New Intelligent Optimisation Systems for Job Shop Scheduling Problems in the Manufacturing Industry

A Thesis Submitted for the Degree of Doctor of Philosophy

By Atefeh Momenikorbekandi

Department of Electronic and Electrical Engineering,

College of Engineering, Design and Physical Sciences

Brunel University London

August, 2024

Abstract

This research focuses on scheduling systems in manufacturing and developing new optimisation techniques which are capable of dealing with scheduling problems. Scheduling is an important factor in manufacturing systems and aims to optimise the production system, reducing time and energy consumption. In this regard, numerous researchers have studied Job shop Scheduling Problems (JSSPs). In JSSPs, there are several jobs and machines, and depending on the type of job shops, schedules and related penalties, each job needs to be executed on machines on specific orders. Finding the best schedule is challenging, and there is still a need to improve and develop advanced and optimised scheduling models.

This work designs optimisation models based on hybridising Genetic Algorithm (GA) techniques and Reinforcement Learning (RL) for scheduling a furnace model and simulated job shops. Hence, several sophisticated algorithms are developed for this proposal, namely Stochastic GA, Sexual GA, Ageing GA, Parthenogenetic algorithm and Ethnic GA. These algorithms are employed to establish a new metaheuristic hybrid parthenogenetic algorithm (NMHPGA) based on the combinations of the different selections to hybridise the basic GAs; moreover, two types of advanced RL, including off-policy Q-learning and on-policy RL based on State-Action-Reward-State-Action (SARSA) are developed. Following that, all algorithms are tested on two categories of scheduling job shops, including 10 single-machine job shops and 19 multi-machine job shops; all the job shops are simulated in MATLAB, and the aim is to reduce the makespan of the job shops. Results which are compared to basic GA, show that the developed models attain superior results with a faster convergence rate. As a case study, a reheating furnace model is used to optimise material heating schedules, finding the most efficient schedule to minimise time and energy consumption. The models improve the efficiency on average to 40 % on job shops and furnace fuel consumption by up to 3.20 % and operation time by 3.79 %.

Statement of Originality

I certify that the thesis I have submitted to Brunel University London is entirely my creation, developed under the guidance of my supervisor, Professor Maysam Abbod. The thesis has appropriately credited all data sources used. I hereby affirm that the content presented in this thesis has not been submitted to obtain a degree from any other academic institution. I will be solely responsible if any evidence is found against my declaration.

Atefeh Momenikorbekandi
Brunel University London

Dedication

This work is dedicated to my parents for their unwavering help and support; I am genuinely grateful for their constant support and affection throughout the years.

Acknowledgements

First and foremost, I sincerely thank everyone who helped make this thesis possible. Without my supervisor, Professor Maysam Abbod, who provided me with the opportunity to learn many scheduling and AI techniques, my work would not have been feasible. His encouragement, direction, and support from the beginning to completing my PhD made the process much easier and more interactive. I would like to express my sincere thanks to my parents for their constant love, support, and encouragement.

I would also like to thank my colleagues for their insightful discussions regarding the subject and the work done. Lastly, I sincerely thank my teachers, friends, and everyone who helped me develop my skills.

Glossary

AI Artificial Intelligence

ABC Artificial Bee Colony

ACO Ant Colony Optimisation

AGA Ageing Selection Genetic Algorithm

APGA Ageing Parthenogenetic Algorithm

CS Cuckoo Search

DRL Deep Reinforcement Learning

DE Differential Evolution

EA Evolutionary Algorithm

EGA Ethnic Selection GA

FA Firefly Algorithm

FJSP Flexible Job Shop Scheduling Problem

GWO Grey Wolf Optimisation

GA Genetic Algorithm

HS Harmony Search

HPGA Hybrid Parthenogenetic Algorithm

JSSP Job Shop Scheduling Problem

ML Machine Learning

MM Multi-Machine

MH Metaheuristic

NMHPGA New Metaheuristic Hybrid Parthenogenetic Algorithm

NP-hardness Non-Deterministic Polynomial-Time Hardness

PSO Particle Swarm Optimisation

QRL Q-Learning RL

RL Reinforcement Learning

RGA Roulette Selection Genetic Algorithm

SPT Shortest Processing Time

SGA Sexual Selection Genetic Algorithm

SM Single-Machine

SARSA State-Action-Reward-State-Action

STPGA Stochastic Selection Parthenogenetic Algorithm

SPGA Sexual Parthenogenetic Algorithm

STGA Stochastic Selection Genetic Algorithm

TS Tabu Search

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

The manufacturing process consists of several intricate operations that must be meticulously planned and scheduled to be executed effectively, it is an integral part of manufacturing systems [1, 2]. Production scheduling aims to find the best possible schedule to maximise one or more performance parameters such as time or energy.

Scheduling fundamentally concerns ordering a known series of stages or actions along a timeline for effective and potentially optimal implementation subject to constraints and uncertainties. As a consequence, a particular set of resources is allocated to the series of operations in a way that satisfies specific efficiency or optimal requirements. Scheduling problems are one of the most challenging optimisation problems [3] and have emerged as an essential academic field spurring the publication of an enormous literature volume [4, 5]; in this context, Job Shop Scheduling Problems (JSSPs) pertain to industrial system scheduling are concerned with maximising operational efficiency by reducing production time and costs [5, 6, 7, 8].

A common feature among research studies on optimisation is the absence of consistent and efficient solution algorithms, and many researchers have applied different algorithms to find the best solution for JSSPs [7, 8, 9, 10, 11]. Practical algorithms need to be created to address scheduling problems optimally in an acceptable amount of time [3]. Johnson [12] first presented his work on optimal two and three-stage production plans with

setup times in 1954, which marked the beginning of the study of scheduling optimisation problems. The development of branch and bound approaches, pure and mixed integer programming formulations, and other scheduling application fields followed. In many industrial systems, an efficient production plan and optimal job execution sequence are essential to increase production and resource efficiency.

This chapter explains the nature of the salient characteristics of intelligent scheduling systems and the current manufacturing paradigm. It provides a basic conceptualisation of JSSPs, which are the particular field of concern in this research. The subsequent sections will explain the motivation for undertaking this study, expound its aim and objectives, identify the study's contributions to knowledge and associated publications, and provide an outline of the chapter structure of this thesis.

1.2 Manufacturing and Scheduling

1.2.1 The Emerging Paradigm

Manufacturing is seen as the source of all products which may be applied for production purposes and is entirely dependent on contemporary technologies [13, 14]. Manufacturing has a critical role in developing and developed countries; it is crucial for societies since it contributes significantly to the global economy [15, 16]. It is strategically necessary to shift the current manufacturing paradigm to one that emphasises sustainability and reducing energy [13, 17, 18, 19, 20]. To achieve sustainability, manufacturers are rethinking and changing their manufacturing processes. Given that natural resources are finite and cannot meet future generations' demand, manufacturing sustainability is a crucial concern [13, 16, 17, 18, 21].

An essential element of any manufacturing setup is scheduling, which directly affects the system's efficiency, productivity and cost-effectiveness [5, 22]. Scheduling allocates, manages, and maximises tasks and workloads. Scheduling is a crucial technique to effectively assign machinery and equipment and optimise production processes focusing on time and energy reduction. In manufacturing, scheduling provides the objective of si-

multaneously optimising production time and cost; it is a strategic process that seeks to minimise the makespan [5, 22]. Production scheduling has several types of machine environments, such as Single-Machines (SM) and Multi-Machines (MM), parallel machines, flow shops, and flexible job shops. These types depend on the jobs' technological needs and the type of facilities available [23, 24]. Studies on JSSP used to be primarily oriented on a single target, such as completion time, but modern scheduling takes multiple objectives into account [2]. For instance, there may be a focus on JSSP's energy and environmental aspects and makespan [25] or more academic inquiries, such as applying improved optimisation algorithms [26].

Although traditional scheduling methods have contributed significantly to this domain, they often need to handle complex and dynamic environments encountered in contemporary industrial settings [1, 5, 13, 22, 27, 28]. Artificial Intelligence (AI) has brought about substantial changes to the domain of JSSPs, which has historically been plagued by intricate decision-making procedures and the requirement for efficiency in the manufacturing and service sectors. AI offers a cutting-edge answer to these difficulties by its capacity to effectively process extensive datasets and intricate algorithms; this improves operational effectiveness and enables the implementation of flexible and responsive scheduling approaches, essential in settings marked by unpredictability and swift fluctuations in demand [29, 30].

The application of AI in the domain of JSSPs is readily apparent through the advancement of intricate models and algorithms. These models and algorithms possess the capability to acquire knowledge from data, identify trends, and enhance scheduling decisions in real-time. These innovations enhance operational efficiency and contribute to enhanced resource utilisation and decreased lead times. Industries' increasing automation and intelligence have significantly impacted JSSPs [29, 30]. Despite numerous research on JSSPs, there is still a need to find an improved and advanced model to optimise JSSPs. As a result, emphasis is now being given to more capable and modern technology. Various scheduling methods have been applied to solve JSSPs, and there are encouraging indicators of the current spike in interest in Metaheuristic (MH) algorithms. These high-

level algorithmic frameworks give developers a set of rules or approaches to follow. They are a desirable solution for scheduling problems due to their proficiency in navigating challenging search spaces [28, 31, 32].

1.2.2 Job Shop Scheduling Problems

The Job Shop Scheduling Problem (JSSP) is one of the most challenging problems in the manufacturing process. It consists of ordering a series of operations to make the execution process optimal [1, 8]. JSSPs are the central part of the manufacturing system, as they affect the efficiency of the production system. The initial attempts to solve the JSSP may be traced back to the 1950s [12, 33] and 1960s [34, 35]. Significant modern advancements have occurred since the 1980s. In 1984, optional machines were considered, with the initial development of Flexible Job Shop Problems (FJSP) [36]. With the extensive use of the flexible manufacturing system in recent years [37, 38], the problem of FJSP has become a research hot spot [1, 39]. Currently, studies on FJSP often concentrate on three aspects: problem definition, optimisation model, and implementation approach [39, 40].

Intelligent optimisation algorithms are widely utilised to address large-scale combinatorial optimisation issues, such as the classic FJSP and many extended scheduling problems. They can efficiently identify sub-optimal solutions, but the efficacy of various methodologies strongly depends on the optimisation objectives [41, 42]. Optimisation is a design problem that necessitates suitable approaches and methodologies to deliver positive outcomes over a respectable (i.e., timely) period with minimal costs. Simply put, optimisation identifies the variables or parameters affecting an objective function, whether it has a single target (single-objective optimisation) or several objectives (multi-objective optimisation) [32, 43].

Nevertheless, despite these developments and their potential, many uncharted territories and problems still need to be solved, highlighting the need for extensive research in various directions pertaining to JSSPs [6, 32]. To fine-tune the search scope, Meng et al. [44] devised a hybrid Artificial Bee Colony (ABC) method. Furthermore, Nouiri et al. [38] introduced a distributed Particle Swarm Optimisation (PSO) that can be integrated

into a manufacturing system. The JSSPs were addressed in another work using the ABC [45]. In a hybrid algorithm proposed by Li and Liu [37], global and local searches are controlled by Genetic Algorithm (GA) and Tabu Search (TS), respectively. Furthermore, Amelian et al. [46] applied a multi-objective optimisation model for JSSPs. Wei et al. [1] proposed a hybrid scheduling method for FJSPs. Xu et al. used an improved Firefly Algorithm (FA) for JSSPs [22].

This work intends to address the identified research gaps by investigating the use of scheduling techniques in the context of JSSPs. It is necessary to thoroughly analyse earlier works in this field to pinpoint any research gaps in applying scheduling algorithms. Even if significant progress has been accomplished, there are still many opportunities for improvement and creativity to optimise scheduling in manufacturing systems.

1.3 Motivations

Based on the importance of reducing energy consumption and time in the manufacturing system, this thesis investigates different intelligence systems that can optimise the production system and scheduling problems. As a result of optimising the system, industry can reduce energy consumption, time and cost in production lines. JSSP comprises a popular research subject in the scheduling discipline, attracting significant attention and investigation from researchers in engineering and academia [24]. Several studies have used MH to manage system optimisation, but there is an ongoing need to improve optimisation systems by any possible method.

This research seeks to contribute to this emerging field of studies and to address specific industrial optimisation problems. With this research background in mind, the following are the primary motivations for undertaking this thesis.

The financial effectiveness, operational effectiveness, and productivity of a manufacturing system are significantly influenced by scheduling for manufacturing job shops. A highly optimised scheduling process can shorten lead times for manufacturing, which has a positive economic impact [47, 48, 49, 50]. Manufacturing industries face increasingly intensive competitiveness in global markets, and there is an imperative to optimise

manufacturing efficiency[47, 48, 49].

Environmental and sustainability considerations are growing and are evident in the governance and regulatory frameworks that govern business operations and customer expectations. Optimised scheduling can help make manufacturing operations more ecologically friendly by reducing energy use. By increasing operational efficiency, intelligent scheduling can provide better workflow and less energy consumption [47, 48, 49, 51].

1.4 Aim and Objectives

This thesis aims to develop intelligent systems for optimising JSSPs. To achieve the above-defined aim, it undertakes to achieve the following objectives:

1. To investigate the state-of-the-art intelligent methods applied to JSSPs.
2. To develop and design new models of intelligent algorithms to optimise the JSSPs.
3. To generate and create simulated job shops to validate the proposed scheduling methods.
4. To design a novel scheduling algorithm based on advanced GA algorithms and test it using benchmarks (models' execution time, cost, and energy consumption after optimisation).
5. To create learning-based scheduling and Reinforcement Learning (RL) models.
6. To test the RL model on simulated job shops.
7. To test the designed novel models on an industrial furnace model to reduce the time and energy consumption and optimise the schedule.

1.5 Contributions to Knowledge

This thesis presents new optimisation models developed in this research. This system makes use of a variety of intelligent methods, including hybridised systems and novel

models of GAs; these methods include the combination of different selections, allowing the less accurate algorithm to pass along its results to the faster and more accurate algorithm, which will benefit from the faster convergence speed and more accurate results; these models specifically reduce the makespan of the schedules. In addition, simulated FJSPs generated by generator code in MATLAB are used to assess each algorithm.

The research demonstrates that new optimal solutions can be discovered by employing various efficient techniques, which prevent the system from producing infeasible solutions. Furthermore, the advanced RL techniques are proposed in the second contribution of this study. The RL models are tested to optimise FJSPs.

Additionally, a case study of the furnace model is used for optimisation proposals; the novel models are applied to optimise the schedule of the furnace model. The time and fuel consumption may be precisely measured using these methodologies. Finally, the results of designed algorithms, including hybrid GA and RL, are compared.

The current overview covers the domain of job shop scheduling and optimisation, emphasising the significance and utilisation of advanced computational models. The primary objective of this study is to highlight the distinct contributions made within this particular subject while conducting an in-depth analysis of the obtained data and engaging in a comprehensive discussion regarding their broader implications. The following are the main contributions of this research:

1. Advancements in GA Methodologies:

   The present study presents new selection strategies within the framework of GAs, which demonstrate a substantial enhancement in the diversity and efficacy of the solutions. This novel methodology combines the most advantageous characteristics from distinct categories, resulting in solutions demonstrating enhanced effectiveness over a wide range of scheduling scenarios.

   Significant progress has been made in algorithmic techniques by rediscovering conventional GA functions. This is especially crucial in intricate operating environments where traditional approaches may prove less efficacious.

2. Advancements in RL Models:

   The study has built advanced RL models to address intricate scheduling challenges. These models demonstrate proficiency in optimising key scheduling parameters such as makespan and resource allocation, hence enhancing the overall efficiency of the scheduling process. The models display notable efficacy in managing production schedules, notably in attaining elevated rates of completion. The inherent qualities of robustness and flexibility make them very suitable for demanding conditions commonly seen in industrial environments. The methods discussed in this study can be applied in both simulated and real-world settings.

3. Diverse Range of Simulated Tests: This study incorporated a wide variety of simulated job shops, creating a realistic and comprehensive testing environment. This guarantees that the constructed models possess not only theoretical validity but also practical feasibility in diverse settings. The utilisation of these models in practical industrial case studies constitutes a vital element of this research, thereby establishing a connection between theoretical research and practical usefulness. The rapid identification of appropriate solutions in complex scheduling assignments is of the highest priority.

   Significant progress has been made in the realm of enhancing resource utilisation and scheduling efficiency, resulting in notable reductions in overall completion durations within production schedules. These enhancements highlight the capacity of these models to enhance resource efficiency and time effectiveness in industrial operations. The academic and practical significance of the study refers to its importance and relevance in both scholarly and real-world contexts.

   The insights and achievements elucidated in this comprehensive research represent a notable advancement in the domain of job shop scheduling and optimisation. The implementation and utilisation of these sophisticated computational models not only bring significant insights to scholarly communities but also provide efficient and effective resolutions for intricate industrial scheduling difficulties. These

technological developments offer novel insights and approaches to tackle scheduling challenges in several industrial domains effectively.

1.6 Publications

The following list shows the publications of this research.

1. Atefeh Momenikorbekandi and Maysam abbod. "Multi-ethnicity Genetic Algorithm for job shop scheduling problems". UKSIM conference, Cambridge University, April 14, 2022. https://ijssst.info/Vol-22/No-1/cover-22-1.htm.

2. Atefeh Momenikorbekandi and Maysam abbod. "A Novel Metaheuristic Hybrid Parthenogenetic Algorithm for Job Shop Scheduling Problems: Applying an Optimisation Model". IEEE Access, vol. 11, pp. 56027-56045, 2023, doi: 10.1109/ACCESS.2023.3278372.

3. Atefeh Momenikorbekandia and Maysam Abbod. "Intelligent Scheduling Based on Reinforcement Learning Approaches: Applying Advanced Q-Learning and State –Action–Reward–State– Action Reinforcement Learning Models for the Optimisation of Job Shop Scheduling Problems". Electronics, MDPI, 2023, 12(23), 4752; https://doi.org/10.3390/electronics12234752.

1.7 Thesis Structure

This thesis contains several chapters, as adumbrated briefly below.

1. Chapter 1 outlines the research's initial context, which explains the background of the research context about manufacturing and JSSPs. The context for the topic's importance in the larger academic and industrial scene is provided in this section. The chapter summarises the precise aims and objectives and clarifies the rationale for choosing this research topic. It explains the study's motivation, aims and objectives, and distinctive contributions it intends to make to the field.

2. Chapter 2 provides a thorough literature review on JSSP, tracing the development of scheduling algorithms and examining their substantial impact on JSSP and production processes. This chapter delves deeply into the approaches frequently used in the academic and industrial worlds to investigate the optimisation systems of JSSPs in production systems. This chapter critically reviews different types of MH algorithms and learning-based and RL applied on JSSPs.

3. The theory of GA as it pertains to scheduling is covered in Chapter 3. The chapter introduces the GA's fundamental ideas, terminology, and workings. It also emphasises the algorithm's uses, benefits, and potential drawbacks in scheduling and optimisation.

4. Chapter 4 describes the characteristics and applications of the advanced Ethnic selection GA (EGA) and the Novel Metaheuristic Parthenogenetic Algorithm (NMHPGA) developed in this research. The novel model is presented with a thorough analysis of its structure and capabilities. The performance of this unique algorithm in comparison to the fundamental GA is examined in a comparative analysis part that draws on simulated job shop scenarios.

5. Chapter 5 introduces the RL framework, explaining its fundamental ideas and operations. The chapter focuses on the advanced RL models and investigates how RL techniques are tailored for scheduling problems. Later in this chapter, NMHPGA and advanced RL model results are compared critically.

6. Chapter 6 explains the case study of the reheating model and explains the complexities of the furnace model. The design and functions of the furnace are briefly explained in this chapter. The novel algorithms expounded in the preceding Chapters (4 and 5) are tested on the furnace model to optimise its scheduling.

7. Chapter 7 summarises the salient outcomes of this research, identifying its major conclusions and contributions and their theoretical and practical ramifications. The chapter concludes with a forecast of future study areas, advancements, and

difficulties in manufacturing system optimisations. This framework seeks to lead readers through the research's multilayered arguments and conclusions by assuring coherence and clarity throughout each chapter.

Chapter 2

Literature Review

2.1 Introduction

This study's primary objective is to review the industrial applications of intelligent system algorithms in manufacturing and propose novel models to solve JSSPs. This chapter aims to critically examine different optimisation algorithms and investigate what algorithms will be established in this thesis to optimise the JSSPs. To achieve this, it reviews literature related to the field of JSSPs and examines the application of different intelligent systems proposed to solve them.

The structure of this chapter is as follows: Section 2.2 reviews definitions of JSSPs and establishes the description that will be used for this study. Section 2.3 discusses the different types of job shops and justifies the rationale for selecting those appropriate for this project. Section 2.4 reviews optimisation algorithms and scheduling methods pertinent to this research. Section 2.5 examines scheduling using learning-based methods and RL. Section 2.6 discusses the existing research gap, analyses different scheduling methods, and selects appropriate models to develop novel models for this research and finally Section 2.7 provides a summary of the chapter.

2.2 Job Shop Scheduling Problems

Job Shop Scheduling Problems involve scheduling jobs that require multiple operations on different machines or workstations[8]. The standard JSSP can be characterised as a collection of jobs to be carried out on various machines, whereby each job consists of many processes carried out in a predetermined order and on particular machines [52, 53]. JSSPs are categorised as Non-deterministic polynomial-time hardness (NP-hard) problems, which are considered the most complex class of problems in computational complexity theory. These are challenging to solve, and they usually pertain to sophisticated types of optimisation problems [10, 54, 55].

2.3 Job Shop Types

Production systems have several distinct job shops; some of the most popular types are reviewed in the subsequent sections.

2.3.1 Single-Machine Job Shop Scheduling

Single-machine job shop scheduling is the simplest form of JSSP, in which one workstation is used to complete all jobs, each of which consists of a sequence of operations with a specific duration. SM job shop scheduling aims to minimise the total completion time (i.e., the time taken to complete all jobs) [56].

SM scheduling and its learning effects have been the subject of numerous studies. For instance, Wang et al. [57] analysed an SM scheduling problem with the time-dependent learning effect to minimise the weighted sum of completion times and the maximum lateness. A job's computation time is a function of the total average processing time of all the other jobs scheduled up front in the job. In addition, Lee et al. [58] studied SM problems, including the learning effect and released time, to reduce the makespan. A branch-and-bound algorithm was also created to find the best answer [59, 60].

SM job shop scheduling can be solved using various algorithms, including the Shortest Processing Time (SPT) and the Earliest Due Date (EDD) algorithms. The SPT and

EDD algorithms, as their names imply, prioritise jobs with the fastest processing time and earliest due date (respectively) [56, 61]. One of the main advantages of SM job shop scheduling is its simplicity, whereby it requires less computational power and can be solved using basic algorithms. This characteristic makes it an attractive option for small-scale manufacturing industries with limited resources. However, it is hampered by its fundamental inefficiency, including long waiting times, resulting in low productivity and increased production costs. Furthermore, it is unsuitable for large-scale manufacturing industries requiring high-volume production. Therefore, SM scheduling is commonly used in small-scale manufacturing industries such as job and repair shops and maintenance workshops [56]. In contrast to SM, MM job shops have several workstations to be executed, which will be discussed later in this chapter.

2.3.2 Parallel Job Shop Scheduling

Another type of job shop scheduling is parallel job shop scheduling, which is used when multiple machines or workstations work to complete jobs simultaneously. In this type of scheduling, each job consists of a sequence of operations performed simultaneously on different machines or workstations. Parallel job shop scheduling aims to minimise the total completion time (i.e., the time taken to complete all jobs). It can be solved using various algorithms, such as the branch-and-bound algorithm, GA, and Simulated Annealing algorithm (SA). The branch-and-bound algorithm systematically explores all possible solutions to find the optimal solution. GA is based on natural selection and evolution and is used to find the best answers [56, 62].

One of the main advantages of parallel job shop scheduling is its efficiency because jobs are performed on different machines simultaneously [56]. It can significantly reduce waiting times, increasing productivity and lowering production costs. Furthermore, it suits large-scale manufacturing industries requiring high-volume production. However, its disadvantage is its complexity because it requires significant computational power and large-scale problems can be challenging. Furthermore, using multiple machines and workstations can increase the risk of machine breakdowns, leading to production delays

[56, 62].

2.3.3 Flexible Job Shop Scheduling Problem

In a practical model of JSSPs, a machine may perform more than one type of operation, and every procedure may then be performed on various machines, giving it greater flexibility than standard JSSP; this problem is referred to as FJSP [52, 53, 63]. FJSP is an extension of the traditional JSSP, which reduces constraints on machine selection by allowing each operation to be processed on many machines within its alternative machine set. Due to the addition of new choice content to the sequencing and the fact that it comprises more problems than JSSP, FJSPs are more difficult combinatorial optimisation problems. In FJSPs, the objectives are to solve two subproblems: operation sequencing and machine assignment. The goal of the FJSP is to obtain an allocation for each operation and define the order of operations on each machine to reduce the maximum processing workload time (makespan) [10, 38, 42, 63, 64].

The FJSP problem may be formulated as described below:

- In FJSP, there is a set of independent jobs $J = J_1, J_2, ..., J_n$.
- Each job J_i is formed by a sequence $O1, O2, ..., On_i$ of operations to be processed one after the other.
- There is a set $U = M_1, M_2, ..., M_m$ of machines as well.
- Each operation O_{ij} is executed among a subset $U_{ij} \subset U$ of compatible machines.
- Each operation has to be executed to complete the job.

  Each operation j of job i (O_{ij}) needs one machine out of a set of given machines M_{ij}.

In general, the following assumptions are considered in FJSSP:

1. Machines are available at time t = 0.
2. Jobs are available at time t = 0.

3. Each operation can be executed only by one machine at a time.

4. There are no precedence constraints in the execution of different jobs, and jobs are independent of each other.

5. Pre-emption of operations is not allowed; an action, once begun, cannot be interrupted.

6. Transportation time of jobs between available machines and time, which is required to set up the machine for processing the operations, are included in the processing time [10, 38, 42, 63, 64, 65, 66].

Many optimisation techniques have been devised to address FJSPs; some major studies pertaining to FJSPs are displayed in Table 2.1.

Table 2.1: Application of different optimisation methods on FJSPs.

Reference	Year	Method
[67]	2009	Hybrid approach combining PSO and TS
[68]	2010	Knowledge-based ant colony optimisation (ACO)
[69]	2011	GA
[70]	2017	Multi-agent-based PSO and a two-stage PSO
[71]	2017	Hybrid ABC based on TS
[72]	2018	Multi-objective evolutionary algorithm
[73]	2019	Mathematical modelling
[42]	2019	Novel MH method
[74]	2020	Effective search algorithm
[75]	2020	Reinforcement learning
[63]	2020	self-learning genetic algorithm
[76]	2021	Advanced GA
[77]	2021	Hybrid GA and TS
[78]	2022	Novel approach for FJSPs
[41]	2022	Improved GA
[19]	2022	A hybrid iterated greedy algorithm
[1]	2022	Hybrid scheduling measures

2.3.4 Flow Shop Scheduling

Flow shop scheduling is a particular example of job shop scheduling in which every operation must be done in a specific order. In specific flow shop scheduling, no machine can carry out more than one task at once, and an execution time is given for each job's operation. There has been much research into the flow shop scheduling problem. For example, Lin et al. [79] solved the flow shop scheduling problem for changeable processing parameters and low carbon emissions, thoroughly examining the effects of machines and scheduling levels on production throughput and the environment. Using renewable energy

and processing time as constraints, Wu et al. [80] developed a mathematical model of multi-objective optimal scheduling. Lu et al. [81] investigated energy consumption of the flow shop scheduling problem with the sequence-dependent setup, and controlled transit time was investigated. Figure 2.1 depicts some fundamental categories of job shop scheduling methods reviewed in this study.

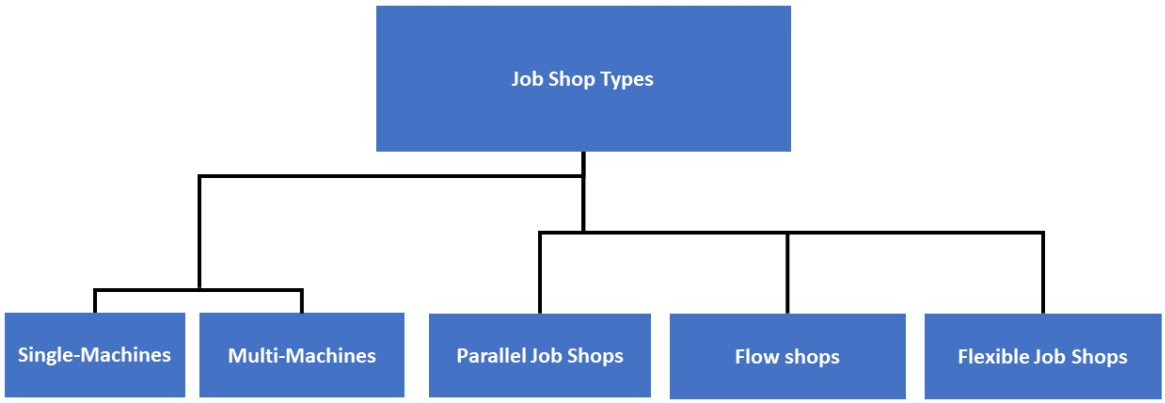


Figure 2.1: Job shop types.

JSSP, the primary production and manufacturing system's topic, has been discussed in the previous sections. Based on the above review of job shop types, this thesis focuses on FJSPs due to their flexibility in the scheduling process. Due to the features of FJSPs and their importance in production scheduling and manufacturing, much research has been done in this field [11]; however, there is still a need to improve and find advanced models. The review of prior research indicates that the central emphasis on improving JSSPs and introducing novel models mainly focuses on verifying and evaluating optimisation algorithms' performance. The following sections review various intelligent methods applications in the FJSPs (Section 2.4).

2.4 Optimisation Algorithms and Scheduling Methods

In relation to the JSSPs, there are four distinct activities that can be undertaken to address an optimisation problem effectively. The first step is identifying the problem's parameters. The optimisation problem can be categorised as either continuous or discrete based on the nature of its parameters. Continuous optimisation involves parameters that can involve any value within a given range, whereas discrete optimisation involves parameters that can only take on specific, distinct values, often representing countable or categorical choices. The second step consists of distinguishing between a constrained and an unconstrained optimisation problem by deciding what limitations should be placed on the parameters. Thirdly, the problem's goals must be carefully examined and incorporated. Optimisation problems may be classified as single-objective problems (having one objective) or multi-objective problems (having multiple objectives). Lastly, a compatible optimiser should be selected to solve the problem depending on the parameters, constraints, and number of objectives [32, 43].

Many optimisation challenges are inherently complicated, and traditional mathematical optimisation techniques cannot quickly identify optimal solutions. Current studies demonstrate a growing demand for optimisation techniques that are more accurate and efficient in terms of associated time and financial costs. Metaheuristic (MH) are one of these optimisation techniques [32, 82]. To optimise JSSPs, there are different scheduling methods. Figure 2.2 [83, 84] indicates popular scheduling methods (i.e., MH, heuristic, simulation, and mathematical programming). Each category includes different models using discrete and continuous problems; this thesis focuses on FJSPs and continuous problems. This section will discuss various scheduling methods based on MH algorithms, which are the most popular techniques for scheduling problems.

Figure 2.2: Scheduling methods.

2.4.1 Metaheuristic Methods and Optimisation

Glover [85] proposed the term "metaheuristic" in the 1980s, derived from "meta" (indicating something superseding the usual or natural bounds) and "heuristics" ("to discover"), referring to the methodology of solving problems within systems. MH optimisation seeks to identify the best solutions to defined problems while avoiding local optima. MHs are optimisation solution techniques incorporating higher-level strategies within search procedures [85, 86, 87].

