الجمهورية الجزائرية الديمقراطية الشعبية

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التعليم العالي و البحث العلمي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

جامعة أبي بكر بلقايد – تلمسان –

Université Aboubakr Belkaïd – Tlemcen –
Faculté de TECHNOLOGIE

THESE

Présentée pour l'obtention du **grade** de **DOCTORAT 3ème Cycle**

En : Génie Industriel

Spécialité : Génie Industriel et Productique

Par : BOUFELLOUH Radhwane

Sujet

Métaheuristiques pour la résolution des problèmes d'ordonnancement sur les systèmes à machines en série

" Metaheuristics for Solving Flow shop Scheduling Problems"

Soutenue publiquement, le 21 / 02 / 2024 , devant le jury composé de :

CHIKH Mohammed Amine	Professeur	Université de Tlemcen	Président
BELKAID Fayçal	Professeur	Université de Tlemcen	Directeur
TRIQUI Lamia	MCA	Université de Tlemcen	Co-Directeur
BRAHIMI Nadjib	Prof	Rennes School of Business, France	Examinateur 1
HACHEMI Khalid	Prof	Université d'Oran 2	Examinateur 2
MALIKI Fouad	MCA	École Supérieure en Sciences Appliquées de Tlemcen	Examinateur 3
HASSAM Ahmed	MCB	Université de Tlemcen	Invité

Acknowledgments

I would like to express my profound gratitude to all those who have played a significant role in the completion of this doctoral thesis.

First and foremost, I am deeply thankful to my supervisor, ***Mr BELKAID Fayçal***, Professor at the University of Tlemcen for his unwavering support, guidance, and mentorship throughout this academic journey. His expertise, dedication, and patience have been invaluable in shaping the research and my growth as a scholar.

I would like to extend my gratitude to my co-supervisor, Mme ***TRIQUI Lamia***, Professor at the University of Tlemcen for her invaluable guidance and insightful discussions throughout this research undertaking.

I give my sincere thanks to Mr ***ARBAOUI Taha*** Professor at the Université de Technologie de Troyes, France. Throughout our collaborative efforts on various research projects, he has offered invaluable insights and shared his invaluable experience, enriching the quality of this work.

I am deeply thankful to Mr ***CHIKH Mohammed Amine***, Professor and Dean of the University of Tlemcen for giving me the honner to serve as the chair of my dissertation committee. His insightful feedback and constructive criticism have been instrumental in strengthening the theoretical foundations of my work.

I extend my sincere appreciation to Mr ***BRAHIMI Nadjib***, Professor at the Rennes School of Business in France for his valuable contributions as a member of my dissertation committee. His expertise has added a unique perspective to my research. I am grateful for the thoughtful discussions, encouragement, and the meticulous attention to detail that have significantly enhanced the quality of this dissertation.

I would like to express my gratitude to Mr ***HACHEMI Khalid***, Professor at Oran 2 University for his insightful feedback and constructive comments that have played a pivotal role in refining the methodology and analysis of my research. I appreciate the time and effort he devoted to reviewing my work and providing thoughtful suggestions for improvement.

I am also grateful to Mr ***MALIKI Fouad***, Professor at the University of Tlemcen for his mentorship, dedication, and generous commitment of his time and expertise to reviewing and providing detailed comments on multiple drafts and ensuring the quality and rigor of this work.

I also would like to thank Mr ***HASSAM Ahmed***, Professor at the University of Tlemcen for giving me the honner by accepting the invitation and providing valuable input and precious advices that only added value to the final product of this work.

I am grateful to all the staff members of Manufacturing and Engineering Laboratory of the Technology Faculty of Tlemcen for providing an environment conducive to research and learning. The resources, laboratories, and support services have been instrumental in facilitating my academic endeavors.

I extend my appreciation to my colleagues and fellow graduate students who have been a source of inspiration and camaraderie during this challenging undertaking. Your intellectual exchanges and camaraderie have enriched this experience in countless ways.

My heartfelt thanks go to my friends and extended family for their unwavering belief in me, their encouragement, and unconditional assistance during the demanding phases of this journey. Your support has been my anchor.

Last but not least, I dedicate this thesis to my parents and my older siblings my new-born nephew and my nieces, whose unwavering love, understanding, and sacrifices have been the cornerstone of my academic pursuit.

This dissertation would not have been possible without the collective contributions and support of all those mentioned and the many others who have been part of this academic voyage. Thank you for being an integral part of this significant milestone in my life.

Contents

List of Figures

List of Tables

Authors' Publications

- Boufellouh, R., & Belkaid, F. (2020). Bi-objective optimization algorithms for joint production and maintenance scheduling under a global resource constraint: Application to the permutation flow shop problem. Computers & Operations Research, 104943. https://doi:10.1016/j.cor.2020.104943 IF: **4.6**

- Boufellouh, R., & Belkaid, F. (2023). Multi-objective optimization for energy-efficient flow shop scheduling problem with blocking and collision-free transportation constraints. Applied Soft Computing, 110884. https://doi.org/10.1016/j.asoc.2023.110884 IF: **8.7**

International conferences

- Boufellouh, R., & Belkaid, F. (2019). Minimizing peak power load in flowshop scheduling under unavailability constraints. First International Conference on Materials, Environment, Mechanical and Industrial Systems (ICMEMIS'19) http://www.univ-djelfa.dz/icmemis19/

- Boufellouh, R., & Belkaid, F. (2019). Multi-objective approach to optimize production and maintenance scheduling in flow shop environment under non-renewable resources constraints. 2019 International Conference on Advanced Electrical Engineering (ICAEE). https://doi:10.1109/icaee47123.2019.9014781

- Boufellouh, R., & Belkaid, F. (2020). Green scheduling of permutation flow shop with tool deterioration and constrained tool replacement: A case study from metalworking industry. 2020 IEEE 13th International Colloquium of Logistics and Supply Chain Management (LOGISTIQUA). https://doi:10.1109/logistiqua49782.2020.9353905

National conferences

- Boufellouh, R., & Belkaid, F. (2019). Bi-objective flowshop scheduling under unavailability constraints for minimizing total tardiness and peak power consumption. 2019 2nd Conference on Informatics and Applied Mathematics (IAM2019) https://labstic.univ-guelma.dz/sites/labstic.univ-guelma.dz/files/Files/proceedings.pdf

- Sekkal N., Belkaid, F., Boufellouh, R. (2019). Multi objective simulated annealing for identical parallel machines with deterioration effect and resources consumption. 2019 2nd Conference on Informatics and Applied Mathematics (IAM2019) https://labstic.univ-guelma.dz/sites/labstic.univ-guelma.dz/files/Files/proceedings.pdf

Introduction

Managers in the manufacturing industry are faced with the critical tasks of production scheduling and maintenance planning when implementing decisions on the shop floor. Allocating resources efficiently for production is another significant consideration for managers. These challenges pose difficulties in the planning process. This study is concerned with the flow shop scheduling problem under shared renewable and nonrenewable resources where in addition to the traditional processing machines, the system requires additional resources which can come either in forms of consumables or transportation resources - mainly automated guided vehicles - tasked with the material handling of the jobs throughout the system. This study focuses on two applications of the flow shop scheduling problem under shared renewable and nonrenewable resources: (1) a Flow shop and Maintenance Scheduling Problem under a Shared Nonrenewable Resource constraint and (2) an Energy-efficient Flow hop Scheduling Problem with Blocking and Collision-Free Transportation constraints. The study presents mathematical formulation for both cases and presents heuristic and metaheuristic approaches to find high quality solutions. Moreover, a thorough analysis of the impact of the additional resources on the scheduling is presented.

I Background and Motivation

Modern manufacturing companies face daily challenges in effectively managing production systems and optimizing the utilization of scarce resources. As soon as these resources become available, companies must efficiently exploit them. Additionally, the current industrial landscape demands the maintenance of consistent systems that can meet consumer requirements, including the delivery of high-quality products, adherence to strict deadlines, and the reduction of production costs to remain competitive in the global market. To address these challenges, enterprises have embraced new methodologies and philosophies within their facilities. Approaches such as Lean manufacturing, Just-In-Time manufacturing, and the Theory of Constraints have been adopted to help industrial companies achieve their goals by enhancing their processes and promoting efficient resource utilization. These methodologies aim to eliminate waste, reduce inventory levels, and identify and resolve bottlenecks that hinder system performance. The integration of these ideas has brought about significant changes in operations management and has given rise to new problem areas that require attention from operational and tactical decision-makers. These decision-makers must now grapple with the complexities associated with implementing and optimizing these methodologies to improve overall operational efficiency and achieve desired outcomes.

The scheduling and planning of operations play a pivotal role in determining the success or failure of a manufacturing organization. Well-designed and flexible schedules have the potential to increase throughput, enhance resource availability, and reduce costs. In a typical manufacturing process, there are two types of resources involved: renewable resources such as machines, transportation vehicles and operators, and nonrenewable resources such as energy and raw materials, which are necessary to support or complement ongoing operations. Production scheduling primarily involves the allocation of resources to a set of tasks according to a predefined timing plan. The objective is to optimize one or multiple performance measures, such as completion time, total production costs, and total energy consumption. However, scheduling problems are generally known to be challenging, even for simple configurations and small instances, as evidenced by the difficulty in finding efficient methods capable of generating optimal solutions within reasonable computational time. As production environments become increasingly complex and problem sizes continue grow, exact solution methods are becoming computationally less useful. Furthermore, conflicts arise when different departments or services within the organization set conflicting objectives, adding another layer of complexity to the scheduling models. Production scheduling and maintenance planning, in particular, emerge as the main sources of these conflicts. Balancing the requirements and objectives of these two areas further complicates the scheduling process. Overall, the challenges associated with production scheduling and maintenance planning do highlight the need for advanced methodologies and techniques to effectively address these conflicts and optimize scheduling decisions in complex manufacturing environments.

