more than one machine can process the job, then assign the job to machine which has least cumulative workload.
- STEP 13: Update the workload on that particular machine with sum of sequence-dependent setup time and processing time of the job.
- STEP 14: Check if all the jobs have been assigned, If yes, stop the process otherwise go to step 3.

3.4 Genetic Algorithm to Solve SPM-LBSDS on Semi-Related Parallel Machines

According to Fowler et al.[7] GA has been widely applied to parallel machine problems mainly because of two reasons.

1. In each generation, GA is capable of producing feasible solutions, whereas in mathematical modeling the possibility of getting the feasible solution largely depends on the complexity and nature of the problems.

2. GA maintains a set of feasible solutions.

Fowler et al. [7] cited that "The knowledge of a family of good solutions is far more important than obtaining an isolated optimum". The problem considered in this thesis is strongly NP-hard as explained in section 3.2. Even the simplified version of this problem (single machine with the objective of minimizing make span) is a Traveling Salesman Problem (TSP), which is NP-hard problem. Hence, no polynomial time algorithm has been developed. This research makes use of hybridized GA to solve the problem. Fowler et al. [7] cited in his literature that Davis (1991) stated that conventional GA will not guarantee an optimal solution to a problem. He also stated that mixing GA with problem oriented heuristics will give better results than solving the problem with heuristics only. Levine [24] solved an airline crew scheduling problem by combining GA with local search heuristics. His algorithm performed better for the problem with complicated objectives with more number of variables. The heuristics being hybridized with GA in this research is Cumulative Processing Time (CPT). After chromosome (sequence) has been developed by GA, the assignment of jobs to machines has been prepared by CPT heuristics. Once the assignment has been made its fitness value has been calculated for deciding the chromosomes for next generation. The process has been repeated until the required number of generations have been created.

The following are the components of the proposed GA:

Representation (Coding): It has a crucial role in development of GAs. A good representation scheme is necessary to describe the problem-specific characteristics in detail. The representation method also plays a major role in the subsequent steps of GA. The genes can be represented in several forms like binary, real integer number or a combination of characters [50]. Tiwari and Vidyarthi [50] has discussed several representation schemes like adjacency, permutation, mixed-based for solving the scheduling problems in a machining center. After analyzing all the methods,

we found that the sequence-oriented natural number representation scheme will be the right choice for this kind of problems. We consider jobs as genes and the length of the chromosome is equal to the total number of jobs to be scheduled in all the machines. The genes are generated randomly from 1 to total number of jobs to form a sequence (Chromosome). For example: If there are six jobs to be scheduled, then the chromosome may be represented as [5 1 4 3 2 6], where the numbers in the brackets represent the jobs and the sequence represents the order in which has to be scheduled.

Initialization: Fowler et al.[7] reported that assigning jobs based on some pre-determined rules may provide better solutions and reduce the computational time than generating random solutions. Hence, an initial set of solutions are generated using the RN-CPT, LPT-CPT, LPT-MRAF-CPT and LPT-MRAF-CPT-SA heuristics presented in Section 3.3.

Evaluation of Fitness Function: The fitness function f_p for a chromosome p can be calculated as a function of the objective function value:

$$f_p(C_{total}) = C_{total}^{\gamma-1} \exp^{-C_{total}}$$

where $\gamma = 0.5$.

Cross-over: Once the parents are selected, cross-over operation is applied to generate a new solution. It is the process by which two parent string unite together to form a new off-spring string. The most commonly used types of cross-over are (i) Single-point Cross-over (ii) Two-point Cross-over and (iii) Uniform cross-over. In GA, cross-over operation will not be implemented always. Cross-over operation will occur on a pair of strings only if a random number generated is less than or equal to the selected cross-over probability. The cross-over probability is set to as 0.95. The type of cross-over employed is single point cross-over. Two parent strings are selected randomly from the population. A random number between 0 to l-1 is generated (l is the length of the chromosome) to determine the locus of the gene in the chromosome. When crossover is done, the genes before the crossover point in chromosome 1 is the first part of the child chromosome. The second part of the child chromosome is generated as follows: Examine the genes from the second chromosome one by one. If a gene is not already in the child chromosome, add that gene to the child chromosome. Otherwise, compare the next gene in the second chromosome with the existing genes in the child

chromosome.

The following example illustrate the single point cross-over briefly.

Consider two chromosomes (i.e., the parents) with 7 genes: [2 7 3 1 6 5 4] and [7 1 2 5 4 6 3]. Assume that the cross-over point is after gene 2. So the child chromosome will have 2 as the first gene (i.e., the gene(s) from the first chromosome). All other genes will be selected from the second chromosome. The first gene of the second chromosome is 7 and it is not available in the child chromosome. So we add gene valued 7 to the child chromosome. The resulting chromosome at this point of time is [2 7....]. Next, consider the gene valued 1 which is at the 2nd position in the second chromosome. This gene is added in the next position (3rd position) of the child chromosome. i.e., [2 7 1....]. Now, consider the next gene valued 2 in the 3rd position of the second chromosome. The gene 2 is already available in position one of the child chromosome hence, it is ignored. When all other genes from second chromosome are analyzed, the following child is obtained [2 7 1 5 4 6 3]. The same procedure has to be repeated for the second chromosome (assume the same cross-over point as after the first gene and the second child is obtained as [7 2 3 1 6 5 4].

Mutation: Once the cross-over has been completed, chromosomes are subjected to mutation. As like cross-over operation, mutation also occurs only if the random number generated is less than or equal to the mutation probability. The mutation probability is set to as 0.05. Mutation creates a new chromosome by altering the locus of the genes. Mutation operator applies individually to each chromosome. The chances of getting muted for a particular chromosome will not affect another chromosome. Tiwari and Vidyarthi [50] has discussed several mutation operators like Insertion (INS), Reciprocal Exchange (RE), Inversion (INV) and Displacement (DIS). We have used reciprocal exchange type of mutation operator in which two positions in a chromosome are randomly generated and the genes in that position are then exchanged to form a new chromosome. For example: In chromosome [1 5 2 6 3 4] with 6 genes, two positions are selected randomly as 2 and 4. The genes in these positions are 5 and 6 and are exchanged to form a new chromosome, [1 6 2 5 3 4].

Selection: The selection model should reflect the nature's law of survival of fittest. Normally, in a genetic algorithm chromosomes with a better fitness value will receive more chances to survive in the next generations. In this paper, the roulette wheel system is used to select the parents for

crossover and mutation. The process of selection of the parents for generating the next population is described below.

- STEP 0: Determine the fitness function value for different chromosomes in an initial population.
- STEP 1: Arrange the chromosomes in the descending order of the fitness value.
- STEP 2: The probability function $Prob(f_p)$ for individual chromosomes can be calculated as a function of the fitness function values.

$$Prob(f_p) = \frac{f_p}{\sum_{p'} f_{p'}}$$

- STEP 3: Use roulette wheel method to select candidates for cross over and mutation using the probability values found in step 2.

Reproduction: Among the old and the new population created by cross-over and mutation operation, we have to select the next best generation equal to size of initial population. Jahangirian and Conroy [18]have discussed several replacement strategies to replace the worse chromosomes from the old population. According to them the most commonly used method is the elitist method which selects the best chromosomes from the old population and the current population and passes it on to the next generation. By doing so, only the chromosomes having better performance will be given the chance to survive in the next generation. This methodology has been adopted in this research. After the new population had been generated, every individual chromosome has to be compared with the corresponding chromosome (from which it is born) in the old population. If the objective function value of the new chromosome is better than or equal to the old chromosome, then it is selected as a candidate for next generation otherwise the old chromosome will be selected as a candidate. The comparison process has to be done for all the new chromosomes. Now, all the candidate solutions have to be arranged in the ascending order with respect to their fitness value. Destroy the chromosomes which are repeated more than once and among the existing population select the ten best chromosomes as an initial solution to the next generation. If there are more than one chromosome repeated in a generation and if the cross-over operation happened to occur in those same chromosomes, the newly generated chromosome will also have the same value and hence

no improvement occurs. Moreover, at a point if all the chromosomes happened to be same, then the solution will strike at local optimum. Hence, by avoiding the same chromosome from repeating more than once, we can move the solution to the global optimum.