2.4.2 Brief History of metaheuristic

The history of MH usage is categorised into five main periods [82, 85]. MH approaches were not formally introduced during the first period, prior to the 1940s, and only straightforward optimisation problems were resolved using these techniques. MH was first formally used during the second period, from 1940 to 1980. Many MHs were used for various applications during the third phase (1980 to 2000). This approach was effectively introduced in the fourth period, which spans from 2000 to the present. In the fifth phase, referred to as the scientific period, the creation of new MHs will become an increasingly specialised scientific endeavour [82, 85]. MHs fall into four primary categories: The first group is Evolutionary Algorithms (EAs), which include GAs [88], memetic algorithms, Differential Evolution (DE) [89], and evolution methods [90], all of which are based on biological evolution [82, 91]. Darwinian principles (i.e., natural selection) to solve scientific problems first emerged in the 1940s, before computers were developed [82].

The cooperative behaviour of decentralised and self-organised natural or artificial systems provides the basis for the second category of swarm intelligence-based algorithms, including ACO [92], PSO [93], ABC [94] and Cuckoo Search (CS)[95]. Physical principles drive the third group of algorithms, and the fourth category includes techniques based on human and animal behaviour [82].

All of these types of MHs offer promising models for FJSPs [39]. The following section reviews and discusses the application of the most popular MH algorithms in scheduling problems.

2.4.3 Evolutionary Algorithms

Evolutionary Algorithm (EA) is a fundamental population-based optimisation algorithm and a subset of the MH evolution account. EA uses biologically inspired mechanisms, including selection, recombination, mutation, and reproduction. The fitness function establishes the quality of the candidate solutions, which act as members of a population in the improvement issue. Population growth occurs after the relevant operators are applied repeatedly.

Computational complexity is a prohibited element in most real-world EA applications. The evaluation of the fitness function causes this computational complexity; simple EAs can frequently handle complex models. Application areas for EAs include planning, design, and simulation. Different types of EAs, including GA and DE, are discussed in the subsections [32, 96].

2.4.4 Genetic Algorithm

Among EAs, GA is a popular optimiser scheduling system. The first GA was created by John Holland in 1975 [90, 97]. It is a well-known optimisation technique that uses the theories of evolution and natural selection to address challenging optimisation problems. It begins with a random initial population in which each member is referred to as a chromosome (potential solution). Evaluating an individual's performance using an objective function initiates the algorithm's primary iterative cycle. Higher fitness values are given a higher likelihood of selection for creating a new generation (offspring) than lower fitness values since they represent better solutions. The most promising individuals are more likely to be chosen for reproduction, while individuals with low fitness scores are commensurately eliminated. As a result, the performance of the new generation of individuals is expected to improve. The selected individuals are then recombined to create offspring by sharing information. After reproduction, mutation further messes with the progeny [98]. A new generation will subsequently emerge based on the fitness of these new offspring. This selection, reproduction, mutation, and evaluation cycle continues until the optimisation requirement is met.

GAs have been widely employed to address challenging JSSPs [90]. GAs encourage solution space exploration through crossover and mutation operators [90]. Furthermore, GAs can become trapped in local optimums and are susceptible to premature convergence. They also need the parameters for population size, mutation rate, and crossover rate to be carefully tuned. The size and complexity of the task can significantly lengthen the computing time [99]. The following are some salient features of GA:

1. Optimisation: Natural selection and genetics are the foundations of GAs, making them effective at finding the best solutions in vast and complex problem spaces, such as those seen in scheduling difficulties [98].

2. Adaptability: GAs are suitable for flexible scheduling problems where conditions can change over time because they can handle changes in the problem environment [98]. Selection is the core part of the GA process, and there are different types of selections, such as elitism, fittest, sexual, tournament, and roulette wheel selection. Some essential types of selections, which are the main topic of this thesis, are discussed in more detail in Chapter 3. Table 2.2 summarises key literature published from 1996 to 2003 concerning different GA selection types applied in optimising FJSPs, which is the principal aim of this research. The review demonstrates the potential of GAs to address FJSPs by employing a range of selection types, as displayed in Table 2.2.

Table 2.2: Application of different types of selections of GA on FJSPs.

Reference	Year	Algorithm	Selection types
[100]	1996	GA	Ageing
[101]	2001	GA	Fittest
[102]	2003	GA	Fittest
[103]	2003	GA	Sexual selection
[104]	2006	GA	Elitism
[105]	2005	GA	Elitism
[7]	2008	GA	Tournament
[106]	2008	GA + TS	Tournament selection

Continued on next page

Table 2.2 (continued)

Ref	Year	Algorithm	Selection
[107]	2009	GA	Linear ranking
[108]	2010	GA	Random
[109]	2010	PSO + GA	Hybrid
[110]	2010	GA + TS	Hybrid
[111]	2010	Parthenogenetic	Roulette wheel
[112]	2011	GA	Tournament selection
[113]	2011	GA + SA	Roulette wheel
[114]	2011	GA + ACO	Linear scaling, stochastic universal sampling
[115]	2012	GA	Elitisim
[116]	2012	GA + PSO	Roulette wheel
[117]	2014	GA	Tournament selection
[118]	2014	GA	Roulette wheel
[119]	2015	GA	Roulette wheel
[120]	2015	GA	Roulette wheel
[121]	2015	GA + TS	Tournament selection
[122]	2015	Improved Parthenogenetic	Greedy selection
[123]	2016	Neighbourhood GA + TS	Fitness neighbourhood selection operator
[124]	2016	A Heuristics-Based Parthenogenetic	Roulette wheel selection
[125]	2017	GA	Tournament selection
[126]	2017	A hybrid GA	Elitism
[127]	2018	Parthenogenetic	Parthenogenetic

Continued on next page

Table 2.2 (continued)

Ref	Year	Algorithm	Selection
[128]	2018	List-scheduling-based multiobjective parthenogenetic (LS-MPGA)	Pareto-Ranking and Selection
[129]	2019	GA	Tournament selection
[130]	2019	RCGA	Roulette wheel
[131]	2020	GA	Tournament selection
[132]	2020	Parthenogenetic algorithm	Parthenogenetic
[133]	2022	Hybrid immune GA with TS	Tournament, Roulette-wheel, linear-rank
[131]	2020	IGA	Maximum priority selection method for remaining processing time
[134]	2021	Learning interactive GA	Edge selection encoding
[135]	2021	Adaptive GA Based on Individual Similarity	Binary tournament
[136]	2021	Parthenogenetic	Parthenogenetic
[137]	2022	GIFA	Ranking based on Fitness
[138]	2022	Taguchi method	GA and Parthenogenetic
[41]	2022	MILP and IGA	Two-vector encoding scheme to represent the configuration selection and operation sequencing
[139]	2022	Elite GA	Binary tournament selection and the elitism method

Continued on next page

Table 2.2 (continued)

Ref	Year	Algorithm	Selection
[140]	2023	HGA	A hybrid selection of tournament selection and the elite selection
[141]	2023	Improved GA with a population diversity check method	Elitist selection and the binary tournament selection
[142]	2023	MGA multi-start GA	Biased on random parthenogenetic algorithm

2.4.5 Particle Swarm Optimisation

Particle Swarm Optimisation (PSO) is a well-known swarm intelligence model created by Eberhart and Kennedy [93], based on the social behaviour of swarming birds and fish. Swarm intelligence models refer to computational algorithms which take inspiration from the collective behaviour observed in natural systems, such as the coordinated movements of bird flocks or the organised activities of ant colonies. These models employ basic principles for individual agents to engage with their surroundings and one another, resulting in the formation of intricate, intelligent global patterns that are advantageous for addressing optimisation challenges. In PSO, the agent known as particles represents the candidate solution to the optimisation problem. Position and velocity describe particles which are free to move about the search space.

In the startup phase of PSO, each particle is given a randomly chosen initial position and velocity. The following iteration will change the particle's position depending on its rate [93]. By comparing the particle's present fitness to both its past best placements and its neighbour's best solution, the PSO technique determines the particle's new position. To solve JSSPs, PSO is frequently taken into consideration by researchers in hybridisation with other MH algorithms, including TS [143], ACO [144], harmony search (HE) [145],

and CS [146].

Unlike GA, PSO lacks a systematic calculation approach and evolutionary operators like crossover, selection, and mutation. Consequently, PSO models are more straightforward to build and have fewer parameters to alter [83]. Fontes et al. [147] proposed a hybrid PS and SA algorithm for the JSSPs. PSO, which has unique advantages for resolving issues with FJSP, has a quick search speed and a limited number of parameters [148]. Furthermore, in another work by [149], the scheduling of embedded real-time production systems was categorised as an FJSP and a distributed PSO approach was proposed as a solution; in this work, the elements affecting speed, position, and learning remained the same. Notably, the PSO algorithm's ability to balance local search with global exploration largely depends on the control settings that the algorithm uses [150].

2.4.6 Differential Evolution

Differential Evolution (DE) is a global optimisation algorithm developed by Storn and Price [88]. It can be considered an EA, which solves optimisation problems by evolving a population of candidate solutions using biology-inspired crossover, mutation, and selection operations [88]. The DE approach provides better results for multi-dimensional optimisation problems, including neural network learning, than other EAs, such as GA. In addition, DE methods have been suggested to solve Holland's primary problem of poor local search performance [90].

2.4.7 Simulated Annealing

Simulated Annealing (SA) is a local and random search MH algorithm [95]. This algorithm mimics the annealing process in metallurgy. During an annealing process, a metal is heated to a specified temperature and then cools and freezes at a determined cooling rate into a crystalline state with minimum energy to avoid defects. The optimisation procedures of SA always start by generating an initial solution space, after which a proper initial value of the annealing parameter is set. In each iteration step, the algorithm determines whether the neighbouring solution is better or worse than the current solution.

The adjacent solution will be accepted if it improves the cost function value, and only specific worse points will be taken based on an acceptance function. After that, the temperature decreases slowly, and the probability of accepting a worse solution is the same. Lastly, the iteration ends while a stopping criterion is met.

A significant advantage of SA is that it has high flexibility and robustness and can approach global optimality compared to other local search methods [95]. Its key drawback is that the computation time tends to grow dramatically with the size of the problem. Furthermore, picking a suitable cooling schedule is a crucial SA parameter, but it can be difficult and significantly affect the algorithm's performance. Zhang and Wu [151] proposed a functional SA model for JSSPs.

The selection operations used by GA and DE algorithms differ markedly [88]. One of the advantages of the DE approach is its simplicity because the search process is only controlled by three input parameters: size of the population, scale factor, and crossover parameter. However, its efficiency depends on the control parameter [152]. In addition, obtaining the optimum operations, such as crossover and mutation in the DE approach, is usually time-consuming [152]. Different variants of DE have been implemented in various industrial applications for better performance [153, 154, 155].

2.4.8 Tabu Search

Glover created Tabu Search (TS), an MH algorithm to solve optimisation problems [85]. This method employs responsive exploration and memory structures to find an optimisation solution. Memory structures like the tabu list can effectively search the solution space by specifying the visited solutions. Responsive exploration provides a fundamentally enhanced technique employing the search history in TS.

The tabu list prevents the seen solutions and directs the search towards the undiscovered solution space for possible solutions. It stores the keys studied throughout the search space. TS effectively solves larger and more complex problems. Numerous control parameters should be established, and the parameter setting significantly impacts achieving a global optimum [85]. TS has been applied in hybridisations with other optimisation

techniques for scheduling, including SA [156, 157] and ABC [158].

2.4.9 Artificial Bee Colony Algorithm

The Artificial Bee Colony (ABC) algorithm is an MH algorithm based on bees' foraging behaviour [159]. A food source's position and nectar content symbolise a potential solution to a problem and the fitness that goes along with it. The program uses three types of artificial bees: employed, bystanders, and scout bees. The available food and the number of bees at work are equal. All employed bees belong to groups that use dancing to find, transport, and communicate information about food sources. Scout and observer bees are always looking for new food sources. Scout bees only blindly explore the search area, while spectator bees can gather information from employed bees by remaining in the dance area while searching for a food source [160].

The ABC method solves issues in various applications, such as improving wireless sensor networks, maximising heat transfer rates, power plant optimisation, machining process improvement, and JSSPs [155]. Compared to other MH algorithms, the ABC method has been shown to efficiently tackle engineering problems with high dimensionality [161]. To improve performance, the ABC algorithm has also been combined with other algorithms, including the hill-climbing [162], ACO [163], and DE [164, 165] algorithms. As a result, in recent years, JSSP has been solved using ABC and its advancement [166] Problems including no-wait constraints [167, 168], rescheduling strategy [71] or FSSP [169] having deteriorated, or a hybrid. However, due to the search process, ABC has shortcomings in solving the scheduling problems [170].

2.4.10 Harmony Search

The intriguing Harmony Search (HS) algorithm is designed to overcome problems according to the inspirations of musical performances [171, 172]. Musicians constantly seek the ideal harmony, and the search process in HS is modelled on their behaviour while seeking the optimal balance to improve their melodies. The HS iteratively improves the solutions during optimisation to optimise the objective function. Initialising the HS parameter,

which includes the amount of harmony memory, harmony considering rate, and pitch adjusting rate, is the first step in the HS algorithm technique.

Three different iterative techniques must be used to improvise and update the new solution to reach the best outcome [171]: (a) Consideration of memory: to use HM, a new solution is created from an old one; (b) Considering memory, a new solution is devised by slightly changing the pitch; (c) Randomisation process: a new, improvised solution is constructed. The subsequent HM acceptance rate determines its strength. These three factors can thus be used to achieve good performance in HS with a balance of intensification and diversification.

HS is straightforward to implement because it does not involve complicated calculations. The HS method has also been modified to enhance its convergence capabilities. This approach is frequently used in many applications, including scheduling in manufacturing processes [173], and medical applications [174]. Owing to these benefits, the HS has been effectively adapted to address many optimisation issues in various domains [175]. Besides the benefits of HS for solving JSSPs, HS can experience stagnation throughout the optimisation process, resulting in less-than-ideal solutions [176]. The algorithm's complexity rises due to the requirement for precise parameter adjustment to produce the best results [177].

2.4.11 Cuckoo Search

According to Yang and Deb [178], Cuckoo Search (CS) optimisation MH algorithm imitates the cuckoo's breeding habits, which are defined by laying fertilised eggs to be hatched in the nests of other birds. Cuckoos typically prefer newly produced nests to enhance the likelihood of hatching eggs; consequently, the host birds will care for the cuckoo offspring. To increase their access to food delivered by the foster parents, the cuckoo chick pushes the native eggs out of the nest after hatching. The host bird either destroys or quits the nest and creates a new one elsewhere if it learns that the egg that has been hatched is alien [178]. As a result, the cuckoo chick has a unique ability to imitate the look and sound of its host bird to maximise its reproductive success and prevent abandonment.

To solve the JSSP, Singh et al. [179] suggested CSO with several individual enhancement schemes. In terms of finding the global optimum, CS has a faster convergence speed in finding optimal solutions. Compared to other algorithms, it simplifies the tuning process because there are fewer parameters to change [178, 180]. CS excels at solving continuous or combinatorial problems, making it adaptable to various JSSPs. However, it is exposed to premature convergence in areas with complicated problems [181].

2.4.12 Firefly Algorithm

Firefly Algorithm (FA) is an MH algorithm inspired by the flashing behaviour of fireflies [182]. A firefly is a species of bug that may emit natural light to entice a mate or ward off predators. The FA generally follows three guidelines: 1. Since fireflies are unisex, they usually gravitate toward the brighter and more appealing mating partner. 2. Since air absorbs light, the attraction is inversely related to brightness and diminishes as the distance between two fireflies grows. 3. The brightness of a firefly will depend on the geography of the objective function.

FA is a simpler algorithm compared to other swarm-based algorithms. Additionally, the benefits of autonomous subdivision have improved its effectiveness [182]. As documented in the literature [182, 183, 184, 185], this approach has been used in numerous applications since the inception of the firefly algorithm. Studies have shown that FA outperforms algorithms like ABC and PSO and can obtain the world's best solutions [183].

FA has been used effectively in a variety of applications, but in contrast to other MHs, not much study has been done on applying the FA with regard to the FJSP [186], in the study by [186], an integrated approach using FA has been proposed to solve FJSPs. In the field of scheduling, an FA was provided by [187]to minimise makespan in the workflow scheduling problem with deadline constraints. Recently, [188] applied improved FA for FJSPs.

2.4.13 Gravitational Search Algorithm

The Gravitational Search Algorithm (GSA) is a physical-based heuristic search algorithm that draws inspiration from the Newtonian principle of mass interaction [189]. It is widely used to solve nonlinear optimisation problems. According to Newton's gravity equation, any two particles will be attracted to one another by a gravitational force [190]. The gravitational force varies inversely with the square distance between the particles and directly with the product of the particle masses. A group of agents in search space are drawn together by gravity in GSA. While their performance is correlated with their packs, these agents behave as objects. Due to the pull of gravity, all things in the search space gravitate toward those with heavier masses [83].

Researchers' interest in this algorithm is growing because it can produce better results than other nature-inspired algorithms and identify solutions close to the global optimum. In various applications, it is useful when combined with other computational techniques to overcome its slow convergence and searching speed [83].

On the basis of current knowledge, the GSA has abundant reported applications for scheduling problems. For instance, [191] introduced a highly efficient GSA for addressing the study's permutation flow shop scheduling problem. In another study, [192] proposed an enhanced GSA for addressing the scheduling problem in hybrid flow shop environments with parallel machines. Furthermore, [193] suggested discrete GSA for a kind of flow shop problem with total flow time minimisation.

2.4.14 Ant Colony Optimisation

Ant Colony Optimisation (ACO) is another effective swarm intelligence method, first put forth to categorise problems in the medical industry and achieve success in continuous optimisation[194]. The ACO algorithm's application to scheduling problems, such as SM scheduling problems, was the subject of preliminary studies [195] or JSSP in general [196]. Researchers have employed strategies for combining the ACO algorithm with specific JSSPs, such as local searches[197, 198].

JSSPs have been successfully solved using ACO [199]. Heuristics customised to a

particular situation can be incorporated into ACO algorithms to improve the search process [200]. ACO's distributed computing model makes parallel processing possible, speeding up the problem-solving process [201]. One of the disadvantages of ACO is that the algorithm needs to be fine-tuned to perform well because it is highly sensitive to parameter adjustments [202]. ACO performance degrades with more significant problems [199].

2.5 Scheduling Using Learning-Based Methods and Reinforcement Learning

One of the promising models for JSSPs and FJSPs is RL [8], wherein agents make decisions while receiving little input and each decision is rewarded or penalised based on a given reward policy; RL is the subfield of machine learning (ML) wherein the agent aims to maximise the reward by starting with arbitrary trials [203]. A Markov decision process has been used to model the primary reinforcement [203]. RL is frequently employed in autonomous robotic operation manufacturing [204]; furthermore, Q-learning and deep learning are often used to create RL agents [205].

A growing number of RL techniques are being used to improve JSSPs. For instance, Shahrabi et al. [206] created an RL with the Q-factor method to solve JSSP. Shen et al. [207] suggested a multi-objective dynamic software project scheduling based on Q-learning. Chen et al. [208] proposed a self-learning GA for addressing the FJSP. This method incorporates both the State-Action-Reward-State-Action (SARSA) algorithm and Q-learning within the self-learning framework. Shi et al. [209] adopted a DRL strategy for intelligently scheduling discrete automated production lines. In another work, Chen et al. [63] suggested using a self-learning genetic algorithm (SLGA) based on RL to reduce the FJSP makespan.

2.6 Discussion

2.6.1 Identifying the Knowledge Gap

Various intelligent optimisation methodologies have been discussed in this chapter, including the GA, PSO, SA, DE, TS, ABC, CS, GSA, ACO, and RL. Having reviewed the relevant literature on JSSPs and, more specifically FJSPs, it can be noticed that there is a research gap indicating the need for more clarity on the effectiveness of intelligent algorithms to optimise job shop scheduling. This thesis gives particular attention to FJSPs due to their flexibility. Numerous algorithms have been used to solve the FJSPs. However, the effectiveness and quality of the solutions can be improved by parameter modifications in the algorithms. The key features of the knowledge gap in this area are outlined below.

1. Scheduling using MH algorithms: MH algorithms are promising methods for locating excellent answers to optimisation issues. JSSPs are ideally suited to them, mainly when the solution space is large and complex. However, depending on the problem's features and the particular algorithm's settings, the efficacy of MH can vary dramatically [28, 31, 32, 40, 48, 210, 211].

2. Hybrid models: There are numerous methods of solving scheduling problems, each with benefits and drawbacks. Additionally, very little research and development is being done on hybrid models, which combine the benefits of many methods to improve performance overall. The following section compares the advantages and disadvantages of some popular MHs discussed earlier in this chapter for job shop scheduling.

Compared to ACO and GA, CS and HS could be more straightforward to deploy and require less parameter adjustment [171, 178]. Unlike the more recent CS and HS, GA and ACO have been extensively used for job shop schedulings and are backed by substantial research outlining strategies to improve their performance [90, 199]. GA and ACO are known to be computationally expensive because of their complicated processes like crossover and mutation [99, 199].

Careful parameter tuning is necessary for GA and ACO to work properly [202]. Com-

paring CS to other algorithms, the tuning procedure is more straightforward because there are fewer parameters to change [180]. Additionally, HS needs precise parameter tweaking [177]. Moving on to PSO, it has several advantages over other algorithms, such as being easy to create, fast on computers, and requiring fewer changes to its parameters [93]. However, it frequently becomes trapped in local optima, and the quality of the solutions may decline as the size of the problem grows [148]. Regarding DE, like many EAs, they tend to be slow, particularly for challenging problems. Additionally, it can be difficult to determine the ideal control parameters for DE [152].

3. Scheduling using reinforcement learning models: RL effectively solves job shop schedulings. The most significant advantage of developing RL models is their ability to enhance the system's performance without using many EA functions [8]. It is noted that the RL technique in the literature above solves fewer studies for FJSP. As a result, it is possible to determine the best FJSP scheme via RL, which is also one of this thesis's key innovations.

2.6.2 Implications

RL and GA have demonstrated promising results in the context of FJSPs. Consequently, this research presents advanced hybrid GA and RL models. The efficacy and benefits of the new and advanced methods for solving FJSP are shown by extensive computer testing and comparisons. As a result, they can be competitive algorithms for solving FJSPs. The advanced models are tested on FJSPs in the following chapters.

2.7 Summary

This chapter reviewed different types of JSSPs, including SM job shops, parallel job shop scheduling, FJSPs and flow shop schedulings. JSSPs are typically categorised as NP-hard problems and have been demonstrated to be inherently complex, related to many constraints, a set of objectives, and the scope of the search area. Later in this chapter, the most popular approaches to addressing various scheduling problems have been reviewed.

FJSPs are more flexible than other scheduling problems due to the enormous search space; consequently, this thesis's primary focus is optimising FJSPs. Based on research in the literature and comparing several hybrid algorithmic properties, advanced hybrid GA and RL models are designed to optimise FJSPs and a furnace model.

The new application of RL techniques is also employed to test and compare the outcomes with the established hybrid GA. More specifically, the advanced GA algorithm utilises a combination of various selections to improve the convergence speed. The next chapter will review the Principles of GAs for scheduling. The suggested techniques and simulation results are expounded in detail in Chapters 4 and 5; these techniques are proposed to reduce the makespan of the schedules.

Chapter 3

Principles of Genetic Algorithm for Scheduling

3.1 Introduction

This chapter reviews the principles of GA, explaining its underlying theoretical basis and explanations of different types of selections. It also points out the benefits and limitations of GA and discusses its various stages, including initialisation, selection, crossover and mutation. It first introduces such algorithms in Section 3.2, and Sections 3.3 and 3.4 discuss GA theory and fundamental operators. Moreover, Section 3.5 discusses the simulated benchmarks and essential GAs used in this chapter to create comparable reference results for the upcoming chapters. More precisely, in this chapter, we get the results from standard GAs to have comparable references for Chapters 4 and 5 in order to test the performance of the established models. Lastly, Section 3.6 provides a summary of this chapter.

3.2 Genetic Algorithm and Biological Background

Genetic algorithm is the most widely used method in research on evolutionary computation, whereby each position in a string represents a particular characteristic of an individual [98, 212]. Individuals are inside the string, and the population is a group of

solutions. Genetics is the field of study that examines the processes underlying a species' similarities and differences. GAs now borrow ideas from natural evolution [98, 212]. The following are the key terms used to describe a species' biological history.

3.2.1 Reproduction and Natural Selection

The idea of survival of the fittest, which entails the preservation of favourable variations and the rejection of detrimental ones, is the foundation of the theory of evolution by natural selection. Within a species, as well as among the offspring of the same parents, variations can occur. Those with beneficial features have a higher chance of living and passing on their genes because they have improved access to resources and increased survivability and reproduction. Natural selection, therefore, plays a crucial role in determining which traits persist over time in a species [98, 212].

3.3 Genetic Algorithm in Brief

Rechenberg first discussed evolutionary computers in the 1960s in his article named Evolution Strategies. Several researchers later developed this concept, including John Holland, the inventor of GAs, who expanded on this concept in his 1975 book "Adaptation in Natural and Artificial Systems." He presented GA as a heuristic approach based on "survival of the fittest," which was a helpful tool for search and optimisation issues [98, 212].

3.3.1 Search Space

In many applications, the objective is to select the best option from a predetermined list of workable options, whereby all possible solutions are set by the search space. A fitness function based on the problem statement can be used to assess each point in the search space as a possible solution. GAs are used to find the best solution, typically requiring the minimum objective function, from a set of potential solutions represented by various points in the search space [98, 212].

The GA brings up a few significant features. It is, first and foremost, a stochastic

algorithm and depends heavily on chance. Reproduction and selection both require randomness. The fact that GAs constantly consider a population of solutions is crucial, and many benefits are associated with keeping many solutions in memory throughout each iteration. Such features make GA a highly effective tool for optimisation. The straightforward technique of random search involves choosing solutions at random, determining their fitness, and then exploring the search space [98, 212].

3.3.2 Benefits and Limitations of Genetic Algorithm

Advantages of GA:

1. The solution area is large.
2. Simple to find the global optimum.
3. The problem is a multi-objective function.
4. Easily adaptable to different problems.
5. Handles huge, complex search spaces with ease.

Limitations:

1. Identifying fitness function.
2. Premature convergence in some cases.
3. Choosing various parameters and functions such as population size, mutation and cross over rates and the selection method.

[98, 212].

3.4 Fundamental Operators of Genetic Algorithms

The two main components of a GA are populations and individuals; in contrast to the population, the group of individuals involved in the search process, an individual is a

single solution. The fitness represents the quality of the solution and how near the chromosome is to the ideal one. A population is a collection of individuals being tested. The formation of the initial population and the size of the population are two crucial components of the population that are highlighted in GAs. The population size refers to the total number of individuals determined randomly. Several approaches may be used based on the problem being solved, as described below, and these two considerations substantially impact the algorithm's success [98, 212].

3.4.1 Search Strategies

The search begins by initialising the population and breeding new individuals until the termination condition is satisfied. There is always a chance that the following search iteration will produce a more desirable result. The GA's centre is where the breeding process occurs, whereby the selection process produces new, ideally more fit individuals. There are three phases in the breeding cycle, as discussed in the following subsections: selecting parents, mating the selected parents to produce offspring, and substituting the old population with the new individuals [98, 212]

3.4.2 Selection

Two parents from the population are selected for crossover. Subsequently, the next stage is to decide how to choose or pick individuals within the population who will produce offspring for the following generation and how many offspring each will produce. The greater the fitness function, the more likely a candidate will be chosen. Variation from crossover and mutation must be balanced with selection; slow evolution will result from limited selection, while over-selection will conversely result in suboptimal, highly fit individuals dominating the population. The following subsections discuss the commonly used selection techniques in this context [212, 213, 214].

3.4.2.1 Roulette Wheel Selection Genetic Algorithm

Roulette selection is one of the classic GA selection methods. The proportionate reproductive operator, which selects a string from the mating pool with a probability proportional to fitness, is the most widely used reproduction operator. The roulette selection concept is a linear search across a roulette wheel with slots weighted according to a candidate solution's fitness values. The population refers to the potential solutions in the optimisations, and the population is increased until the desired value is attained. This is a somewhat effective selection method. In the roulette procedure, an individual's expected value is calculated by dividing their fitness level by the population's actual fitness level, whereby each candidate solution is given a slice of the roulette wheel whose size corresponds to the individual fitness level. N times are spun on the wheel, where N is the total population. The candidate solution chosen as a potential parent for the following generation is under the wheel's marker for each spin[98, 212, 213, 215].

3.4.3 Stochastic Selection Genetic Algorithm

Stochastic selection is a popular technique in GAs whereby individuals are randomly chosen depending on their fitness; it has been widely used as an alternative to roulette wheel selection. This is so that each candidate solution is given an equal place on the roulette wheel based on fitness, using a set random number [98, 212, 213, 215].

3.4.4 Sexual Selection Genetic Algorithm

An enhanced variant of the GA is sexual GA (SGA). Choosing parent chromosomes for reproduction is the foundation of the selection process in traditional GA. SGA is motivated by the notion of masculine endeavour, and female choice is based on an algorithm that divides the population into males and females. Each female is used as the basis for the selection process, and several males compete to be chosen for reproduction. The final processes resemble those of traditional GA. SGA optimisation method is more advanced than conventional GA, which uses a single-selection approach to choose parent chromosomes for reproduction [103, 216, 217, 218]. Algorithm 1 shows the pseudo-code for SGA

[219, 220].

Algorithm 1: Sexual GA

Input: Population size, Fitness function

Output: Best solution

Randomly Initialise population;

Evolution population fitness;

Divided the individuals into males and females;

while *Stopping criteria not met* **do**
    Reproduction using the idea of male effort and female choice

end

Return the best solution;

3.4.5 Ageing Selection Genetic Algorithm

Ageing Selection Genetic Algorithm (AGA) is another modified form of conventional GA, with the goal of performance enhancement. To produce a better generation or population that approximates the ideal answer, GA acts on a population of potential solutions based on the survival of the fittest principle. AGA differs from traditional GAs in that it more closely mimics the natural genetic system and considers how an individual's age impacts performance. When a new candidate solution enters a population, their age is immediately presumed to be zero, and each candidate solution's age increases by one with each repetition. Young and old people are considered less fit than adults, just as in a natural genetic system [98].

According to this strategy, individuals who have been a part of the population for a long time have a reduced probability of being chosen for mating. In contrast, younger people have a higher possibility. If a proportional selection method is utilised, a particular individual that becomes fit continues to have opportunities to generate offspring up to the algorithm's conclusion, increasing the likelihood of producing similar offspring. Less fit individuals typically do not survive, whereas the more fit ones usually do [100, 221, 222].

Algorithm 2: Aging GA

Input: Population size, Fitness function

Output: Best solution

Randomly Initialise population;

while *Stopping criteria not met* **do**

1. Compute the age factor for each
2. Adjust fitness according to age factor (older individuals have less chance)
3. Select the best individuals for reproduction (based on adjusted fitness)
4. Breed new generation through crossover and mutation
5. Compute fitness of new generation

end

return the best solution;

3.4.6 Crossover and Recombination

Crossover is the process of creating a child from two-parent solutions. The selection (reproduction) process adds better individuals to the population. The mating pool is given a crossover operator to produce better progeny. A recombination operator called crossover goes through three steps [212].