Production and maintenance services play vital roles in the success of business management within an industrial enterprise, despite having distinct objectives. The primary objective of production is to maximize resource and equipment utilization to meet customer demand and agreed-upon delivery dates. However, this pursuit of maximum utilization can increase the risk of machine breakdown, thereby reducing machinery availability. On the other hand, maintenance services aim to maximize equipment availability by implementing scheduled inspections and preventive actions, such as filter replacements, adjustments, and lubrication. These activities are critical for ensuring equipment reliability. However, they can also disrupt production plans and schedules. To mitigate conflicts between production and maintenance, effective communication and synchronized cooperation between the two services are vital. Establishing clear channels of communication and fostering collaboration helps align the objectives of both services. This ensures that production plans take into account maintenance requirements, and maintenance activities are strategically scheduled to minimize unexpected disruptions to production. By promoting a harmonious relationship between production and maintenance services, industrial enterprises can strike a balance between maximizing resource utilization and ensuring equipment availability. This balance ultimately leads to improved operational efficiency and customer satisfaction.

Flow shop scheduling in particular is a prevalent problem in various industries such as semiconductor manufacturing, steel-rolling, and pharmaceutical and chemical processes. These industries involve sequential transformations of raw materials throughout the system. Each job requires specific resources at each stage, like energy in forms of heat or mechanical forces, chemical additives and transportation resources to carry it from one stage to the next. However, processors involved in the transformation can experience degradation over time which makes them susceptible to breakdowns, necessitating an optimal maintenance policy. Moreover, the transportation resources may run out of energy and require a recharging process to keep them functional. Therefore, the proper allocation of renewable and nonrenewable resources is a necessary task to achieve the best possible performance of the system. Improper assignments of these resources can result in system unavailability and idleness, as operations are compelled to wait for the arrival of resources before commencing their processes. This not only leads to delays but also increases waste levels, consequently driving up costs. Managing these renewable and nonrenewable resources requires careful consideration of maintenance responsibilities as well. The allocation and utilization of resources must be synchronized with maintenance activities to ensure smooth operations. Neglecting maintenance requirements can have adverse effects on the availability and performance of machines, potentially leading to breakdowns, inefficiencies, and additional costs. Therefore, an integrated approach is necessary, where the allocation of resources and maintenance tasks are coordinated effectively. By aligning these two aspects, companies can optimize resource utilization, reduce waste, minimize downtime, and ultimately improve overall operational efficiency while keeping costs in check [1].

From a sustainability point of view, modern manufacturing facilities heavily depend on diverse energy resources to sustain their production operations. The industrial sector accounts for roughly 54% of global energy consumption, as reported in the 2016 International Energy Outlook [2]. To illustrate, in Germany, the manufacturing sector alone contributes to over 47% of the total national energy usage [3]. Meanwhile, in China, manufacturing enterprises consume approximately 50% of the country's electricity [4]. Moreover, the rapid growth of the manufacturing sector in Africa is projected to increase its industrial energy consumption by 30% by 2040 [5]. It's important to note that a significant portion of this energy is sourced from traditional nonrenewable fuels like oil and gas, which play a substantial role in environmental pollution and global warming. Due to these environmental concerns and the recent surge in energy prices, manufacturing companies are under significant economic pressure to reduce their reliance on polluting energy sources, promote the use of renewable energy, and enhance overall energy efficiency. Efforts to address these challenges are already underway. For instance, energy efficiency initiatives within the European Union aim to achieve a 27% increase in efficiency by 2050, with a focus on reducing primary energy demands [6]. However, to systematically improve energy efficiency, it is imperative to identify and distinguish between energy requirements associated with different components and phases of the production process [7]. Monitoring energy consumption modules linked to processes such as material transformation, machine setup, and material handling is essential for devising a comprehensive energy efficiency strategy for a production system.

The majority of previous research aimed at reducing energy consumption in manufacturing has primarily concentrated on equipment design [8 – 11], product design [11], and, more recently, production management. However, both equipment and product redesign necessitate substantial financial investments. Making such decisions, especially in light of increasing taxes and regulations related to energy and carbon emissions, is often impractical for most small and medium-sized manufacturing enterprises [12]. Consequently, there is a growing interest in shifting focus towards green scheduling as a more practical approach to enhancing carbon efficiency in manufacturing systems [13]. In recent years, research on green scheduling has predominantly centered on energy conservation in the context of production management and machine scheduling. However, transportation operations and material handling systems have not received a commensurate level of attention. Remarkably, approximately 21% of greenhouse gas emissions in the European Union in 2017 were attributed to transportation activities [14]. Surprisingly, only 5% of the published works on green scheduling address transportation energy efficiency within production systems [15]. In a typical automated production system, jobs are retrieved from an input device and transported between machines situated in various areas of the shop floor to fulfill the processing requirements. Each machine contributes partially to the production of the final product, which is then collected by an output device and removed from the system. The transportation tasks are carried out by material handling resources, such as Automated Guided Vehicles (AGV). AGVs are autonomous material handling devices widely employed to support various production environments, automated storage and retrieval systems, and port terminals [16].

Many enterprises worldwide are actively pursuing greater flexibility and reduced carbon emissions by adopting battery-powered Automated Guided Vehicles (B-AGVs) instead of conventional fuel-powered AGVs. B-AGVs are not only more environmentally friendly by consuming electrical energy, but they also hold substantial economic potential. This potential arises from their lower maintenance costs, increased energy efficiency in their powertrain, and the relatively lower electricity expenses compared to diesel costs. These factors make B-AGVs a more sustainable and cost-effective choice [17, 18]. However, due to their limited battery capacity, B-AGVs require recharging or battery replacement when their energy levels become critically low, hindering their availability for certain transportation tasks. During this downtime, the AGV's absence reduces the overall transportation system's capacity, affecting AGV dispatching and production scheduling. Furthermore, the method of recharging batteries varies depending on the implemented battery management system within the facility. The two most common approaches for AGV battery charging are battery swapping, where the AGV's battery is replaced with a fully charged one at a designated station, and automatic charging, where the depleted battery remains onboard during recharging, rendering the AGV unavailable at the charging station [19]. Regardless of the chosen charging scheme, if not managed efficiently, the unavailability of some AGVs can disrupt the production system. Consequently, considering battery management in AGV dispatching and routing can yield several advantages, including reduced transportation energy consumption and the uninterrupted flow of production [20].

II Objectives and Importance of the Study

This research addresses two distinct cases of the flow shop scheduling problem under additional resources. The first concerns the Flow shop and Maintenance Scheduling Problem with a Shared Nonrenewable Resource constraint (FMSP-SNR). It considers a global nonrenewable resource scarcity that is shared among all production machines throughout the planning horizon. The objective is to minimize the maximum completion time, often denoted as makespan ($Cmax$) and total production costs (TPC) which includes corrective and preventive maintenance costs engendered from both the job processing machines and the resource production center. Other costs such as resource consumption cost and resource holding costs are also taken into account. Secondly, this research sets out to investigate an intricate problem known as the Energy-efficient Flow shop Scheduling Problem with Blocking and Collision-Free Transportation constraints (EFSP-BCFT). Specifically, it addresses the simultaneous scheduling of a permutation flow shop system with sequence-dependent setup times (SDST) and the scheduling of a single-load multi-AGV transportation system as a secondary resource with the objectives of minimizing both the makespan, and the total energy consumption (TEC) of the scheduling. Regarding machine scheduling, this study takes into account energy consumption during setups and adheres to zero-buffer constraints. Concerning AGV scheduling, it considers battery energy consumption during loaded and empty travels, loading and unloading of jobs on and off the machines, and energy usage during idle states. Additionally, the research incorporates collision prevention constraints among individual AGVs and includes AGV speed control measures.

From an academic perspective, the authors believe that this research represents a pioneering effort as it is – To our current understanding - the first of its kind to consider the FMSP-SNR with consideration of resource supply control, maintenance planning affecting the nonrenewable resource supply equipment, and resource inventory costs and also consider the EFSP-BCFT with AGV speed control and battery management aspects. Much of the existing literature on joint production scheduling and maintenance planning and literature on production and transportation scheduling often focuses solely on machine scheduling independently, neglecting resource production control, resource holding costs and material handling aspects or assuming that nonrenewable resources are pre-fixed both in terms of their total amount or delivery patterns and that they are unaffected by system breakdowns. Other previous works presuppose that transportation resources are perpetually available without considering issues like collision prevention, speed control, or battery management. Another body of research places substantial emphasis on AGV dispatching, routing, and control, while giving less attention to machine scheduling or assuming that production planning is predetermined and simply aims to optimize transportation scheduling. Furthermore, a significant portion of previous research primarily concentrates on single-objective optimization (e.g., Makespan, Total tardiness, Total traveled distance of AGVs), whereas the complexity of multi-objective scheduling problems, characterized by conflicting objectives, demands the discovery of multiple, non-unique solutions [16]. From a practical standpoint, this thesis endeavors to tackle real-world challenges commonly encountered in various industrial settings. It investigates the influence of maintenance planning and system reliability on resource supply and explores the impact of AGV battery management and speed control, which are often overlooked aspects [21, 22], on the energy efficiency and productivity of the system.

III Research Questions and Contributions

This study endeavors to address the flow shop scheduling problem with shared renewable and unrenewable resources through an exploration of the following research questions:

1. In light of various distinctive features present in the FMSP-SNR such as resource production control, maintenance planning on the resource production center and resource consumption and holding costs and features present in the EFSP-BCFT like zero-buffer requirements between consecutive machines, sequence-dependent setup times and energy considerations, AGV collision avoidance, battery capacity constraints, AGV guide-path layouts, and transportation speed control, how can we formulate a comprehensive mathematical model for both cases?
2. What are the limitations of both mathematical formulations in terms of generating optimal solutions?
3. Given the inherent computational complexity of FMSP-SNR and EFSP-BCFT, what represents the most efficient approaches for obtaining high-quality solutions?
4. What is the impact on the scheduling of both machines and additional resources when exposed to different scenarios involving variations in model parameters and data?
5. What are the limitations of the proposed solution approaches and what can be proposed to mitigate such limitations?

In response to the research questions, this study makes the following significant contributions:

- *Flow shop and Maintenance Scheduling Problem with a Shared Nonrenewable Resource constraint:*
    1. Proposal of a novel integrated model that simultaneously considers production, maintenance, and global resource constraints. To the best of our knowledge, no existing study in the literature has addressed aspects such as the maintenance planning, resource supply control and resource holding costs in a flow shop scheduling environment in a unified model.
    2. Development of a fast computational method for decoding, constructing and evaluating solutions. This method uses priority rules for scheduling individual operations iteratively and computing the total resource holding cost simultaneously in a reasonable amount of time.