The flowchart in figures 3.1, 3.2, 3.3 and 3.4 explains the complete process of GA schematically.

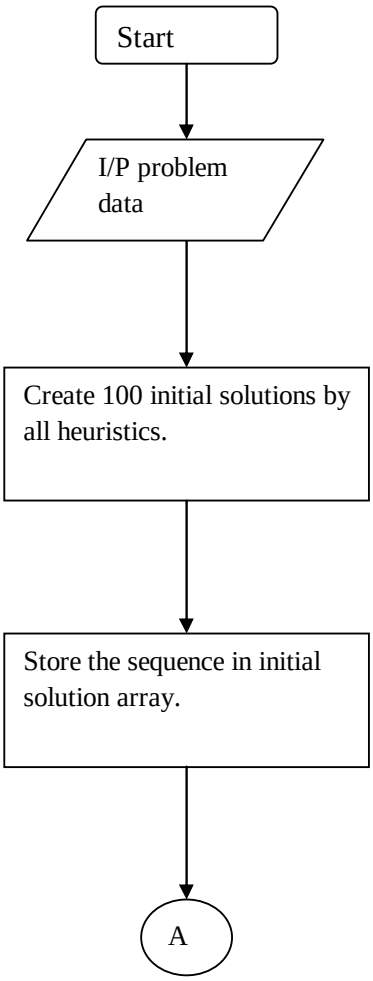

Figure 3.1: Genetic algorithm flow chart

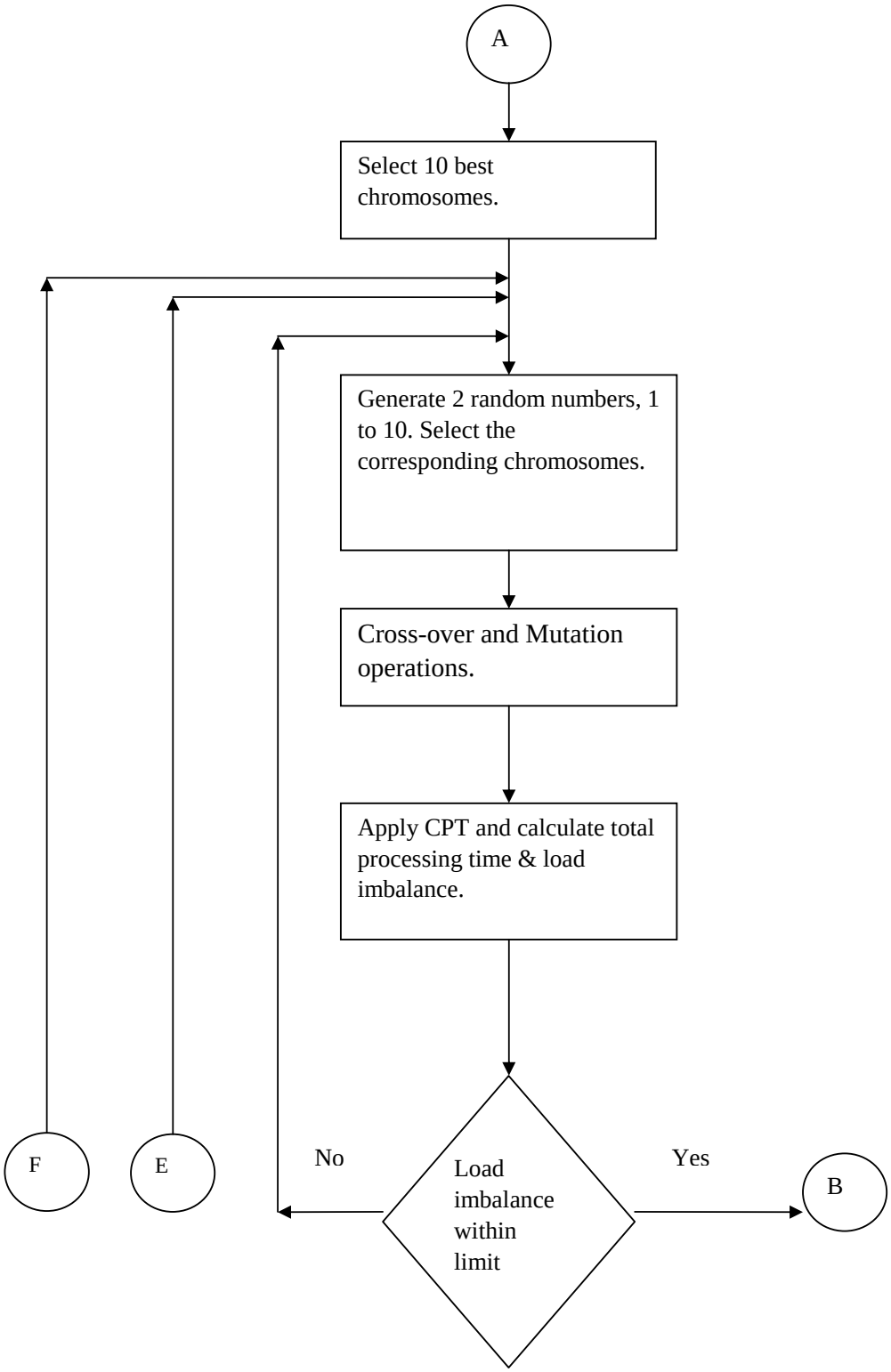

Figure 3.2: Genetic algorithm flow chart (continued)