1. A pair of individual strings is randomly chosen for mating through the reproduction operator.
2. A cross-site is randomly selected along the string length.
3. Lastly, the position values are exchanged between the two strings after the cross-site.

There are different crossover types, such as single- and double-point crossover. The following subsections describe some popular types of crossovers.

3.4.6.1 Single Point Crossover

The conventional GA employs single-point crossover, in which the matching parts of the two mating chromosomes are cut once at each location and are subsequently switched. In

this case, bits close to the cross-sites are swapped after a random cross-site or crossover point is chosen along the length of the matched strings. If the right location is chosen, good parents can combine to produce superior offspring; otherwise, the string quality will be greatly limited [212].

3.4.6.2 Multi Point Crossover

Besides single-point crossover, various crossover algorithms have been developed, often involving multiple cut points. It should be mentioned that increasing the number of crossover points reduces the effectiveness of the GA: increasing the number of crossover locations has the drawback of making it more likely that construction blocks may be disturbed while having more crossing points can lead to a more exhaustive search of the study area. Two crossover points are selected in a two-point crossover, and the content between them is shared between two mated parents [212].

Parent 1	1	0	1	1	0	0	1	0
Parent 2	1	0	1	0	1	1	1	1
Child 1	1	0	1	1	0	1	1	1
Child 2	1	0	1	0	1	0	1	0
Parent 1	1	1	0	1	1	0	1	0
Parent 2	0	1	1	0	1	1	0	0
Child 1	1	1	0	0	1	1	1	0
Child 2	0	1	1	1	1	0	0	0

The parents swap the contents between these points to create new offspring for mating in the following generation. One-point crossover, which divides two chromosomes into two pieces and splices them together to form new ones, was the first method used by GAs [212, 213, 214].

3.4.7 Mutation

The strings are subjected to mutation after crossing. Thanks to mutation, the algorithm cannot become stuck at a local minimum, which can restore lost genetic material or randomly disrupt genetic information. Mutation represents a form of protection against the permanent loss of genetic material. Traditionally, the mutation is understood to be a straightforward search operator designed to aid in investigating the entire search space. In contrast, crossover is supposed to take advantage of the existing solution, seeking to identify better ones. It randomly alters genetic building blocks, introducing new genetic structures into the population [212, 213, 214].

The likelihood of a mutation determines how frequently specific chromosomal regions will change. Without a mutation, children are produced immediately after a crossover (or are directly duplicated) without any modifications. When a mutation occurs, a chromosome's one or more sections are altered. If the likelihood of a mutation is 100%, the entire chromosome is limited; if it is 0%, nothing is changed. There shouldn't be too many mutations because then GA will switch to random search [212, 213, 214].

3.4.8 Search Termination and Convergence Criteria

The stopping conditions for GAs are:

1. Maximum generations: The algorithm terminates when the specified number of generations has been reached.

2. Elapsed time: The algorithm terminates when a specified time has elapsed. However, if the maximum number of generations has already been reached, the algorithm will terminate regardless of the elapsed time.

3. No change in fitness: The algorithm terminates if the population's best fitness remains unchanged for several generations. However, if the maximum number of generations has already been reached, the algorithm will terminate regardless of the number of generations with no change in fitness.

The more-or-less standard procedure for running the standard GA algorithm is as follows:

Algorithm 3: Standard GA

Input: Population size, Fitness function

Output: Best solution

randomly generate population;

while *Stopping criteria not met* **do**

Select parents (using fitness function);

Crossover parent chromosomes;

Mutate offspring chromosomes;

Add offspring back into the population; Implement elitism (select parents);

end

Return the best solution;

3.5 Job Shops Case Studies

This section applies four standard GAs based on STGA, RGA, SGA, and AGA to optimise the sequencing orders of the simulated job shops that are modelled in MATLAB. These four GAs are based on the basic GA. Still, they differ in selection criteria, as explained previously (i.e., STGA employs stochastic selection, RGA employs roulette wheel selection, SGA uses sexual selection type, and AGA exploits the ageing selection method).

3.5.1 Simulated Benchmarks

Single-machine and multi-machine (SM and MM) benchmarks are used with regard to three basic flexible simulated job shops, with early arrival (earliness) and delay (tardiness) penalties. The job shop examples are SM, MM_1 with similar scale of earliness and tardiness penalties, and MM_2 with a different scale of earliness and tardiness penalties,

The Total Penalty is given by equation 3.1.

$$\text{Total Penalty}(n) = \text{Earliness} \times 0.3 + \text{Tardiness} \times 0.7 \tag{3.1}$$

The benchmark sequencing orders are tested and optimised using the above GAs to have primary results as a reference to be compared with the performance of advanced models whose designs are presented in the following chapters. In the classical $n \times m$ job shop type, there are n jobs J_1, J_2 of $, \ldots, J_n$ of different processing times scheduled on m machines; each job has a set of operations $O_1, O_2, \ldots, O_n$ processing in a specific order [98, 212, 223, 224].In the simulated job shops in this thesis, each job has only 1 operation because we aim to make the job shops as straightforward as possible to compare the performance of the algorithms.

3.5.2 Job Shop Parameters with Earliness and Tardiness Penalties

The following is a description of the JSSP. There exists a collection of m machines, denoted as $M = \{M_1, M_2, \ldots, M_m\}$, and a collection of n jobs, denoted as $J = \{J_1, J_2, \ldots, J_n\}$ which need to undergo processing. Let i denote the index for tasks and k denote the index for machines. Specifically, we have i ranging from 1 to n, and k ranging from 1 to m. Every job J_i necessitates a collection of n_i operations, $O_i = \{O_{i1}, O_{i2}, \ldots, O_{in_i}\}$, which must be executed in a sequential order. Let j denote the index for operations, where j takes on values from 1 to n_i. The operation O_{ij} is executed on a designated machine $M(O_{ij}) \in M$, with a processing time denoted by p_i. Let M_k be a machine in the set M. The notation $O(M_k)$ denotes the set of all operations that are executed on M_k.

Every operation O_{ij} is associated with a specific due date d_{ij}, and any completion of the operation that is either early or late will result in a penalty that is directly proportionate to the extent of deviation from d_{ij}. Every operation O_{ij} is associated with two penalty coefficients, α_{ij} and β_{ij}, which are utilised to impose penalties for its early and tardy completion, respectively.

Let c_{ij} denote the scheduled completion time of operation O_{ij}, e_{ij} represent its earliness, and t_{ij} denote its tardiness and $e_{ij} = \max(0, d_{ij} - c_{ij})$ and $t_{ij} = \max(0, c_{ij} - d_{ij})$.

The primary goal of JSSP is to identify an optimal schedule in terms of minimising the overall cost resulting from the divergence of completion times for all operations from their respective due dates. This cost is represented by the equation $\sum_{i=1}^{n} \sum_{j=1}^{m} (\alpha_{ij} e_{ij} + \beta_{ij} t_{ij})$. The problem can be mathematically formulated as follows [225]:

Objective:

$$\text{Minimize} \sum_{i=1}^{n} \sum_{j=1}^{m} (\alpha_{ij} e_{ij} + \beta_{ij} t_{ij}) \tag{3.2}$$

Subject to:

$$e_{ij} \geq d_{ij} - c_{ij} \quad \forall i, j \tag{3.3}$$

$$t_{ij} \geq c_{ij} - d_{ij} \quad \forall i, j \tag{3.4}$$

$$c_{i1} \geq p_{i1} \quad \forall i \tag{3.5}$$

$$c_{ij} \geq c_{ij-1} + p_{ij} \quad \forall i, j : j \neq 1 \tag{3.6}$$

$$\forall i, j, i', j', k : O_{ij} \in O(M_k), O_{i'j'} \in O(M_k), i \neq i' \tag{3.7}$$

$$e_{ij} \geq 0, t_{ij} \geq 0 \quad \forall i, j. \tag{3.8}$$

Constraints 3.3 and 3.4 establish a correlation between the timeliness of each operation, as measured by its completion time and due date, and its earliness and tardiness. The constraint 3.5 indicates that the initial operation of every job commences after the time zero. Constraint 3.6 enforces a precedence relationship between successive operations inside the same job.

3.5.3 Simulation Results

In this work, the GAs are set to a population size of 300, a generational age of 1000, a crossover rate of 95% and a mutation rate of 5% and regeneration was considered when calculating the generation gap. The characteristics of simulated job shops are shown in

Table 3.1 regarding the number of jobs and machines. SM has 32 jobs and one machine, while MM_1 and MM_2 are flexible MM job shops with eight jobs and four machines.

Table 3.2 compares the convergence speed of the applied GAs on the simulated benchmarks. Table 3.3 indicates the objective functions of different GAs. The objective function represents the makespan (time) to complete the job shop schedule. This section shows the simulated benchmarks established from running STGA, RGA, SGA, and AGA. These results show the objective functions applying essential GAs; when the objective function is less, the cost and time used are less. Converging speed refers to identifying the number of generations required to stabilise the error (cost). Once the best solution is reached and the error remains constant, the algorithm can be terminated, as further iterations will not result in any changes. This approach saves time and resources by quickly arriving at the optimal solution.

Table 3.1: Simulated benchmarks.

Job Shop Type	**Number of Machines**	**Number of Jobs**
SM	1	32
MM_1 with similar penalties	4	8
MM_2 with different penalties	4	8

Table 3.2: Basic GAs convergence speed.

Job Shop Type	**STGA**	**RGA**	**SGA**	**AGA**
SM	130	62	26	910
MM_1 with similar penalties	180	30	38	700
MM_2 with different penalties	910	120	230	920

Table 3.3: Basic GA objective function results for different job shops.

Job Shop Type	**STGA**	**RGA**	**SGA**	**AGA**
SM	38,977	43,146	40,588	40,247
MM_1 with similar penalties	7,900	7,850	7,893	7,761
MM_2 with different penalties	4,194	4,187	4,358	4,042

Figures 3.1, 3.2 and 3.3 illustrate the comparison of four algorithms applied to the mentioned three benchmarks, including SM, MM_1 and MM_2. These figures compare

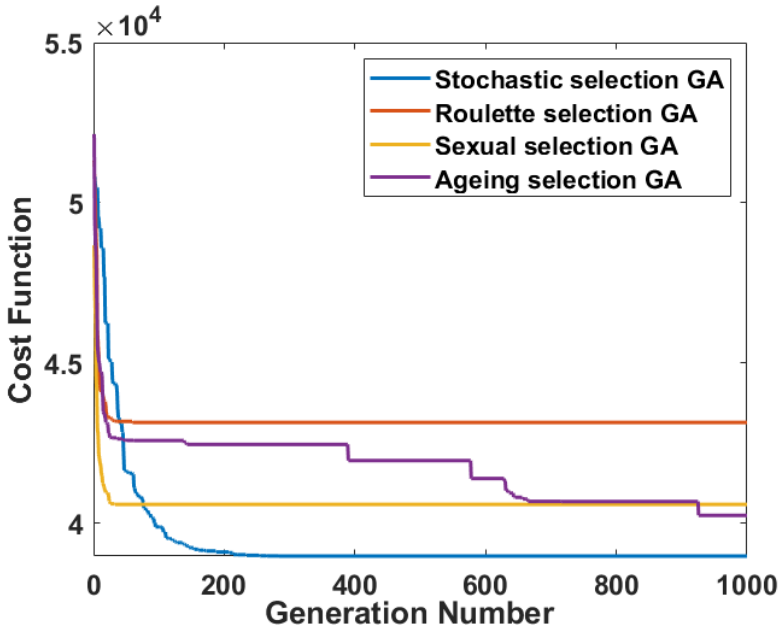


Figure 3.1: SM benchmark results.

the performance of different GAs to reach the best objective function. The algorithm terminates when finding the best schedule with the lowest objective function.

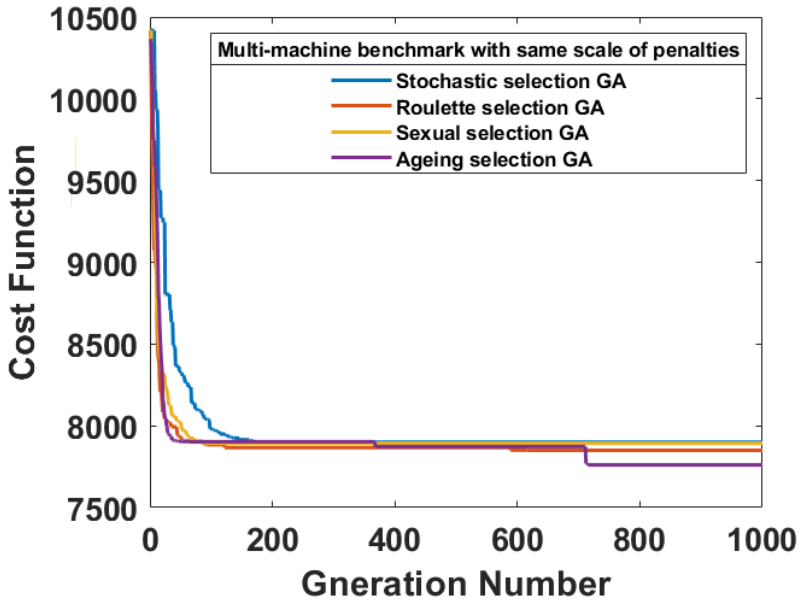


Figure 3.2: MM_1 benchmark results with similar earliness and tardiness scales.

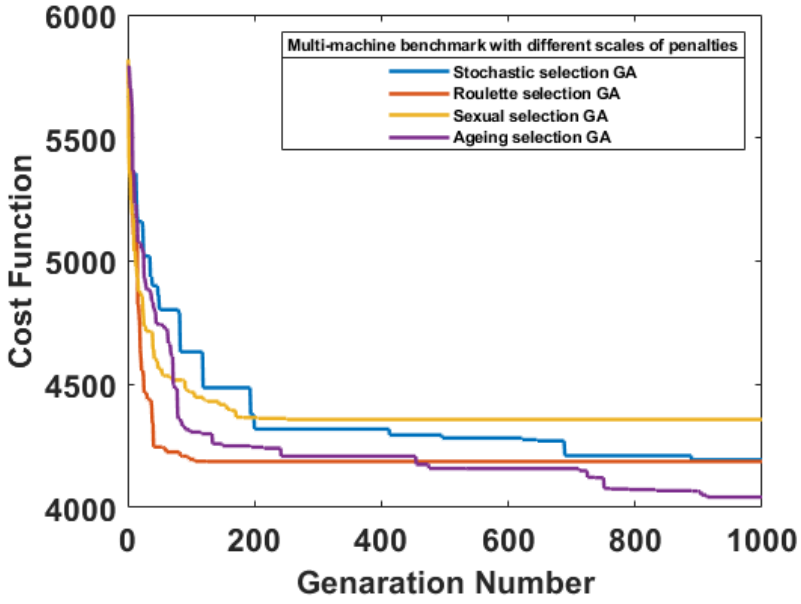


Figure 3.3: MM_2 benchmark results with different earliness and tardiness scales.

3.6 Summary

This chapter has explained the theory of GAs and the biological metaphors used to understand these algorithms. A brief explanation of several selection techniques, including STGA, RGA, SGA, and AGA, have also been included to illustrate the working principles and performance of different types of selections of GA. Also, the benefits and restrictions of GA have been discussed, concluding that it is a valuable tool for global optimisation.

Building on the identified advantages of various GA selections, this study creates advanced GAs to solve JSSPs to reduce elapsed time. To test the performance of new and advanced algorithms in the following chapters, three benchmarks in this chapter have been simulated, and primary results are provided as a reference to be compared in the following chapters with the proposed new algorithms.

Chapter 4

Ethnic and Parthenogenetic Algorithms

4.1 Introduction

This chapter presents a new model named NMHPGA. This approach is built on the fundamental ideas of GAs, which also underlies a new procedure based on EGA. NMHPGA uses parthenogenetic operations, including swap, reverse and inserts, which are examined in this work, along with an EGA. The established algorithm's results are compared to those of other standard GAs using different selection methods, and it becomes clear that the NMHPGA produces superior objective functions with a faster convergence rate.

In Section 4.2, the concept of the proposed EGA is introduced. The ethnic selection process employs a variety of selection operators, including stochastic, roulette, sexual, and ageing. The best individuals are then chosen from each technique and blended. SM and MM job shops with tardiness, earliness and due date penalties are used to test the ethnic selection technique. Later, this concept is tested in this section using similar case studies from Chapter 3, and the results are compared with the basic GAs. Since the results from the experiments outperform the essential GAs, the parthenogenetic algorithm and NMHPGA approaches are introduced in Sections 4.3 and 4.4, combining EGA and PGA to improve the algorithm's accuracy. Section 4.5 uses this new approach in multiple case

studies, and the results are presented and critically analysed. Lastly, section 4.6 provides the summary of this chapter.

4.2 Ethnic Genetic Algorithm

This study proposes EGA. The concept behind EGA is based on combining the diverse populations produced by various selection techniques [226]. The combinations impact the speed of convergence and the overall solution. To verify the convergence speed, an EGA in this section mixes four different forms of selections, including stochastic, ageing, sexual, and roulette selections. Converging speed refers to the process of identifying the number of generations required for the error (cost) to stabilise.

The suggested EGA has been set to a population size of 300, a generational age of 1000, a cross-over rate of 95%, and a mutation rate of 5%, and regeneration is considered when calculating the generation gap.

In this section, the results of EGA are compared with results from essential GAs from Chapter 3. The simulated job shops include SM, MM_1 and MM_2 from Chapter 3. All these three flexible job shops have earliness, tardiness and due date penalties. SM includes 1 machine and 32 jobs, and MM_1 and MM_2 are flexible MM job shops, including 8 jobs and 4 machines. Figure 4.1 shows the flow chart of EGA. It includes different steps; below are the various stages in this algorithm:

1. Initialisation: Creating an initial population of random solutions (schedules in the context of JSSP).

2. Evaluation: Calculating each individual's fitness within the population. This usually entails figuring out each schedule's makespan in the JSSPs.

3. Selection (Continuous Process): Applying a variety of selection techniques on the population, in the EGA, selections are as follows:

    - Stochastic Selection: Selecting individuals with a bias toward better answers.

- Ageing Selection: To add diversity, preferring more recent solutions and steer clear of more traditional ones.
- Sexual Selection: Selection is based on female and male solutions.
- Roulette Selection: Choosing individuals where the likelihood of selection is inversely correlated with fitness.

4. Crossover and Mutation: To generate offspring, performing crossover procedures on the chosen parents. Applying mutations to some of the new population to introduce random changes.

5. Every 10 Generations: Reviewing and Combination:

   - Reviewing: Considering the effectiveness of the individuals, each selection approach is chosen before making the decision.
   - Combination: Determining the best performers from each selection approach based on the review.
   - Combining these top individuals to create a new subset.

6. Termination

Up until a termination requirement is satisfied, the previous procedures are repeated. This might be the algorithm improving a little over numerous cycles, achieving a desirable makespan.

4.2.1 Benefits and Limitations of Ethnic Genetic Algorithm

Below are some benefits and limitations of EGA:

4.2.1.1 Benefits

1. Diversity Preservation: The EGA helps the population remain diversified by employing various selection techniques and reevaluating every 10 generations. Premature convergence may be prevented in this way.

2. Utilising the Best Options: The EGA ensures that reasonable solutions are not lost and can further develop and combine their strengths by integrating the best answers from each method.

   The EGA outlined here aims to take advantage of various selection procedures while routinely analysing and changing to ensure the best solutions are being spread. This strategy can be extremely helpful in complex issues like JSSPs, where the search space is broad and challenging to navigate.

4.2.1.2 Limitations

1. **Computational Cost:** The diversity preservation mechanisms and the continuous reevaluation process every ten generations can be computationally expensive, especially for large populations and complex problems.

2. **Parameter Sensitivity:** The performance of the EGA can be highly sensitive to the choice of parameters such as population size, mutation rate, and generation numbers.

3. **Convergence Speed:** Although EGA aims to prevent premature convergence, it might sometimes lead to slower overall convergence towards the optimal solution.

4.2.2 Case Study: Testing Ethnic Genetic Algorithm

Simulation results comparing STGA, RGA, SGA, AGA, and EGA tested on the single SM and MM job shops (MM_1 and MM_2) are shown in Figures 4.2., 4.3, and 4.4 respectively. Each graph displays the objective function of the generation, representing the cost function to complete the job shop schedule. It can be seen that ethnic selection has the best results in locating the global minima (i.e., the best time) compared to other types of selection, as shown in Table 4.1, which compares the various types of suggested selections. The convergence speed required by each algorithm to find the overall answer is shown in Table 4.2.

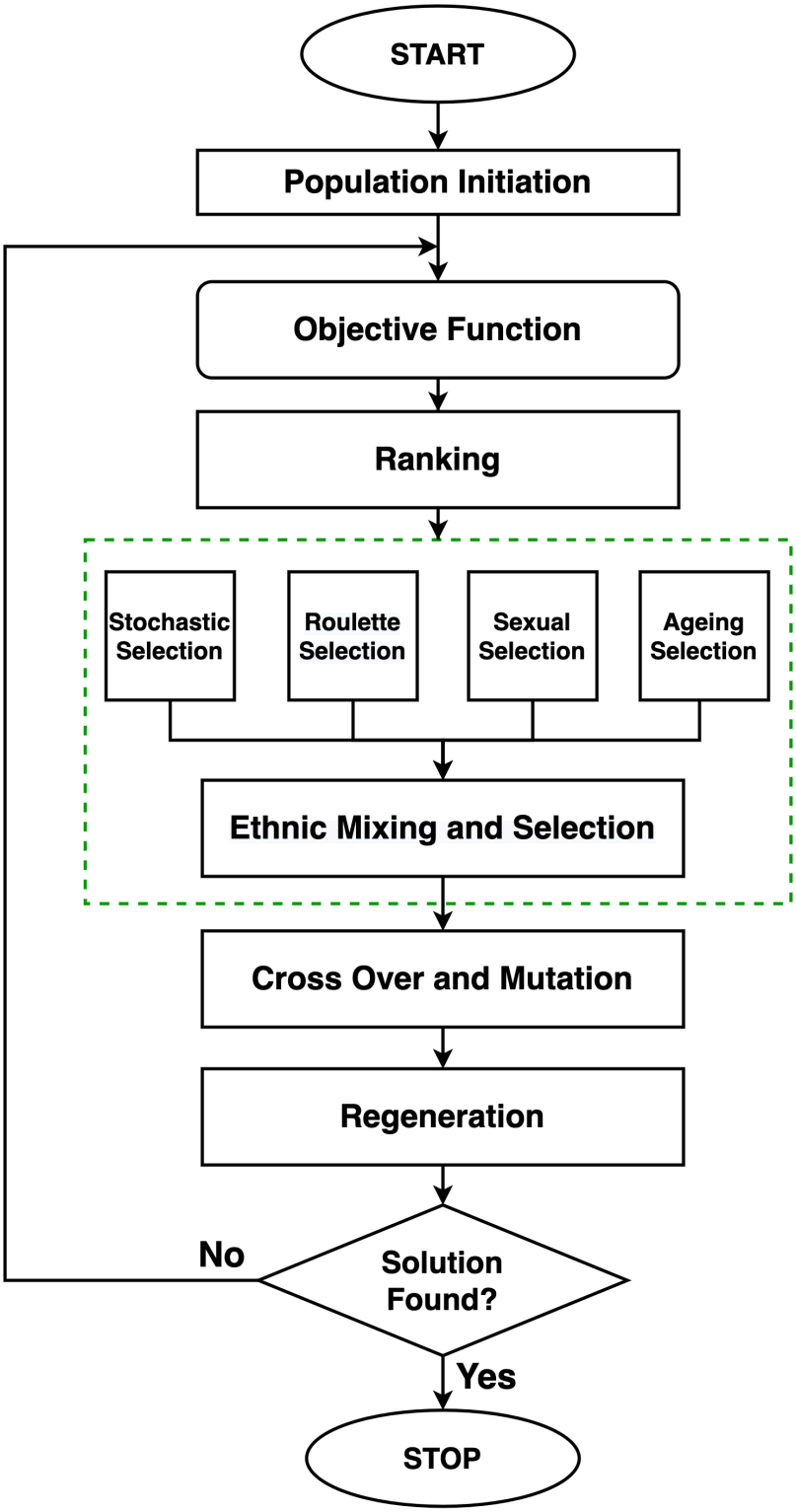


Figure 4.1: Flowchart of EGA

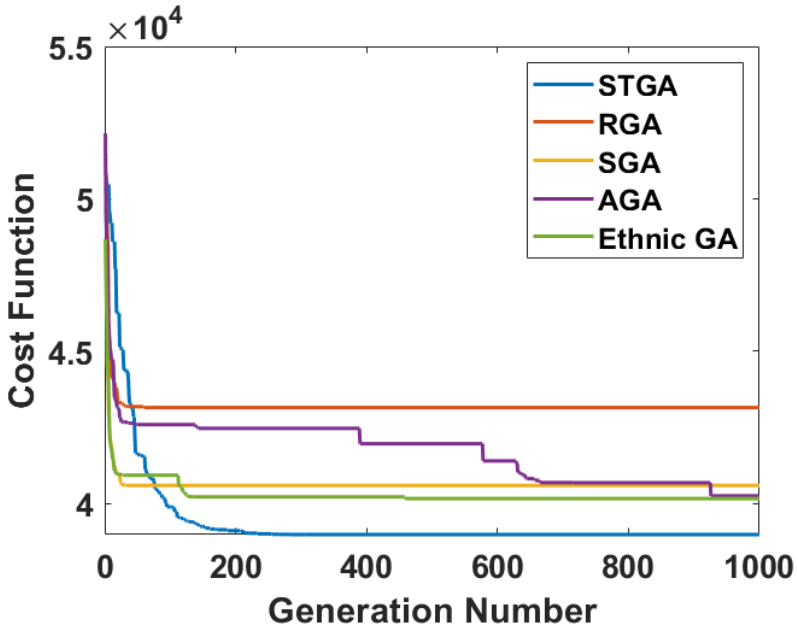


Figure 4.2: SM simulation results

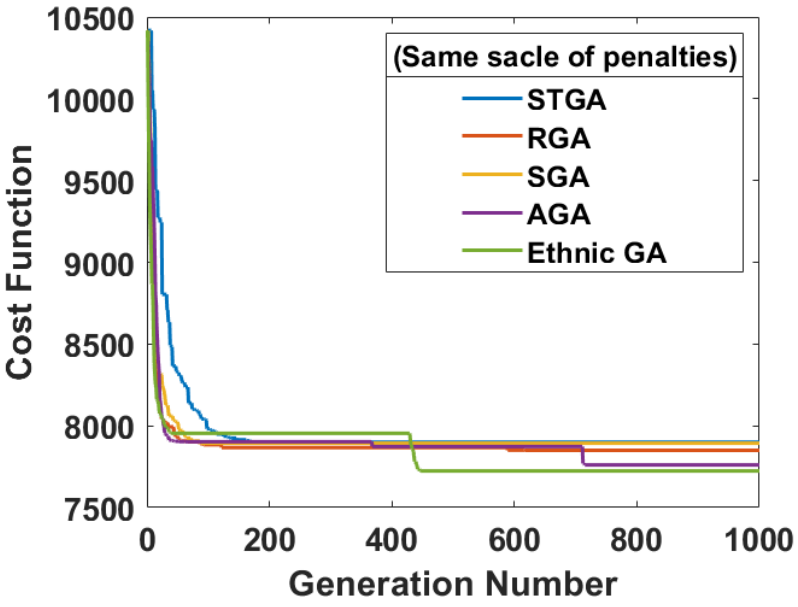


Figure 4.3: Results of MM_1 with similar penalties

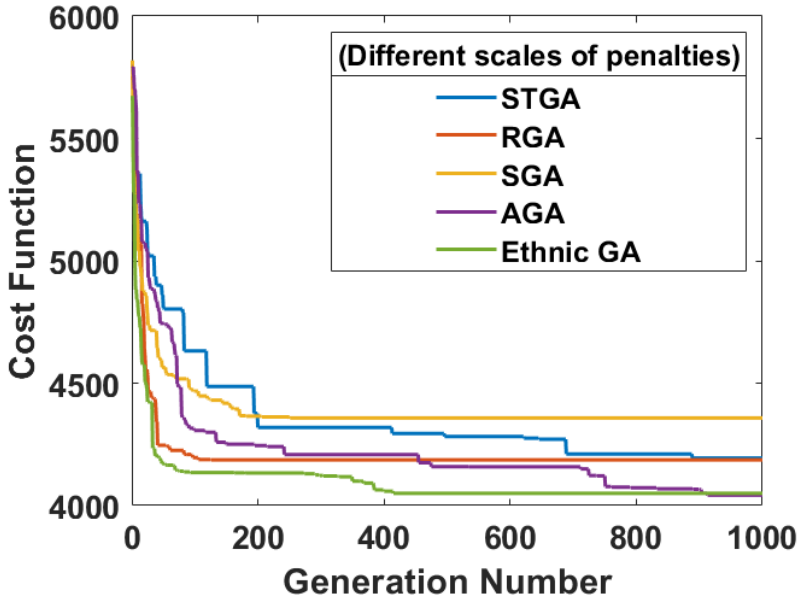


Figure 4.4: Results of MM_2 with different penalties

SM, MM_1 and MM_2 have better results generated from EGA than basic GAs. For instance, MM_1 has the objective function of 7724 on EGA, which is lower than other GA results. MM_2 has the objective function of 4050 on EGA. Having compared the convergence speed of the algorithms, it can be concluded that EGA has better results because it also improves the convergence speed besides having a low objective function value. EGA reduces the objective function and improves the convergence speed simultaneously.

Table 4.1: Comparison of the objective functions of basic GAs and EGA tested on different job shops.

Job Shop Type	**STGA**	**RGA**	**SGA**	**AGA**	**EGA**
SM	38,977	43,146	40,588	40,247	40,161
MM_1 with similar penalties	7,900	7,850	7,893	7,761	7,724
MM_2 with different penalties	4,194	4,187	4,358	4,042	4,050

Table 4.2: Comparison of the convergence speed of basic GAs and EGA tested on different job shops.

Job Shop Type	**STGA**	**RGA**	**SGA**	**AGA**	**EGA**
SM	130	62	26	910	135
MM_1 with similar penalties	180	30	38	700	410
MM_2 with different penalties	910	120	230	920	390

4.3 Parthenogenetic Algorithm

GA is founded on evolution, and NP-hard problems have frequently been solved using the survival of the fittest strategy. A population of chromosomes (individuals) of a series of genes make up the candidate solutions in GAs. The crossover operator is the leading genetic operator to produce new offspring when two parents are combined. Unique individuals can inherit specific characteristics from their parents. Traditional crossover operators come in various forms, such as one-point crossover, two-point crossover, scattered crossover, etc [213].

The Parthenogenetic Algorithm (PGA) produces offspring by gene recombination and selection rather than the conventional crossover operator. By eliminating the crossover operator, PGAs address the problem above and enhance the effectiveness and performance of the GA. This is because the shift operator, which is only performed on a single chromosome, prevents the offspring from the crossover operator from jumping to the invalid solutions area. Swap, reverse, and insert are the three PGA operators. These three operators create a new chromosome by rearranging the genes on an existing one, Figure 4.5 indicates the flow chart of PGA [82, 122, 127].

PGA Operators for JSSP are as follows:

1. **Swap Operation in PGA:**
   - Concept: Changing the locations of two jobs in the encoding.
   - Application in PGA: Creating a child from a parent solution by switching two

tasks in the order. This generates a new potential solution without the requirement for crossover by simulating a slight modification to a machine's task sequence.

2. **Reverse Operation in PGA:**

   • Concept: Choosing a set of jobs, then reversing the order of the jobs within the encoding.