3. Adaptation of two well-known metaheuristic algorithms: The Nondominated Sorting Genetic Algorithm (NSGA-II) and a developed bi-objective version of the Particle Swarm Optimization algorithm (BOPSO) to generated near-optimal solutions.
4. Proposition of a novel Bi-Objective Randomized Local Search (BORLS) procedure to generate multiple high-quality nondominated solutions in a single run. These solutions can be used to enhance the initial population of a metaheuristic or as an independent search algorithm. The BORLS exploits the bi-objective nature of the problem by obtaining a high-quality initial solution of minimum makespan value, then gradually explores the near-optimal front of solutions until a final solution with minimum total production cost is achieved. A comprehensive set of computational experiments demonstrate the effectiveness of the proposed BORLS method both as an independent algorithm as well as a metaheuristic-enhancement procedure.

- *Energy-efficient Flow shop Scheduling Problem with Blocking and Collision-Free Transportation constraints:*
  1. Development of an integrated energy-efficient scheduling model that takes into consideration all the previously mentioned problem characteristics and objectives. Notably, this model incorporates collision-free transportation constraints on AGV movements, AGV speed control, and battery constraints simultaneously. To the best of our information, no existing study in the literature has addressed these three critical aspects in a unified model.
  2. In light of the problem's complexity, the proposal of an Enhanced Multi-Objective Ant Colony Algorithm (EMOACA) which is thoroughly evaluated through a series of computational experiments and statistical assessments, in comparison to various other metaheuristics. The proposed algorithm employs an array of innovative problem-specific heuristics and search techniques, facilitating both efficient exploration and exploitation of the search space. The results demonstrate that the enhancements integrated into the algorithm significantly enhance its capabilities in terms of diversity, spread, convergence, and overall performance when compared to alternative metaheuristic algorithms.
  3. Utilization of the EMOACA algorithm's results to analyze the impact of diverse problem parameters, including transportation times, idle power demand, battery capacity, and AGV fleet size, on the scheduling of machines and AGVs. Numerous scenarios are meticulously examined, accompanied by comprehensive multi-objective analyses. The findings underscore that the energy-conscious AGV scheduling and speed control approach introduced in this thesis effectively leads to improved optimization solutions for EFSP-BCFT.

IV Thesis Outline

- Chapter 1 is dedicated to discussing the relevant basic concepts related to the production scheduling problem under shared resource constraints which are fundamental to understanding the problem and its various aspects with the purpose of establishing the theoretical and empirical foundation necessary for the study.
- Chapter 2 focuses on reviewing the existing literature related to the FMSP-SNR and EFSP-BCFT with an aim to summarize and synthesize the key findings, methodologies, and insights from previous research to help the reader grasp the context in which the research is situated and highlight the gaps or areas where the study aims to contribute new insights and solutions.
- Chapter 3 investigates the flow shop and maintenance scheduling problem under a shared nonrenewable resource constraint. A mathematical model of the problem is presented and a bi-objective randomized local search procedure is proposed along with the NSGA-II and BOPSO metaheuristics as solution approaches. Then a series of comprehensive experimental tests are carried out to assess the potential of the proposed solution methods to effectively solve the problem.
- Chapter 4 investigates the energy-efficient flow shop scheduling problem with blocking and collision-free transportation constraints. A mathematical model of the problem is presented and the proposed enhanced multi-objective ant colony algorithm is systematically detailed. Then a series of comprehensive experimental tests are carried out to assess the energy-saving potential of the proposed methodology and approving its performance.
- Chapter 5 concludes the study and outlines potential avenues for future research

Chapter 1

Basic Concepts

This chapter is devoted to the exploration of fundamental concepts pertinent to the production scheduling problem within the context of shared resource constraints. These concepts are integral to comprehending the intricacies of the problems treated in this study and their multifaceted dimensions. The objective is to establish both the theoretical underpinnings and the empirical and practical groundwork essential for the ensuing study.

1.1 Production Scheduling Problems

Scheduling is a fundamental decision-making process routinely employed across various manufacturing and service industries. Its primary purpose is to allocate resources to specific tasks within predefined time frames, with the overarching aim of optimizing one or more objectives [22]. Within an organization, resources and tasks can assume diverse forms. Resources might encompass machinery within a workshop, runways at an airport, construction crews at a worksite, processing units in a computational environment, and various other possibilities. Conversely, tasks (or more formally known as jobs) can manifest as operations in a production process, aircraft take-offs and landings at an airport, project milestones in construction, execution of computer programs, and more [23]. Each task typically possesses attributes like priority levels, resource requirements, earliest feasible start times, and due dates [24]. The objectives pursued in scheduling can exhibit great variability as well. For instance, one objective might revolve around minimizing the total time required for the last task's completion, while another objective might center on reducing the number of tasks completed after their respective due dates [22].

In the next subsections, the different basic concepts and characteristics of scheduling problems are detailed including job specific attributes, machine environments and objective functions that are often treated in the literature.

1.1.1 Job Specific Parameters

In all the scheduling problems, it is assumed that there are a finite number of jobs and machines involved. We represent the number of jobs as n and the number of machines as m. Typically, the index j is used to refer to a job and the index i to refer to a machine. When a job involves multiple processing steps or operations such as in flow shop and job shop environments, the pair OP_{ij} is used to denote the specific processing step or operation of job j on machine i. The following data parameters are often linked to job j [22].

- **Processing time (P_{ij}):** The parameter P_{ij} signifies the time required for job j to be processed on machine i. The index i is omitted if the processing time of job j is independent of the machine or if job j is designated to be processed on a single machine.
- **Release date (r_j):** The release date, denoted as r_j, for job j is sometimes referred to as the ready date. This represents the moment when job j enters the system, signifying the earliest possible time at which job j can commence its processing.
- **Due date (d_j):** The due date d_j associated with job j signifies the promised shipping or completion date, which is the date on which the job is committed to be delivered to the customer. While it is permissible to complete a job after its due date, it incurs a penalty. When strict adherence to a due date is necessary, it is referred to as a deadline and is considered as a constraint on the scheduling problem.
- **Weight (w_j):** The weight, denoted as w_j for job j, essentially serves as a prioritization factor, indicating the significance of job j in comparison to other jobs within the system. For instance, this weight might represent the real cost associated with retaining the job within the system. This cost could encompass holding or inventory expenses, or it could reflect the value already added to the job.

1.1.2 Nontraditional Job-Specific Attributes

Depending on the specific scheduling problem, jobs may have other nontraditional attributes and requirements such as:

- **Transportation:** Some jobs may require access to specific material handling equipment with particular capabilities and load capacities such as automated guided vehicles forklifts conveyors...etc [25]. This requirement will translate in forms of transportation times between machines which depends on several factors:

  (1) Distance between machines
  (2) Transportation speed
  (3) Routing and congestion issues
  (4) Job specific load

- **Personnel:** Jobs may require skilled labor or personnel with specific qualifications. This is often seen in service scheduling, where tasks may require employees with particular skills or certifications [26].

- **Material:** Certain jobs may involve the use of specific materials or resources that need to be available in sufficient quantities. This is crucial in industries like construction, where materials need to be on-site for tasks to commence [27].
- **Energy or Power:** Jobs may have energy or power consumption requirements. For example, some tasks may require access to a certain amount of electricity, gas, or other energy sources [28, 29].
- **Space or Location:** Certain jobs may need specific workspace configurations or access to particular locations. This is relevant in industries like logistics, where jobs require access to loading docks, storage areas, or specific facilities [30].
- **Tooling or Equipment:** Jobs may require access to specialized tools, equipment, or instrumentation. For instance, scientific experiments or research tasks may necessitate specific laboratory equipment [31].

These resource requirements add an additional layer of complexity to scheduling problems. Balancing job sequencing while considering these constraints is crucial to achieving efficient resource utilization and minimizing idle time. Scheduling algorithms must take into account not only job attributes like processing times and due dates but also the availability and constraints of the required resources to generate feasible and efficient schedules.

1.1.3 Machine Scheduling Environments

A scheduling problem is often defined using a triplet notation $\alpha|\beta|\gamma$, introduced by Graham et al. [32]. The α component characterizes the machine environment and comprises a single entry. The β component furnishes information about processing attributes and constraints, which may include no entry, a single entry, or multiple entries. The γ component outlines the objective to be minimized, typically containing a single entry.

The potential machine environments that can be specified in the α field are:

- **Single machine (1):** The scenario involving a single machine is the most straightforward among all possible machine setups, and it serves as a fundamental case from which all other more intricate machine environments can be derived.
- **Identical machines in parallel ($\boldsymbol{Pm}$):** In this scenario, there are m identical machines operating simultaneously. Each job j necessitates a single operation and can be executed on any of the m machines or on any machine within a predefined subset.
- **Machines in parallel with different speeds ($\boldsymbol{Qm}$):** In this setting, there are m machines operating in parallel, each with its own unique speed denoted by v_i. The processing time P_{ij} for job j on machine i is calculated as P_j/v_i, assuming that job j exclusively receives processing from machine i. This configuration is often termed as uniform machines.
- **Unrelated machines in parallel ($\boldsymbol{Rm}$):** This environment represents a broader extension of the previous scenario. Here, m distinct machines operate in parallel, and each machine i can process job j at a specific speed denoted as v_{ij}. The processing time P_{ij} for job j on

machine i is computed as P_{ij}/v_{ij}, under the assumption that job j exclusively receives its processing from machine i.