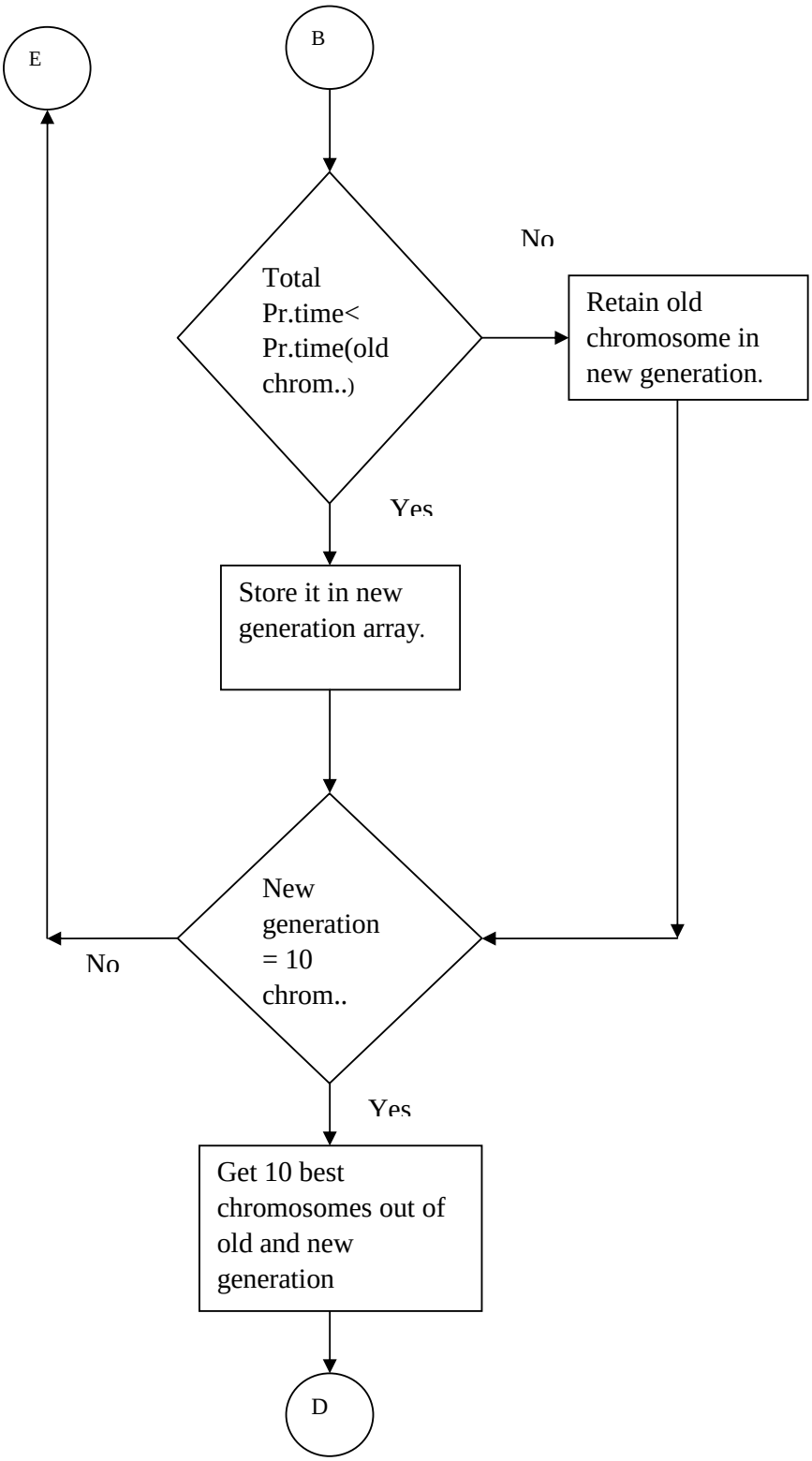

Figure 3.3: Genetic algorithm flow chart (continued)

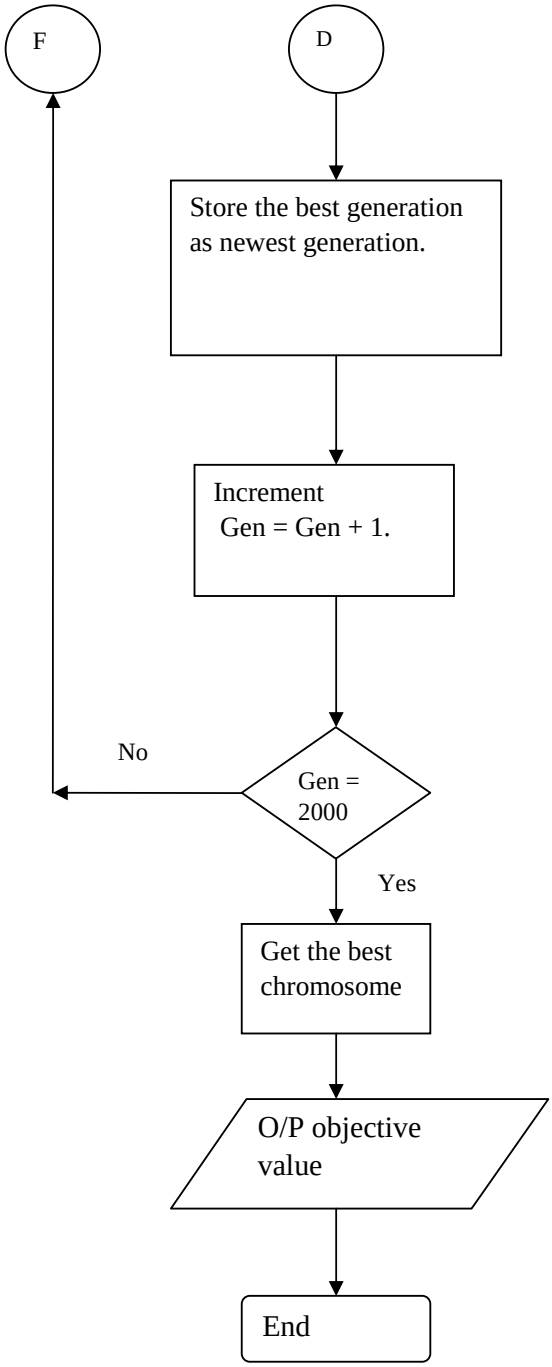

Figure 3.4: Genetic algorithm flow chart (continued)

3.5 Summary

This chapter provides a complete description of the problem considered in this research. It also presents the mathematical formulation and explains the complexity involved in solving problems of this kind. In addition, it explains the proposed heuristics and the developed GA to solve this problem.

CHAPTER 4

IMPLEMENTATION

4.1 Introduction

In order to validate the developed methodology, it must be tested under different problem scenarios. This chapter presents the experimental design and creates different problem scenarios by varying important parameters like Processing time, Setup time, Number of jobs and Percentage of total processing time under different job categories. In addition, the effectiveness of the developed methodology has been tested for the data obtained from a real-life problem.

This chapter is organized as follows: Section 4.1 provides a detailed explanation of the experimental design and discusses the performance of heuristics and the GA under different parameter settings. Section 4.2 gives an application of methodology developed to a real life problem. Finally, Section 4.3 concludes with a summary.

4.2 Computational Experimentation

This section discusses how to generate different scenarios to test the effectiveness of proposed heuristics in two machine and five machine environment. The heuristics and GA are programmed using Visual C++ 6.0 and the statistics are collected using Microsoft Excel. Below, we describe the experimental design which is followed by results and discussions.

4.2.1 Experimental Design

In the experimental setup, we have considered the following factors: the number of machines (two machines (2M) and five machines (5M)); the number of jobs (20, 40 and 60); processing time-setup time ratio, ρ, (low ($\rho = 0.1$), medium ($\rho = 1$) and high ($\rho = 10$)); and categories of jobs which can be processed on semi-related machines (2 levels for 2M case and 7 levels for 5M case). As a result, the total number of experimental designs are 27 for 2M case and 63 for 5M case.

In two machine setting, there are two categories of jobs. The first category can be processed on both machines while the second category can only be processed on machine two (i.e., the more capable machine). On the other hand, in the five machines setting, machines three, four and five

are the most capable machines and they are identical. Machine two can process all of the jobs that machine one can process and is less capable than machines three-five. As a result, there are three categories of jobs. The first category can be processed on machines 1-5; similarly, second and third categories of jobs can be processed on machines 2-5 and 3-5, respectively. The proportion of total processing and setup time allocated to each of the categories is experimented on seven levels (see Table 4.1).

Table 4.1: Experimental Setup for Types of Job Categories

	Scenarios									
	2M			5M						
Category	1	2	3	1	2	3	4	5	6	7
1	45	50	55	50	50	50	25	40	25	40
2	55	50	45	10	25	40	50	50	25	10
3				40	25	10	25	10	50	50

We have experimented on several values of imbalance. When $\alpha = 0.05$ or $\alpha = 0.10$, the proportion of infeasible solutions by the proposed heuristics was significantly high. We observed that when $\alpha > 0.20$, the problem behaved similar to an unrelated parallel machine scheduling problems with completion time objective. As a result, we choose $\alpha = 0.15$ for our experimentation. Note that finding a partition of jobs which satisfy the load-balancing constraint is similar to the set partitioning problem which is NP-hard.