   • Application in PGA: Creating a child solution from a parent solution by flipping a series of tasks. Due to the order of numerous jobs changing, this results in a more noticeable shift than a swap. It gives the PGA another way to delve into the large JSSP solution space.

3. **Inserting Operation in PGA:**

   • Concept: Taking a job from its current location in the encoding and place it in another position.

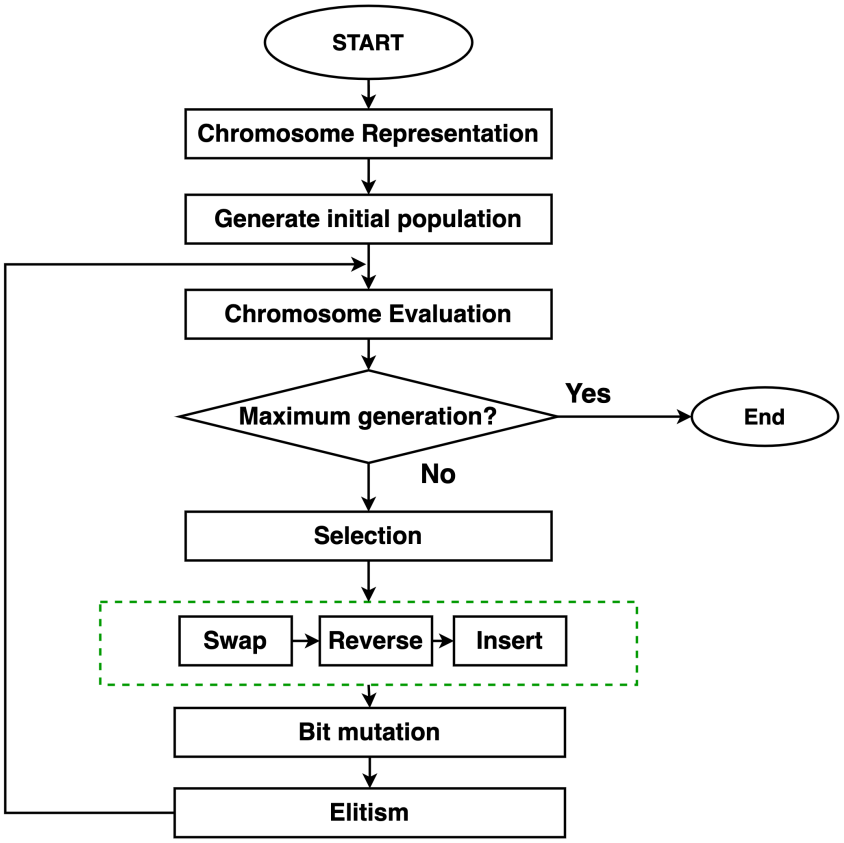


Figure 4.5: Flowchart of PGA.

4.4 New Metaheuristic Hybrid Parthenogenetic Algorithm

A New Metaheuristic Hybrid Parthenogenetic Algorithm (NMHPGA) is developed in this research. This algorithm integrates a variant of existing selections with the PGA to conduct a reliable comparative analysis and assess the overall effectiveness of the innovative NMHPGA [82].

Five distinct algorithms with different selection types solely use mutation, swap, reverse, and insert functions instead of crossover operators to test the algorithm. Stochastic selection based on random selection is used in the stochastic selection PGA (STPGA). Additionally, the sexual PGA (SPGA) is based on sexual selection; in contrast, the APGA wish is an ageing PGA based on ageing selection; and finally, the NMHPGA is based on ethnic selection, which is a combination of stochastic selection, roulette selection, sexual selection, and the ageing above selection. The NMHPGA algorithm's steps are depicted as follows [82]: Standard selection techniques are used in the first testing phase. The second stage, ethnic selection, contrasts this by combining the top individuals chosen using various styles into a single group. The crossover function, which takes a lot of time since it must check for duplicate chromosome genes, is not used by the NMHPGA.

The algorithm's accuracy is improved using the most recent and enhanced selection versions. Also, the intrinsic parameters of the algorithm play a significant role in how quickly they converge. This algorithm is tested on 29 simulated job shops. Figure 4.6 indicates different stages of NMHPGA, which are described below:

1. **Initialisation:** Starting with a population of potential solutions for JSSPs that have been randomly initialised.

2. **Ethnic Selection:** Determining an individual's fitness for each ethnic population using the JSSP objective (e.g., minimising makespan).

3. **Parthenogenetic Operations:**

   a. Reverse: Applying the reverse process on some of the solutions inside each

subset of chosen solutions. This entails selecting tasks from a solution and flipping their placement. This may introduce diversity and lead to improved job processing sequences.

b. Insert: Applying the insert operation to some solutions. A job is chosen from the present position of the sequence and is relocated to another position. This operation determines whether altering the relative priority of some tasks enhances the solution's fitness.

c. Swap: A swap operation is performed on solutions by which two jobs trade places in the sequence. Swapping can change job priorities, which could result in shorter makespans. The core of parthenogenetic reproduction is preserved after these operations because each solution in the population received a modification without crossover.

4. **Mutation:** Introducing a typical mutation phase after the parthenogenetic procedures, here, randomly chosen genes (i.e., jobs, in the context of JSSP) may experience slight alterations in a solution. This could involve altering a job's machine assignment in the sequence. This mutation ensures that the population is even more variable and facilitates the discovery of sections of the solution space that the parthenogenetic operations might not have been able to reach.

5. **Evaluation and Selection** Evaluating the fitness of all modified solutions following the mutation phase, keeping the top performers for the following generation and replacing or discarding the less efficient ones.

6. **Termination:** For a given number of generations or until a specific convergence criterion is satisfied, repeating the ethnic selection, parthenogenetic operations, mutation, and evaluation phases. The best response across all populations is chosen once the algorithm has finished.

Generation Gap: The generation gap is a crucial factor; a more significant generation gap may accelerate convergence, but there is a chance that it will result in less desirable solutions. A smaller generation gap may allow for greater diversity and a more

thorough exploration of the problem space, but it may also result in a slower convergence rate. Finding the best value for a given issue frequently necessitates tuning, as with many factors in GAs. The selection and replacement phases of the GA involve the generation gap as follows:

Selection:

An individual is chosen as a parent to produce the following generation based on their fitness values. The generation gap affects both the selection process and the number of candidates. If the generation is 100%, enough parents are chosen to create an entirely new population. Fewer parents are chosen if it is less.

Replacement:

After crossover and mutation procedures produce new offspring, the GA must choose which members of the old population to replace with these children. The generation gap determines the number of replacements.

4.5 Comparative Analysis of NMHPGA

To test the performance of NMHPGA, objective function values and convergence rates of various algorithms, including (STPGA, APGA, SPGA and RPGA) are used. To achieve this, the following subsections compare the convergence rates of several algorithms evaluated on the simulated job shops.

4.5.1 Simulated Job Shop Schedules

In this work, three categories of flexible job shops are tested, designed to make workshops as straightforward as possible to focus on optimisation results and compare the efficacy achieved. Three categories of job shops are tested: Category SM job shops, consisting of 10 SM job shops with earliness, tardiness, and due date penalties (Table 4.3). Category-A MM job shops, consisting of 10 MM job shops, have four machines and eight jobs with earliness, tardiness, and due dates. Category-B consists of nine MM job shops with earliness, tardiness, and due date (Table 4.4).

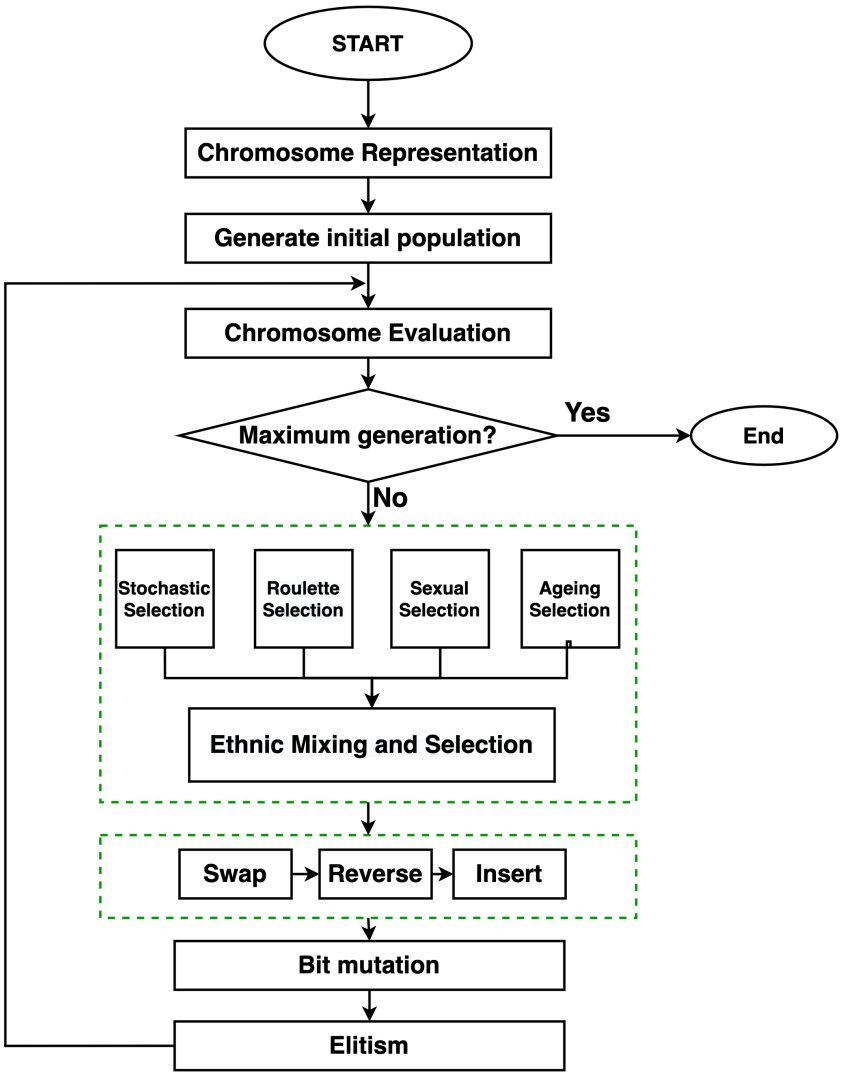


Figure 4.6: Flow chart of NMHPGA.

Tables 4.3 and 4.4 present features of the various job shops employed in this study. Also, the beginning state of random selection is evenly chosen to provide a comparable model. Appendix A illustrates the earliness and tardiness of the job shops, which are created randomly in the MATLAB code.

The purpose of the scheduling problem is to minimise execution time and penalties. The population is set to 300, and the generation size is set at 1000. The test examines the NMHPGA's performance with simple mutation and advanced regeneration and the impact of the chosen forms of roulette, sexual, ageing, and ethnic selection. MATLAB R2021a software was utilised for this testing. Based on the analysis of the provided data, the NMHPGA algorithm's performance was compared to the best performances of the other algorithms (STPGA, RPGA, SPGA, and APGA) across different job shop types. The improvement percentages were calculated by identifying the lowest (best) objective function value from the other algorithms for each job shop type and then computing the percentage reduction achieved by NMHPGA relative to this best value. For instance, in the SM category, the NMHPGA algorithm demonstrated an overall average improvement of approximately 0.32-4.6 per cent in the SM category. This indicates that while NMHPGA showed significant improvements in some cases, Despite these variations, the NMHPGA algorithm generally contributed to better optimisation outcomes, reflecting its potential effectiveness in enhancing job shop scheduling solutions.

Table 4.3: SM job shops attribute.

Job Shop Type	**Number of Machines**	**Number of Jobs**
SM1	1	32
SM2	1	40
SM3	1	60
SM4	1	80
SM5	1	100
SM6	1	120
SM7	1	150
SM8	1	200
SM9	1	250
SM10	1	300

Table 4.4: Category-A MM job shops 4 machines, 8 jobs (MM1-MM10) and category-B MM job shops (MM11-MM19) attributes.

Job Shop Type	Number of Machines	Number of Jobs
MM1	4	8
MM2	4	8
MM3	4	8
MM4	4	8
MM5	4	8
MM6	4	8
MM7	4	8
MM8	4	8
MM9	4	8
MM10	4	8
MM11	4	10
MM12	4	20
MM13	4	30
MM14	4	40
MM15	4	100
MM16	4	150
MM17	4	200
MM18	4	250
MM19	4	300

4.5.2 Simulation Results

Figures 4.7, 4.8 and 4.9 present the SM and MM simulation results. Figure 4.7 shows the simulated result of the applied algorithms to SM1-SM10. Figure 4.8 shows the simulated result of the applied algorithms to MM1-MM10 (MM-A). Lastly, Figure 4.9 shows the simulated result of the applied algorithms to MM11-MM19 (MM-B).

Each graph displays the generation number's objective function. The objective function represents the cost function required to finish the job shop. The comparison of objective functions of various algorithms tested on job shops for the SM category is shown in Table 4.5. Also, Table 4.6 compares the convergence rates of several algorithms evaluated on job shops in the SM category. The number of generations shown to arrive at the optimal solution indicates that the objective function for NMHPGA has the lowest value and the fastest convergence speed. As NMHPGA requires fewer generations, this improves convergence speed. As shown by SM1 results, the suggested algorithm provides a lower objective function; for instance, the objective function of SM1 is 39444, and the

Table 4.5: Comparison of objective functions of different algorithms tested on job shops in SM category.

Job Shop Type	STPGA	RPGA	SPGA	APGA	NMHPGA
SM1	41,251	39,982	39,947	40,225	39,444
SM2	159,697	181,734	187,035	159,802	160,871
SM3	479,639	599,840	584,763	471,184	474,325
SM4	1,154,879	1,280,561	1,301,604	1,118,500	1,120,440
SM5	2,405,190	2,566,361	2,609,437	2,299,699	2,297,979
SM6	14,135,674	11,540,670	11,591,499	11,849,996	11,009,721
SM7	9,342,492	10,172,814	9,832,276	8,546,759	8,549,596
SM8	19,156,403	19,650,883	19,532,952	15,980,532	15,952,550
SM9	39,430,578	40,136,718	40,668,916	31,142,579	30,845,566
SM10	74,190,404	61,548,166	60,259,784	52,838,449	52,260,543

Table 4.6: Comparison of convergence speed of different algorithms tested on job shops in SM category.

Job Shop Type	STPGA	RPGA	SPGA	APGA	NMHPGA
SM1	859	113	113	800	500
SM2	311	53	143	210	747
SM3	894	113	150	203	142
SM4	906	145	362	143	178
SM5	899	173	159	272	271
SM6	917	740	943	975	581
SM7	984	475	259	396	338
SM8	991	382	302	604	394
SM9	993	728	518	647	578
SM10	988	636	353	747	606

convergence speed is 500. Results from the APGA are 40225 for the objective function and 800 for convergence. If the above results are compared with SPGA, it can be shown that the objective function of SPGA is 39947, which is almost in the same range near 4000; however, NMHPGA has a lower objective function with a desirable convergence speed. Moving forward to other job shops, the general procedure of NMHPGA is the same, and it concentrates on both the objective function and convergence speed.

Table 4.7: Comparison of objective functions of different algorithms tested on job shops in MM category-A

Job Shop Type	STPGA	RPGA	SPGA	APGA	NMHPGA
MM1	7,931	7,850	7,893	7,700	7,758
MM2	9,481	9,469	9,523	9,485	9,465
MM3	7,895	8,514	8,277	8,023	7,913
MM4	7,931	7,850	7,893	7,700	7,758
MM5	7,104	7,582	7,306	7,054	7,039
MM6	7,849	7,907	7,918	7,785	7,798
MM7	9,428	9,387	9,399	9,210	9,181
MM8	8,728	9,039	9,209	8,824	8,727
MM9	6,255	6,258	6,237	6,198	6,195
MM10	7,931	7,850	7,893	7,700	7,758

Table 4.8: Comparison of convergence speed of different algorithms tested on job shops in MM category-A.

Job Shop Type	STPGA	RPGA	SPGA	APGA	NMHPGA
MM1	991	597	127	55	818
MM2	435	152	177	115	50
MM3	219	385	67	85	76
MM4	908	590	99	60	812
MM5	583	664	382	410	336
MM6	429	406	447	420	543
MM7	938	136	120	147	90
MM8	565	733	905	343	212
MM9	952	141	81	230	635
MM10	986	592	104	76	816

Table 4.9: Comparison of objective functions of different algorithms tested on job shops in MM category-B.

Job Shop Type	STPGA	RPGA	SPGA	APGA	NMHPGA
MM11	17,043	16,503	16,634	16,546	16,421
MM12	137,736	131,970	131,274	133,618	131,115
MM13	468,806	463,734	465,696	461,970	461,538
MM14	1,269,516	1,258,531	1,246,911	1,238,453	1,236,658
MM15	18,603,560	17,521,764	17,555,825	17,503,567	17,432,983
MM16	67,602,116	62,639,047	62,988,711	62,694,341	62,354,613
MM17	147,200,593	133,407,491	133,306,194	133,005,781	131,840,576
MM18	305,201,786	274,169,417	275,108,844	273,802,285	270,317,679
MM19	544,267,800	482,106,036	480,728,988	482,871,193	475,082,286

Table 4.10: Comparison of convergence speed of different algorithms tested on job shops in MM category-B.

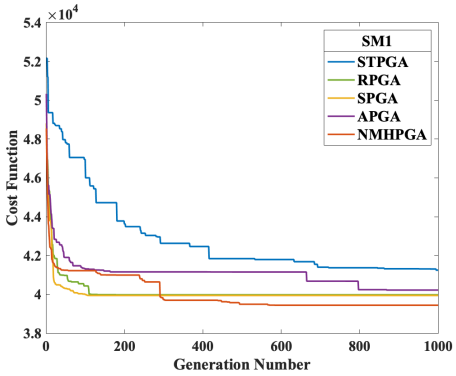
Job Shop Type	**STPGA**	**RPGA**	**SPGA**	**APGA**	**NMHPGA**
MM11	995	364	661	915	143
MM12	993	825	744	816	401
MM13	993	396	445	283	279
MM14	993	329	403	518	302
MM15	998	454	641	532	493
MM16	990	715	908	878	774
MM17	992	942	954	922	920
MM18	986	982	984	979	979
MM19	993	984	988	988	933

NMHPGA can produce superior outcomes in terms of the cost function because NMHPGA objective function results are generally better with lower values. The proposed algorithm chooses the best answers from multiple selection methods, providing thorough and varied solutions that may be combined to produce new answers using missing chromosomes from various search locations. On the other hand, due to the random nature of the solution generation restricted to a specific search area of the solution space, the stochastic and roulette selection approaches exhibit slower convergence and do not reach the global minima.

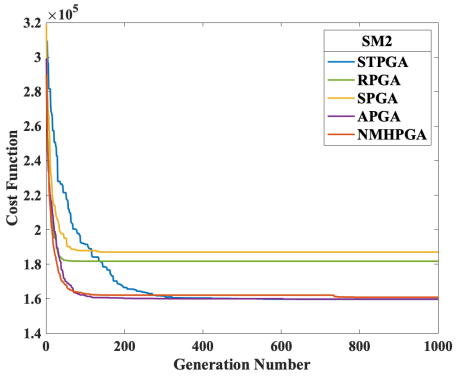
The objective functions of the algorithms tested on job shops in MM category A are compared in Table 4.7. A comparison of the convergence rates of the algorithms tested on job shops in MM category-A is also included in Table 4.8. The findings of job shops category-B are shown in Tables 4.9 and 4.10.

Comparing the convergence speed and objective functions run on different selected algorithms shows the excellent performance of NMHPGA. For instance, testing algorithms on MM1, NMHPGA produces an objective function of 7758; compared to STPGA, RPGA, SPGA and APGA, the objective function of NMHPGA is lower with improved convergence speed, which means cost reduction in a faster period.

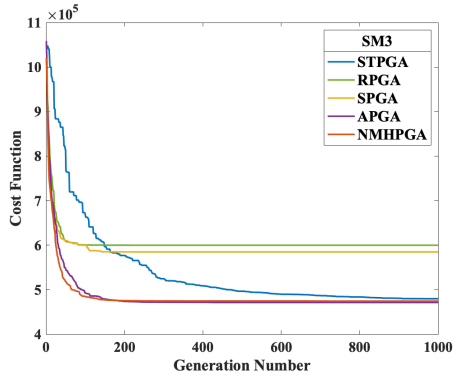
MM11-MM19 are more complicated job shops than MM1-MM10 because the number of jobs and machines are increased in this category; they are simulated to test the performance of NMHPGA on the more complex and more significant scale of job shops.



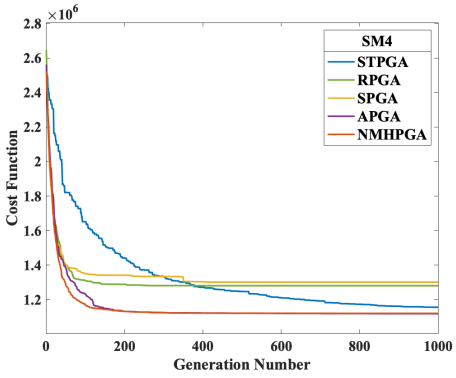
(a) Results of SM1 model



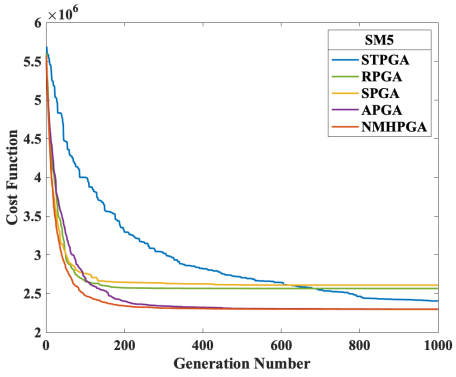
(b) Results of SM2 model



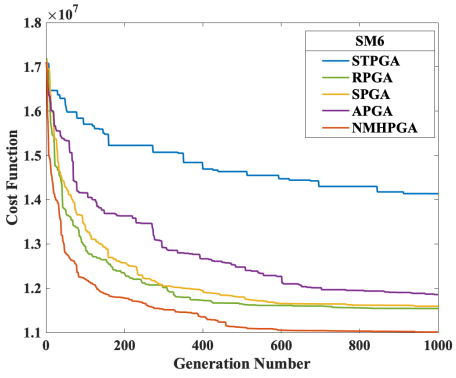
(c) Results of SM3 model



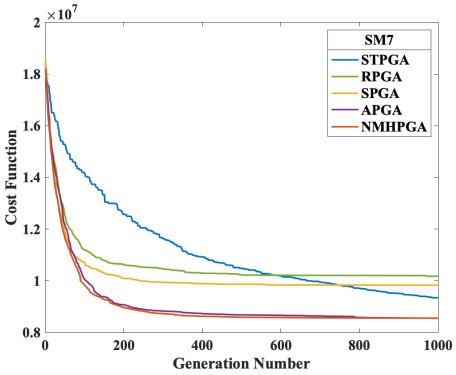
(d) Results of SM4 model



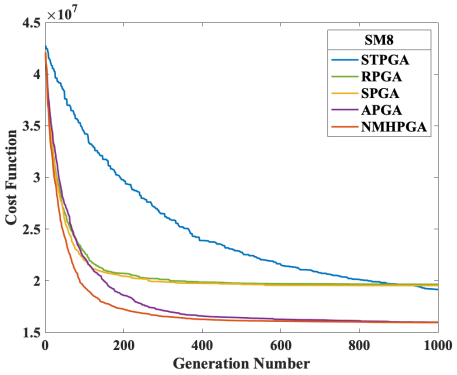
(e) Results of SM5 model



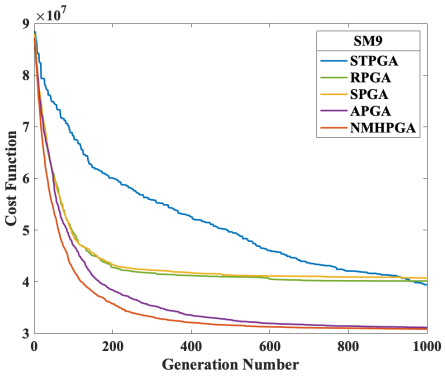
(f) Results of SM6 model



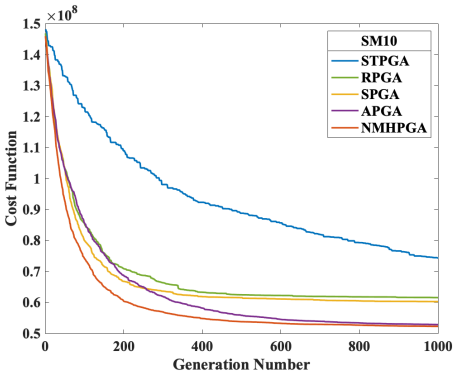
(g) Results of SM7 model



(h) Results of SM8 model



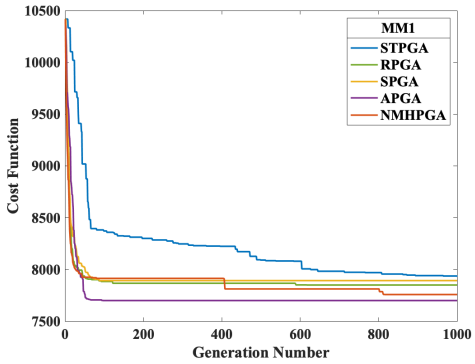
(i) Results of SM9 model



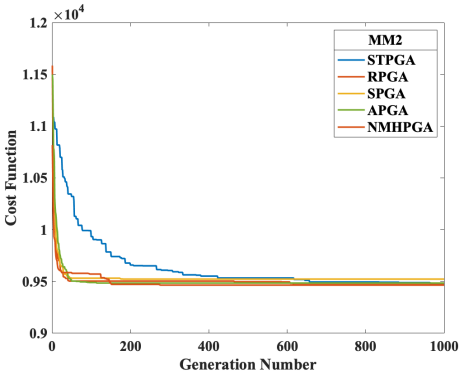
(j) Results of SM10 model

Figure 4.7: Simulation results tested on (SM1-SM10).

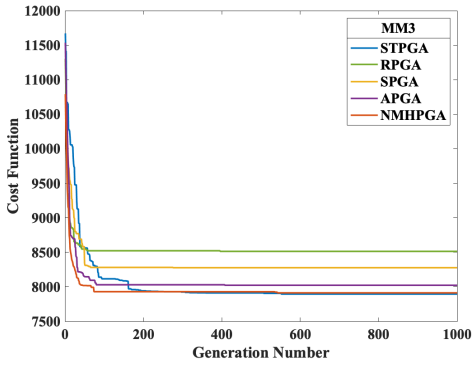
Similar to the MM1-MM10, NMHPGA outperforms the primary GAs. Due to the large scale of MM11-MM19, the primary GAs are time-consuming, and NMHPGA performs faster.



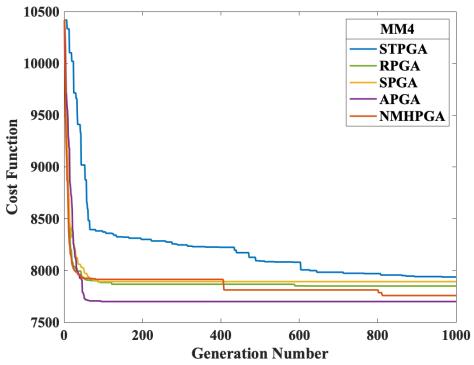
(a) Results of MM1 model



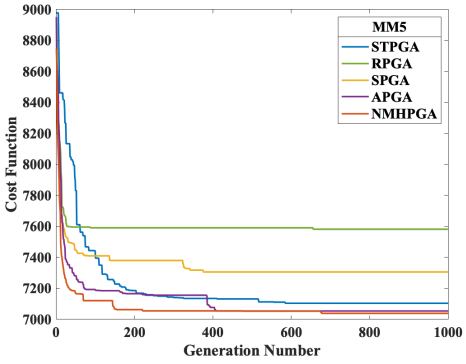
(b) Results of MM2 model



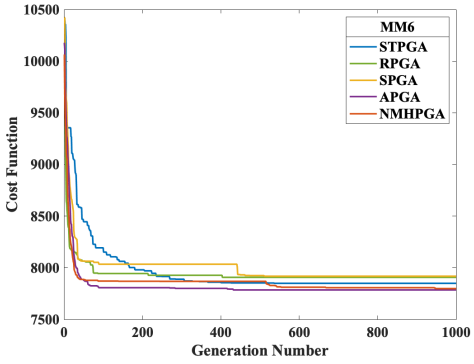
(c) Results of MM3 model



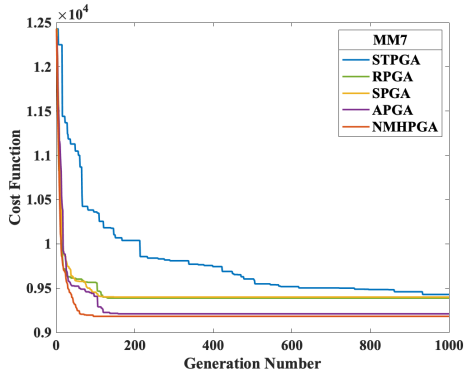
(d) Results of MM4 model



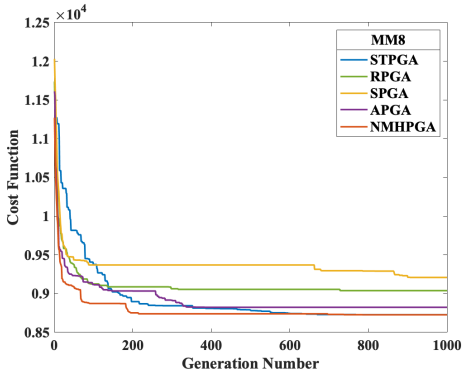
(e) Results of MM5 model



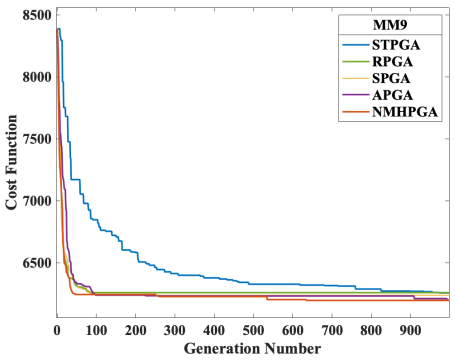
(f) Results of MM6 model



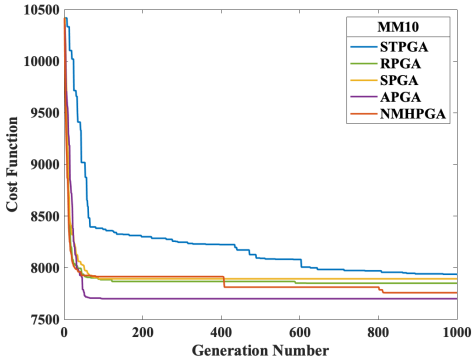
(g) Results of MM7 model



(h) Results of MM8 model



(i) Results of MM9 model



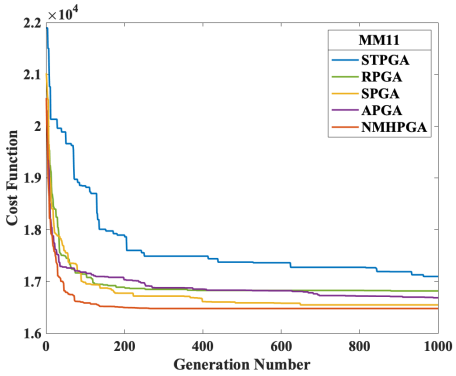
(j) Results of MM10 model

Figure 4.8: Simulation results of MM category-A (MM1-MM10).