- **Flow shop (Fm):** In this configuration, there are m machines arranged in series. Each job undergoes processing on all m machines in a predetermined sequence, where every job must adhere to the same sequence: starting with machine 1, progressing to machine 2, and so forth. Following completion on one machine, a job joins the queue at the next machine. Typically, all queues operate under the First In First Out (FIFO) rule, which means that a job cannot bypass another while waiting in a queue.
- **Flexible flow shop (FFc):** A flexible flow shop represents a broader adaptation of both the flow shop and parallel machine environments. Instead of having m machines in series, it consists of c stages arranged in series, each stage housing multiple identical machines in parallel. Each job follows a specific sequence, starting with stage 1, then proceeding to stage 2, and so forth. In each stage, a group of parallel machines is available, and job j requires processing on only one of these machines, with any machine capable of performing the task. The number of parallel machines in each stage may differ from one another. The queues between different stages may or may not adhere to the First Come First Served (FCFS) discipline. It's worth noting that in the literature, flexible flow shops have at times been referred to as hybrid flow shops and also as multi-processor flow shops.
- **Job shop (Jm):** In a job shop featuring m machines, each job follows a specific pre-established route. There are two categories of job shops: those in which each job visits each machine at most once, and those in which a job may visit a machine more than once.
- **Flexible job shop (FJc)**: A flexible job shop represents an expanded version of both the job shop and parallel machine environments. Instead of having m machines in series, it features c work centers, each equipped with multiple identical machines operating in parallel. Each job has its unique path to traverse within the shop, with job j needing processing at each work center on just one machine, and any machine can perform the task.
- **Open shop (Om):** In an open shop configuration, there exist m machines, and each job must undergo processing on all m machines. Importantly, some of these processing times may be zero. There are no constraints on how each job should be routed through the machine environment. The scheduler is granted the authority to establish a unique route for each job, and different jobs may follow distinct routes as per the scheduling requirements.

1.1.4 Scheduling Restrictions

The β field can contain various entries to describe processing restrictions and constraints, and these entries may include multiple specifications. In cases where these entries are not mentioned, the corresponding restrictions are not applicable. Potential entries in this field include:

- **Release dates (r_j):** When the symbol r_j is present in the β field, it signifies that job j is not permitted to commence its processing prior to its specified release date, denoted as r_j.
- **Preemptions ($prmp$):** The presence of $prmp$ in the β field indicates that preemptions are permissible. In such cases, it is not obligatory to continue a job on a machine once it has

started processing until its completion. The scheduler has the authority to interrupt the processing of a job (preempt) at any point and replace it with a different job on the machine. Importantly, the progress made by a preempted job is not lost. When a preempted job is subsequently placed back on the machine (or on another machine in the case of parallel machines), it only requires the machine for the remaining processing time.

- **Precedence constraints ($prec$):** Precedence constraints can be found in either a single-machine or parallel-machine environment. They dictate that one or more jobs must be completed before another job is permitted to commence its processing. There are various specialized forms of precedence constraints:

    (1) *Chains*: If each job has at most one predecessor and at most one successor;
    (2) *Intree*: When each job has at most one successor;
    (3) *Outtree*: Conversely, if each job has at most one predecessor**.**

- **Sequence-dependent setup times (ST_{jk}):** The variable ST_{jk} signifies the setup time that is associated with the specific sequence of processing between jobs j and k. To elaborate, ST_{0k} represents the setup time for job k if it is the initial job in the sequence, and ST_{j0} denotes the cleanup time following the processing of job j if it is the final job in the sequence. It is important to note that these setup times, ST_{0k} and ST_{j0}, may indeed be zero. If the setup time between jobs j and k is contingent upon the machine being used, the subscript i is added, resulting in ST_{ijk}.

  Machine setups and sequence-dependent setup times play a crucial role in scheduling problems, particularly in the context of flow shop scheduling and job shop scheduling. Machine setups refer to the activities required to prepare a machine for processing a specific job or task. These setups involve activities such as changing tools, adjusting machine parameters, loading materials, or configuring the machine for a different operation. These activities not only require time but also other resources, energy and costs, such as tool change costs or material change costs, which can be modelled using specific parameters for each. For example the parameter SE_{ijk} designates the setup energy required by machine i during the setup period between processing job j and job k.

  Examples of setup times include the following:

    - **Manufacturing:** In a manufacturing environment, changing a machine's tooling and reconfiguring its settings between different production runs represents a setup. For example, transitioning from cutting one material to another on a CNC machine requires a setup [33].
    - **Printing:** In a print shop, changing the printing plates on a press to switch from one print job to another or to change color involves a setup. The time and effort required for this activity can vary [34].
    - **Food processing:** In food processing plants, setup times for cleaning and configuring machines can vary based on the specific products being processed. Changing from processing one type of food product to another may require more extensive cleaning and setup [35].
    - **Chemical industry:** In chemical manufacturing, the need for setup times is common. For instance, setups are essential when it becomes necessary to clean a

machine between producing two different products. This step is crucial whenever the remnants from the initial product could disrupt the production of the second one [36].

Setup times can be classified in two categories:

(1) **Anticipatory setup times:** In this case the setup of the machine for the next job k can begin immediately after the completion of the current job j and does not require the arrival of k [37].
(2) **Non-anticipatory setup times**: In this case the starting of the setup of the machine for the next job k requires both the completion of jobs j and job k from the previous machine [33, 38, 39].

Other variants of the sequence dependent setup times may exist depending on the shop floor specificities [40]. Sometimes setups require that current job j is removed and transported from the site before the setup for next job k can begin. There can also be costs associated with the setups [41].

- **Job families ($\boldsymbol{fmls}$):** In this scenario, the n jobs are divided into F different job families. Within each family, jobs may exhibit varying processing times, but they can be consecutively processed on a machine without necessitating any setup in between. However, when the machine switches from processing jobs in one family to another, for instance, transitioning from family g to family h, a setup is required.
- **Batch processing ($\boldsymbol{batch(b)}$):** In this case, a machine is capable to concurrently process a certain number of jobs, denoted as b This means it can handle a batch comprising up to b jobs simultaneously. It's important to note that the processing times of the individual jobs within a batch may not be uniform. The entire batch is considered complete only when the last job in the batch has finished processing; signifying that the overall batch completion time is determined by the job with the longest processing time.
- **Breakdowns ($\boldsymbol{brkdwn}$):** Machine breakdowns indicate that a machine may not be consistently accessible for production. During these times, the machine is temporarily unavailable. It is assumed that these periods of unavailability are predetermined and fixed, possibly due to factors like shifts or scheduled maintenance. Machine breakdowns are occasionally referred to as machine availability constraints.
- **Machine eligibility restrictions ($\boldsymbol{M_j}$):** The presence of the symbol M_j in the β field is relevant when the machine environment consists of m machines operating in parallel (Pm). When M_j is specified, it means that not all m machines are suitable for processing job j. The set M_j represents the group of machines capable of handling job j.
- **Permutation ($\boldsymbol{prmu}$):** In a flow shop environment, one possible constraint is the adherence to the FIFO discipline for the queues in front of each machine. This means that the original order or permutation in which the jobs enter the first machine is preserved and maintained consistently throughout the system. When the FIFO discipline is enforced, the flow shop is known as a permutation flow shop.

- **Blocking ($block$):** Blocking is a phenomenon that can occur in flow shops. In a flow shop where there is a limited buffer between two consecutive machines, it's possible that when the buffer is full, the upstream machine cannot release a completed job. This situation results in blocking, where the completed job must remain on the upstream machine, preventing it (i.e., blocking it) from working on the next job. The most commonly examined form of blocking involves having zero buffers between any two successive machines. In such cases, a job that has finished processing on a particular machine cannot leave that machine if the preceding job has not yet completed its processing on the next machine. Consequently, the blocked job also hinders (or blocks) the commencement of processing for the next job on the same machine.

  In the model with blocking which is explored in chapter 4, the assumption is that the machines operate according to the FIFO principle. In this context, $block$ implies $prmu$.
- **No-wait (nwt):** The no-wait requirement is another aspect that can manifest in flow shops. In this scenario, jobs are not permitted to pause or wait between two consecutive machines. This necessitates delaying the start time of a job at the first machine to guarantee that it can smoothly traverse the entire flow shop without encountering any delays at any machine.
- **Recirculation ($rcrc$):** In a job shop or flexible job shop, recirculation can occur when a job is allowed to revisit a machine or work center more than once.

Other entries that may appear in the β field are typically self-explanatory. For instance, $d_j = d$ implies that all due dates are equal. Due dates are typically not explicitly stated in this field; the nature of the objective function provides enough information to determine whether jobs have due dates or not.

1.1.5 Objective Functions

The objective function to be minimized always hinges on the completion times of the jobs, which naturally rely on the schedule [22]. Specifically, the completion time of the operation of job j on machine i is represented as C_{ij}. The time at which job j departs from the system (essentially, its completion time on the final machine it requires processing on) is denoted as C_j. It's worth noting that the objective function can also take into account due dates.

The lateness of job j is formally defined as:

$$L_j = C_j - d_j$$

This expression yields a positive value when job j is completed after its designated due date (i.e., it is late) and a negative value when it is completed earlier than its due date (i.e., it is early).

The tardiness of job j is described by the equation:

$$T_j = \max(C_j - d_j, 0)$$

The key distinction between tardiness and lateness lies in the fact that tardiness is never negative, whereas lateness can be either positive (when a job is late) or negative (when it is early).

The unit penalty of job j is characterized by the following definition:

$$U_j = 1 \quad \text{if} \quad C_j > d_j, \qquad 0 \quad \text{otherwise}$$

Examples of potential objective functions to minimize include:

- **Makespan ($\boldsymbol{Cmax}$):** The makespan, represented as $\max(C_1, \ldots, C_n)$, corresponds to the completion time of the last job to exit the system. Achieving a minimum makespan typically signifies efficient utilization of the machine(s).
- **Maximum Lateness ($\boldsymbol{Lmax}$):** $Lmax$ is defined as $\max(L_1, \ldots, L_n)$. It provides a measure of the worst-case deviation from the due dates.
- **Total weighted completion time ($\sum \boldsymbol{w_j C_j}$):** The summation of the weighted completion times of all n jobs offers insights into the cumulative holding or inventory costs associated with the schedule. The summation of completion times is commonly known as flow time.
- **Total weighted tardiness ($\sum \boldsymbol{w_j T_j}$):** This cost function is more comprehensive than the total weighted completion time and offers a broader measure of the associated costs.
- **Weighted number of tardy jobs ($\sum \boldsymbol{w_j U_j}$):** The weighted number of tardy jobs serves as not only an academically significant measure but is also a practical objective, given its ease of recording and relevance in real-world scenarios.