The setup time matrix is asymmetric (i.e., s_{ijk} may not be equal to s_{jik}). The job data, processing time and setup time is generated using a method similar to Fowler et al. [7]. The method for each combination is as follows

High ratio ($\bar{p}/\bar{s} = 10$): The processing time of jobs are generated randomly between the intervals 0 and 20 and added with 60. The setup time of the jobs are generated randomly between the interval 0 and 7 and multiplied by 2. The ratio of the average of total processing time generated to the setup time will be 70/7, which equals to 10.

Moderate ratio ($\bar{p}/\bar{s} = 1$): The processing time of jobs are generated randomly between the intervals 0 and 20 and added with 40. The setup time of the jobs are generated randomly between the interval 0 and 7 and multiplied by 15. The ratio of the average of total processing time generated to the setup time will be approximately equal to 50/50 ,which equals to 1.

Low ratio ($\bar{p}/\bar{s} = 0.1$)**:** The processing time of jobs are generated randomly between the intervals 0 and 20 and added with 10. The setup time of the jobs are generated randomly between the interval 0 and 7 and multiplied by 43. The ratio of the average of total processing time generated to the setup time will be approximately equal to 15/150, which equals to 0.1.

Selection of GA parameters plays a vital role in GA experiments. We performed several experiments to determine the sensitivity of the problem to GA parameters:

- **Population size:** The size of the population has been varied from 5, 10 and 20 and we found out that the population size of 10 worked better.
- **Maximum generation:** The number of generations has been varied from 500, 1000, 2000 and 3000 and we found out that after 2000 generation the solution got saturated and there is no improvement. Therefore we selected the maximum generation as 2000.
- **Cross-over and mutation probability:** we selected 0.95 and 0.05 as the cross-over and mutation probability respectively.

The graph 4.1 shows the objective function value obtained for different number of generations like 500, 1000, 2000, and 3000.

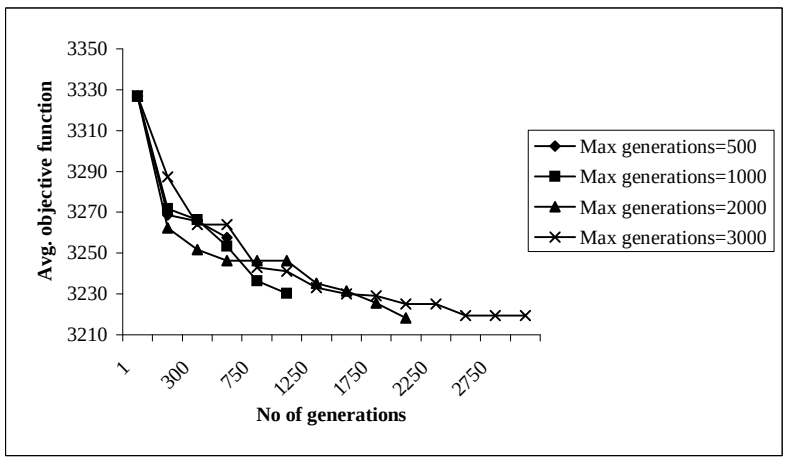

Figure 4.1: Effect of varying the number of generations with respect to the average objective function

The GA has been run for three times for every parameter and the average value has been considered for selecting the maximum number of generations. The graph shows that the average objective values for the number of generation 500 and 1000 are 3257.7 and 3230.3 respectively.

There is a significant improvement in the objective function value when the number of generations is increased until 2000 iterations. Later, the objective function value remains unchanged. Hence, we selected the maximum number of generation as 2000.

The graph 4.2 shows the objective function value obtained for different number of population sizes like 5, 10, 15, 20, and 25.

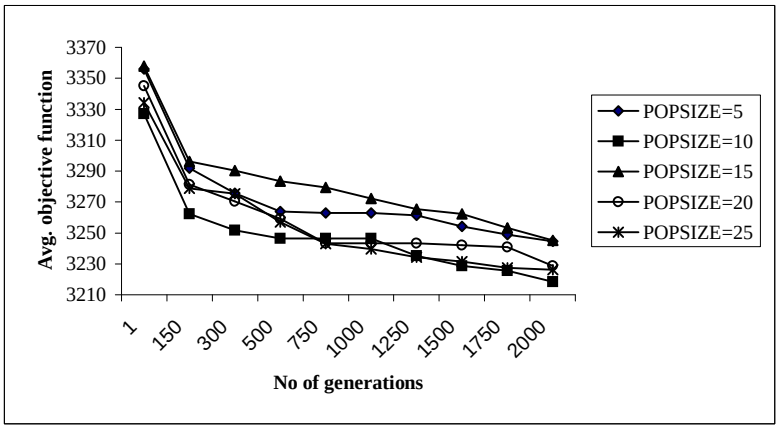

Figure 4.2: Effect of varying the size of population with respect to the average objective function

As like in determining the number of generations we run the GA for 3 times and the average objective function value has been considered to determine the correct population size. The graph shows that the average objective function value for population size 5, 10, 15, 20 and 25 are 3244.7, 3218.3, 3245.3, 3228.7 and 3226.3 respectively. The population size of 10 gave the best average objective function value. Hence, we selected 10 as the population size for further experimentation.

In the genetic algorithm, one hundred initial solutions are generated. Hence, out of the one hundred initial solutions, the ten best solutions based on the fitness value are selected. At any iteration, after the new population is generated, the ten best solutions from both the new and current set of solutions is used to determine the most elite solution. While selecting the ten best solutions, repetition are avoided. This helps to maintain the diversity in the population, and to prevent the solution from being stuck at local optimum.

Table 4.2: Performance of Heuristics to Solve SPM-LBSDS Problem

	$\bar{\sigma}$		$\bar{\alpha}$		$best$	
Heuristic	2M	5M	2M	5M	2M	5M
RN-CPT	3.12	2.15	7.70	3.61	59.2	50.7
LPT-CPT	4.21	4.14	8.26	1.98	0.0	0.0
LPT-MRAF-CPT	3.22	3.04	3.98	2.17	7.4	7.9
LPT-MRAF-CPT-SA	2.76	2.20	8.25	4.19	55.5	47.6

4.2.2 Results and Discussion

Each experimental design is run for five different data sets on a Pentium IV 1.6Ghz machine with 512MB of RAM. Furthermore, for each data set, the genetic algorithm is run five times to obtain different solutions. Although, five different heuristics (i.e., LPT-SA, RN-CPT, LPT-CPT, LPT-MRAF-CPT and LPT-MRAF-CPT-SA) are tested, the SA assignment rule did not produce feasible results for 96% of the cases. Note that the SA assignment rule is widely used in practice [7] for parallel machine scheduling with setups for minimum completion time objective. However, we observed that this rule can not be used to solve problems with load-balancing in parallel machine scheduling. LPT-MRAF-CPT and RN-CPT give feasible solutions in all of the cases that we have experimented. Recall that the LPT-MRAF first prioritizes the most restricted jobs and RN-CPT generates random orders until a feasible solution is determined. On an average, RN-CPT determines a feasible assignment in 0.5 seconds in 2M case and 1.5 seconds in 5M case compared to 5.1 and 5.4 seconds for GA to determine its final solution in 2M and 5M cases, respectively. Other heuristics do not consume any significant CPU time. LPT-CPT and and LPT-MRAF-CPT-SA produce feasible solutions for 75% and 85% of the cases.