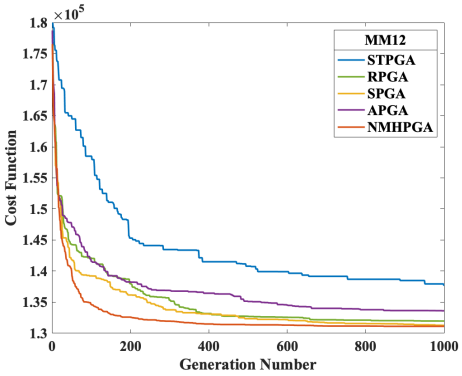
4.6 Summary

This chapter developed the NMHPGA technique based on the fusion of various GA selection techniques. Three kinds of job shops were used to assess the established algorithm's performance. The following are the key conclusions and a summary of the findings:

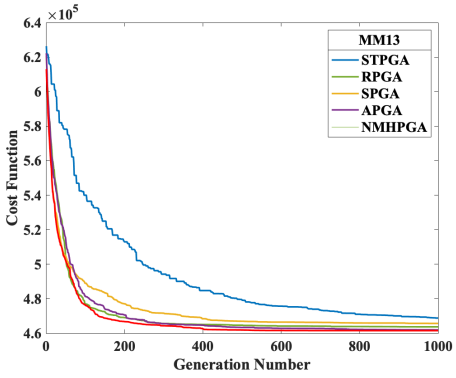
- This work addresses several GA types with various selection types.
- The PGAs are tested using various selection methods, and the results show that a combined solution is superior to an individual one. The ethnic selection is the greatest since it brings together the top individuals.
- Integrating the PGA with ethnic selection demonstrates that superior outcomes can be obtained without time-consuming crossover processes.
- The NMHPGA can improve the global search point and convergence speed.



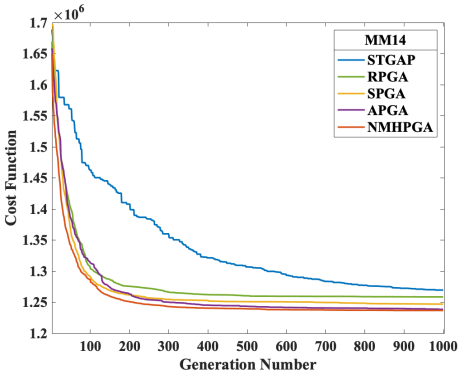
(a) Results of MM11 model



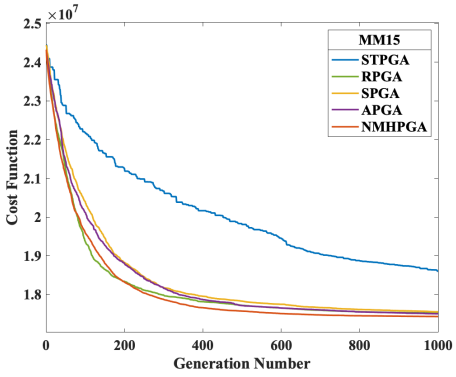
(b) Results of MM12 model



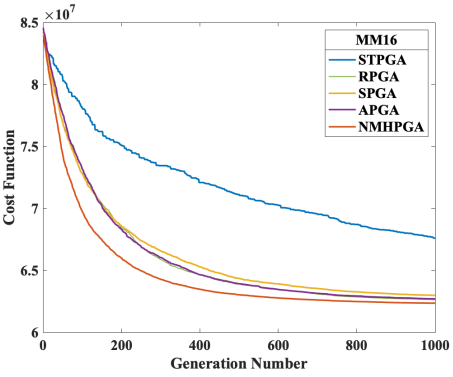
(c) Results of MM13 model



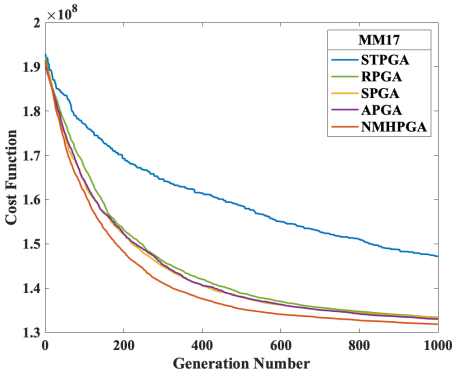
(d) Results of MM14 model



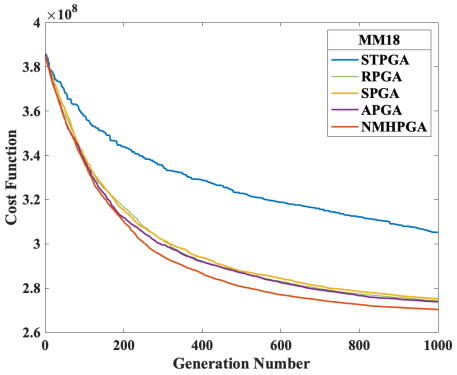
(e) Results of MM15 model



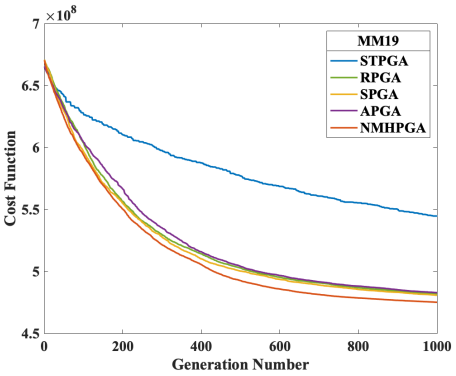
(f) Results of MM16 model



(g) Results of MM17 model



(h) Results of MM18 model



(i) Results of MM19 model

Figure 4.9: Simulation results of MM category-B (MM11-MM19).

- The NMHPGA eliminates the crossover function and replicates it with swap, insert, and reverse functions combined with ethnic selection, enhancing efficacy and performance.

The established NMHPGA has been tested using three different job shop categories. More selection and combination processes can be incorporated into upcoming research to increase efficiency with fewer genes. Moreover, more challenging benchmarks and case studies from the industry might be used to test the algorithm.

Converging speed refers to determining the number of generations until the error (cost) is steady. The best solution has been found, and the algorithm can be stopped because the error will not change. This saves time and resources by enabling the best solution to be found quickly. Such selection approaches indeed impact the speed of convergence and the global solution. The method will thus identify the best and quickest convergence compared to the basic GAs tested in this study. In the next chapter, two RL-based methods are designed, and the results are compared with the technique in this chapter.

Chapter 5

Application of Reinforcement Learning for Job Shop Scheduling Problems

5.1 Introduction

This chapter provides intelligent algorithms that control scheduling problems using advanced Q-learning (QRL) and SARSA algorithms. To test and compare the accuracy and convergence speed of the models, the results are compared to the EGA and advanced NMHPGA [82, 226] presented in the previous chapter. RL methodologies are a kind of machine learning algorithms identified as global optimiser methods. The proposed methods are discussed in more detail in the subsequent sections. The most notable benefit of the developed RL models is their ability to enhance the system's performance without using numerous GA functions.

The structure of this chapter is as follows: Section 5.2 provides the RL framework, including related equations. Section 5.3 discusses QRL, Section 5.4 discusses the novel model of SARSA, and Section 5.5 provides simulation results and discussion. Section 5.6 provides a summary of the Chapter.

5.2 Reinforcement Learning Framework

This section discusses conventional RL foundations. The methodology used in this study is discussed in detail in the next section. The RL discussion begins with its foundational definitions and ideas, such as the agent, environment, action, state and reward functions.

In RL, an agent can use a action set $A = \{A_1, A_{\mathbf{2}} \ldots\}$ to interact with the environment. All the potential actions are described in the action set. RL methods are designed to train the agent to interact properly with its surroundings [227, 228, 229]. The agent initially determines the current state of the environment S_t, and the associated reward value R_t, at a relative time step t. The agent decides what to do next based on the state and reward information. To achieve a new state S_{t+1} and reward R_{t+1}, the intended action of the agent A_t, is fed back to the environment [227, 228].

The reward function R generates an immediate reward R_t, based on the status of the environment and sends it to the agent at each time step. The goal is to provide feedback from the environment to the agents; the reward function is based only on the present state, as expressed in Equation 5.1 [227, 228]:

$$R_t = R\left(S_t\right) \tag{5.1}$$

A trajectory in RL is a series of conditions, actions, and rewards that record how the agent interacts with the environment, as expressed in Equation 5.2:

$$\tau = (S_0, A_0, R_0, S_1, A_1, R_1, \ldots) \tag{5.2}$$

The start-state distribution is denoted as 0, from which $S_{\mathbf{0}}$ is randomly sampled to determine the initial state of a trajectory as expressed in Equation 5.3:

$$S_0 \sim \rho_0(\cdot) \tag{5.3}$$

A trajectory, also known as an episode, is the progression from a starting point to an ending point. Two important factors in RL are exploration and exploitation. Utilising

Algorithm 4: Q-learning algorithm

Initialise $Q(s, a)$
for *all state-action pairs and step size* $\alpha \in (0, 1]$ **do**
  for *each episode* **do**
    Initialise S_0
    for *each step* S_t *in the current episode* **do**
      Select A_t Using policy that is based on Q
      $R_{t+1}, S_{t+1} \leftarrow \text{Environment}\,(S_t, A_t)$
      $Q\,(S_t, A_t) \leftarrow Q\,(S_t, A_t) + \alpha\,[R_{t+1} + \gamma \max_a Q\,(S_{t+1}, a) - Q\,(S_t, A_t)]$
    end
  end
end

current knowledge to maximise agent performance is known as exploitation, and the expected reward typically measures an agent's performance. On the other hand, through activity and interaction with the environment, exploration aims to increase the knowledge [227, 228].

5.3 Application of Q-Reinforcement Learning

5.3.1 Overview

One of the most common RL methods is QRL. It trains an action-value function that reflects the anticipated cumulative reward for performing a given action in a given situation. It works by imparting knowledge to an action-value function and expressing the expected cumulative benefit of performing a specific action.

The primary concept of the algorithm is a simple iterative update, wherein each pair of the states and actions (s, a) has a corresponding Q-value. When the agent in state s selects an action (a), the reward for doing so and the best Q-value for the following states are utilised to update the Q-value of the state action pair [228]. Q-value shows the quality of the action in a given state. The basic QRL process is presented in Algorithm 4 [227, 228].

5.3.2 State, Action and Reward in the Designed QRL Model

The initially developed model of this chapter is based on the QRL. Figure 5.1 shows the training phase of the proposed QRL model. Flexible job shops, including 10 SM and 19 MM job shops from Chapter 4, are considered QRL environments. In this model, an agent is connected to each machine to find the best schedules. Here, the agent locally selects the fundamental actions and decides which work among those pending on the appropriate machine must be processed next. The agent considers the jobs processed at each stage based on the limits and challenges. The process depends on the collection of machines performing specific tasks. This approach is rewarded significantly and occupies the top slot when the new makespan is superior. When there is no improvement in the outcomes, the iterations end. The scheduling procedure is further optimised by repeating the optimisation steps of the QRL algorithm.

In this model, the learning rate (alpha) is 95%, the discount rate (gamma) is 0.1, and the max number of iterations is 10000 [229]. Following is a discussion of the parameters in the proposed RL models as follows:

Alpha (α): The agent changes its Q-values depending on new experiences, which is determined by the learning rate represented by (α). Regarding simulated FJSPs in this work, alpha affects how fast the agent modifies its scheduling decisions. When FJSPs are highly interactive, a more significant alpha can help because it speeds up learning. However, an excessively low learning rate can cause slow convergence.

Gamma (γ): The agent's preference for immediate rewards over future rewards is determined by the discount factor, represented by the (γ) symbol. It impacts how the proposed RL models balance short- and long-term goals. This work focuses on short-term goals.

Greedy exploration (ε)-Greedy: The (ε)-Greedy exploration strategy combines exploitation and exploration. It entails choosing between a random action and the action with the highest Q-value (greedy action) with probability 1- (ε). This approach in the applied FJSPs enables the agent to balance the exploitation of proper strategies and the exploration of scheduling choices. More exploration is encouraged by a greater (ε),

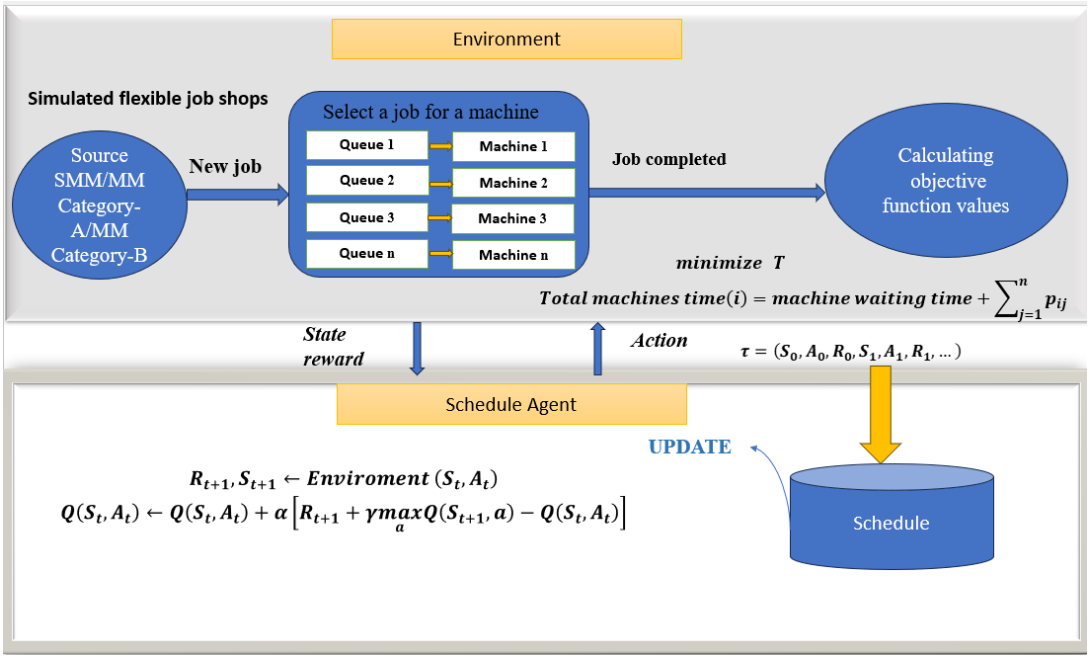


Figure 5.1: The designed QRL model, applying the simulated job shops as the environment.

Algorithm 5: SARSA algorithm

Initialise $Q(s, a)$ for all state-action pairs.
for *each episode* **do**
  Initialize S_0
  Select A_0 using a policy that is based on Q
  for *Each step* S_t *In the current episode* **do**
    Select A_t from S_t Using policy that is based on Q
    R_{t+1}, Environment (S_t, A_t)
    Select A_{t+1} from S_{t+1} using a policy that is based on Q
    $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\right]$
  end
end

which is advantageous in flexible scheduling environments. A smaller indicates more exploitation and the use of try-and-error methods.

5.4 Application of SARSA Model

The second contribution of this chapter is to develop an advanced SARSA model. Algorithm 5 presents the standard SARSA algorithm [227, 228]. Figure 5.1 shows the design of the proposed SARSA model for optimising the job shops applied in this study [229]. The following section discusses the model in more detail.

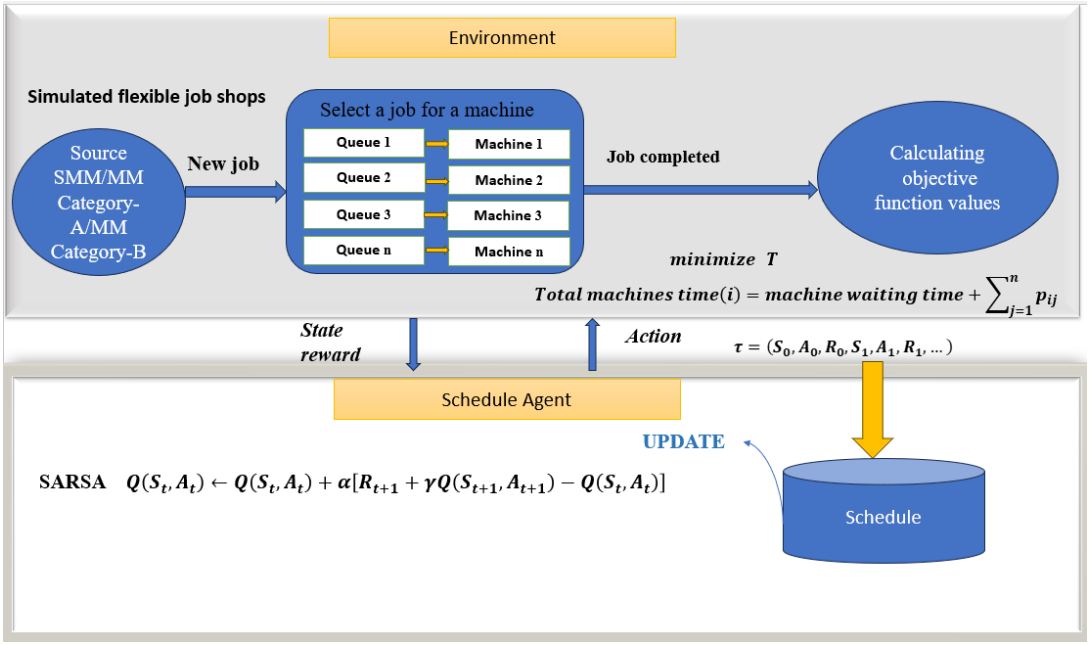


Figure 5.2: The designed SARSA model applying the simulated job shops as the environment.

In this model, the initial environments include 29 simulated flexible job shops from Chapter 4. The initial state begins to schedule the job sequences employing related rewards.

Agent In this model, an agent interacts with the environment, gains knowledge, and makes choices. When a job becomes available, it is selected from the local waiting queue of the machine and processed. When the current operation is completed, each job chooses a machine. At this point, the job becomes a job candidate for the machine and is assigned an agent for task routing. This is performed to address the vast action space of the FJSPs. Thus, the agent decides what is appropriate given the current state of the environment.

States and Rewards The state represents the environment, including various machines and jobs-related aspects. Such aspects include the productivity of machines and how they interact, including the number of machines active and available for use in each operation and the workloads of the jobs completed or in progress. RL aims to maximise returns or the sum of rewards while minimising the makespan required to accomplish the jobs.

Table 5.1: Comparison of developed RL models with EGA

Job Shop Type	EGA	QRL	SARSA
SM	38,980	42,016	23,006
MM_1	7,674	6,711	10,845
MM_2	9,454	5,420	11,236

5.5 Simulation Results and Discussion

The simulated results of the established QRL and SARSA models are discussed in the subsections below; the results are compared to the EGA and NMHPGA, which used identical job shops.

5.5.1 Comparing Developed RL Models with EGA

To compare the findings to basic EGA, three basic job shops presented in Chapter 3 are used (i.e., SM1, MM_1, and MM_2). Table 5.1 compares the results of the established QRL and SARSA models to those of the EGA regarding the objective function. The objective function or cost function is the main character which aims to reduce the makespan of the schedules; when the objective function is the lowest, it means the results are the best with the lowest makespan. The results from the RL models are similar to those from the EGA. However, the RL models are desirable because the EGA is complex and requires numerous functions. The schedule's time is reduced when the objective function is minimised. As demonstrated, SARSA performs better than the QRL on SMs with small schedules. However, the QRL objective functions are preferred in more extensive settings [229].

5.5.2 Comparing Developed RL Models with Advanced Genetic Algorithm

In this part, the results of RL models are compared with NMHPGA, which are tested on identical SM1-SM10 and MM1-MM19 job shops; by reversing the rewards, the minimum time in the program is calculated, wherein the agent increases the reward while the

environment minimises the time. Tables 5.2, 5.3, and 5.4 show the simulation results, comparing the objective functions for the QRL and SARSA models with NMHPGA; the Tables present the simulation results tested on SM (SM: SM1-SM10), MM category-A (MM1-MM10), and MM category-B (MM11-MM19) respectively.

It can be seen from Tables 5.2, 5.3, and 5.4 that the Q agent delivers the maximum reward and lowest objective functions. However, when the size of the job shop is large (MM11-MM19), the proposed QRL algorithm performs poorly compared to the SARSA algorithm. Additionally, as shown in Tables 5.3 and 5.4, by comparing the objective function values, it can be concluded that the NMHPGA technique outperforms the proposed QRL algorithm regarding apparent improvements for MM job shops (because the objective functions of NMHPGA are lower than QRL results).

Results from QRL tested on the SM category are shown in Figure 5.3 (a), and results from MM Categories-A and -B (MM-A, MM-B) are shown in Figures 5.3 (b) and 5.3 (c), respectively. The simulation results for the SM categories SM1–SM10 comprise 10000 episode numbers, which means the agent increases the exploration-exploitation of the environment by 10000 generations. It is indicated that the agent tries exploring-exploitation to learn from the environment to reach the best reward. Figure 5.4 shows simulation results applying the proposed SARSA model on the SM category and MM Category-A and -B. Figure 5.4 (a) shows the rewards of using the SARSA model on SM1-SM10. Figure 5.4 (b) shows the rewards of applying the SARSA model on MM1-MM10 to reach the best rewards. Figure 5.4 (c) shows the rewards of applying the SARSA model on MM11-MM19 to achieve the best.

The SARSA convergence performs better than different algorithms, showing that the RL-based algorithm can solve the FJSP with satisfying solutions. In the RL simulation models, the rewards involve moving between jobs to calculate the makespan, which ultimately becomes an objective function. Regarding the state, the agent moves between jobs to calculate the time, add it to the reward, and calculate the objective function. In this idea, RL learns how to connect environmental states to an agent's actions to maximise the reward. The agent can discover the best action for each state by repeatedly attempting

Table 5.2: SM benchmarks results.

Job shop type	**Number of jobs**	**NMHPGA**	**QRL**	**SARSA**
SM1	32	39,444	42,016	23,006
SM2	40	160,871	297,088	156,270
SM3	60	474,325	816,604	507,100
SM4	80	1,120,440	1,662,800	1,082,100
SM5	100	2,297,979	4,452,100	2,446,362
SM6	120	11,009,721	11,283,000	8,310,500
SM7	150	8,549,596	12,201,000	8,051,900
SM8	200	1,595,2550	26,355,000	20,215,000
SM9	250	30,845,566	58,246,000	39,822,000
SM10	300	52,260,543	90,327,000	65,091,000

Table 5.3: MM category-A results (MM-A).

Job shop type	**NMHPGA**	**QRL**	**SARSA**
MM1	7,758	6,711	7,230
MM2	9,465	5,420	7,490
MM3	7,913	13,558	13,187
MM4	7,758	6,711	7,230
MM5	7,039	10,647	11,542
MM6	7,798	12,863	13,500
MM7	9,181	8,055	8,652
MM8	8,727	17,695	16,094
MM9	6,195	5,379	5,766
MM10	7,758	6,711	7,230

different actions in different states. The agent can also learn the best behaviour through trial-and-error interactions with the environment.

Table 5.4: MM category-B results (MM-B).

Job shop type	Number of jobs	NMHPGA	QRL	SARSA
MM11	4 ×10	16,421	17,700	17,882
MM12	4 ×20	131,115	124,270	125,296
MM13	4 ×30	461,538	429,230	443,460
MM14	4 ×40	1,236,658	1,061,652	1,143,600
MM15	4 ×100	17,432,983	15,698,000	17,035,000
MM16	4 ×150	62,354,613	54567000	60,006,000
MM17	4 ×200	131,840,576	126,430,000	105,410,000
MM18	4 ×250	270,317,679	248,160,000	270,810,000
MM19	4 ×300	475,082,286	417,290,000	420,790,000

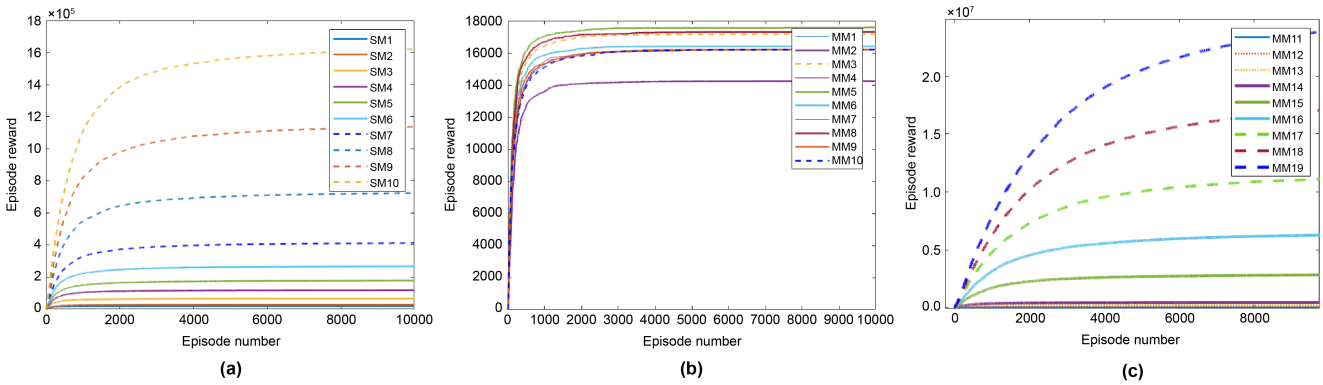


Figure 5.3: QRL training process.

The following are some conclusions from the analysis of this chapter: Based on this study's results, QRL, a fundamental RL model, outperforms the NMHPGA and EGA because GA algorithms are more complex, time-consuming and require additional functions to run the program. Most of the QRL results show the strong performance of the algorithm, except for a few job shops of more considerable sizes. The SARSA model is used to enhance these results. Despite the poor performance of SARSA when there are more job shops, it is still a highly recommended model because it is more straightforward than GA regarding the number of functions and time required to run the program. Future deep RL-based manufacturing research could incorporate additional relevant benchmarks by considering advanced algorithms, optimisations, and parameter tuning in similar simulations. This is because GA techniques are primarily slow and challenging to implement (despite their high performance in optimising scheduling systems). Additionally, different RL models can be integrated with advanced GA models, resulting in a noticeable

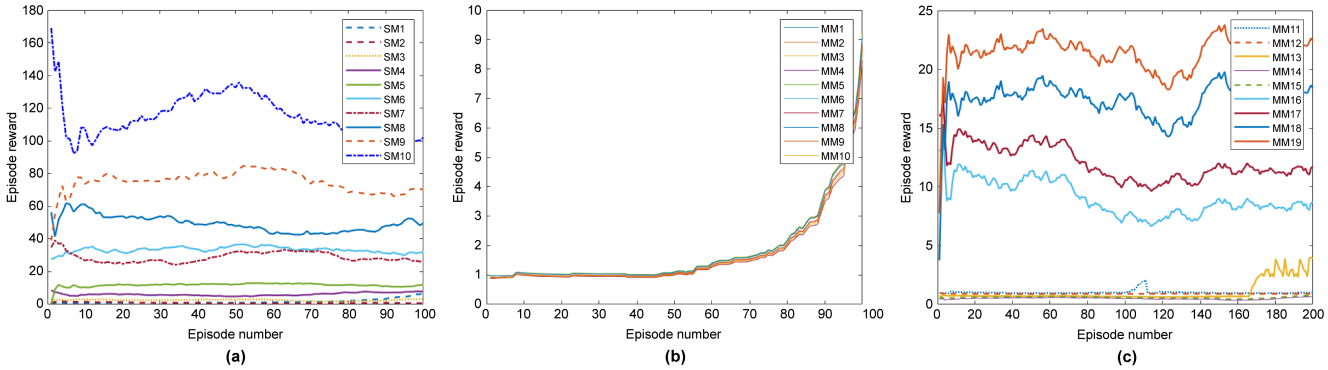


Figure 5.4: SARSA training process.

performance improvement without intensive adaptation efforts.

5.6 Summary

The purpose of this chapter was to present two advanced RL methods for solving FJSPs. These two methods are applied to the simulated benchmarks (job shops) from Chapters 3 and 4, and the results are compared and critically analysed. In the next chapter, the same methods from Chapters 4 and 5 are applied to an industrial case study of the reheating furnace model, and the results are presented. By employing our techniques, the scheduling of the furnace model is optimised. The general aim is to reduce the time and energy consumption in the production process of the furnace.

Chapter 6

Case Study: Reheating Furnace Scheduling

6.1 Introduction

In this chapter, a reheating furnace model is presented as a case study; the same methods of Chapters 4 and 5 are applied to test and optimise the furnace schedule, focusing on reducing the energy and time consumption required for the production process.

To create the optimal site, it is vital in the steel industry to identify the ideal heat treatment regime needed to give steel the correct mechanical qualities. Consequently, creating a system that can choose the best heat treatment regime to get the desired metal qualities (in the shortest amount of time and with the least energy consumption) is crucial. The applied furnace model is derived from a real industrial application.

This chapter provides a general framework of the applied furnace model derived by Yoshitani [230]. The structure of this chapter is as follows:

Section 6.2 describes the features of the furnace model. Section 6.3 provides the GA results testing on the furnace model. Section 6.4 shows the results of the proposed RL models in Chapter 5 tested on the furnace model. Section 6.5 discusses the results, and Section 6.6 summarises the chapter.

6.2 Reheating Furnace

Furnace designs differ depending on their purpose, heating operations, fuel type and technique [230]. In this study, a furnace model is applied to test the optimisers of the thesis. Appendix B shows the equations of the applied furnace model. Figure B.1 shows the design of the used furnace model [230]. Appendix B.2 shows the physical equations of the furnace model. Table B.1 shows the features of the materials used in the furnace. There are 64 strips, and this work aims to find the best order of strips using minimal time and energy. This furnace is intended to heat metal to a temperature of about 1850°C, so high alumina materials were chosen for the walls. This is because its bricks can resist temperatures of up to 1850°C. In this furnace model, a strip of material is heated through a heating process. To prevent quality degradation, the strip is heated according to its composition. The key points of this model can be summarised as follows [230]:

1. **Heating:** Heating occurs as the strip material passes through the furnace. The furnace's temperature and speed are controlled to ensure the strip reaches the desired temperature.

2. **Quenching:** Quenching usually occurs after the strip exits. Air or a cooling liquid would rapidly cool the heated strip.

3. **Tempering:** During tempering, the quenched strip is reheated at a lower temperature than during the initial heating. Various methods could be used, such as passing the strip through a second cooler furnace or using heaters. It has a hierarchical control system with upper- and lower-level controls performing specific tasks. The furnace has several zones, each measuring temperature and fuel flow. The first zone is used for heating jobs, and the second for cooling them. The furnace is quite long (400-500 metres). The strip's temperature can only be measured at the furnace's outlet. An entire strip takes a few minutes to pass through. The control system is made to react to changes in the system, such as variations in fuel flow rate, strip thickness, line speed, etc., in a way that provides the best possible heating of the strip material.

4. **Heat Pattern:** A heat pattern is a plan for heating the strip based on its composition and grade. To ensure the quality of the final product, the strip's actual temperature must stay within the range defined by this heat pattern. The furnace's outlet heat pattern is used as a reference temperature to control the heating process.

5. **Indirect Heating:** Gas-fired radiant tubes heat the strip indirectly. As a result, the furnace's air is heated by the heat of the gas flames.

6. This model also works with various fuels, such as multiple oil grades and petrol fuel. The capacity of the heating system, which can provide a certain amount of heating per unit of time with no energy loss in this situation, can be used to directly compute the net heat available when the energy source is electrical power.