All the aforementioned objective functions are categorized as regular performance measures, which means they are functions that increase or remain the same as $C_1, \ldots, C_n$ increase. However, recent research has delved into objective functions that deviate from regularity. For instance, when job j possesses a due date d_j, it may incur an earliness penalty, where the earliness of job j is determined by:

$$E_j = \max(d_j - C_j, 0)$$

This earliness penalty decreases as C_j increases and is, therefore, a non-increasing function of C_j. An objective such as the total earliness plus the total tardiness:

$$\sum_{j=1}^{n} E_j + \sum_{j=1}^{n} T_j$$

is thus considered non-regular. A more generalized objective that is not regular is the total weighted earliness plus the total weighted tardiness:

$$\sum_{j=1}^{n} w_j' E_j + \sum_{j=1}^{n} w_j'' T_j$$

where the weight associated with the earliness of job j (w_j') may differ from the weight associated with the tardiness of job j (w_j'').

1.1.6 Nontraditional Objective Functions

Production scheduling involves optimizing various aspects of manufacturing processes. While objective functions like makespan and total completion time are widely used, non-traditional objective functions address specific needs and constraints in different industries and scenarios. Next is a list of such objective functions in production scheduling with specific details:

- **Energy Consumption Minimization:** Minimize the total energy consumption of machines during the production process. This objective is relevant in environmentally conscious industries where reducing energy costs is a priority [42, 43].
- **Resource Utilization Balancing:** Balance the utilization of resources (e.g., machines, workers) to avoid bottlenecks and underutilized capacity. It is Essential in scenarios where resource imbalances lead to inefficient production and increased costs [44].
- **Carbon Emission Reduction:** Minimize the carbon footprint or greenhouse gas emissions associated with production processes. This goal is prominent in sustainable manufacturing practices and industries striving for environmental responsibility [29].
- **Supply Chain Cost Minimization:** Minimize the overall cost incurred throughout the entire supply chain, including production, transportation, and inventory holding. This is a useful objective for optimizing end-to-end supply chain operations to reduce costs, wastes and improve efficiency [45].

1.2 Maintenance and Reliability Concepts

Maintenance planning in manufacturing is a crucial aspect of ensuring the smooth and efficient operation of production facilities and equipment. It involves a systematic approach to managing maintenance activities, with the goal of maximizing equipment uptime, extending asset lifecycles, minimizing downtime, and optimizing overall productivity [46, 47]. In the next subsections, the different types of maintenance tasks and reliability models often used to model the state of the machines are briefly reviewed.

1.2.1 Maintenance Models for Production Systems

In the context of production scheduling, several types of maintenance activities can be classified based on their objectives and timing

- **Preventive Maintenance (PM):** Preventive maintenance involves scheduled inspections, servicing, and replacement of components to prevent equipment failures. PM plans are developed based on equipment manufacturer recommendations, historical data, and industry best practices. PM tasks are scheduled at predetermined intervals or based on the equipment's operating hours. They help to reduce the likelihood of unexpected breakdowns, extend equipment lifespan, and enhance overall reliability.
- **Corrective Maintenance (CM):** Corrective maintenance, also known as breakdown or reactive maintenance, involves repairing equipment after it has failed. While it's generally more costly and disruptive than preventive maintenance, it is sometimes necessary for unexpected breakdowns. It is used as quick resolution of issues but can result in significant downtime and higher repair costs.
- **Predictive Maintenance:** Predictive maintenance involves monitoring equipment conditions in real-time using sensors and advanced analytics to predict when maintenance is required. Maintenance activities are performed when monitoring data indicates a potential issue or at the optimal point to avoid unplanned downtime, reduce unnecessary maintenance effort and optimize maintenance resources.
- **Reliability-Centered Maintenance (RCM):** Reliability-Centered Maintenance is a systematic approach to determine the most effective maintenance strategy for each component in a system, balancing performance, cost, and risk. Tailored to the specific needs and criticality of each component, RCM seeks to maximize system reliability, minimize maintenance costs, and focus resources on critical components.
- **Shutdown Maintenance:** Also known as overhaul or turnaround maintenance, Shutdown Maintenance involves a complete shutdown of the production system for extensive maintenance and repairs. It is typically scheduled during planned downtime periods in order to exploit this time for thorough inspection, repair, and replacement of equipment components, ensuring long-term reliability.
- **Opportunistic Maintenance:** Opportunistic maintenance involves taking advantage of planned shutdowns or production pauses between successive shifts to perform maintenance activities in order to minimize disruptions and reduce the impact on production schedules.

An effective maintenance strategy involves a combination of at least some of these maintenance types, tailored to the specific needs of the production system.

1.2.2 Reliability Models

Reliability models are mathematical or statistical tools used to analyze and predict the reliability, availability, and performance of systems, products, or components over time. These models help engineers and decision-makers understand how systems or products will perform in real-world conditions and make informed decisions about maintenance, design improvements, and risk management. Some common reliability models and concepts involve the following:

- **Reliability Function ($R(t)$):** The reliability function, often denoted as $R(t)$, represents the probability that a system, component, or product will perform its intended function without failure up to a specified time t.
- **Failure Rate (λ):** The failure rate, often denoted as λ, is a measure of how quickly a component or system is expected to fail over time. It is the derivative of the reliability function with respect to time.
- **Mean Time Between Failures (MTBF):** MTBF is the average time between consecutive failures of a system or component. It is often used as a reliability metric and can be calculated as the reciprocal of the failure rate (MTBF = $1/\lambda$).
- **Hazard Function ($z(t)$):** The hazard function, denoted as $z(t)$, represents the instantaneous failure rate at a given time t. It provides information about how the failure rate changes over time.
- **Exponential Reliability Model:** The exponential reliability model assumes a constant failure rate over time. It is simple and is often used to model systems with a memoryless failure process. More details are in section 1.2.3.
- **Weibull Reliability Model:** The Weibull distribution is a versatile model used to describe a wide range of failure patterns, from early-life failures (infant mortality) to wear-out failures. It can be adjusted to fit various types of failure data.

1.2.3 The Exponential Reliability Model

The exponential reliability model is a fundamental mathematical model used in reliability engineering to describe the behavior of systems, components, or products over time. It is particularly useful for modeling systems with constant failure rates and is based on the exponential probability distribution. Next is an expansion on the exponential reliability model:

Probability Density Function (PDF): In the exponential reliability model, the probability density function (PDF) is defined as:

$$f(t) = \lambda e^{-\lambda t}$$

where $f(t)$ is the PDF, representing the probability of failure occurring at time t. λ is the failure rate, which is a constant representing the rate at which failures occur over time.

Cumulative Distribution Function (CDF): The CDF gives the probability that a failure will occur before or at a specific time t. It is calculated as follows:

$$F(t) = 1 - e^{-\lambda t}$$

As for the MTBF, the exponential reliability model is characterized by a constant failure rate λ. The MTBF can be calculated as the reciprocal of λ:

$$\text{MTBF} = 1 / \lambda$$

One notable feature of the exponential distribution and the exponential reliability model is its memoryless property. This means that the probability of a component failing in the next time interval is independent of how long it has been in operation. In other words, past performance does not affect future failure probabilities. The exponential reliability model is often applied to systems or components with constant failure rates, where failures occur randomly and independently over time. Examples include electronic components like light bulbs, where the probability of failure remains relatively constant throughout their operational life.

While the exponential reliability model is useful for certain applications, it may not accurately represent the failure behavior of complex systems with varying failure rates. Many real-world systems experience different failure modes over time, such as "infant mortality" failures early in their life and wear-out failures as they age. In such cases, more complex reliability models like the Weibull distribution may be more appropriate.

The exponential reliability model can be used to estimate the reliability of a system or component over a specified time interval. Engineers and reliability analysts can use this model to make predictions about the likelihood of failure within a given period, aiding in maintenance planning and decision-making.

1.3 Transportation within a Manufacturing Facility

Transportation within a manufacturing facility is a critical aspect of the production process that involves the movement of materials, components, and finished products between various locations and workstations within the facility. Efficient transportation plays a key role in optimizing manufacturing operations, reducing lead times, minimizing costs, and ensuring smooth production flows [48].

1.3.1 General Overview

This section presents a brief overview of transportation in a manufacturing facility giving some insight on the different types of transportation systems involved in manufacturing shop floors, some key considerations, challenges and technological solutions when implementing transportation systems.

Transportation systems contribute significantly to the overall productivity, cost-effectiveness, and flexibility of manufacturing operations. Next is a review of some of the commonly used transportation systems used in manufacturing plants:

- **Conveyor systems:** Conveyor systems are mechanical devices designed for the efficient movement of materials, products, or items within a manufacturing plant or other industrial settings. These systems are crucial for streamlining production processes, improving efficiency through the utilization of vertical space, and reducing manual labor. They are versatile and suitable for conveying various types of products, from small components to bulk materials. Conveyors can be of different types namely: Belt, Roller, Chain, Slat, Screw, Bucket and Vibrating conveyors. Due to their low cost, adaptability to different layouts, ease of implementation and control, conveyors are widely used in industries such as manufacturing, agriculture, food processing, chemical, automotive, steel manufacturing, mining, logistics, distribution centers, warehouses, assembly lines, construction and aerospace.

- **Forklift trucks:** Forklift trucks, also known as forklifts or lift trucks, are industrial vehicles designed for the efficient handling and transportation of materials in various settings, including manufacturing plants, warehouses, distribution centers, and construction sites. Different types of forklift trucks include Counterbalance Forklifts, Reach Trucks, Order Pickers and Pallet Jacks.
- **Overhead Crain Systems:** Overhead crane systems, also known as bridge cranes or overhead traveling cranes, are industrial lifting equipment used for transporting heavy loads horizontally within a facility. These cranes are essentially composed of a Bridge, Hoist, Trolley and Runway Rails. These cranes are commonly found in manufacturing plants, warehouses, and other industrial settings such as shipyards and ports for assembling ship components and loading/unloading cargo containers in automated container terminals (ACT)
- **Automated Guided Vehicles:** AGVs are autonomous mobile robots that are designed to move materials or products in a controlled and programmed manner without the need for an onboard operator. More details are presented in section 1.3.2.