Table 4.3: Performance of Heuristics (C_{total} and Total Setup) in 2M and 5M Environments

	C_{total}		Total Setup	
Heuristic	2M	5M	2M	5M
RN-CPT	1989.0	1976.7	118.2	110.3
LPT-CPT	2239.0	2029.5	159.2	145.1
LPT-MRAF-CPT	2002.4	1985.1	131.7	118.8
LPT-MRAF-CPT-SA	1969.0	1956.6	108.6	94.8
GA	1937.9	1918.2	67.2	75.2

We also collect statistics on the number of times that a heuristic performs the best: in 2M case, RN-CPT gives the best solution in 59.2 % of runs while LPT-MRAF-CPT-SA gives the best

solution in 55.5 % of the runs (table 4.2, the $\overline{best}$ columns). In Table 4.2, $\bar{\sigma}$ denotes the average deviation of heuristic solution from the GA solution. In case of 2M, LPT-MRAF-CPT-SA obtains the lowest $\bar{\sigma}$ and LPT-MRAF-CPT obtains the lowest imbalance ($\bar{\alpha}$). Even though the heuristic LPT-MRAF-CPT-SA gives the best average deviation, it fails to give feasible solutions in 15% of the cases. As a result, RN-CPT provides the second best average deviation and the maximum number of best solutions. Note that similar observations holds for the 5M case as well. Note that Rajakumar et al. [38] report that LPT-CPT give best solutions with respect to load-balancing in identical parallel machine environment without setups. However, LPT-CPT performs poorly in case of semi-related machines with sequence-dependent setups.

Table 4.4: Performance of Heuristics with Respect to Number of Jobs

	2M			5M		
Heuristic/n	20	40	60	20	40	60
RN-CPT	982.7	2015.0	2969.3	973.8	1972.0	2984.2
LPT-CPT	1357.3	2168.8	3191.0	1085.6	1984.7	3018.3
LPT-MRAF-CPT	992.0	2031.3	2984.0	983.2	1982.5	2989.8
LPT-MRAF-CPT-SA	965.6	1991.4	2949.9	958.6	1959.9	2951.4
GA	935.6	1973.1	2904.1	944.3	1938.0	2872.2

On an average, Table 4.4 shows that LPT-MRAF-CPT-SA heuristic gives the best average objective function value. This might be due to the fact that the assignment order is determined by considering the restrictions in job processing and sequence dependent setups at the same time. The second best performance is given by RN-CPT heuristic. As the number of jobs increases $n = 20$ to $n = 40$, the objective function value approximately doubles. However, this is not the case, when n is increased from 40 to 60. In this case, C_{total} increases by 150%.

Table 4.5: Effect of Processing - Setup Time Ratio on the Objective Function

	2M			5M		
Heuristic/ρ	0.1	1.0	10	0.1	1.0	10
RN-CPT	919.3	2143.0	2904.7	905.5	2117.2	2907.3
LPT-CPT	1292.3	2226.9	3197.9	1016.7	2138.7	2933.2
LPT-MRAF-CPT	935.1	2157.7	2914.7	921.2	2124.5	2909.8
LPT-MRAF-CPT-SA	910.1	2130.7	2866.1	917.3	2070.5	2882.0
GA	868.5	2092.7	2852.5	876.4	2009.5	2868.7

Similar observations can be made for the effect of processing time -setup time ratio in Table 4.5. When the effect of processing time is minimal (i.e., the $\rho = 10$), the GA and heuristics have similar performance. For smaller ρ the GA outperforms all of the heuristics.

Table 4.6: Effect of Number of Jobs on Imbalance

	2M			5M		
Heuristic/n	20	40	60	20	40	60
RN-CPT	6.90	8.52	7.68	4.43	3.88	2.51
LPT-CPT	6.69	8.95	8.13	1.90	2.04	1.98
LPT-MRAF-CPT	3.94	4.68	3.33	3.04	1.61	1.87
LPT-MRAF-CPT-SA	7.99	7.90	8.79	3.47	4.00	4.02

Table 4.7: Effect of Processing - Setup Time Ratio on Imbalance

	2M			5M		
Heuristic/ρ	0.1	1.0	10	0.1	1.0	10
RN-CPT	6.18	9.57	7.34	4.83	3.30	2.69
LPT-CPT	8.79	7.66	8.44	2.84	1.30	1.79
LPT-MRAF-CPT	3.31	4.66	3.98	2.41	2.22	1.83
LPT-MRAF-CPT-SA	9.90	8.69	5.55	3.87	4.98	2.77

Although the load-balancing objective appears as a constraint in the SPM-LBSDS model, we would like to test the performance of different heuristics when we vary the number of jobs (Table 4.6) and processing time setup time ratio (Table 4.7). In both cases, on average LPT-MRAF-CPT provides the best result. The only exception occurs in the 5M case when $\rho = 1$ and $\rho = 10$. In these cases LPT-CPT provides the lowest imbalance. However, we can not observe the fact that when n increases the imbalance decreases as in the parallel machine scheduling [38].

Table 4.8: GA Performance as Function of Processing - Setup Time Ratio and Number of Jobs with Respect to Best Heuristic Solution

	2M			5M		
ρ	20	40	60	20	40	60
0.1	3.43	4.35	4.75	1.70	1.29	0.41
1.0	0.97	0.88	1.48	1.17	0.61	0.52
10.0	1.62	0.43	0.12	1.36	0.11	0.19

In Table 4.8, we present the effect of the number of jobs and processing time -setup time ratio on the performance of the GA. When the processing time-setup time ratio, ρ, is small, i.e., the

processing time is relatively less important than the setup time (setups dominate the processing time), the GA's performance is higher compared to the cases where ρ is larger. When 2M, $n = 20$ and $\rho = 0.1$, the GA performs 3.43% better than the best heuristic solution. In case of 2M, the lowest average percentage deviation with respect to GA is found at the $\rho = 1$ and $n = 20$. The combination of $\rho = 10$ and n =40 and 60 gave the lowest average percentage deviation with respect to GA. In case of 5M, the combination of $\rho = 1$ and $n = 20$ and the combination of $\rho = 10$ and $n = 40$ and 60 give the lowest average percentage deviation with respect to GA. In 2M case, there is generalizable trend about the performance of the GA, while in 5M case, when the number of jobs increase, generally the performance of the heuristics improve.

4.3 Case Study at an Aluminum Processing Plant in Turkey

A large aluminum casting plant located at Istanbul, Turkey is considered to illustrate the implementation in a practical setting. The data was collected by Yildirim et al. (2006). This factory has a five non-identical machines (casting lines). The jobs are characterized by sequence-dependent setups as well as machine dependent setups. The products ordered by the customers have different properties like alloy types and widths. The sequence dependent setups will occur when there is a change of alloy and widths. Whenever there is a change in the alloy type, a significant setup may be required. In most of the cases, no setup is required if impure alloy has to be processed after pure alloy. On the other hand, a significant setup of 10hrs will be required in first casting line and 15 hrs will be required in other casting lines if pure alloy has to be processed after an impure alloy. In addition to this, if there is a change in width, an additional setup is also required. For the two consecutive jobs, if the width of the previous job is narrower than the width of the present job, both roller and shaper have to be changed to process a wider job which takes 6 hrs of setup. If the present job is narrower than the previous job then the setup time is 2.5 hrs. In order to decrease the setup times due to job widths, the company had standardized the widths into 14 sizes as 1220mm, 1280mm, 1320mm, 1400mm, 1450mm, 1550mm, 1600mm, 1650mm, 1700mm, 1830mm, 2080mm, 2120mm, 2140mm and 2200mm. The width of the casting lines vary from 1400mm to 2200mm. The casting line 1 can process jobs up to the width of 1400mm and casting line 2 can process jobs up to the width of 1700mm and casting line 3,4 and 5 can process jobs up to width 2200mm. The

goal is to minimize the sum of sequence-dependent setups between the alloy and width changes and also to balance the workload.