The total amount of heat consumed in this situation will equal the amount lost and the amount of heat absorbed by the workspaces. The following section indicates the results of the furnace schedule testing NMHPGA and developed RL models from previous chapters.

6.3 Genetic Algorithm Testing

Optimum schedules are developed for materials to reduce the amount of time and energy in the heating process. Five algorithms presented in Chapter 4 (i.e., STPGA, RPGA, SPGA, APGA, and NMHPGA) are compared to optimise the furnace model's schedule. The NMHPGA model achieves superior results in reducing the time required to run the furnace model. Moreover, in the next stage, optimisation outcomes from the developed QRL and SARSA models applied in Chapter 5 are contrasted with the GAs outcomes. The applied furnace model is a multi-objective function with three objectives: (1) reducing the time of the schedule, (2) reducing the furnace's energy consumption, and (3) simultaneously reducing the time by 30% and energy by 70%. The latter is the most critical in terms of efficiency. Based on the objectives, the furnace consumes 86.8673 $\times 10^4$ (m^4/h) and takes 139.2167 (h) to operate. Following are the equations related to the furnace objectives.

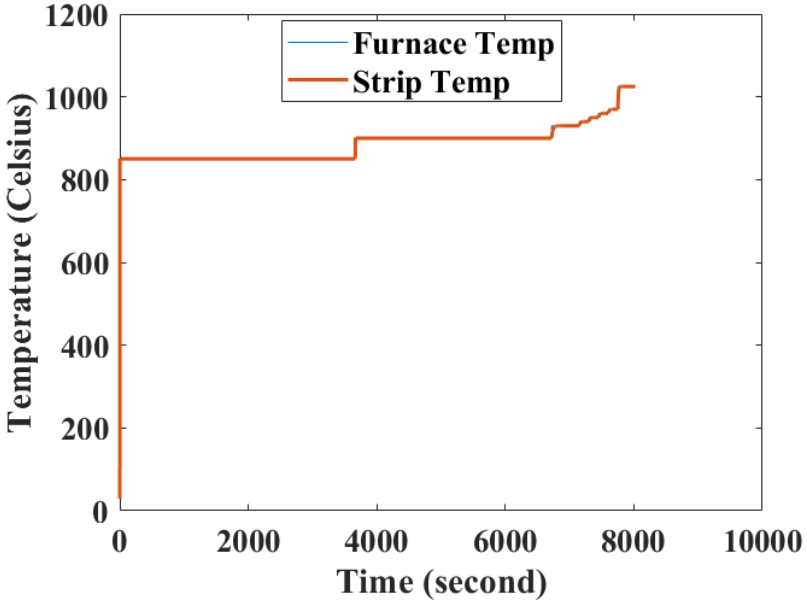


Figure 6.1: NMHPGA results tested on the furnace model using Objective 1.

$$\text{Objective 1} = \sum_{i=1}^{n} T_i \tag{6.1}$$

$$\text{Objective 2} = \sum_{i=1}^{n} F_i \tag{6.2}$$

$$\text{Objective 3} = \sum_{i=1}^{n} (0.3 \times T_i + 0.7 \times F_i) \tag{6.3}$$

where T_i is the time taken for each job, F_i is the fuel consumption per job, and n is the number of jobs.

Figure 6.1 shows the changes in temperature on the furnace by applying NMHPGA schedules based on Objective 1, which concentrates on reducing time. At the beginning of the process, the temperature heats up to 820°C and then increases by 860°C; this process continues, and the schedules are terminated. Similarly, Figures 6.2 and 6.3 show the same method of applying NMHPGA on furnace Objective 2 and Objective 3. The graphs show the elapsed time after optimising the schedules and achieving more efficient results using NMHPGA.

In the next section, the furnace model is used as an environment in the RL models, and RL models optimise the schedule of the furnace to find the best schedule, reducing

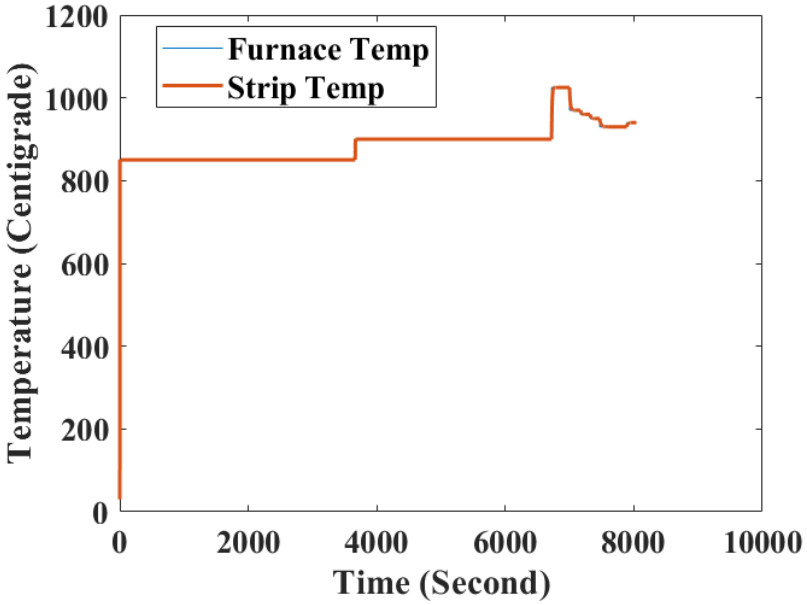


Figure 6.2: NMHPGA results tested on the furnace model using Objective 2.

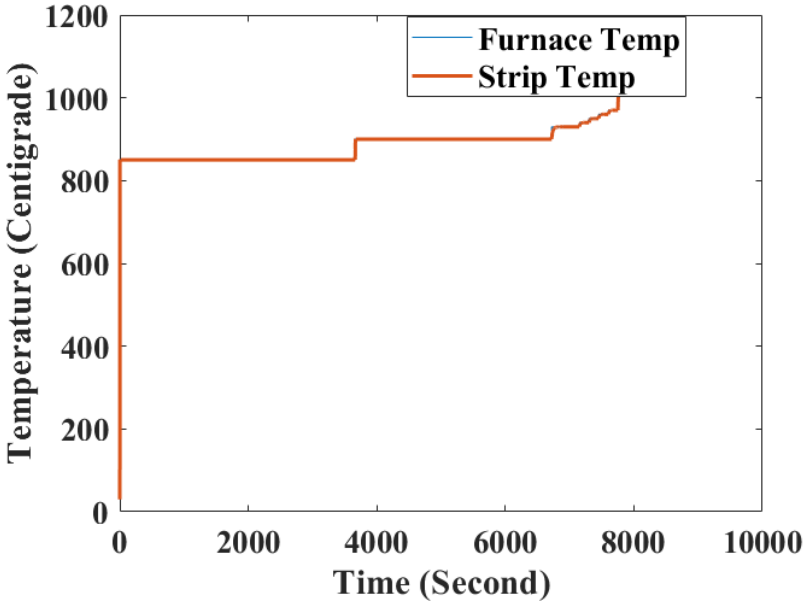


Figure 6.3: NMHPGA results tested on the furnace model using Objective 3.

time and energy.

6.4 Reinforcement Learning Testing

This section uses advanced QRL and SARSA models for the furnace model environment. The following are the model's initial parameters. The environment is a furnace model; the agent initiates the state, learns from the environment, and calculates rewards at each state. The initial state is chosen randomly, and the agents try to learn from the environment.

RL models aim to use less time and energy in the furnace model. The maximum number of iterations is 1000, the learning rate (alpha) is 1, the discount rate (gamma) is 0.09, and the greedy factor (epsilon) is 0.9. Evaluated results on NMHPGA, QRL and SARSA are shown in Tables 6.1, 6.2 and 6.3; Table 6.1 shows the comparison of objective functions of the model applying STPGA, RPGA, SPGA, APGA, NMHPGA, QRL and advanced SARSA.

In Table 6.1, the first row shows Objective 1, which focuses on time reduction only. The objective function is a crucial factor in optimisation, and when it is lower, it shows that the optimisation has a better result in terms of reducing the objective function; here, SARSA has the lowest objective function of 133.9333, which means that the time reduction is more desirable. Moving forward to the second row, Objective 2, which aims at fuel reduction, SARSA has the lowest value of 84.1481 compared to other algorithms, which shows the model's accuracy. Lastly, SARSA has more desirable results for Objective 3, a multi-objective model with the lowest value of 99.0837.

Based on the objectives prior to optimisation, the furnace consumes 86.8673 $\times 10^4$ (m^3/h) of fuel and takes 139.2167 hours to operate. Post-optimization, NMHPGA reduces the operation time to 134.9500 hours, 3.06% improvement, and fuel consumption to 84.0847 $\times 10^4$ (m^3/h), 3.20% improvement. SARSA further reduces the operation time to 133.9333 hours, marking 3.79% improvement over the original and 0.75% better than NMHPGA

The quantity of energy utilised in this model is displayed in Table 6.2, along with

Table 6.1: Comparison of 3 objective functions minimisation of the furnace model.

Function	STPGA	RPGA	SPGA	APGA	NMHPGA	QRL	SARSA
Objective 1	136.1167	136.0167	135.2500	135.7667	134.9500	137.9167	133.9333
Objective 2	85.4268	85.3400	84.9302	85.1881	84.7118	85.5716	84.1481
Objective 3	100.6338	100.5430	100.0261	100.3617	99.7833	100.6319	99.0837

the elapsed time results following schedule optimisation in Table 6.3. According to Table 6.2, SARSA model uses 84.1481 $\times 10^4$ (m^3/h) of fuel, compared to NMHPGA's 83.96 $\times 10^4$ (m^3/h). Despite these variations in fuel usage, SARSA is still preferred due to its simplicity and excellent performance over QRL and complicated GAs.

Table 6.2: Comparison of consumed fuel $\times$ 10^4 (m^3/h) using three objective functions.

Function	STPGA	RPGA	SPGA	APGA	NMHPGA	QRL	SARSA
Objective 1	83.9666	84.0094	84.0703	84.0942	83.9666	84.1481	84.1481
Objective 2	84.3083	84.3982	84.2927	84.2316	84.0847	86.2792	84.1481
Objective 3	83.9666	84.1739	84.0167	84.1451	83.9666	85.2274	84.1481

As indicated in Figure 6.4, the QRL agent increases the exploration-exploitation trade-off. At the same time, the number of episodes grows to learn from the environment and enhance the reward at each state. The graph fluctuation depicts the agent's learning rate for the best reward. The blue line indicates the furnace training model for Objective 1, the red line indicates the training of QRL on furnace Objective 2, and the yellow line indicates Objective 3 activity.

In Figure 6.5, average reward findings using the SARSA model are displayed. Figure 6.5 (a) illustrates the average rewards of the SARSA model when applied to furnace Objective 1, revealing the fluctuations and changes in the rewards to reach the best reward. Figure 6.5 (b) shows the average rewards of the SARSA model applied to furnace Objective 2, highlighting the fluctuations and changes in the rewards to reach the best reward and maximise performance. Figure 6.5 (c) shows the average rewards of the SARSA model applied to the furnace Objective 3 to obtain the best reward.

Table 6.3: Comparison of the elapsed time (h) using the 3 objectives.

Function	STPGA	RPGA	SPGA	APGA	NMHPGA	QRL	SARSA
Objective 1	133.7500	133.8167	133.9833	133.9500	133.7500	133.9333	133.9333
Objective 2	134.4500	134.5833	134.4000	134.3167	134.0000	137.9667	133.9333
Objective 3	133.7500	134.2167	133.8333	134.1500	133.7500	135.6333	133.9333

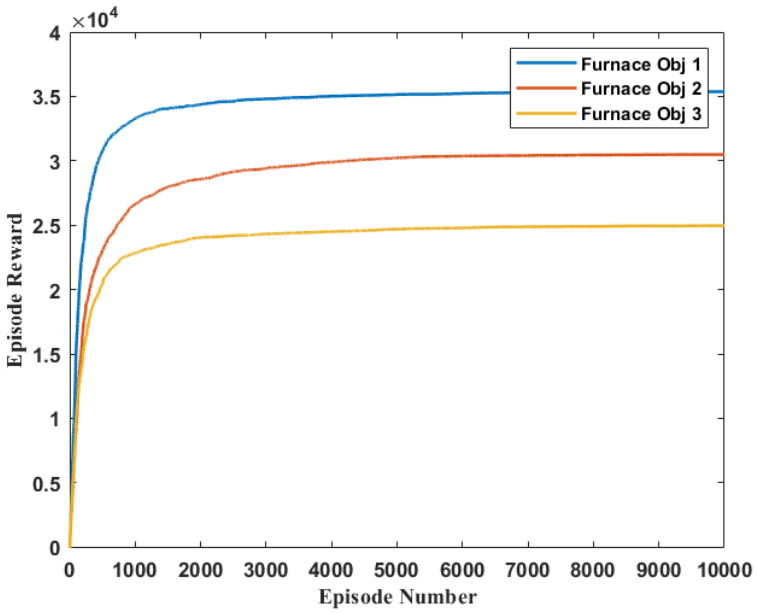


Figure 6.4: QRL model tested on furnace model using 3 objective functions.

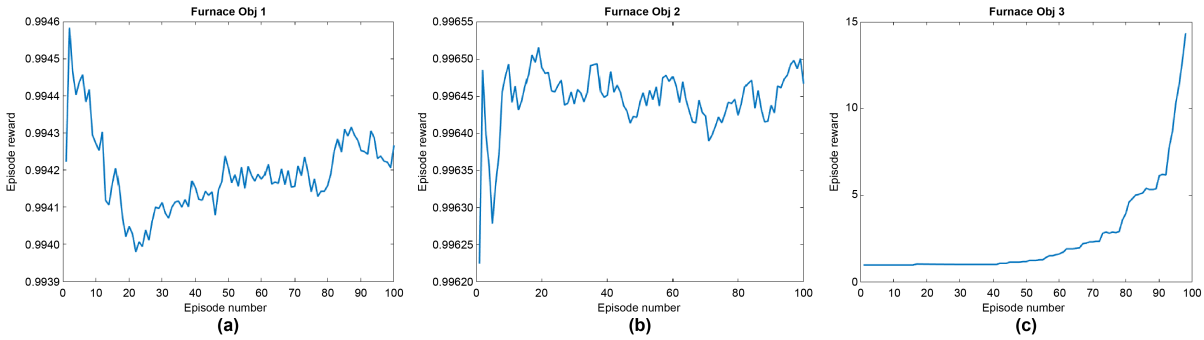


Figure 6.5: SARSA model tested on the reheating furnace using 3 objective functions.

The results for the simulated furnace temperature are shown in Figure 6.6, applying QRL and SARSA. In Figure 6.6, (a), (b), and (c) illustrate the QRL model results, and (d), (e), and (f) indicate the SARSA model results for the furnace model for objectives 1, 2, and 3 (respectively).

Objectives 1 and 2 are still relevant because they optimise for time, regardless of how long it takes (i.e., for urgent tasks) and fuel, respectively. Depending on the consumer's demands, the operator might select a schedule. The corporation will benefit most from Objective 3 because it uses less time and fuel. In conclusion, SARSA always discovers the optimal solution, the best option, for every aim, whereas QRL finds sub-optimal solutions.

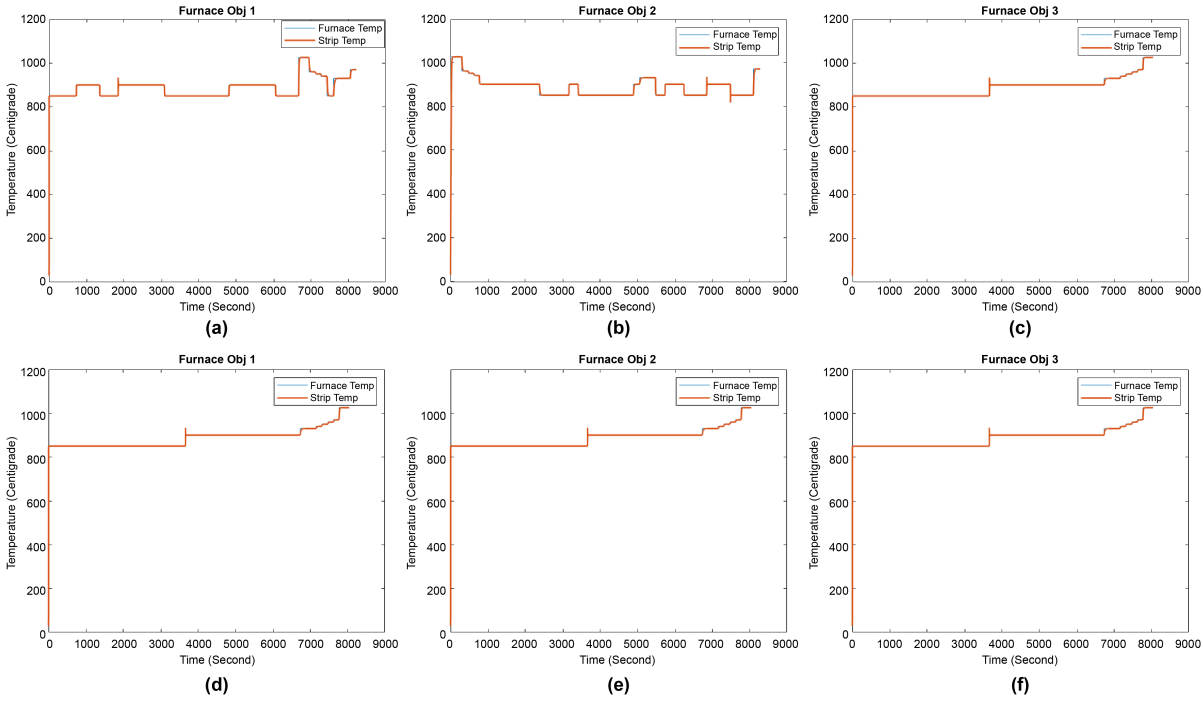


Figure 6.6: Furnace temperature changes applying QRL and SARSA models, Figures (a), (b) and (c) illustrate QRL results testing Objectives 1, 2 and 3. Figures (d), (e) and (f) indicate the results of the SARSA model tested on the furnace testing Objectives 1,2 and 3.

6.5 Discussion of Results

It is evident from the findings displayed in Table 6.2 that SARSA and NMHPGA perform similarly in terms of fuel utilisation. The appeal of SARSA, as opposed to QRL and advanced GAs, rests in its simplicity and higher performance. SARSA provides a good balance between performance and complexity. The third objective in the furnace schedule appears to produce the best results because it minimises time and fuel use.

It is important to note that the first and second objectives (i.e., concerning time and fuel) have particular advantages that warrant careful consideration for actual deployment in practice. For instance, the first target, which minimises time, would be more appropriate for urgent jobs. In situations when there are no vital jobs, and the emphasis is more on energy efficiency, the second objective, which optimises for fuel, would be more pertinent.

Additionally, it is possible that the performance of the algorithms cannot be accurately compared using only one indicator (fuel usage). Future studies may also consider

additional elements, including computing complexity sensitivity to beginning conditions. Moreover, due to its simplicity, reliable performance, and capacity to find the best solutions for various objectives, the results make a compelling argument for using SARSA to optimise industrial production lines. Future studies should examine this performance in different circumstances.

SARSA is fast and precise enough to optimise industrial production lines with reasonable schedules. Although the RL cannot yet be fully implemented in production systems due to the abovementioned restrictions and the challenges of the models described, more studies could hasten this process and enable effective real-world deployments.

6.6 Summary

This chapter presented the case study of a real industrial furnace model to test the methods expounded in Chapter 4 and Chapter 5 to optimise the heat treatment processes. The simulation results were displayed with a detailed description of how this model counts the time and fuel consumption. All primary GAs, NMHPGA, and proposed RL methods have been utilised to optimise the furnace and reduce time and energy costs. The NMHPGA and RL have been successfully developed and operated efficiently. Both methods reached desirable results; however, RL outperformed NMHPGA. The next Chapter concludes the research presented in this thesis and identifies recommendations for future work.

Chapter 7

Conclusions and Future Work

This chapter presents the primary conclusions of this thesis and identifies areas requiring additional research. The research projects discussed in earlier chapters are summarised to reach concluding remarks, and the overall success in achieving the study's goals and objectives is described. Instead of purely theoretical advancements, this research concentrates on the integration problem of process planning in JSSPs and the environment of steel heat treatment operations. Section 7.1 discusses the optimisers reviewed in this thesis; more specifically, Sections 7.2 and 7.3 discuss two types of thesis methodologies, including NMHHPGA and advanced RL models for optimising JSSPs. Section 7.4 discusses final thoughts and upcoming work, including future work recommendations for this thesis. Section 7.5 discusses a general conclusion of this work.

7.1 The Optimisers

This thesis reviewed literature pertaining to intelligent scheduling applications, including GA and RL, for scheduling problems. It demonstrated the importance of FJSP as an essential subset of production scheduling and identified the factors rendering it challenging to solve. Most reviewed studies revealed significant advancements in AI approaches to solve JSSPs and FJSPs. This study provided an overview of scheduling problems and addressed FJSPs in detail. The analysis demonstrates the intricacy of scheduling problems, typically categorised as NP-hard problems. To tackle scheduling problems, this research

focused on FJSPs and has devised several algorithms, including GA, STGA, AGA, RGA, SGA, EGA and PGA, STPGA, APGA, SPGA, RPGA, NMHPGA and RL.

In practice, the identified complexity is mainly caused by the search space size, limitations, and objectives. These algorithms' performance has been enhanced through multiple successful selections introduced in Chapters 4 and 5. These algorithms were then tested utilising a furnace model and SM and MM job shops. The NMHPGA and RL techniques proposed in this thesis outperform the standard optimisers; these methods are discussed briefly in the next section.

7.2 NMHPGA and EGA

To optimise FJSPs and the reheating furnace schedule, this thesis proposed NMHPGA in Chapter 4. Combinations of various selection types support this GA-based approach, suggested EGA and hybrid PGAs. This thesis examined the PGA (which includes the parthenogenetic operators swap, reverse, and insert) and an EGA. To create an ethnic population, the ethnic selection process employed a variety of selection operators (including stochastic, roulette, sexual, and ageing). The best individuals were then chosen from each technique and were blended. The established algorithm's results were compared to those of other selection methods, and it became clear that the NMHPGA was producing superior objective functions with a faster convergence rate. This work addressed several GA types with various selection types.

PGA and ethnic selection demonstrate that superior outcomes can be obtained without time-consuming crossover processes. This method can improve the global search point and convergence speed. NMHPGA, which replicates the crossover function using the swap, insert, and reverse functions, increased the efficiency and performance of the job shop schedules.

Convergence speed involves determining the iteration where the error (cost) reaches a steady state; this indicates that the best solution has been found, and the algorithm can stop because the error will not change. This practice helps find the best solution quickly, requiring less time and effort.

7.3 Reinforcement Learning

The performance of the scheduling methods was improved, as presented in Chapters 5 and 6, using the developed RL models, including the QRL and advanced SARSA algorithm. The findings were compared with EGA and NMHPGA. Compared to NMHPGA and basic GA from earlier works (as presented in Chapter 4), the simulation findings based on various performance metrics showed that the proposed SARSA algorithm could be a competitive approach for JSSPs; this is because of the reduction of makespan of the schedules and increasing the convergence speed of the models. The RL application in this thesis can be summarised as follows: To validate the performance of the novel models, three simulated environments were used for reference comparisons—a furnace model, ten single flexible machines job shops, and 19 flexible MM job shops. The results of EGA and NMHPGA and other preceding studies were compared with the simulated outputs of these environments. The results of the SARSA show how the proposed model enhances system optimisation efficiency.

7.4 Final Thoughts and Future Work

The following are some salient outcomes of the work presented in this thesis.

1. According to the presented results, QRL, a fundamental RL model, outperforms NMHPGA and EGA. This is attributable to GA algorithms being more complex and time-consuming and requiring additional functions to run the program.

2. Although SARSA performs poorly when there are more job shops, it is still a highly recommended model because it is more straightforward than GA regarding the number of functions and the time needed to run the program. In addition, SARSA performs better than NMHPGA and QRL by optimising the furnace model. It can be said that SARSA is quick and precise enough to optimise industrial production lines for schedules of a reasonable size.

3. The relevant system standards might be created to enable more complex and ad-

vanced industrial models, facilitating an accelerated escalation from simulation to real-world application.

4. By considering cutting-edge algorithms, optimisations and parameter tuning inside the same simulations, future advanced RL-based manufacturing, including DRL research, can incorporate more pertinent benchmarks.

5. Additionally, advanced GA models can be combined with a variety of RL models, and performance could be noticeably improved without the need for extensive adaptation efforts.

7.5 Conclusion

This study investigated interactions between intelligent scheduling and factory scheduling, emphasising the roles of GA and RL. The dynamic and intricate nature of manufacturing scheduling was acknowledged throughout the study, emphasising how urgent it is to create more effective and flexible algorithms than those currently available in practice. The analysis of novel GA and RL models revealed their potential to transform how businesses address scheduling problems. These algorithms' potential was demonstrated through real-world use in a model of an industrial furnace, offering insightful information for upcoming industrial uses.

The research addressed scheduling and provided a more profound knowledge of the difficulties in balancing many goals in a scheduling problem. The comparison of various scheduling algorithms revealed the advantages and disadvantages of each approach, offering a foundation for future research. Hybrid models were created and tested, and the results showed that performance could be improved by combining the advantages of various algorithms.

This thesis closes a substantial knowledge gap by offering a thorough grasp of the application of scheduling methods for challenging dynamic scheduling problems. Future studies could explore several areas after looking into multi-objective scheduling, hybrid models and more complicated real-world applications. The road to intelligent industrial

systems is a difficult and drawn-out one. Still, this thesis's framework can offer guidance for many intriguing and promising opportunities for further studies.

Bibliography

[1] Z. Wei, W. Liao, and L. Zhang, "Hybrid energy-efficient scheduling measures for flexible job-shop problem with variable machining speeds," *Expert Systems with Applications*, vol. 197, 7 2022.

[2] G. Ambrogio, R. Guido, D. Palaia, and L. Filice, "Job shop scheduling model for a sustainable manufacturing," *Procedia Manufacturing*, vol. 42, pp. 538–541, 1 2020.

[3] R. Cheng, M. Gen, and Y. Tsujimura, "A tutorial survey of job-shop scheduling problems using genetic algorithms, part ii: hybrid genetic search strategies," *Computers Industrial Engineering*, vol. 36, pp. 343–364, 4 1999.

[4] F. Zhang, G. S. Member, Y. Mei, S. Member, S. Nguyen, M. Zhang, and S. N. is, "Correlation coefficient-based recombinative guidance for genetic programming hyperheuristics in dynamic flexible job shop scheduling," *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, vol. 25, 2021. [Online]. Available: https://doi.org/10.1109/TEVC.2021.3056143.

[5] J. M. Fernandes, S. M. Homayouni, and D. B. Fontes, "Energy-efficient scheduling in job shop manufacturing systems: A literature review," *Sustainability (Switzerland)*, vol. 14, p. 6264, 5 2022. [Online]. Available: https://www.mdpi.com/2071-1050/14/10/6264/htmhttps://www.mdpi.com/2071-1050/14/10/6264

[6] Y. Zhang, H. Zhu, D. Tang, T. Zhou, and Y. Gui, "Dynamic job shop scheduling based on deep reinforcement learning for multi-agent manufacturing systems," *Robotics and Computer-Integrated Manufacturing*, vol. 78, p. 102412, 12 2022.

[7] F. Pezzella, G. Morganti, and G. Ciaschetti, "A genetic algorithm for the flexible job-shop scheduling problem," *Computers and Operations Research*, vol. 35, pp. 3202–3212, 10 2008.

[8] R. Chen, W. Li, and H. Yang, "A deep reinforcement learning framework based on an attention mechanism and disjunctive graph embedding for the job-shop scheduling problem," *IEEE Transactions on Industrial Informatics*, vol. 19, pp. 1322–1331, 2 2023.

[9] H. Ishibuchi, T. Yoshida, and T. Murata, "Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling," *IEEE Transactions on Evolutionary Computation*, vol. 7, pp. 204–223, 4 2003.

[10] M. R. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *https://doi.org/10.1287/moor.1.2.117*, vol. 1, pp. 117–129, 5 1976. [Online]. Available: https://pubsonline.informs.org/doi/abs/10.1287/moor.1.2.117

[11] A. Ebrahimi, H. W. Jeon, S. Lee, and C. Wang, "Minimizing total energy cost and tardiness penalty for a scheduling-layout problem in a flexible job shop system: A comparison of four metaheuristic algorithms," 2020. [Online]. Available: https://doi.org/10.1016/j.cie.2020.106295

[12] S. M. Johnson, "Optimal two- and three-stage production schedules with setup times included," *Naval Research Logistics Quarterly*, vol. 1, pp. 61–68, 3 1954.

[13] L. Waltersmann, S. Kiemel, J. Stuhlsatz, A. Sauer, and R. Miehe, "Artificial intelligence applications for increasing resource efficiency in manufacturing companies—a comprehensive review," *Sustainability 2021, Vol. 13, Page 6689*, vol. 13, p. 6689, 6 2021. [Online]. Available: https://www.mdpi.com/2071-1050/13/12/6689/htmhttps://www.mdpi.com/2071-1050/13/12/6689

[14] J. Ahuja, T. K. Panda, S. Luthra, A. Kumar, S. Choudhary, and J. A. Garza-Reyes, "Do human critical success factors matter in adoption of sustainable manufacturing

practices? an influential mapping analysis of multi-company perspective," *Journal of Cleaner Production*, vol. 239, p. 117981, 12 2019.

[15] G. Seliger, H. J. Kim, S. Kernbaum, and M. Zettl, "Approaches to sustainable manufacturing," *International Journal of Sustainable Manufacturing*, vol. 1, pp. 58–77, 2008.

[16] P. Renna, S. M. A. Sciences, and undefined 2021, "A literature review of energy efficiency and sustainability in manufacturing systems," *mdpi.comP Renna, S MateriApplied Sciences, 2021•mdpi.com*, vol. 11, 8 2021. [Online]. Available: https://www.mdpi.com/2076-3417/11/16/7366

[17] C. G. Machado, M. P. Winroth, and E. H. D. R. da Silva, "Sustainable manufacturing in industry 4.0: an emerging research agenda," *International Journal of Production Research*, vol. 58, pp. 1462–1484, 3 2020. [Online]. Available: https://www.tandfonline.com/action/journalInformation?journalCode=tprs20

[18] J. Malek and T. N. Desai, "A systematic literature review to map literature focus of sustainable manufacturing," *Journal of Cleaner Production*, vol. 256, p. 120345, 5 2020.

[19] J. Q. Li, Y. Du, K. Z. Gao, P. Y. Duan, D. W. Gong, Q. K. Pan, and P. N. Suganthan, "A hybrid iterated greedy algorithm for a crane transportation flexible job shop problem," *IEEE Transactions on Automation Science and Engineering*, vol. 19, pp. 2153–2170, 7 2022.

[20] L. Meng, Y. Ren, B. Zhang, J. Q. Li, H. Sang, and C. Zhang, "Milp modeling and optimization of energy-efficient distributed flexible job shop scheduling problem," *IEEE Access*, vol. 8, pp. 191 191–191 203, 2020.

[21] T. Stock and G. Seliger, "Opportunities of sustainable manufacturing in industry 4.0," *Procedia CIRP*, vol. 40, pp. 536–541, 1 2016.

[22] Q. Xu, L. Zhang, and W. Yu, "A localization method of ant colony optimization in nonuniform space," *Sensors 2022, Vol. 22, Page 7389*, vol. 22, p. 7389, 9 2022.