The choice of transportation systems in manufacturing plants depends on the specific requirements of the production process, including the types of materials being handled, the layout of the facility, and the desired level of automation. Often, a combination of different transportation systems is employed to create a comprehensive and efficient material handling strategy within manufacturing plants. Manufacturers must carefully evaluate their unique needs and consider factors such as cost, flexibility, and scalability when selecting, acquiring and implementing transportation systems in their facilities. Safety considerations are also paramount when moving heavy materials or products. Proper training, equipment maintenance, and safety protocols are essential. Effective space planning and organization are crucial to ensure that transportation routes and storage areas are optimized for smooth material flow. Finally, reducing the environmental impact of transportation, such as by using energy-efficient equipment, can be a priority for many manufacturing facilities [49].

1.3.2 Automated Guided Vehicles

Automated Guided Vehicles (AGVs) are autonomous or semi-autonomous mobile robots that are widely used in manufacturing facilities to handle various material handling tasks. They are designed to transport materials, components, and products within a facility without the need for human intervention. Fig. 1.1 shows an example of a battery-powered AGV.

AGVs offer several benefits, including increased efficiency, reduced labor costs, improved safety, and enhanced overall productivity. Next is an overview of the commonly used types of AGVs in a manufacturing facility [50]:

- **Tow Vehicles:** These AGVs are used to tow trailers, carts, or other containers loaded with materials or products. They are commonly employed in moving heavy loads or larger quantities of materials.
- **Unit Load Carriers:** Unit load carriers are AGVs designed to carry a single item or product at a time. They are suitable for precise and controlled transportation tasks.
- **Fork AGVs:** These AGVs are equipped with forks or clamps to lift, stack, and transport pallets or other types of containers. They are often used in warehouse and distribution center settings.
- **Assembly Line AGVs:** These AGVs are specifically designed for assembly line processes, such as delivering components to workstations or transporting products between production stages.

Fig. 1.1 A typical battery-powered AGV.

AGVs can move raw materials to production lines, transport work-in-progress (WIP) between workstations, and shuttle finished products to storage or shipping areas. AGVs play a crucial role in managing and organizing inventory within a manufacturing facility, ensuring that materials and products are stored and retrieved efficiently. Also, AGVs can support Just-In-Time (JIT) principles by delivering materials and components to production lines precisely when needed, reducing excess inventory. Moreover, AGVs can be employed in hazardous environments or situations where human safety is a concern.

AGVs use various guidance systems, including laser navigation, magnetic tape, vision systems, and inertial navigation, to navigate their environment. AGVs may communicate with other equipment, such as conveyor systems or automated storage systems, to coordinate material flow. In addition, AGVs can be integrated with manufacturing execution systems (MES) and enterprise resource planning (ERP) systems to optimize material flow and production scheduling making them an integral part of modern manufacturing facilities, contributing to increased efficiency and competitiveness. Their use continues to evolve with advancements in robotics, sensors, and artificial intelligence, making them more versatile and adaptable to various manufacturing environments and tasks [51].

The use of AGVs has brought about many benefits. For instance, AGVs operate continuously and consistently, reducing wait times and bottlenecks in material handling processes. They can be programmed to follow precise paths and perform tasks with high accuracy, or can be easily reprogrammed or reconfigured to adapt to changing production needs or facility layouts reducing the risk of errors and product damage [52]. By automating material handling tasks, they can lead to reduced labor costs and increased operational efficiency. In the meantime, employing an efficient AGV system can also be challenging. Implementing AGV systems can require a significant upfront investment in terms of hardware, software, and infrastructure. AGVs require regular maintenance to ensure their continued reliability and performance. Personnel may need training to operate, program, and maintain AGVs effectively. Facility layouts and infrastructure need to be designed or adapted to accommodate AGVs, including charging stations and navigational aids.

1.4 Complexity

Computational complexity theory is a branch of computer science that focuses on the study of the resources required to solve computational problems. It provides a framework for analyzing the efficiency and feasibility of algorithms by considering how their running time or space requirements grow with input size [53]. In the context of scheduling problems, computational complexity plays a crucial role in understanding the difficulty of finding optimal or near-optimal schedules for various types of tasks or jobs.

1.4.1 Computational Complexity Classes

Computational complexity theory provides a framework for understanding the difficulty of scheduling problems and other computational tasks. The classification of problems into complexity classes helps researchers and practitioners assess the feasibility of finding optimal solutions and select appropriate algorithms to tackle these problems effectively. The complexity of optimization problems include the following:

(1) **P (Polynomial Time):** Problems in this class can be solved in polynomial time, meaning the algorithm's running time is bounded by a polynomial function of the input size. These problems are considered efficiently solvable.

(2) **NP (Nondeterministic Polynomial Time):** Problems in this class may not have polynomial-time algorithms, but if given a proposed solution, one can verify its correctness in polynomial time. The question of whether P = NP is one of the most famous unsolved problems in computer science.

(3) **NP-Hard (Nondeterministic Polynomial Hard):** These are some of the most challenging problems in computational complexity. They are at least as hard as the hardest problems in NP and do not have known polynomial-time algorithms unless P = NP.

(4) **NP-Complete (Nondeterministic Polynomial Complete):** These problems are both in NP and NP-Hard. They are considered among the most difficult problems in NP and serve as benchmarks for the difficulty of other problems.

1.4.2 Computational Complexity of Scheduling Problems

A substantial body of research in deterministic scheduling has been dedicated to the pursuit of efficient algorithms, often referred to as polynomial time algorithms, to address scheduling problems. Nonetheless, it's crucial to acknowledge that numerous scheduling problems do not possess polynomial time algorithms and fall into the category of so-called NP-hard problems. Establishing the NP-hardness of a problem necessitates a formal mathematical proof [22].

The complexity of scheduling problems varies depending on the specific problem and constraints involved. Some scheduling problems are polynomial-time solvable, while others are NP-Hard. For example: The classical 2-machine flow shop scheduling problem with makespan minimization as objective can be solvable in polynomial time using the Johnson algorithm. However, the general case with 3 machines is NP-Hard. In general, scheduling problems belong to a class of combinatorial optimization problems, and there are typically two distinct approaches to solving them. The first approach involves employing exact algorithms, which are suitable for handling smaller problems and yield optimal solutions. In contrast, the second approach is characterized by the utilization of approximate methods, which are employed for tackling more intricate problems and result in solutions that are close to being optimal [54].

1.5 Exact Methods

Exact methods refer to solution techniques that can yield optimal solutions for combinatorial problems through intelligent exploration of solution spaces. However, it's important to note that they do not guarantee systematic polynomial-time solutions. Unfortunately, there are no general algorithms available that can solve typical optimization problems, such as scheduling problems, efficiently within polynomial time, as these problems are often NP-hard. Consequently, these exact methods find limited application in industrial settings. The realm of exact methods is diverse and encompasses various techniques, including linear programming, dynamic programming, and branch and bound methods [55].

1.5.1 Mixed Integer Linear Programming (MILP)

Linear programming is a versatile approach rooted in mathematical modeling for combinatorial optimization problems. In this method, the objective function, constraints, and decision variables are all linear. Essentially, linear programming seeks to determine the most optimal outcome for a given set of conditions. Many practical problems in the field of operations research can be expressed as linear programming problems. When dealing with continuous variables in a linear program, the most commonly utilized algorithms are the simplex method and the interior-point's method.

However, modeling scheduling problems often necessitates incorporating integer decision variables into the mathematical model. This introduces the concept of an integer linear programming model, which can have a notable drawback due to the potential proliferation of constraints and variables. Consequently, a significant portion of scheduling problems, typically NP-hard, cannot be solved optimally using integer programming alone. Conversely, relaxing these problems into continuous linear models allows for the derivation of lower bounds for minimization problems. The most effective approaches for solving these relaxed models include branch and cut, or specialized solvers like CPLEX.

Various approaches exist for modeling scheduling problems. For a specific scheduling problem, one can develop several MILP formulations employing different sets of decision variables. Nonetheless, the suitability of each formulation, as well as its strengths and weaknesses, largely depends on the specific problem under examination within a given context. In the modeling of scheduling problems, binary variables play a crucial role in representing the scheduling decisions and constraints. Pan [56] reviewed the different types of binary variables commonly used:

1.5.1.1 Position Variables

Position variables are used to determine the order in which jobs are scheduled in a sequence. For example, X_{ik} represents whether job or operation i is scheduled at position k in a job or operation sequence. Following this representation requires defining n^2 binary variables where n is the number of jobs or operations. In a permutation flow shop scheduling problem, each job has its own set of position variables. If there are n jobs, each job has n position variables. Only one of these variables can take the value 1 for each job, representing its position in the sequence. Each job i should therefore be scheduled at exactly one position k, ensuring that all jobs are scheduled exactly once. And no two jobs can be scheduled at the same position. i.e., $\sum_{i=1}^{n} X_{ik} = 1$ and $\sum_{k=1}^{n} X_{ik} = 1$ for all $i, k = 1, \dots, n$.

1.5.1.2 Precedence Variables

Precedence variables are used to model the order in which jobs are processed relative to one another. For example, X_{ij} represents whether job i is scheduled before job j. In a flow shop scheduling problem with non-renewable resource constraints or sequence dependent setup times, precedence variables help represent dependencies between jobs or operations. Constraints between each pair of jobs must be ensured so that jobs are scheduled in the correct order and prevent circular dependencies. For example, $X_{ij} + X_{ji} = 1$ for all $i \neq j$.

1.5.1.3 Time Variables

Time variables are used to model the temporal allocation of jobs to specific time slots. For example, X_{it} represents whether job i is being processed at time t. In resource-constrained scheduling or real-time scheduling, time variables are crucial for representing when and for how long each job is processed. Time variables ensure that a job is either actively being processed ($X_{it} = 1$) or not ($X_{it} = 0$) at a given time t. Resource constraints are often added to ensure that the resource is not overused at any given time.

1.5.2 Dynamic Programming

This technique hinges on breaking down a problem recursively into simpler subproblems. Consequently, the system comprises multiple steps to be sequentially solved. Each step corresponds to a subproblem that is solved optimally by considering the information acquired in previous steps. This necessitates formulating the criterion as a recurrence relation linking two consecutive levels. However, the transition from one stage to the next can be guided by the system's evolution and a decision-making process. This procedure circumvents the need for exhaustive exploration of the search space by selectively pruning decision sequences that cannot lead to an optimal solution [57].