The data have been collected for ten months from January to October 2004. The heuristics and GA are also programmed using Visual C++ 6.0 on a 2.8GHZ Pentium IV processor with 512 MB of RAM and the statistics are collected using Microsoft Excel. The initial solution to GA has been created successfully for the value of α =0.30 and by proposed heuristics like LPT-SA, RN-CPT, LPT-CPT, LPT-MRAF-CPT and LPT-MRAF-CPT-SA. For every month, GA has been run for five times and the average objective function value has been calculated.

Table 4.9: Percentage of Feasible Solutions

Heuristics	(%)
RN-CPT	100
LPT-CPT	60
LPT-MRAF-CPT	100
LPT-MRAF-CPT-SA	100

Statistics is collected on performance measure of the heuristics and GA. The LPT-SA heuristics did not provide any feasible solution. RN-CPT, LPT-MRAF-CPT, LPT-MRAF-CPT-SA gave feasible solution for all of the months (refer to table 4.9). It is because that the heuristics LPT-MRAF-CPT and LPT-MRAF-CPT-SA first prioritizes the most restricted jobs and RN-CPT generates random orders until a feasible solution is determined. All of the heuristics didn't consume any significant amount of CPU time and on an average GA took 5.7 seconds to determine its final solution. The heuristic LPT-CPT gave feasible solutions only for six months out of ten months.

The following table 4.10 gives the average total processing and setup time of all heuristics for all the months. Even though the heuristic LPT-CPT gave the lowest average total completion time, it gave feasible solutions only for six months. The heuristic RN-CPT gave the second lowest value and also it gives feasible solution for all of the months. Hence we conclude that the overall performance of RN-CPT heuristic is the best.

Each of the heuristics RN-CPT and LPT-MRAF-CPT-SA gave best solutions in 50% of the cases. In table 4.11 $\bar{\sigma}$ denotes the average deviation of the heuristics with respect to GA. The heuristic LPT-MRAF-CPT gives the lowest imbalance ($\bar{\alpha}$). The heuristic RN-CPT and LPT-MRAF-CPT-

Table 4.10: Performance of Heuristics (C_{total} and Total Setup)

Heuristics	C_{total}	Total Setup
RN-CPT	3574.3	327.7
LPT-CPT	3538.167	385.3
LPT-MRAF-CPT	3633.4	386.8
LPT-MRAF-CPT-SA	3580.5	333.9
GA	3458.2	222.3

SA give the lowest average deviation with respect to GA and also give feasible solutions for all the months. The heuristic LPT-MRAF-CPT-SA give the maximum value of imbalance.

Table 4.11: Performance of Heuristics to Solve SPM-LBSDS Problem

Heuristics	$\bar{\sigma}$	$\bar{\alpha}$	$\overline{best}$
RN-CPT	3.4	6.5	50
LPT-CPT	4.9	4.8	0
LPT-MRAF-CPT	5.1	1.8	0
LPT-MRAF-CPT-SA	3.5	8.9	50

Table 4.12 gives the percentage deviation of the best GA solution from the best heuristic solution. The heuristic LPT-MRAF-CPT-SA gave the lowest deviation with respect to GA and the heuristic RN-CPT gave the second best solution.

Table 4.12: GA Performance with Respect to Best Heuristic Solution

Heuristics	$\bar{\sigma}$
RN-CPT	2.739291
LPT-CPT	6.234905
LPT-MRAF-CPT	4.677768
LPT-MRAF-CPT-SA	2.199059

In addition to the results and discussions of heuristics and GA, we have compared the company's existing schedule and the results generated by four phase methodology of Yildirim et al. (2006) with our GA's solution. The proposed methodology outperformed the company's schedule and the results of four phase methodology in most of the cases. Tables 4.13 and 4.14 present the GA performance for all the months. On an average, the proposed methodology has improved the company's schedule and the four phase methodology by 5.3% and 1.7% respectively.

Table 4.13: Comparison Between Company's Schedule, 4Phase and GA (Jan-May)

	Jan	Feb	Mar	Apr	May
Company's schedule	3688.2	3310.6	3712.7	3590.6	3712.7
4 Phase methodology	3617.4	3248	3611.3	3503.2	3544.4
GA	3415.8	3218.8	3756	3671.2	3693.4
% Improvements (Cmpy)	8.0	3.0	-1.1	-2.1	0.6
% Improvements (4Ph)	6.0	1.0	-3.8	-4.5	-4.0

Table 4.14: Comparison Between Company's Schedule, 4Phase and GA (Jun-Oct)

	Jun	Jul	Aug	Sep	Oct	Avg
Company's schedule	3565.6	3696	3707.7	3590	3701.6	3627.57
4 Phase methodology	3446.1	3548.9	3479.8	3443.6	3616.5	3505.92
GA	3616.4	3146.8	3288.4	3297.2	3478	3458.2
% Improvements (Cmpy)	-1.4	17.5	12.8	9.0	6.5	**5.3**
% Improvements (4Ph)	-4.7	12.8	5.8	4.4	4.0	**1.7**

4.4 Summary

This chapter gives a detailed description of the experimental design and discusses the results of heuristics and the GA for different problem scenarios. It also provides an application of this developed methodology to a real-life problem.

CHAPTER 5
CONCLUSIONS AND FUTURE RESEARCH

5.1 Conclusions

In this study, the scheduling of semi-related machines with sequence-dependent setups and load-balancing constraints was solved, with the objective of minimizing the sum of completion time on all machines. A mathematical programming model was proposed. Due to the computational complexity involved in solving the mathematical model, heuristics and a genetic algorithm were developed to generate solutions within a reasonable period of time. Heuristics were also used to generate the initial set of solutions for the genetic algorithm, which resulted in reducing computational time as well. In order to test the effectiveness of the developed methodology, several randomly generated scenarios were tested and it was found that the genetic algorithm improves heuristic solutions significantly when the processing-setup time ratio is small. Furthermore, the most promising heuristic was the LPT-MRAF-CPT-SA which switched from CPT to SA (or vice a versa) based on the current relative imbalance. This also signifies that the proposed methodology can be applied to real-life problems. The heuristics and GA were tested on a real data obtained from an aluminum processing plant in Turkey. The developed methodology outperformed the company's existing procedure and the four-phase methodology.

5.2 Future Research

In this section, multi-objective load-balancing problems and a new solution methodology for future research is presented.

5.2.1 New Formulations

A direct extension of this research would be to have a multi-objective mathematical model in which both minimization of total completion time and imbalance could be considered. A multi-objective genetic algorithm approach could be developed to solve this problem as part of future research. The second objective of imbalance minimization can be added in two methods.

Two important factors contributing to load imbalance are maximum and minimum workload machines. Hence, if the difference between the maximum and minimum workload is reduced, then the imbalance could be minimized [11]. The following is a multi-objective mathematical programming formulation which includes this load-balancing objective:

$$\begin{aligned}
\min \quad & C_{total} \\
\min \quad & C_{max} - C_{min} \\
& C_{total} = \sum_{k \in K} C_k && (5.1) \\
& C_k = \sum_{i=1}^{n} y_{ik} p_{ik} + \sum_{i \in J_k} \sum_{j \in J_k} x_{ijk} s_{ijk} \quad \forall k \in K && (5.2) \\
& \sum_{k \in K_i} y_{ik} = 1 \quad i = 1..n && (5.3) \\
& x_{ijk} \leq y_{ik} \quad \forall i, j \in K_i, k \in K && (5.4) \\
& \sum_{i \in J_k} x_{ijk} = 1 \quad \forall k \in K, j \in J_k && (5.5) \\
& \sum_{j \in J_k} x_{ijk} = 1 \quad \forall k \in K, i \in J_k && (5.6) \\
& \sum_{i \in S_k} \sum_{j \in S_k, j \neq i} x_{ijk} \leq |S_k| - 1 \quad S_k \subseteq J_k && (5.7) \\
& C_{min} \leq C_k \leq C_{max} && (5.8)
\end{aligned}$$

The first and second objective functions represent the minimization of total completion time and difference between maximum and minimum workloads respectively. Equations from (5.1) to (5.7) hold the same explanation as found in Chapter 3, section 3.2. Equation (5.8) identifies maximum and minimum workload.