[Online]. Available: https://www.mdpi.com/1424-8220/22/19/7389/htmhttps://www.mdpi.com/1424-8220/22/19/7389

[23] Z. Chang, S. Song, Y. Zhang, J. Y. Ding, R. Zhang, and R. Chiong, "Distributionally robust single machine scheduling with risk aversion," *European Journal of Operational Research*, vol. 256, pp. 261–274, 1 2017.

[24] H. Xiong, S. Shi, D. Ren, and J. Hu, "A survey of job shop scheduling problem: The types and models," *Computers Operations Research*, vol. 142, p. 105731, 6 2022.

[25] T. Ning, Z. Wang, P. Zhang, and T. Gou, "Integrated optimization of disruption management and scheduling for reducing carbon emission in manufacturing," 2020. [Online]. Available: https://doi.org/10.1016/j.jclepro.2020.121449

[26] K. Fang, N. Uhan, F. Zhao, and J. W. Sutherland, "A new approach to scheduling in manufacturing for power consumption and carbon footprint reduction," *Journal of Manufacturing Systems*, vol. 30, pp. 234–240, 10 2011.

[27] A. Giret, D. Trentesaux, and V. Prabhu, "Sustainability in manufacturing operations scheduling: A state of the art review," *Journal of Manufacturing Systems*, vol. 37, pp. 126–140, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.jmsy.2015.08.002

[28] B. L. Maccarthy and J. Liu, "Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling," *https://doi.org/10.1080/00207549308956713*, vol. 31, pp. 59–79, 2007.

[29]

[30] Y. Li, C. Liao, L. Wang, Y. Xiao, Y. Cao, and S. Guo, "A reinforcement learning-artificial bee colony algorithm for flexible job-shop scheduling problem with lot streaming," *Applied Soft Computing*, vol. 146, p. 110658, 10 2023.

[31] Y. Mauergauz, "Advanced planning and scheduling in manufacturing and supply chains," *Advanced Planning and Scheduling in Manufacturing and Supply Chains*, pp. 1–570, 4 2016.

[32] D. Oliva, E. H. Houssein, and S. Hinojosa, "Studies in computational intelligence 967 metaheuristics in machine learning: Theory and applications," 2021. [Online]. Available: http://www.springer.com/series/7092

[33] W. Karush and L. A. Moody, "Determination of feasible shipping schedules for a job shop," *Operations Research*, vol. 6, pp. 35–55, 2 1958.

[34] P. Mellor, "A review of job shop scheduling," *OR*, vol. 17, p. 161, 6 1966.

[35] A. S. Manne, "On the job-shop scheduling problem," *Operations Research*, vol. 8, pp. 219–223, 4 1960.

[36] G. Chryssolouris, S. Chan, and W. Cobb, "Decision making on the factory floor: An integrated approach to process planning and scheduling," *Robotics and Computer-Integrated Manufacturing*, vol. 1, pp. 315–319, 1 1984.

[37] Z. Li and J. Liu, "A multi-agent genetic algorithm for community detection in complex networks," *Physica A: Statistical Mechanics and its Applications*, vol. 449, pp. 336–347, 5 2016.

[38] M. Nouiri, A. Bekrar, Abderezak, J. Smail, N. Ahmed, C. Ammari, M. Nouiri, A. Jemai, A. C. Ammari, A. Bekrar, and S. Niar, "An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem," *J Intell Manuf*, vol. 29, pp. 603–615, 2018. [Online]. Available: https://doi.org/10.1007/s10845-015-1039-3

[39] R. Liu, R. Piplani, and C. Toro, "Deep reinforcement learning for dynamic scheduling of a flexible job shop," *International Journal of Production Research*, vol. 60, pp. 4049–4069, 7 2022. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/00207543.2022.2058432

[40] I. A. Chaudhry and A. A. Khan, "A research survey: Review of flexible job shop scheduling techniques," *International Transactions in Operational Research*, vol. 23, pp. 551–591, 5 2016.

[41] J. Fan, C. Zhang, Q. Liu, W. Shen, and L. Gao, "An improved genetic algorithm for flexible job shop scheduling problem considering reconfigurable machine tools with limited auxiliary modules," *Journal of Manufacturing Systems*, vol. 62, pp. 650–667, 1 2022.

[42] K. Gao, Z. Cao, L. Zhang, Z. Chen, Y. Han, and Q. Pan, "A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems; a review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, 2019. [Online]. Available: http://ieeexplore.ieee.org.

[43] W. Han, Q. Deng, G. Gong, L. Zhang, and Q. Luo, "Multi-objective evolutionary algorithms with heuristic decoding for hybrid flow shop scheduling problem with worker constraint," *Expert Systems with Applications*, vol. 168, p. 114282, 4 2021.

[44] T. Meng, Q. K. Pan, and H. Y. Sang, "A hybrid artificial bee colony algorithm for a flexible job shop scheduling problem with overlapping in operations," *https://doi.org/10.1080/00207543.2018.1467575*, vol. 56, pp. 5278–5292, 8 2018. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/00207543.2018.1467575

[45] K. Z. Gao, N. Suganthan, Q. K. Pan, M. F. Tasgetiren, and A. Sadollah, "Knowle dge-base d systems artificial bee colony algorithm for scheduling and rescheduling fuzzy flexible job shop problem with new job insertion," vol. 109, pp. 1–16, 2016. [Online]. Available: http://dx.doi.org/10.1016/j.knosys.2016.06.014

[46] S. S. Amelian, S. M. Sajadi, M. Navabakhsh, and M. Esmaelian, "Multi-objective optimization for stochastic failure-prone job shop scheduling problem via hybrid of nsga-ii and simulation method," *Wiley Online LibrarySS Amelian, SM Sajadi,*

M Navabakhsh, M EsmaelianExpert Systems, 2022•Wiley Online Library, vol. 39, 2 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/exsy.12455

[47] P. Ciepliński, S. Golak, M. Blachnik, K. Gawryś, and A. Kachel, "Production scheduling methodology, taking into account the influence of the selection of production resources," *Applied Sciences 2022, Vol. 12, Page 5367*, vol. 12, p. 5367, 5 2022. [Online]. Available: https://www.mdpi.com/2076-3417/12/11/5367/htmhttps://www.mdpi.com/2076-3417/12/11/5367

[48] D. Alemão, A. D. Rocha, and J. Barata, "Smart manufacturing scheduling approaches—systematic review and future directions," *Applied Sciences 2021, Vol. 11, Page 2186*, vol. 11, p. 2186, 3 2021. [Online]. Available: https://www.mdpi.com/2076-3417/11/5/2186/htmhttps://www.mdpi.com/2076-3417/11/5/2186

[49] L. Monostori, "Cyber-physical production systems: Roots, expectations and rd challenges," *Procedia CIRP*, vol. 17, pp. 9–13, 1 2014.

[50] A. Nassehi, R. Y. Zhong, X. Li, and B. I. Epureanu, "Review of machine learning technologies and artificial intelligence in modern manufacturing systems," *Design and Operation of Production Networks for Mass Personalization in the Era of Cloud Technology*, pp. 317–348, 1 2022.

[51] G. E. Vieira, J. W. Herrmann, and E. Lin, "Rescheduling manufacturing systems: A framework of strategies, policies, and methods," *Journal of Scheduling*, vol. 6, pp. 39–62, 1 2003. [Online]. Available: https://link.springer.com/article/10.1023/a:1022235519958

[52] P. Brucker and R. Schlie, "Job-shop scheduling with multi-purpose machines," *Computing*, vol. 45, pp. 369–375, 12 1990.

[53] P. Brandimarte, "Routing and scheduling in a flexible job shop by tabu search," *Annals of Operations Research*, vol. 41, pp. 157–183, 1993.

[54] D. Biskup, "Single-machine scheduling with learning considerations," *European Journal of Operational Research*, vol. 115, pp. 173–178, 5 1999.

[55] J. Wang, D. Lei, and M. Li, "A q-learning-based artificial bee colony algorithm for distributed three-stage assembly scheduling with factory eligibility and setup times," *Machines 2022, Vol. 10, Page 661*, vol. 10, p. 661, 8 2022. [Online]. Available: https://www.mdpi.com/2075-1702/10/8/661/htmhttps://www.mdpi.com/2075-1702/10/8/661

[56] M. L. Pinedo, "Scheduling: Theory, algorithms, and systems: Fourth edition," *Scheduling: Theory, Algorithms, and Systems: Fourth Edition*, vol. 9781461423614, pp. 1–673, 2 2012.

[57] J. B. Wang, C. T. Ng, T. C. Cheng, and L. L. Liu, "Single-machine scheduling with a time-dependent learning effect," *International Journal of Production Economics*, vol. 111, pp. 802–811, 2 2008.

[58] C. Y. Lee, S. Piramuthu, and Y. K. Tsai, "Job shop scheduling with a genetic algorithm and machine learning," *https://doi.org/10.1080/002075497195605*, vol. 35, pp. 1171–1191, 2010. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/002075497195605

[59] Y. Y. Lu, C. M. Wei, and J. B. Wang, "Several single-machine scheduling problems with general learning effects," *Applied Mathematical Modelling*, vol. 36, pp. 5650–5656, 11 2012.

[60] C. Y. Cheng, S. F. Li, K. C. Ying, and Y. H. Liu, "Scheduling jobs of two competing agents on a single machine," *IEEE Access*, vol. 7, pp. 98 702–98 714, 2019.

[61] M. de Weerdt, R. Baart, and L. He, "Single-machine scheduling with release times, deadlines, setup times, and rejection," *European Journal of Operational Research*, vol. 291, pp. 629–639, 6 2021.

[62] D. Lei and S. He, "An adaptive artificial bee colony for unrelated parallel machine scheduling with additional resource and maintenance," *Expert*

Systems with Applications, vol. 205, p. 117577, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417422008909

[63] R. Chen, B. Yang, S. Li, and S. Wang, "A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem," 2020. [Online]. Available: https://doi.org/10.1016/j.cie.2020.106778

[64] I. Kacem, S. Hammadi, and P. Borne, "Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 32, pp. 1–13, 2 2002.

[65] T. Yamada and R. Nakano, "Fusion of crossover and local search," *Proceedings of the IEEE International Conference on Industrial Technology*, pp. 426–430, 1996.

[66] Y. Yoshitomi, "A genetic algorithm approach to solving stochastic job-shop scheduling problems," *International Transactions in Operational Research*, vol. 9, pp. 479–495, 2002.

[67] G. Zhang, X. Shao, P. Li, and L. Gao, "An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem," *Computers Industrial Engineering*, vol. 56, pp. 1309–1318, 5 2009.

[68] L. N. Xing, Y. W. Chen, P. Wang, Q. S. Zhao, and J. Xiong, "A knowledge-based ant colony optimization for flexible job shop scheduling problems," *Applied Soft Computing*, vol. 10, pp. 888–896, 6 2010.

[69] G. Zhang, L. Gao, and Y. Shi, "An effective genetic algorithm for the flexible job-shop scheduling problem," *Expert Systems with Applications*, vol. 38, pp. 3563–3573, 4 2011.

[70] M. Nouiri, A. Bekrar, A. Jemai, D. Trentesaux, A. C. Ammari, and S. Niar, "Two stage particle swarm optimization to solve the flexible job shop predictive scheduling problem considering possible machine breakdowns," *Computers Industrial Engineering*, vol. 112, pp. 595–606, 10 2017.

[71] X. Li, Z. Peng, B. Du, J. Guo, W. Xu, and K. Zhuang, "Hybrid artificial bee colony algorithm with a rescheduling strategy for solving flexible job shop scheduling problems," *Computers and Industrial Engineering*, vol. 113, pp. 10–26, 11 2017.

[72] E.-D. Jiang and L. Wang, "An improved multi-objective evolutionary algorithm based on decomposition for energy-efficient permutation flow shop scheduling problem with sequence-dependent setup time," 2018. [Online]. Available: https://doi.org/10.1080/00207543.2018.1504251

[73] L. Meng, C. Zhang, X. Shao, and Y. Ren, "Milp models for energy-aware flexible job shop scheduling problem," *Journal of Cleaner Production*, vol. 210, pp. 710–723, 2 2019.

[74] R. H. Caldeira, A. Gnanavelbabu, and T. Vaidyanathan, "An effective backtracking search algorithm for multi-objective flexible job shop scheduling considering new job arrivals and energy consumption," *Computers and Industrial Engineering*, vol. 149, 11 2020.

[75] B. Li, H. Zhang, P. Ye, and J. Wang, "Trajectory smoothing method using reinforcement learning for computer numerical control machine tools," *Robotics and Computer-Integrated Manufacturing*, vol. 61, p. 101847, 2 2020.

[76] H. Wei, S. Li, H. Quan, D. Liu, S. Rao, C. Li, and J. Hu, "Unified multi-objective genetic algorithm for energy efficient job shop scheduling," *IEEE Access*, vol. 9, pp. 54 542–54 557, 2021.

[77] W. Xu, Y. Hu, W. Luo, L. Wang, and R. Wu, "A multi-objective scheduling method for distributed and flexible job shop based on hybrid genetic algorithm and tabu search considering operation outsourcing and carbon emission," *Computers and Industrial Engineering*, vol. 157, 7 2021.

[78] N. Rakovitis, D. Li, N. Zhang, J. Li, L. Zhang, and X. Xiao, "Novel approach to energy-efficient flexible job-shop scheduling problems," *Energy*, vol. 238, 1 2022.

[79] W. Lin, G. Tian, Z. Li, Y. Zhang, and C. Zhang, "Flow shop scheduling with low carbon emission and variable machining parameters," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 233, pp. 1561–1572, 4 2019. [Online]. Available: https://journals-sagepub-com.ezproxy.brunel.ac.uk/doi/10.1177/0954405418782290

[80] X. Wu, J. Zhao, and Y. Tong, "Big data analysis and scheduling optimization system oriented assembly process for complex equipment," *IEEE Access*, vol. 6, pp. 36 479–36 486, 7 2018.

[81] C. Lu, L. Gao, X. Li, Q. Pan, and Q. Wang, "Energy-efficient permutation flow shop scheduling problem using a hybrid multi-objective backtracking search algorithm," *Journal of Cleaner Production*, vol. 144, pp. 228–238, 2 2017.

[82] A. Momenikorbekandi and M. Abbod, "A novel metaheuristic hybrid parthenogenetic algorithm for job shop scheduling problems: Applying optimization model," *IEEE Access*, pp. 1–1, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10130171/

[83] H. Y. Chong, H. J. Yap, S. C. Tan, K. S. Yap, and S. Y. Wong, "Advances of metaheuristic algorithms in training neural networks for industrial applications," *Soft Computing*, vol. 25, no. 16, pp. 11 209–11 233, 2021.

[84] Z. Jiang, S. Yuan, J. Ma, and Q. Wang, "The evolution of production scheduling from industry 3.0 through industry 4.0," *https://doi.org/10.1080/00207543.2021.1925772*, vol. 60, pp. 3534–3554, 2021. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/00207543.2021.1925772

[85] F. Glover, "Future paths for integer programming and links to ar tifi cial intelli g en ce," *Comput. Ops. Res*, vol. 13, pp. 533–549, 1986.

[86] Q. V. Dang, T. van Diessen, T. Martagan, and I. Adan, "A matheuristic for par-

allel machine scheduling with tool replacements," *European Journal of Operational Research*, vol. 291, pp. 640–660, 6 2021.

[87] K. Tamssaouet, S. Dauzère-Pérès, and C. Yugma, "Metaheuristics for the job-shop scheduling problem with machine availability constraints," *Computers and Industrial Engineering*, vol. 125, pp. 1–8, 11 2018.

[88] R. Storn and K. Price, "Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, pp. 341–359, 1997.

[89] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies a comprehensive introduction," *ACM Computing Classification*, vol. 1, pp. 3–52, 2002.

[90] J. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*, 1992. [Online]. Available: https://books.google.com/books?hl=en&lr=&id=5EgGaBkwvWcC&oi=fnd&pg=PR7&ots=mJmn63Frqj&sig=glY5NJEG20wcdCO1sAhtqvxS6Ck

[91] D. B. Fogel, "Evolutionary computation: The fossil record. selected readings on the history of evolutionary algorithms," p. 641, 1998.

[92] M. K. Tiwari and F. T. Chan, "Swarm intelligence, focus on ant and particle swarm optimization," *Swarm Intelligence, Focus on Ant and Particle Swarm Optimization*, p. 548, 3 2019.

[93] R. Eberhart and J. Kennedy, "New optimizer using particle swarm theory," *Proceedings of the International Symposium on Micro Machine and Human Science*, pp. 39–43, 1995.

[94] S. Walton, O. Hassan, K. Morgan, and M. R. Brown, "A review of the development and applications of the cuckoo search algorithm," *Swarm Intelligence and Bio-Inspired Computation: Theory and Applications*, pp. 257–271, 1 2013.

[95] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, 5 1983. [Online]. Available: https://www.science.org/doi/10.1126/science.220.4598.671

[96] E. da Jiang and L. Wang, "Multi-objective optimization based on decomposition for flexible job shop scheduling under time-of-use electricity prices," *Knowledge-Based Systems*, vol. 204, p. 106177, 9 2020.

[97] J. H. Holland, "Genetic algorithms and adaptation," pp. 317–333, 1984. [Online]. Available: https://link.springer.com/chapter/10.1007/978-1-4684-8941-5_21

[98] D. E. Goldberg and J. H. Holland, "Genetic algorithms in search, optimization, and machine learning david e. goldberg the university of alabama t," *Machine Learning*, vol. 3, pp. 95–99, 1979.

[99] D. Whitley, "An overview of evolutionary algorithms: practical issues and common pitfalls," *Information and Software Technology*, vol. 43, pp. 817–831, 12 2001.

[100] A. Ghosh, S. Tsutsui, and H. Tanaka, "Individual aging in genetic algorithms," *Proceedings of the Australian and New Zealand Conference on Intelligent Information Systems*, pp. 276–279, 1996.

[101] C. Zhao and Z. Wu, "A genetic algorithm approach to the scheduling of fmss with multiple routes," *International Journal of Flexible Manufacturing Systems*, vol. 13, pp. 71–88, 2 2001. [Online]. Available: https://link.springer.com/article/10.1023/A:1008148313360

[102] I. Kacem, "Genetic algorithm for the flexible job-shop scheduling problem," *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, vol. 4, pp. 3464–3469, 2003.

[103] K. S. Goh, A. Lim, and B. Rodrigues, "Sexual selection for genetic algorithms," *Artificial Intelligence Review*, vol. 19, pp. 123–152, 4 2003. [Online]. Available: https://link.springer.com/article/10.1023/A:1022692631328

[104] K. F. Guimarães and M. A. Fernandes, "An approach for flexible job-shop scheduling with separable sequence-dependent setup time," *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, vol. 5, pp. 3727–3731, 2006.

[105] F. T. Chan, S. H. Chung, and L. Y. Chan, "An introduction of dominant genes in genetic algorithm for fms," *http://dx.doi.org/10.1080/00207540600632190*, vol. 46, pp. 4369–4389, 8 2008. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/00207540600632190

[106] M. Zhang, W. Luo, and X. Wang, "Differential evolution with dynamic stochastic selection for constrained optimization," *Information Sciences*, vol. 178, pp. 3043–3074, 8 2008.

[107] L. D. Giovanni and F. Pezzella, "An improved genetic algorithm for the distributed and flexible job-shop scheduling problem," *European Journal of Operational Research*, vol. 200, pp. 395–408, 1 2010.

[108] P. Unachak and E. Goodman, "Solving multiobjective flexible job-shop scheduling using an adaptive representation," *Proceedings of the 12th Annual Genetic and Evolutionary Computation Conference, GECCO '10*, pp. 737–742, 2010. [Online]. Available: https://dl.acm.org/doi/10.1145/1830483.1830615

[109] X. H. Xu, L. L. Zeng, and Y. W. Fu, "Hybrid particle swarm optimization for flexible job-shop scheduling problem and its implementation?" *2010 IEEE International Conference on Information and Automation, ICIA 2010*, pp. 1155–1159, 2010.

[110] G. Zhang, L. Gao, and Y. Shi, "A genetic algorithm and tabu search for multi objective flexible job shop scheduling problems," *2010 International Conference on Computing, Control and Industrial Engineering, CCIE 2010*, vol. 1, pp. 251–254, 2010.

[111] D. Hu and Z. Yao, "Stacker-reclaimer scheduling for raw material yard operation," *3rd International Workshop on Advanced Computational Intelligence, IWACI 2010*, pp. 432–436, 2010.

[112] M. Wan, X. Xu, and J. Nan, "Flexible job-shop scheduling with integrated genetic algorithm," *Proceedings of 4th International Workshop on Advanced Computational Intelligence, IWACI 2011*, pp. 13–16, 2011.

[113] J. Jiang, M. Wen, K. Ma, X. Long, and J. Li, "Hybrid genetic algorithm for flexible job-shop scheduling with multi-objective," *JOURNAL OF INFORMATION &COMPUTATIONAL SCIENCE*, vol. 8, no. 11, pp. 2197–2205, 2011.

[114] L. N. Xing, Y. W. Chen, and K. W. Yang, "Multi-population interactive coevolutionary algorithm for flexible job shop scheduling problems," *Computational Optimization and Applications*, vol. 48, pp. 139–155, 1 2011. [Online]. Available: https://link.springer.com/article/10.1007/s10589-009-9244-7

[115] R. K. Phanden, A. Jain, and R. Verma, "A genetic algorithm-based approach for flexible job shop scheduling," *Applied Mechanics and Materials*, vol. 110-116, pp. 3930–3937, 2012. [Online]. Available: https://www.scientific.net/AMM.110-116.3930

[116] L. Lin, M. Gen, Y. Liang, and K. Ohno, "A hybrid ea for reactive flexible job-shop scheduling," *Procedia Computer Science*, vol. 12, pp. 110–115, 1 2012.

[117] H. Na and J. Park, "Multi-level job scheduling in a flexible job shop environment," *https://doi.org/10.1080/00207543.2013.848487*, vol. 52, pp. 3877–3887, 2014. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/00207543.2013.848487

[118] T. K. Liu, Y. P. Chen, and J. H. Chou, "Solving distributed and flexible job-shop scheduling problems for a real-world fastener manufacturer," *IEEE Access*, vol. 2, pp. 1598–1606, 2014.

[119] H. C. Chang, Y. P. Chen, T. K. Liu, and J. H. Chou, "Solving the flexible job shop scheduling problem with makespan optimization by using a hybrid taguchi-genetic algorithm," *IEEE Access*, vol. 3, pp. 1740–1754, 2015.

[120] X. Yang, J. Wang, M. Hou, and X. Fan, "Job shop scheduling based on genetic algorithm using matlab," *Proceedings of 2015 IEEE Advanced Information Technology, Electronic and Automation Control Conference, IAEAC 2015*, pp. 772–775, 3 2016.

[121] J. J. Palacios, M. A. González, C. R. Vela, I. González-Rodríguez, and J. Puente, "Genetic tabu search for the fuzzy flexible job shop problem," *Computers Operations Research*, vol. 54, pp. 74–89, 2 2015.

[122] J. Wang, W. Huang, G. Ma, and S. Chen, "An improved partheno genetic algorithm for multi-objective economic dispatch in cascaded hydropower systems," *International Journal of Electrical Power Energy Systems*, vol. 67, pp. 591–597, 5 2015.

[123] H. E. Nouri, O. B. Driss, and K. Ghédira, "Simultaneous scheduling of machines and transport robots in flexible job shop environment using hybrid metaheuristics based on clustered holonic multiagent model," *Computers Industrial Engineering*, vol. 102, pp. 488–501, 12 2016.

[124] C. Shi, T. Li, Y. Bai, and F. Zhao, "A heuristics-based parthenogenetic algorithm for the vrp with potential demands and time windows," *Scientific Programming*, vol. 2016, 2016.

[125] Y. He, W. Weng, and S. Fujimura, "Improvements to genetic algorithm for flexible job shop scheduling with overlapping in operations," *Proceedings - 16th IEEE/ACIS International Conference on Computer and Information Science, ICIS 2017*, pp. 791–796, 6 2017.

[126] A. Costa, F. A. Cappadonna, and S. Fichera, "A hybrid genetic algorithm for minimizing makespan in a flow-shop sequence-dependent group scheduling

problem," *Journal of Intelligent Manufacturing*, vol. 28, pp. 1269–1283, 8 2017. [Online]. Available: https://link.springer.com/article/10.1007/s10845-015-1049-1

[127] X. Yu, X. Liao, W. Li, X. Liu, and Z. Tao, "Logistics automation control based on machine learning algorithm," *Cluster Computing*, vol. 22, pp. 14 003–14 011, 11 2019. [Online]. Available: https://link.springer.com/article/10.1007/s10586-018-2169-0

[128] B. Wang and H. Wang, "Multiobjective order acceptance and scheduling on unrelated parallel machines with machine eligibility constraints," *Mathematical Problems in Engineering*, vol. 2018, 2018.

[129] T. Biswas, P. Kuila, and A. K. Ray, "A novel scheduling with multi-criteria for high-performance computing systems: an improved genetic algorithm-based approach," *Engineering with Computers*, vol. 35, pp. 1475–1490, 10 2019. [Online]. Available: https://link.springer.com/article/10.1007/s00366-018-0676-5

[130] K. C. Bhosale and P. J. Pawar, "Material flow optimisation of production planning and scheduling problem in flexible manufacturing system by real coded genetic algorithm (rcga)," *Flexible Services and Manufacturing Journal*, vol. 31, pp. 381–423, 6 2019. [Online]. Available: https://link.springer.com/article/10.1007/s10696-018-9310-5

[131] G. Zhang, Y. Hu, J. Sun, and W. Zhang, "An improved genetic algorithm for the flexible job shop scheduling problem with multiple time constraints," *Swarm and Evolutionary Computation*, vol. 54, p. 100664, 5 2020.

[132] Z. Yang, L. Jiacheng, and L. Lei, "Time-dependent theme park routing problem by partheno-genetic algorithm," *Mathematics 2020, Vol. 8, Page 2193*, vol. 8, p. 2193, 12 2020. [Online]. Available: https://www.mdpi.com/2227-7390/8/12/2193/htmhttps://www.mdpi.com/2227-7390/8/12/2193

[133] S. Shi and H. Xiong, "A hybrid immune genetic algorithm with tabu search for minimizing the tool switch times in cnc milling batch-processing,"

Applied Intelligence, vol. 52, pp. 7793–7807, 5 2022. [Online]. Available: https://link.springer.com/article/10.1007/s10489-021-02869-3

[134] W. Guo, Q. Lei, Y. Song, and X. Lyu, "A learning interactive genetic algorithm based on edge selection encoding for assembly job shop scheduling problem," *Computers Industrial Engineering*, vol. 159, p. 107455, 9 2021.

[135] X. Liang, J. Chen, X. Gu, and M. Huang, "Improved adaptive non-dominated sorting genetic algorithm with elite strategy for solving multi-objective flexible job-shop scheduling problem," *IEEE Access*, vol. 9, pp. 106 352–106 362, 2021.

[136] J. Park, J. Chun, S. H. Kim, Y. Kim, and J. Park, "Learning to schedule job-shop problems: representation and policy learning using graph neural network and reinforcement learning," *https://doi.org/10.1080/00207543.2020.1870013*, vol. 59, pp. 3360–3377, 2021. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/00207543.2020.1870013

[137] M. S. Viana, R. C. Contreras, and O. M. Junior, "A new frequency analysis operator for population improvement in genetic algorithms to solve the job shop scheduling problem," *Sensors 2022, Vol. 22, Page 4561*, vol. 22, p. 4561, 6 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/12/4561/htmhttps://www.mdpi.com/1424-8220/22/12/4561

[138] S. Saidat, A. K. Junoh, W. Z. A. W. Muhamad, and Z. Yahya, "Modified job shop scheduling via taguchi method and genetic algorithm," *Neural Computing and Applications*, vol. 34, pp. 1963–1980, 2 2022. [Online]. Available: https://link.springer.com/article/10.1007/s00521-021-06504-7

[139] N. Chen, N. Xie, and Y. Wang, "An elite genetic algorithm for flexible job shop scheduling problem with extracted grey processing time," *Applied Soft Computing*, vol. 131, p. 109783, 12 2022.

[140] B. Tutumlu and T. Saraç, "A mip model and a hybrid genetic algorithm for flexible

job-shop scheduling problem with job-splitting," *Computers Operations Research*, vol. 155, p. 106222, 7 2023.

[141] L. Meng, W. Cheng, B. Zhang, W. Zou, W. Fang, and P. Duan, "An improved genetic algorithm for solving the multi-agv flexible job shop scheduling problem," *Sensors 2023, Vol. 23, Page 3815*, vol. 23, p. 3815, 4 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/8/3815/htmhttps://www.mdpi.com/1424-8220/23/8/3815

[142] S. M. Homayouni, D. B. Fontes, and J. F. Gonçalves, "A multistart biased random key genetic algorithm for the flexible job shop scheduling problem with transportation," *International Transactions in Operational Research*, vol. 30, pp. 688–716, 3 2023. [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1111/itor.12878https://onlinelibrary.wiley.com/doi/abs/10.1111/itor.12878https://onlinelibrary.wiley.com/doi/10.1111/itor.12878

[143] Q. Shen, W. M. Shi, and W. Kong, "Hybrid particle swarm optimization and tabu search approach for selecting genes for tumor classification using gene expression data," *Computational Biology and Chemistry*, vol. 32, pp. 53–60, 2 2008.

[144] M. Mandloi and V. Bhatia, "A low-complexity hybrid algorithm based on particle swarm and ant colony optimization for large-mimo detection," *Expert Systems with Applications*, vol. 50, pp. 66–74, 5 2016.

[145] H. B. Ouyang, L. Q. Gao, X. Y. Kong, S. Li, and D. X. Zou, "Hybrid harmony search particle swarm optimization with global dimension selection," *Information Sciences*, vol. 346-347, pp. 318–337, 6 2016.

[146] J. Chen, Q. Do, H. H. Algorithms, and undefined 2015, "Training artificial neural networks by a hybrid pso-cs algorithm," *mdpi.com*, vol. 8, pp. 292–308, 2015. [Online]. Available: https://www.mdpi.com/102080

[147] D. B. Fontes, S. M. Homayouni, and J. F. Gonçalves, "A hybrid particle swarm optimization and simulated annealing algorithm for the job shop scheduling problem

with transport resources," *European Journal of Operational Research*, vol. 306, pp. 1140–1157, 5 2023.

[148] M. R. Bonyadi and Z. Michalewicz, "Particle swarm optimization for single objective continuous space problems: A review," *Evolutionary Computation*, vol. 25, pp. 1–54, 3 2017.

[149] M. Nouiri, A. Bekrar, A. Jemai, S. Niar, and A. C. Ammari, "An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem," *Journal of Intelligent Manufacturing*, vol. 29, pp. 603–615, 3 2018. [Online]. Available: https://link.springer.com/article/10.1007/s10845-015-1039-3

[150] C. M. Wang and Y. F. Huang, "Self-adaptive harmony search algorithm for optimization," *Expert Systems with Applications*, vol. 37, pp. 2826–2837, 4 2010.

[151] H. Zhang, Z. Deng, Y. Fu, L. Lv, and C. Yan, "A process parameters optimization method of multi-pass dry milling for high efficiency, low energy and low carbon emissions," *Journal of Cleaner Production*, vol. 148, pp. 174–184, 4 2017.

[152] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, pp. 4–31, 2 2011.

[153] W. Yi, Y. Zhou, L. Gao, X. Li, and J. Mou, "An improved adaptive differential evolution algorithm for continuous optimization," *Expert Systems with Applications*, vol. 44, pp. 1–12, 2 2016.