1.5.3 Branch and Bound

Branch and Bound is a general algorithmic technique used for solving optimization problems, typically in the context of combinatorial optimization. It is employed to find the optimal solution to a problem by systematically exploring the search space and eliminating subproblems that are guaranteed to contain suboptimal solutions

The procedure starts with an initial solution or an empty solution and an initial lower bound on the optimal solution's cost and initializes an empty priority queue (usually a min-heap) to keep track of the subproblems. Then, it divides the problem into smaller subproblems by making decisions at each step. These decisions create branches in the search tree, which represent different possible solutions. These branches are explored systematically. For each subproblem, a lower bound on the optimal solution's cost is calculated. This lower bound is typically obtained by solving a relaxed or simplified version of the subproblem. If the lower bound is greater than the current best solution found so far, the subproblem can be pruned, since it is guaranteed to not contain an optimal solution. The procedure then adds the remaining subproblems to the priority queue, ordered by their lower bounds. The subproblem with the lowest lower bound is chosen for further exploration. This helps prioritize the search and potentially find the optimal solution more quickly. A subproblem is picked from the priority queue and explored further by branching and bounding. This process continues iteratively until the priority queue is empty or other termination conditions are met. A record keeps track of the best solution found so far and updates it whenever a better solution is discovered. [58].

1.5.4 Johnson's Algorithm

The Johnson algorithm [59] is an efficient method for solving the 2-machine and a special case of the 3-machine flow shop scheduling problems. It can be summarized as follows:

Step 1: Set Formation

1. Form a set J_1 containing all the jobs with $P_{1,j} < P_{2,j}$;
2. Form a set J_2 containing all the jobs with $P_{1,j} > P_{2,j}$;
2. The remaining jobs with $P_{1,j} = P_{2,j}$ may be put arbitrarly in either set.

Step 2: Sorting

1. Sort the jobs in set J_1 in increasing order of $P_{1,j}$, ties are brocken arbitrarly;
2. Sort the jobs in set J_2 in decreasing order of $P_{2,j}$, ties are brocken arbitrarly;

Step 3: Schedule the Jobs

Schedule the jobs in set J_1 then the jobs in set J_2

1.6 Heuristic Algorithms

Heuristic algorithms are useful tools to obtain satisfactory solutions to complex scheduling problems in very short time or to generate high-quality initial solutions to local search methods or single-solution-based metaheuristics such as simulated annealing and tabu search. These heuristic algorithms can be either constructive heuristics that iteratively build a complete solution or schedule starting from scratch by selecting and adding new jobs or operations at each step, or improvement heuristics that start from a complete solution and explores its neighborhood by introducing changes to it and selecting the best neighbor as the new updated solution [22]. In the next subsections we explore some well-known constructive and improvement heuristics that are often used to search for near-optimal solutions to scheduling problems.

1.6.1 Nawaz, Enscore, and Ham (NEH) Heuristic

The Nawaz, Enscore, and Ham [60] (NEH) heuristic is a widely used and effective algorithm for solving permutation flow shop scheduling problems. It was proposed as an efficient approach for minimizing makespan.

NEH starts with an initial list of jobs (L). This list is often generated using a simple heuristic, such as the Shortest Processing Time (SPT) rule (see section 1.6.2). NEH maintains a partial sequence that enventually becomes the complete final sequence. Initially, it is empty. The algorithm iteratively constructs the sequence by inserting a new job from the list L. in each iteration. NEH inserts the job at each possible position of the current partial sequence and computes the makespan of the resulting sequence. The position that leads to the lowest makespan is selected as the new patial sequence. The algorithm continues iterating, evaluating, and updating the partial sequences until a complete seuquence is obtained.

The following is a pseudocode of the NEH heuristic algorithm.

NEH Heuristic for Flow Shop Scheduling

Input: A set of n jobs J with processing times on m machines.
Output: A near-optimal job sequence S for the makespan objective.

Arrange the jobs in a list L using a heuristic method, e.g., SPT (Shortest Processing Time)
Initialize partial sequence S as empty
For each job j not in L :
  For each position p in S:
    Create a new sequence S'_p by inserting job j at position p in S.
    Compute the Makespan of S'_p and save it in a separate list C_p
  End For
  $S \leftarrow S'_k \mid k = \arg\min_{\forall p}\{C_p\}$
End For
Output S (the best sequence found).

The NEH heuristic is known for its effectiveness in producing high-quality solutions for permutation flow shop scheduling problems. While it may not guarantee an optimal solution, it often provides solutions that are very close to optimality in a relatively short amount of time, making it a valuable tool in production scheduling and manufacturing optimization.

1.6.2 Sequencing Rules

Sequencing rules are essential tools in scheduling that determine the order in which jobs are processed on machines and can be used as bases for heuristic solutions for NP-Hard problems when exact methods fail. Traditional sequencing rules have been widely used, while new and advanced rules and combinations of rules have emerged to address specific challenges and objectives in various industries [22, 24]. Some well-known rules include the following:

- **Shortest Processing Time (SPT):** The SPT rule prioritizes jobs with the shortest processing times first. It aims to minimize the average job completion time which make it effective in reducing overall job flow time and enhancing machine utilization and also suitable for minimizing makespan.
- **The Longest Processing Time First (LPT):** The LPT heuristic is a scheduling rule that arranges tasks by sorting them in descending order of their processing durations. It is widely recognized as the most effective dispatch rule for addressing parallel machine scheduling challenges. Consequently, LPT is employed to allocate shorter-duration tasks towards the end of the scheduling process, helping to refine the workload balance between machines.
- **Earliest Due Date (EDD):** The EDD rule schedules jobs based on their due dates, with the earliest due date jobs given priority. It is often used in problems with minimization of maximum lateness or total tardiness objectives and reducing the number of tardy jobs.
- **First-Come-First-Served (FCFS):** Jobs are processed in the order they arrive in the system, with the earliest arrival jobs processed first.
- **Critical Ratio (CR):** The CR rule calculates a critical ratio for each job, defined as the ratio of time remaining until the due date to remaining processing time. Jobs with the

lowest critical ratios are scheduled first. It balances the importance of due dates while considering processing times.

- **Least Slack Time (LST):** The LST rule prioritizes jobs with the least slack time, which is the difference between the due date and the time remaining until the due date. Effective in reducing job lateness.
- **The Longest Processing-Time-to-Resources-Consumption Ratios First (L-PT/RC)**: The L-PT/RC heuristic, introduced by Carrera [61], was designed for solving a single machine scheduling problem with non-renewable resources. The rule involves prioritizing tasks by sorting them in descending order of the ratio between their processing times and resource consumption.

1.6.3 Local Search

In the two preceding sections, we delved into heuristics of the constructive variety, methodologies that begin without a preexisting schedule and incrementally build one by incorporating jobs one at a time. In contrast, this section, as well as the subsequent one, delves into algorithms of the improvement category, which exhibit a fundamentally different approach. Improvement-type algorithms commence with a fully developed schedule, often arbitrarily chosen, and then endeavor to enhance it through adjustments to the existing schedule. A prominent subset of improvement-type algorithms is known as local search procedures. Notably, local search procedures do not guarantee optimal solutions; instead, they aim to identify schedules that are superior to the current one within the immediate vicinity [62]. Schedules are considered neighbors if one can be derived from the other through well-defined modifications. During each iteration, a local search procedure explores this neighborhood, evaluating the potential solutions therein. The procedure then accepts or rejects a candidate solution as the next schedule to transition to, guided by a predefined acceptance-rejection criterion. These local search procedures can be assessed according to four fundamental design criteria:

(i) The manner in which the schedule is represented.
(ii) The design of the neighborhood, specifying the set of feasible modifications.
(iii) The process of navigating this neighborhood during the search.
(iv) The acceptance-rejection criterion governing the transition to a new schedule.

The layout of the neighborhood plays a pivotal role in the local search procedure within a local context. When considering a single machine schedule, a neighborhood can be conceptually defined as comprising all schedules that can be generated by executing a solitary pairwise interchange between adjacent elements. This definition implies the presence of $n - 1$ schedules within the neighborhood relative to the original schedule. Expanding the scope of the neighborhood for a permutation flow shop schedule involves selecting a job from the schedule and repositioning it elsewhere within the schedule. It is evident that each job can be inserted into $n - 1$ alternative positions. However, it should be noted that the entire neighborhood contains fewer than $n(n - 1)$ neighbors since some of these neighbors are essentially duplicates. In cases involving more intricate machine environments, the neighborhood of a schedule typically exhibits greater complexity [22].

1.7 Overview on Optimization Problems

Optimization problems are mathematical or computational challenges where the goal is to find the best solution or solutions from a set of possible solutions, often subject to certain constraints. In simpler terms, optimization pertains to the identification of conditions that yield the optimal outcome for one or more functions under specific circumstances. Optimization problems can be classified into different categories based on various factors, including the number of variables and objectives, the linearity or non-linearity of objectives and constraints, and the definition of decision variables (binary, integer or continuous).

To formulate an optimization problem, a collection of decision variables; each characterized by its domain and associated constraints is employed. According to Talbi [62], this problem can be represented as a triplet $(\mathcal{S}, \Omega, f)$, where: $\mathcal{S}$ denotes the research domain defined over a set of decision variables. Ω represents the set of constraints that must be satisfied. $f: \mathcal{S} \rightarrow \mathbb{R}$ signifies the objective function. In the standard minimization form, the goal is to find an optimal solution $x^* \in \mathcal{S}$ such that $f(x^*) \leq f(x), \forall x \in \mathcal{S}$.

1.7.1 Multi-Objective Optimization Problems

In single-objective optimization, there is only one objective function to be minimized or maximized. The goal is to find a single optimal solution that optimizes this objective while satisfying any given constraints. Examples include linear programming and nonlinear optimization problems. On the other hand, Multi-objective optimization deals with problems where there are multiple conflicting objectives to be optimized simultaneously. The aim is to find a set of solutions that represent a trade-off between these objectives, known as the Pareto front (PF). Multi-objective optimization is commonly used in decision-making scenarios where there is no single best solution, but a range of feasible alternatives [63]. Before describing the various optimization algorithms, brief overview of multi-objective problems is necessary to the understanding of subsequent sections.