Imbalance could also be minimized by decreasing the workload on a machine with respect to average workload. In this case, the following multi-objective model could be used to determine optimal total completion time and load balance.

$$\min \quad C_{total}$$

$$\min \quad \alpha$$

$$C_{total} = \sum_{k \in K} C_k \tag{5.9}$$

$$C_k = \sum_{i=1}^{n} y_{ik} p_{ik} + \sum_{i \in J_k} \sum_{j \in J_k} x_{ijk} s_{ijk} \quad \forall k \in K \tag{5.10}$$

$$\alpha \geq \mid C_k - \bar{C} \mid \quad \forall k \in K \tag{5.11}$$

$$\bar{C} = \frac{1}{m} C_{total} \tag{5.12}$$

$$\sum_{k \in K_i} y_{ik} = 1 \quad i = 1..n \tag{5.13}$$

$$x_{ijk} \leq y_{ik} \quad \forall i, j \in K_i, k \in K \tag{5.14}$$

$$\sum_{i \in J_k} x_{ijk} = 1 \quad \forall k \in K, j \in J_k \tag{5.15}$$

$$\sum_{j \in J_k} x_{ijk} = 1 \quad \forall k \in K, i \in J_k \tag{5.16}$$

$$\sum_{i \in S_k} \sum_{j \in S_k, j \neq i} x_{ijk} \leq |S_k| - 1 \quad S_k \subseteq J_k \tag{5.17}$$

The first and second objective functions represent the minimization of total completion and load imbalance. Equations (5.9) to (5.10) and equations (5.13) to (5.17) have the same interpretation as found in Chapter 3, section 3.2. Equation (5.11) represents the minimization of workload on all machines with respect to average workload, and equation (5.12) calculates the average workload.

5.2.2 Solving the Load-Balancing Problem

In addition to the multi-objective mathematical model, improvement in the solution quality might be further improved by employing Path Re-linking (PR) [54], which is a population-based meta-heuristic. As a GA, PR creates newer solutions with the help of current solutions, and based on their fitness values, a new set of candidate solutions are generated. According to Yavuz et al. [54], PR consists of three steps:

- Create the initial set of solutions by using the heuristics specifically designed for the problem. Select the subset of best solutions as a reference set.

- Among the subsets of the reference set, create subsets each consisting of two solutions. Term one of the solutions as "initiating solution" and the other as "guiding solution". Create a path from the initiating solution to the guiding solution by using the method called "distance measure based on neighborhood search." This is basically an iterative neighborhood search process that identifies neighbor solutions that decrease the distance to the guiding solution.

- Finally, evaluate the solutions that have been created in the second stage and the reference solutions based on their fitness values. Only allow the solutions with the fittest values to participate in the next iteration. Repeat the last two stages until the required number of iterations is attained.

In the problem setting, 100 initial solutions could be created by heuristics like LPT-SA, RN-CPT, LPT-CPT, LPT-MRAF-CPT and LPT-MRAF-SA-CPT. Out of 100 solutions, ten best solutions could be selected as a reference set. Then five subsets in the reference sets, each consisting of two solutions called initiating solutions and guiding solutions could be created. The solution between the initiating and guiding solutions could be explored in such a way that the new solution would decrease the distance to the guiding solution. The process for five subsets should be repeated. The new set of solutions and the reference solutions would then be examined for further iteration. The fitness values for the newly created solutions and the reference solutions would be calculated and the solutions having the highest values would be considered for further iterations. The last two stages would be repeated until the solution quality becomes saturated.

REFERENCES

REFERENCES

[1] Allahverdi, A., Gupta, J.N.D., Aldowaisan, T., "A review of scheduling research involving setup consideration," *International Journal of Management Science*, 1999, 27, pp.219-239.

[2] Arnaout, M. J., and Rabada, G., "Minimizing the total weighted completion time on unrelated parallel machines with stochastic times," *Proceedings of the 2005 Wnter Simulation Conference*, 2005, pp.2141-2147.

[3] Becker, C., Scholl, A. and Dolgui, A. "A survey on problems and methods in generalized assembly line balancing," *European journal of Operational Research*,(2006) 168, pp. 694-715

[4] Cao, J., and Bedworth, D.D., "Flow shop scheduling in a serial multi-product processes with transfer and setup times," *International Journal of Production Research*, 1992, 30(8), pp. 1819-1830.

[5] Chen, Z.L., and Powell, W. B., "Exact algorithms for scheduling multiple families of jobs on parallel machines," *Naval Research Logistics*, 2003, 50, pp. 823-840.

[6] Dietrich, B. L., and Escudero, L. F., "On solving a 0-1 model for workload allocation on parallel unrelated machines with setups," *Proceedings 3rd ORSA/TIMS Conference on Flexible Manufacturing Systems: Operations Research Models and Applications*, 1989, pp.181-186.

[7] Fowler, J.W., Horng, S. M., and Cochran, J.K., "A hybridized genetic algorithm to solve parallel machine scheduling problems with sequence dependent setups," *International Journal of Industrial Engineering*, 2003, 10(3), pp. 232-243.

[8] Gascon, A., and Leachman, R. C., "A dynamic programming solution to the dynamic, multi-item, single-machine scheduling problem," *Operations Research.*, 1998, 36 (1), pp.50-56.

[9] Glover, F. and Greenberg, H.J., "New approaches for heuristic search: A bilateral linkage with artificial intelligence," *European Journal of Operational Research*, 1989, 39, pp.119-130.

[10] Goldberg, E. D., Genetic Algorithm, 2001, Pearson Education, Inc.

[11] Guerrero, F., Lozano, S., Koltai, T., and Larraneta, J., "Machine loading and part type selection in flexible manufacturing systems," *International Journal of Production Research*, 1999, 37 (6), pp.1303-1317.

[12] Gung, R.R., and Steudel, H. J., "A workload balancing model for determining set-up time and batch size reductions in GT flow line work cells," *International Journal of Production Research*, 1999, 37 (4), pp.769-791.

[13] Gunther, O. H., Kulak, O., and Yilmaz, O. I., "Application of minimum setup strategy in PCB assembly", *35th International Conference on Computers and Industrial Engineering*, pp.845-850.

[14] Herrmann, J. W., Lee, Y. C. and Hinchman, J., "A global job-shop scheduling with genetic algorithm," *Production and Operations Management*, 1995, 4(1), pp.30-45

[15] Hillier, M. S. and Brandeau, M. L. "Cost minimization and workload balancing in printed circuit board assembly," *IIE Transactions*, 2001 33, 7, pp. 547-557.

[16] Holland, J.H. Adaptation in Natural and Artificial Systems, The University of Michigan Press, Ann Arbor, (1975).

[17] Hu, J., and Caramanis, M.," Near optimal setup scheduling for flexible manufacturing systems," *Proceedings 3rd International Conference on Computer Integrated Manufacturing*, Troy, NY, May, 20-22, 1992, pp.192-201.

[18] Jahangirian, M., and Conroy, G. V., "Intelligent dynamic scheduling system: the application of genetic algorithms," Integrated manufacturing systems, 2000, 11(4), pp.247-257.