[154] Q. Fan and Y. Zhang, "Self-adaptive differential evolution algorithm with crossover strategies adaptation and its application in parameter estimation," *Chemometrics and Intelligent Laboratory Systems*, vol. 151, pp. 164–171, 2 2016.

[155] S. Das, S. S. Mullick, and P. N. Suganthan, "Recent advances in differential evolution – an updated survey," *Swarm and Evolutionary Computation*, vol. 27, pp. 1–30, 4 2016.

[156] K. Lenin, B. R. Reddy, and M. Suryakalavathi, "Hybrid tabu search-simulated annealing method to solve optimal reactive power problem," *International Journal of Electrical Power Energy Systems*, vol. 82, pp. 87–91, 11 2016.

[157] S. M. Sait, F. C. Oughali, M. Al-Asli, S. M. Sait, F. C. Oughali, and M. Al-Asli, "Design partitioning and layer assignment for 3d integrated circuits using tabu search and simulated annealing," *Journal of applied research and technology*, vol. 14, pp. 67–76, 2 2016. [Online]. Available: http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S1665-64232016000100067&lng=es&nrm=iso&tlng=en

[158] K. Buyukozkan, I. Kucukkoc, S. I. Satoglu, and D. Z. Zhang, "Lexicographic bottleneck mixed-model assembly line balancing problem: Artificial bee colony and tabu search approaches with optimised parameters," *Expert Systems with Applications*, vol. 50, pp. 151–166, 5 2016.

[159] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," 2005. [Online]. Available: http://abc.erciyes.edu.tr/pub/tr06_2005.pdf

[160] B. Yuce, M. S. Packianather, E. Mastrocinque, D. T. Pham, and A. Lambiase, "Honey bees inspired optimization method: the bees algorithm," *mdpi.com*, vol. 4, pp. 646–662, 2013. [Online]. Available: https://www.mdpi.com/60386

[161] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (abc) algorithm," *Applied Soft Computing*, vol. 8, pp. 687–697, 1 2008.

[162] A. L. A. Bolaji, A. T. Khader, M. A. Al-Betar, and M. A. Awadallah, "University course timetabling using hybridized artificial bee colony with hill climbing optimizer," *Journal of Computational Science*, vol. 5, pp. 809–818, 9 2014.

[163] S. Janakiraman, "A hybrid ant colony and artificial bee colony optimization algorithm-based cluster head selection for iot," *Procedia Computer Science*, vol. 143, pp. 360–366, 1 2018.

[164] E. ZorarpacI and S. A. Özel, "A hybrid approach of differential evolution and

artificial bee colony for feature selection," *Expert Systems with Applications*, vol. 62, pp. 91–103, 11 2016.

[165] S. S. Jadon, R. Tiwari, H. Sharma, and J. C. Bansal, "Hybrid artificial bee colony algorithm with differential evolution," *Applied Soft Computing*, vol. 58, pp. 11–24, 9 2017.

[166] N. Sharma, H. Sharma, and A. Sharma, "Beer froth artificial bee colony algorithm for job-shop scheduling problem," *Applied Soft Computing*, vol. 68, pp. 507–524, 2018. [Online]. Available: https://doi.org/10.1016/j.asoc.2018.04.001

[167] J. Gao, X. Zhu, K. Bai, and R. Zhang, "New controllable processing time scheduling with subcontracting strategy for no-wait job shop problem," *International Journal of Production Research*, vol. 60, pp. 2254–2274, 4 2022. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/00207543.2021.1886368

[168] S. Sundar, P. N. Suganthan, C. T. Jin, C. T. Xiang, and C. C. Soon, "A hybrid artificial bee colony algorithm for the job-shop scheduling problem with no-wait constraint," *Soft Computing*, vol. 21, pp. 1193–1202, 3 2017. [Online]. Available: https://link.springer.com/article/10.1007/s00500-015-1852-9

[169] Q.-K. Pan, M. F. Tasgetiren, P. N. Suganthan, and T. J. Chua, "A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem," 2010.

[170] Q. K. Pan, L. Wang, K. Mao, J. H. Zhao, and M. Zhang, "An effective artificial bee colony algorithm for a real-world hybrid flowshop problem in steelmaking process," *IEEE Transactions on Automation Science and Engineering*, vol. 10, pp. 307–322, 2013.

[171] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: Harmony search," *Simulation*, vol. 76, pp. 60–68, 2001.

[172] K. S. Lee and Z. W. Geem, "A new structural optimization method based on the harmony search algorithm," *Computers Structures*, vol. 82, pp. 781–798, 4 2004.

[173] M. A. Awadallah, M. A. Al-Betar, A. T. Khader, A. L. Bolaji, and M. Alkoffash, "Hybridization of harmony search with hill climbing for highly constrained nurse rostering problem," *Neural Computing and Applications*, vol. 28, pp. 463–482, 3 2017.

[174] A. Panchal, "Harmony search in therapeutic medical physics," *Studies in Computational Intelligence*, vol. 191, pp. 189–203, 2009.

[175] D. Manjarres, I. Landa-Torres, S. Gil-Lopez, J. D. Ser, M. N. Bilbao, S. Salcedo-Sanz, and Z. W. Geem, "Survey paper a survey on applications of the harmony search algorithm," 2013. [Online]. Available: http://dx.doi.org/10.1016/j.engappai.2013.05.008

[176] M. Mahdavi, M. Fesanghary, and E. Damangir, "An improved harmony search algorithm for solving optimization problems," *Applied Mathematics and Computation*, vol. 188, pp. 1567–1579, 5 2007.

[177] O. M. D. Alia and R. Mandava, "The variants of the harmony search algorithm: An overview," *Artificial Intelligence Review*, vol. 36, pp. 49–68, 6 2011. [Online]. Available: https://link.springer.com/article/10.1007/s10462-010-9201-y

[178] X. S. Yang and S. Deb, "Cuckoo search via lévy flights," *2009 World Congress on Nature and Biologically Inspired Computing, NABIC 2009 - Proceedings*, pp. 210–214, 2009.

[179] S. Singh and K. P. Singh, "Cuckoo search optimization for job shop scheduling problem," *Advances in Intelligent Systems and Computing*, vol. 335, pp. 99–111, 2015. [Online]. Available: https://link.springer.com/chapter/10.1007/978-81-322-2217-0_9

[180] S. Walton, O. Hassan, K. Morgan, and M. R. Brown, "Modified cuckoo search: A new gradient free optimisation algorithm," *Chaos, Solitons Fractals*, vol. 44, pp. 710–718, 9 2011.

[181] A. H. Gandomi, X. S. Yang, and A. H. Alavi, "Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems," *Engineering with Computers*, vol. 29, pp. 17–35, 2013.

[182] X.-S. Yang, *Nature-inspired metaheuristic algorithms*, 2010. [Online]. Available: https://books.google.com/books?hl=en&lr=&id=iVB_ETlh4ogC&oi=fnd&pg=PR5&ots=DxhysjzKuc&sig=JPLH4qm4R4QavVr_qs6vm10WsQ8

[183] A. Chatterjee, G. Mahanti, A. C. P. I. Electromagnetics, and undefined 2012, "Design of a fully digital controlled reconfigurable switched beam concentric ring array antenna using firefly and particle swarm optimization algorithm," *jpier.org*, vol. 36, pp. 113–131, 2012. [Online]. Available: https://www.jpier.org/pierb/pier.php?paper=11083005

[184] T. Apostolopoulos, A. V. I. journal of, and undefined 2011, "Application of the firefly algorithm for solving the economic emissions load dispatch problem," *downloads.hindawi.com*, vol. 2011, 2011. [Online]. Available: https://downloads.hindawi.com/archive/2011/523806.pdf

[185] M. H. Horng, "Vector quantization using the firefly algorithm for image compression," *Expert Systems with Applications*, vol. 39, pp. 1078–1091, 1 2012.

[186] N. Álvarez Gil, R. Rosillo, D. de la Fuente, and R. Pino, "A discrete firefly algorithm for solving the flexible job-shop scheduling problem in a make-to-order manufacturing system," *Central European Journal of Operations Research*, vol. 29, pp. 1353–1374, 12 2021. [Online]. Available: https://link.springer.com/article/10.1007/s10100-020-00701-w

[187] K. K. Chakravarthi, L. Shyamala, and V. Vaidehi, "Cost-effective workflow scheduling approach on cloud under deadline constraint using firefly algorithm," *Applied Intelligence*, vol. 51, pp. 1629–1644, 3 2021. [Online]. Available: https://link.springer.com/article/10.1007/s10489-020-01875-1

[188] G. Xu, X. Jiang, D. Sun, B. Yang, R. Deng, and J. Fu, "Flexible job-shop scheduling based on improved firefly algorithm," *2022 2nd International Conference on Computer, Control and Robotics, ICCCR 2022*, pp. 90–97, 2022.

[189] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "Gsa: A gravitational search algorithm," *Information Sciences*, vol. 179, pp. 2232–2248, 6 2009.

[190] B. Schutz and D. Schutz, *Gravity from the ground up: An introductory guide to gravity and general relativity*, 2003. [Online]. Available: https://books.google.com/books?hl=en&lr=&id=P_T0xxhDcsIC&oi=fnd&pg=PR13&ots=e0Iqj7hHmb&sig=wHNhzrt34d0WzNhZqM_ThFBc0sg

[191] X. Li, J. Wang, J. Zhou, and M. Yin, "An effective gsa based memetic algorithm for permutation flow shop scheduling," *2010 IEEE World Congress on Computational Intelligence, WCCI 2010 - 2010 IEEE Congress on Evolutionary Computation, CEC 2010*, 2010.

[192] C. Cao, Y. Zhang, X. Gu, D. Li, and J. Li, "An improved gravitational search algorithm to the hybrid flowshop with unrelated parallel machines scheduling problem," *International Journal of Production Research*, vol. 2021, pp. 1–17, 9 2021. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/00207543.2020.1788732

[193] F. Zhao, F. Xue, Y. Zhang, W. Ma, C. Zhang, and H. Song, "A discrete gravitational search algorithm for the blocking flow shop problem with total flow time minimization," *Applied Intelligence*, vol. 49, pp. 3362–3382, 9 2019. [Online]. Available: https://link.springer.com/article/10.1007/s10489-019-01457-w

[194] K. Socha and C. Blum, "An ant colony optimization algorithm for continuous optimization: Application to feed-forward neural network training," *Neural Computing and Applications*, vol. 16, pp. 235–247, 5 2007. [Online]. Available: https://link.springer.com/article/10.1007/s00521-007-0084-z

[195] A. Bauer, B. Bullnheimer, R. F. Hartl, and C. Strauss, "An ant colony optimization approach for the single machine total tardiness problem," *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999*, vol. 2, pp. 1445–1450, 1999.

[196] A. Puris, R. Bello, Y. Trujillo, A. Nowe, and Y. Martínez, "Two-stage aco to solve the job shop scheduling problem," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 4756 LNCS, pp. 447–456, 2007. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-540-76725-1_47

[197] I. Chaouch, O. B. Driss, and K. Ghedira, "A modified ant colony optimization algorithm for the distributed job shop scheduling problem," *Procedia Computer Science*, vol. 112, pp. 296–305, 1 2017.

[198] V. P. Eswaramurthy and A. Tamilarasi, "Hybridizing tabu search with ant colony optimization for solving job shop scheduling problems," *International Journal of Advanced Manufacturing Technology*, vol. 40, pp. 1004–1015, 2 2009. [Online]. Available: https://link.springer.com/article/10.1007/s00170-008-1404-x

[199] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: Optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 26, pp. 29–41, 1996.

[200] C. Blum, "Beam-aco - hybridizing ant colony optimization with beam search: An application to open shop scheduling," *Computers and Operations Research*, vol. 32, pp. 1565–1591, 6 2005.

[201] W. J. Gutjahr, "Aco algorithms with guaranteed convergence to the optimal solution," *Information Processing Letters*, vol. 82, pp. 145–153, 5 2002.

[202] M. Birattari, T. Stützle, S. Stützle, L. Paquete, and K. Varrentrapp, "A racing algorithm for configuring metaheuristics," 2002. [Online]. Available: https://dl.acm.org/doi/10.5555/2955491.2955494

[203] R. Bellman, "A markovian decision process," *Indiana University Mathematics Journal*, vol. 6, pp. 679–684, 1957.

[204] H. Oliff, Y. Liu, M. Kumar, M. Williams, and M. Ryan, "Reinforcement learning for facilitating human-robot-interaction in manufacturing," *Journal of Manufacturing Systems*, vol. 56, pp. 326–340, 7 2020.

[205] C. J. C. H. Watkins and P. Dayan, "Q-learning," vol. 8, pp. 279–292, 1992.

[206] J. Shahrabi, M. A. Adibi, and M. Mahootchi, "A reinforcement learning approach to parameter estimation in dynamic job shop scheduling," *Computers and Industrial Engineering*, vol. 110, pp. 75–82, 8 2017.

[207] X. N. Shen, L. L. Minku, N. Marturi, Y. N. Guo, and Y. Han, "A q-learning-based memetic algorithm for multi-objective dynamic software project scheduling," *Information Sciences*, vol. 428, pp. 1–29, 2 2018.

[208] R. Chen, B. Yang, S. Li, and S. Wang, "A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem," *Computers Industrial Engineering*, vol. 149, p. 106778, 11 2020.

[209] D. Shi, W. Fan, Y. Xiao, T. Lin, and C. Xing, "Intelligent scheduling of discrete automated production line via deep reinforcement learning," *International Journal of Production Research*, vol. 58, pp. 3362–3380, 6 2020. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/00207543.2020.1717008

[210] M. Karimi-Mamaghan, M. Mohammadi, P. Meyer, A. M. Karimi-Mamaghan, and E.-G. Talbi, "Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art," *European Journal of Operational Research*, vol. 296, pp. 393–422, 2022. [Online]. Available: https://doi.org/10.1016/j.ejor.2021.04.032

[211] M. A. Rosen and H. A. Kishawy, "Sustainable manufacturing and design: Concepts, practices and needs," *Sustainability 2012, Vol. 4, Pages 154-174*, vol. 4, pp. 154–174,

1 2012. [Online]. Available: https://www.mdpi.com/2071-1050/4/2/154/htmhttps://www.mdpi.com/2071-1050/4/2/154

[212] S. Sivanandam and S. Deepa, "Genetic algorithms," *Introduction to Genetic Algorithms*, pp. 15–37, 2008. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-540-73190-0_2

[213] S. Katoch, . Sumit, S. Chauhan, and V. Kumar, "A review on genetic algorithm: past, present, and future," *Multimedia Tools and Applications*, vol. 80, pp. 8091–8126, 2021. [Online]. Available: https://doi.org/10.1007/s11042-020-10139-6

[214] J. Luo, D. E. Baz, R. Xue, and J. Hu, "Solving the dynamic energy aware job shop scheduling problem with the heterogeneous parallel genetic algorithm," *Future Generation Computer Systems*, vol. 108, pp. 119–134, 7 2020.

[215] T. Gonzalez and S. Sahni, "Open shop scheduling to minimize finish time," *Journal of the ACM (JACM)*, vol. 23, pp. 665–679, 1976.

[216] K. Omori, S. Maekawa, H. Tamaki, and S. Kitamura, "Parallelization of genetic algorithm with sexual selection," *Electrical Engineering in Japan*, vol. 150, pp. 42–49, 1 2005. [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1002/eej.20029https://onlinelibrary.wiley.com/doi/abs/10.1002/eej.20029https://onlinelibrary.wiley.com/doi/10.1002/eej.20029

[217] S. A. Jafari, S. Mashohor, and M. J. Varnamkhasti, "Committee neural networks with fuzzy genetic algorithm," *Journal of Petroleum Science and Engineering*, vol. 76, pp. 217–223, 3 2011.

[218] R. Singh and S. Jagadeeshan, "Charles darwin: Theory of sexual selection," *Encyclopedia of Evolutionary Psychological Science*, pp. 1011–1026, 2021. [Online]. Available: https://link.springer.com/referenceworkentry/10.1007/978-3-319-19650-3_1396

[219] M. M. Raghuwanshi and O. G. Kakde, "Genetic algorithm with species and sexual selection," *2006 IEEE Conference on Cybernetics and Intelligent Systems*, 2006.

[220] N. Shukla, M. K. Tiwari, and D. Ceglarek, "Genetic-algorithms-based algorithm portfolio for inventory routing problem with stochastic demand," *https://doi.org/10.1080/00207543.2011.653010*, vol. 51, pp. 118–137, 1 2012. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/00207543.2011.653010

[221] N. Kubota and T. Fukuda, "Genetic algorithms with age structure," *Soft Computing*, vol. 1, pp. 155–161, 1997.

[222] A. Ghosh, S. Tsutsui, and H. Tanaka, "Function optimization in nonstationary environment using steady state genetic algorithms with aging of individuals," *Proceedings of the IEEE Conference on Evolutionary Computation, ICEC*, pp. 666–671, 1998.

[223] S. Tian, T. Wang, L. Zhang, and X. Wu, "An energy-efficient scheduling approach for flexible job shop problem in an internet of manufacturing things environment," *IEEE Access*, vol. 7, pp. 62 695–62 704, 2019.

[224] W. Shao, Z. Shao, and D. Pi, "Modeling and multi-neighborhood iterated greedy algorithm for distributed hybrid flow shop scheduling problem," *Knowledge-Based Systems*, vol. 194, 4 2020.

[225] P. Baptiste, M. Flamini, and F. Sourd, "Lagrangian bounds for just-in-time job-shop scheduling," *Computers Operations Research*, vol. 35, pp. 906–915, 2008. [Online]. Available: www.elsevier.com/locate/cor

[226] A. Momenikorbekandi and M. F. Abbod, "Multi-ethnicity genetic algorithm for job shop scheduling problems."

[227] M. Sewak, *Deep Reinforcement Learning: Frontiers of Artificial Intelligence*. Springer, 2019.

[228] S. Z. Hao Dong, Zihan Ding, *Deep Reinforcement Learning*. Springer, 2020.

[229] A. Momenikorbekandi and M. Abbod, "Intelligent scheduling based on reinforcement learning approaches: Applying advanced q-learning and state–action–reward–state–action reinforcement learning models for the optimisation of job shop scheduling problems," *Electronics*, vol. 12, no. 23, 2023. [Online]. Available: https://www.mdpi.com/2079-9292/12/23/4752

[230] N. Yoshitani and A. Hasegawa, "Model-based control of strip temperature for the heating furnace in continuous annealing," *IEEE Transaction on Control Systems Technology*, vol. 6, no. 2, pp. 146–165, 1998.

Appendix A

Job Shop Parameters

Table A.1 shows the penalties for the multi-machine and Table A.2 shows the penalties for single machines.

Table A.1: Multi-machine job-shop with due date, earliness and tardiness penalties.

No.	M1	E/T penalty	M2	E/T penalty	M3	E/T penalty	M4	E/T penalty
1	26	3/2	79	1/4	43	4/1	90	2/3
2	19	1/2	27	4/4	15	3/3	14	2/1
3	86	2/2	58	4/4	55	3/3	15	1/1
4	85	2/2	62	3/4	35	4/3	51	1/1
5	40	1/3	8	3/4	25	4/2	13	2/1
6	19	4/3	25	3/1	42	1/4	6	2/2
7	89	2/3	94	4/1	49	3/2	49	1/4
8	35	4/3	89	3/4	37	1/1	12	2/2

Table A.2: Single-machine job-shop including earliness and tardiness penalties.

No.	Job time	Earliness penalty	Tardiness penalty
1	77	3	3
2	10	3	2
3	57	2	1
4	81	2	2
5	65	2	3
6	55	4	4
7	68	2	4
8	77	2	4
9	49	2	1
10	51	3	2
11	64	3	4
12	35	4	2
13	62	2	2
14	47	3	4
15	23	4	3
16	31	2	2
17	90	1	3
18	26	3	1
19	60	3	2
20	30	3	4
21	9	3	2
22	92	1	3
23	24	2	3
24	52	4	1
25	68	1	2
26	5	3	3
27	11	3	2
28	14	2	4
29	49	2	2
30	88	3	4
31	4	1	3
32	90	1	1

Appendix B

Furnace Model Equations

This appendix contains examples and derivations supporting the furnace model in Chapter 6. Each entry indicates the section which refers to it. The model derivations are based on an article published by [230].

The temperature of the furnace must be raised and decreased in order to adjust for the substantial temperature fluctuations between the operations in a series. The procedure will demand a lot of time and energy because the furnace needs to be heated and cooled to the desired item temperature. The optimisation system will help to meet the need for scheduling optimisation for the shortest time possible and use less energy.

B.1 Furnace Modelling

The reheating furnace model is based on the schematics shown in Figure B.1. It consists of two sections: the heating section and the cooling section. The heating section is the part where the temperature is raised and held for a specific period of time due to the reheating requirements of the materials. Figure B.1 indicates the furnace model created by [230]

Equations B.1-B.8 pertain to a dynamic model of the strip temperature [230].

$$y(t+d) = \alpha_0 y(t) + \sum_{i=0}^{m} \beta_i u(t-i) + \sum_{i=1}^{4} d_i v_i(t+d) \tag{B.1}$$

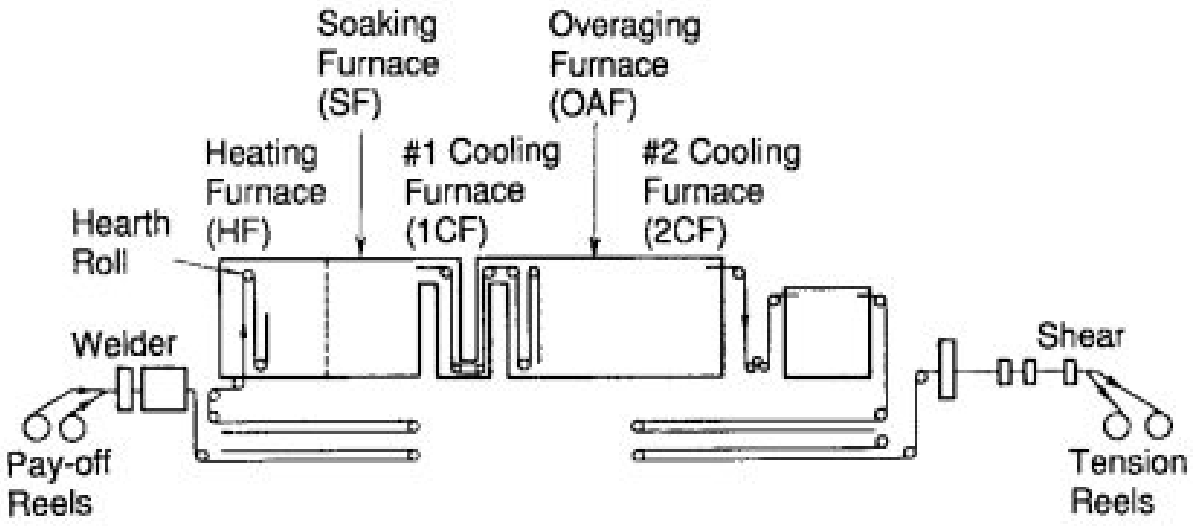


Figure B.1: Furnace Model .

$$y(t) = T_s(t) - T_s^* \tag{B.2}$$

$$u(t) = D_{sf}(t)\,[F(t) - F^*] \tag{B.3}$$

$$v_i(t+d) = \sum_{j=0}^{d-1} \alpha_0^{j/d} w_i(t+d-j)(i = 1, \cdots, 4) \tag{B.4}$$

$$w_1(t) = (1-d_1)\,w_1(t-1) + (1-d_2)\,w_2(t-1) \tag{B.5}$$

$$w_2(t) = [T_f(t) - T_{\sin}]\,[S_f(t) - S_f(t-1)] \tag{B.6}$$

$$w_3(t) = D_{sf}(t)\,[W_d(t)T_h(t)V(t) - W_{tv}^*] \tag{B.7}$$

$$w_4(t) = D_{sf}(t) \tag{B.8}$$

Equations B.9 - B.14 pertain to a dynamic model of the furnace temperature [230].

$$y_f\left(t+d_f\right)=\alpha_{f0}y_f(t)+\sum_{i=0}^{m_f}\beta_{fi} \tag{B.9}$$

$$u_f(t-i)+\sum_{i=1}^{2}d_{fi}v_{fi}\left(t+d_f\right) \tag{B.10}$$

$$y_f(t)=T_f(t)-T_f^* \tag{B.11}$$

$$u_f(t)=F(t)-F^* \tag{B.12}$$

$$v_{fi}\left(t+d_f\right)=\sum_{j=0}^{d_f-1}\alpha_{f0}^{\frac{j}{d_f}}w_{fi}\left(t+d_f-j\right),(i=1,2) \tag{B.13}$$

$$u_f(t)=F(t)-F^* \tag{B.14}$$

where

F is the total fuel flow rate in the heating furnace $[10^4\ \mathrm{Nm^3/h}\ ]$

$w_{f1}(t)=W_d(t)T_h(t)V(t)-W_{tv}^*$

$w_{ff}(t)=1$

t time [min] (One sampling period $=1$ min);

d, d_f dead time [min]; notes m^3 at 0°C;

T_s Out-strip temperature of the heating furnace $[100°\mathrm{C}]$;

m, m_f Non-negative integer (to be determined experimentally);

T_{ss} Out-strip temperature in steady state $[100°\mathrm{C}]$;

T_{sin} Strip temperature at the inlet of the heating furnace (constant) $[100°\mathrm{C}]$;

T_f Furnace temperature of the heating furnace $[100°\mathrm{C}]$;

W_d, T_h Strip width [m] and thickness [mm]

$V_f(t)$ Average line speed during $[t-t_f,t]$; $v_i(i=1,\cdots,4)$; $v_{fi}(i=1,2)$;

Disturbance terms: $F^*, T_s^*, W_{tv}^*, T_v^*, T_f^*$: are Average values of $F, T_s, W_dT_hV, T_v, T_f$, respectively, in normal operations.

$\alpha_0, \beta_0, \cdots, \beta_m, d_1, \cdots, d_4, s_1, \cdots, s_4$: specific model parameters.

The material properties used in the reheating furnace are shown in Tables B.1.

		C	Si	Mn	Cr	Mo	Ni	Diameter (mm)	Temp (C)	Quenching Pressure (bar)	Quenching Temp (C)	Time (min)	Hardness (HV)	Depth Hardness (um)
14 Ni Cr 14	%: 0.14% C, 0.25% Si, 0.55% Mn, 0.75% Cr, 3.50% Ni.	14	25	55	75	0	325	30	850	8	70	20	395	24
		14	25	55	75	0	325	30	850	9	70	30	400	27
		14	25	55	75	0	325	30	850	10	70	30	400	28
		14	25	55	75	0	325	30	850	10	70	20	405	24
		14	25	55	75	0	325	30	850	12	70	20	410	24
		14	25	55	75	0	325	25	850	8	70	18	410	21
		14	25	55	75	0	325	25	850	9	70	20	413	23
		14	25	55	75	0	325	25	850	10	70	22	415	24
		14	25	55	75	0	325	25	850	10	70	23	418	24
		14	25	55	75	0	325	25	850	12	70	20	420	23
14 Ni Cr 14	%: 0.14% C, 0.25% Si, 0.55% Mn, 0.75% Cr, 3.50% Ni.	14	25	55	75	0	325	30	900	8	50	20	415	26
		14	25	55	75	0	325	30	900	9	50	30	425	29
		14	25	55	75	0	325	30	900	10	50	30	428	30
		14	25	55	75	0	325	30	900	10	50	20	435	26
		14	25	55	75	0	325	30	900	12	50	20	440	26
		14	25	55	75	0	325	25	900	8	50	18	411	23
		14	25	55	75	0	325	25	900	9	50	20	414	25
		14	25	55	75	0	325	25	900	10	50	22	417	26
		14	25	55	75	0	325	25	900	10	50	23	420	26
		14	25	55	75	0	325	25	900	12	50	20	422	25
16Mn Cr5	14%C 20%Si 115%Mn 18%S 95 %Cr	16	20	115	95	0	0	16	850	8	70	16	393	16
		16	20	115	95	0	0	16	850	9	70	16	397	16
		16	20	115	95	0	0	16	850	10	70	18	400	16
		16	20	115	95	0	0	16	850	10	70	18	407	16
		16	20	115	95	0	0	16	850	12	70	20	411	16
16Mn Cr5	14%C 20%Si 115%Mn 18%S 95 %Cr	16	20	115	95	0	0	16	900	8	50	16	411	16
		16	20	115	95	0	0	16	900	9	50	16	415	16
		16	20	115	95	0	0	16	900	10	50	18	417	16
		16	20	115	95	0	0	16	900	10	50	18	419	16
		16	20	115	95	0	0	16	900	12	50	20	423	16
16Mn Cr5	14%C 20%Si 115%Mn 18%S 95 %Cr	16	20	115	95	0	0	40	850	8	70	20	330	23
		16	20	115	95	0	0	40	850	9	70	25	340	25
		16	20	115	95	0	0	40	850	10	70	30	345	27
		16	20	115	95	0	0	40	850	10	70	35	350	30
		16	20	115	95	0	0	40	850	12	70	20	360	25
16Mn Cr5	14%C 20%Si 115%Mn 18%S 95 %Cr	16	20	115	95	0	0	40	900	8	50	20	390	25
		16	20	115	95	0	0	40	900	9	50	25	394	27
		16	20	115	95	0	0	40	900	10	50	30	399	30
		16	20	115	95	0	0	40	900	10	50	35	402	33
		16	20	115	95	0	0	40	900	12	50	20	415	28
30 Cr Ni Mo 8	30%C 20%Si 45Mn 2%S 200Cr 40%Mo	30	20	45	200	40	200	9	850	8	70	15	542	9
		30	20	45	200	40	200	9	850	9	70	15	589	9
		30	20	45	200	40	200	9	850	10	70	17	552	9
		30	20	45	200	40	200	9	850	10	70	17	560	9
		30	20	45	200	40	200	9	850	12	70	20	565	9
30 Cr Ni Mo 8	30%C 20%Si 45%Mn 2%S 200Cr 40%Mo	30	20	45	200	40	200	9	900	8	50	14	545	9
		30	20	45	200	40	200	9	900	9	50	14	558	9
		30	2	45	200	40	200	9	900	10	50	15	575	9
		30	20	45	200	40	200	9	900	10	50	15	590	9
		30	20	45	200	40	200	9	900	12	50	17	610	9
15 S20	20%C 25%Si 45%Mn	20	25	45	0	0	0	22	850	8	70	17	735	19
		20	25	45	0	0	0	22	850	9	70	17	752	20
		20	25	45	0	0	0	22	850	10	70	18	768	20
		20	25	45	0	0	0	22	850	10	70	18	771	22
		20	25	45	0	0	0	22	850	12	70	20	786	22
mild carbon	90%C 25%Si 45%Mn	90	24	39	1.97	28	15	546	930	35	0	24	875	29
		90	84	39	1.97	28	15	546	930	15	0	24	875	29
		90	84	39	1.97	28	15	546	930	15	0	36	849	42
		90	56	39	4.98	2.6	99	385	1025	15	0	29	852	48
		79	27	32	3.21	53	28	455	940	35	-40	23	862	24
		79	27	32	3.21	53	28	455	960	35	-40	24	876	25
		90	56	39	4.98	2.6	99	385	1025	15	-85	31	856	44
		80	65	25	5.05	22	18	455	950	30	-85	25	856	28
		80	65	25	5.05	22	18	455	970	30	-85	27	883	34
		90	56	39	4.98	2.6	99	380	1025	15	-140	27	932	34

Table B.1: Materials properties data.