[19] Kim, S.C., and Bobrowski, P.M., "Impact of sequence dependent setup time on job shop scheduling performance," *International Journal of Production Research*, 1994, 32(7), pp. 1503-1520.

[20] Kurz, M. E., and Askin, R.G., "Heuristic scheduling of parallel machines with sequence-dependent setup times," *International Journal of Production Research*, 2001, 39 (16) pp. 3747-3769.

[21] Lee, Y. H., and Pinedo, M., "Scheduling jobs on parallel machines with sequence-dependent setup times," *European Journal of Operational Research*, 1997, 100, 464-474.

[22] Lee, T.R., and Ueng, J.H., "A study of vehicle routing problems with load-balancing," *International Journal of Physical Distribution and Logistics Management*, 1999,29 (10), 646-658.

[23] Lee, W.C., Wu, C.C., and Sung, H. J., "A bi-criterion single-machine scheduling problem with learning considerations," Acta informatica, 2004, 40, pp.303-315.

[24] Levine, D. "Application of genetic algorithm to airline crew scheduling," *Computers Operations Research*,1996 23(6), pp.547-558.

[25] Liu, C.Y., Chang, S. C., Hsu, H. M., and Chen, C. C., "Scheduling for IC testing and assembly," *Production Automation*, Taipei, Taiwan, R.O.C., 1994, 4, pp.59-67.

[26] Liu, J., and Tang, L., "A Modified Genetic Algorithm for Single Machine Scheduling," *Computers & Industrial Engineering*, 1999, (37), 43-46.

[27] Liu, Y.C., and Chang, C. S., "Scheduling fexible flowshop with sequence dependent setup effects," IEEE Transactions on Robotics and Automation, 2000, 16(4), pp. 408-419.

[28] Maimon, Z. O., and Braha, D., "A Genetic Algorithm approach to scheduling PCBs on a single machine," *International Journal of Production Research*, 1998, 36(3), pp. 761-784.

[29] Miller, M. D., Chen, C. H., Matson, J., Liu, Q., "A Hybrid Genetic Algorithm for the Single Machine Scheduling Problem," *Journal of Heuristics*, 1999, 5, pp.437-454.

[30] Modi, B. K., and Shanker, K., "A formulation and solution methodology for part movement minimization and workload balancing at loading decisions in FMS," *International Journal of Production Economics*, 1994, 34, pp.73-82.

[31] Monma, C.L., and Potts, C. N., "On the complexity of scheduling with batch setup times," *Operations Research*, 1989, 37, pp.798-804.

[32] Monma, C, L., and Potts, C, N., "Analysis of heuristics for preemptive machine scheduling with batch setup times", *Operations Research*, Sep-Oct 1993,41(5), pp 981-993.

[33] Ovacik, I.M., and Uzsoy, R., "Rolling horizon algorithm for a single machine dynamic scheduling problem with sequence dependent setup times," *International Journal of Production Research.*, 1994, 32(6), pp.1243-1263.

[34] Uzsoy, R.and Ovacik, I.M., "Rolling horizon procedures for dynamic parallel machine scheduling with sequence dependent setup times," *International Journal of Production Research.*, 1995, 33(11), pp.3173-3192.

[35] Pinedo, Micheal., "Planning and Scheduling in Manufacturing and services," 2005 *Springer Series in Operations Research and Financial Engineering.*

[36] Pinedo, Micheal., "Scheduling: Theory, Algorithms and systems," 1995, *Springer Series in Operations Research and Financial Engineering.*

[37] Rabadi, G., Moraga, J. R., and Al-Salem, A., "Heuristics for unrelated parallel machine scheduling problem with setup times," *Journal of Intelligent Manufacturing*, 2006, 17, pp.85-97.

[38] Rajakumar S, Arunachalam V.P, Selladurai V. Workflow balancing strategies in parallel machine scheduling, *International Journal of Advanced Manufacturing Technology*, 2004, 23, 366-374.

[39] Randhawa, S.U., and Kuo, C-H., "Evaluating scheduling heuristics for non-identical parallel processors," *International Journal of Production Research*, 1997, 35, 969-981.

[40] Rios-Mercado, R. Z., and Bard, F. J., "A Branch and Bound algorithm for permutation flow shops with sequence-dependent setup times", *IIE Transactions*, 1999, 31, pp. 721-731.

[41] Rubin, P., and Ragatz, G., "Scheduling in a sequence dependent setup environment with genetic search," *Computers and Operations Research*, 1995, 22(1), pp.85-95.

[42] Sawik, T., "An LP-based approach for loading and routing in a flexible assembly line," *International Journal of Production Economics*, 2000, 64, pp. 49-58.

[43] Shanker, K and Rajamarthandan, S., "Loading problem in FMS: part movement minimization," *Proc. 3rd ORSA/TIMS conf. on Flexible anufacturing Systems: Operations Research Models and Applications*, 1989, pp. 99-104.

[44] Srivatsan, N., and Greshwin, S.B., "Selection of setup times in a hierarchically controlled manufacturing system," *Proceedings 29th IEEE conference decision and control*, Honolulu, HI, December 1990, pp 575-581.

[45] Stecke, K. E., "Formulation and solution of nonlinear integer production planning problems for flexible manufacturing systems," *Management Science*, 1983, 29, pp. 273-288.

[46] Stowers, C. L., and Palekar, U. S., "Lot sizing problems with strong setup interaction," *IIE transaction*, 1997, 29(1), pp.167-169.

[47] Sule, D.R., Industrial Scheduling. (1997), *PWS Publishing Company.*

[48] Sun, D.S., Lee, T.E., and Kim, K.H., "Component allocation and feeder arrangement for a dual-gantry multi-head surface mounting placement tool," *International Journal of production Economics*, 2005, 95, pp.245-264.

[49] Suresh, V. and Chaudhuri, D., " Bi criteria scheduling problem for unrelated parallel machines," *Computers and Industrial Engineering*, 1996, 30(1), pp.77-82.

[50] Tiwari, M.K., and Vidyarthi, N.K., "Solving machine loading problems in a flexible manufacturing system using a genetic algorithm based heuristic approach," *International Journal of Production Research*, 2000, 38 (14), 3357-3384.

[51] Turkcan, A., Akturk, S, M., and Storer, H, R., "Non-identical parallel CNC machine Scheduling," *International Journal of Production Research*, 2003, 41 (10), pp.2143-2168.

[52] Van Hop, N. and Nagarur, N. N., The scheduling problem of PCBs for multiple nonidentical parallel machines, *European Journal of Operational Research*, (2004),158: 577-594.

[53] Wang, D., Gen, M., and Cheng, R., "Scheduling Grouped Jobs on single machine with Genetic Algorithm," *Computers and Industrial Engineering*, 36 (1999), 309-324.

[54] Yavuz, M., Akcali, E., and Tufekci, S., "A hybrid meta-heuristic for batching problem in Just-In-Time flow shops," *Journal of Mathematical Modelling and Algorithms*, 2006.

[55] Yu, L., Shih. M. H., Pfund. M., Carlyle. M.W., and Fowler.W.J., "Scheduling of unrelated parallel machines: an application to PWB manufacturing", *IIE Transactions*, 2002, 34, pp 921-931.

[56] Zhao, C., and Wu, Z., "A genetic algorithm approach to the scheduling of FMSs with multiple routes," *The international journal of flexible manufacturing systems*, 2001, 13, pp. 71-88.

[57] Zhou, C., and Egbelu, P.J., "Scheduling in a manufacturing shop with sequence dependent setups," *Robotics and Computer-Integrated Manufacturing.*, 1989, 5(1), pp.73-81.