A Multi-objective problem (MOP) can be defined in the standard minimization form as follows:

$$\text{MOP} = \begin{cases} \text{minimize } F(x) = [f_1(x) \quad f_2(x) \quad \cdots \quad f_n(x)] \\ \\ \text{subject to} \qquad\qquad x \in \mathcal{S} \end{cases}$$

where n $(n \geq 2)$ is the number of objectives and the feasible point x represents the decision variables, while $\mathcal{S}$ represents the feasible set, which encompasses all such feasible points. $F(x)$ denotes the vector of objectives that are to be optimized in the problem.

An objective vector $F = (f_1 \ \ f_2 \ \cdots \ f_n)$ is considered to dominate another objective vector $G = (g_1 \ \ g_2 \ \cdots \ g_n)$ where this dominance relation is often denoted as $F \prec G$, if each component of G is greater than or equal to the corresponding component of F, and at least one component of F is strictly smaller than the corresponding component of G.

A Pareto optimal solution implies that there is no feasible solution that can improve one objective without simultaneously decreasing the quality of at least one other objective. A set of points, denoted as $\mathcal{P}^*$ ($\mathcal{P}^* \subseteq \mathcal{S}$), is classified as a Pareto optimal set if there are no other points in $\mathcal{S}$ that dominate any point in $\mathcal{P}^*$. The projection of $\mathcal{P}^*$ onto the objective space is referred to as the Pareto optimal front or surface, PF^*.

1.7.2 Multi-Objective Optimization Methods

Multi-objective optimization (MOO) involves finding a set of solutions that simultaneously optimize multiple conflicting objectives. These problems are common in various fields, including engineering, finance, and decision-making [64]. Solving MOPs is challenging due to the trade-offs between objectives. Next is a review of techniques used to solve MOPs [63]:

- **Pareto-Based Methods:** Pareto dominance forms the basis for many MOO algorithms. Multi-objective genetic algorithms, such as NSGA-II (Nondominated Sorting Genetic Algorithm II), use Pareto dominance to evolve a population of solutions toward the Pareto front. These algorithms maintain diversity among solutions and promote convergence to the front.
- **Weighted Sum Methods:** The weighted sum approach converts a MOP into multiple single-objective problems by linearly combining the objectives with different weights then solving the individual problems using any single-objective optimization techniques.
- **The ε-Constrained Method:** In this approach one objective is optimized while treating the others as constraints. The process is repeated for different constraint values to obtain a set of solutions.
- **Multi-Objective Evolutionary Algorithms (MOEAs):** These algorithms, such as NSGA-II, SPEA2 (Strength Pareto Evolutionary Algorithm 2), and MOEA/D (Multi-Objective Evolutionary Algorithm based on Decomposition), use population-based search to approximate the Pareto front. They incorporate mechanisms like selection, crossover, mutation, and elitism to explore and converge to Pareto-optimal solutions.
- **Decomposition-Based Methods:** There are two main categories in this class of methods: (1) Scalarization Methods where the multi-objective problem in converted into a series of single-objective problems. The solutions are combined into a final Pareto front. (2) Reference Vector-Based Methods where the problem is decomposed into subproblems, each aiming to optimize a single-objective function with respect to a reference vector. MOEA/D is an example of this approach.
- **Hybrid Methods:** Combining MOO with Single-Objective Optimization. Some problems can be divided into both multi-objective and single-objective subproblems. Hybrid

approaches solve the subproblems using appropriate techniques and combine the solutions to form a Pareto front.

1.8 Metaheuristic Algorithms for Scheduling Problems

Metaheuristics are a class of optimization algorithms used to solve complex combinatorial optimization problems, such as production scheduling. These algorithms are designed to find near-optimal solutions in a reasonable amount of time, making them invaluable in various real-world applications. In this extensive review, we will delve into metaheuristics, their categories, and their applications in production scheduling.

1.8.1 Overview and Categorization

Metaheuristics are high-level strategies or frameworks for solving optimization problems. They are particularly useful when exact algorithms, such as linear programming, become impractical due to the problem's complexity. Instead of guaranteeing an optimal solution, metaheuristics aim to quickly find a near-optimal solution. Metaheuristics can be categorized into several types, with the most common ones being [62]:

- **Local Search Algorithms:** These algorithms explore the solution space by iteratively improving an initial solution. Examples include Simulated Annealing and Tabu Search.
- **Population-Based Algorithms:** These algorithms maintain a population of candidate solutions and evolve them over generations. Notable examples are Genetic Algorithms and Particle Swarm Optimization.
- **Trajectory Methods:** Trajectory-based metaheuristics move through the solution space by iteratively adjusting a single solution. Ant Colony Optimization and the Great Deluge Algorithm are examples.
- **Hybrid Metaheuristics:** These combine two or more metaheuristics with other optimization techniques. Hybridization can improve the algorithm's performance. For example, a Genetic Algorithm can be combined with Simulated Annealing.

Advantages of using metaheuristics in production scheduling include their ability to handle complex, real-world scenarios, adapt to changing conditions, and find near-optimal solutions in reasonable timeframes. Another notable advantage is their applicability to problems without requiring in-depth knowledge of the problem's structural properties. These algorithms can be easily implemented and frequently yield satisfactory solutions. On the other hand, challenges include the need for parameter tuning, the possibility of getting stuck in local optima, and the absence of optimality guarantees. Researchers continually work on improving metaheuristic algorithms and developing hybrid approaches to address these challenges [65].

1.8.2 Key Features of Metaheuristics

Metaheuristics are research methodologies grounded in two fundamental principles: diversification and intensification, which correspond to exploration and exploitation, respectively. Diversification entails systematically exploring the search space to generate diverse problem solutions. This is achieved by periodically restarting the search process from a randomly generated or strategically selected solution within an uncharted region of feasible solutions, enhancing solution diversity. In contrast, intensification centers on leveraging the accumulated search experience to focus on areas that have shown promise. Every metaheuristic must strike a delicate balance between diversification and intensification. This equilibrium is critical for a twofold purpose. First, it facilitates the swift exploration and identification of regions within the search space, thereby preventing premature convergence towards local optima and enabling the attainment of high-quality solutions. Second, it prevents excessive exploitation of regions that have been explored previously but have not yielded promising results. Consequently, the optimal combination of intensification and diversification can guide the search process towards global optima, ensuring effective problem solving [66].

Metaheuristics represent a diverse family of algorithms, often inspired by natural processes, and have garnered significant attention from researchers. Despite their diversity, metaheuristics share common elements, particularly in the design phase. The design of iterative metaheuristics hinges on two fundamental concepts, which can be succinctly outlined as follows: (1) Solution representation which is the manner in which solutions are represented within the algorithm; and (2) Objective function definition which is the establishment of the criteria that guide the optimization process, determining the quality and suitability of solutions [62].

1.8.2.1 Solution Representation

The representation of the solution stands as a pivotal stage in the design and development of any metaheuristic, wielding significant influence over its effectiveness. The choice of coding during representation is of paramount importance and should align flawlessly with the specific problem under examination. This choice dictates how search operators are applied to the representation and how the solution's quality is assessed after decoding. Furthermore, any representation must embody the following key attributes: (1) Completeness, i.e., it should have the capacity to represent any feasible solution within the problem domain; (2) Connectivity, i.e., a navigable pathway should exist between any two solutions within the search space. This ensures that the optimal one, can be reached through the representation; and (3) Efficiency, i.e., the representation should be designed for ease of manipulation by search operators, and its temporal and spatial complexity should be minimized [62].

1.8.2.2 Objective Function

The objective function assumes a vital role in the design of a metaheuristic. It serves as the compass that directs the search process towards optimal solutions within the search space. By assigning a real value to each solution, the objective function quantifies its quality. Consequently, a poorly defined objective function can yield suboptimal solutions, irrespective of the metaheuristic employed.

1.8.3 Performance Evaluation of Metaheuristics

This section outlines the procedure for comprehensively evaluating the performance of metaheuristics. To rigorously assess the effectiveness of a metaheuristic, the following three key steps, as advocated by Talbi [62], must be meticulously followed: (1) Definition of an experimental plan where in this initial step, an experimental plan is meticulously crafted to guide the performance assessment process; (2) Selection of performance measures and indicators where relevant performance parameters, metrics, and indicators are carefully chosen. These measures will serve to gauge the efficacy of the metaheuristic in a quantitative and structured manner; and (3) Application of statistical analyses where the obtained experimental results are subjected to thorough statistical analysis. This statistical assessment ensures that the performance of the developed algorithms is statistically significant and not just a result of pure chance.

Once the experimental framework is established, and the results are collected for various performance indicators, statistical analysis can be employed to appraise the efficacy of the metaheuristics. The performance analysis must be done with state-of-the-art optimization algorithms dedicated to the problem. The problem data can be obtained ideally from real-world instances or can be wisely constructed to ensure that all possible scenarios are covered. The experimental framework must also ensure the reproducibility of the computational experiments. Many performance indicators for population based metaheuristics were proposed in the last few decades, yet there is no universally agreed-upon standard for performance evaluation [67]. The primary challenge in assessing MOO lies in the fact that the outcome of the optimization process isn't a single solution, but rather a collection of solutions that approximate the PF. These measures can be categorized into three main classes: Convergence-Based, Diversity-Based, or hybrid indicators. Convergence metrics assess the efficiency of solutions by measuring their proximity to the optimal PF. Diversity indicators gauge the evenness of the distribution of the obtained solutions by considering factors such as dispersion and extension, while hybrid indicators combine convergence and diversity aspects into one single measure. Other measures that are often used to compare the performance of several metaheuristics include the number of obtained solutions and the computational time.

1.8.3.1 Coverage Measure (C)

The C metric, introduced by Zitzler & Thiele [68], is a widely used performance measure in multi-objective research. For two nondominated fronts A and B, the $C(A,B)$ value represents the fraction of solutions in B that are dominated by at least one solution from A. as presented in the equation below:

$$C(A,B) = \frac{|\{b \in B \;/\; \exists a \in A: a \prec b\}|}{|B|} \tag{1.1}$$

The closer the value of $C(A,B)$ is to 1, the more favorable front A is compared to B. However, values close to 0 do not necessarily imply that front B is better than A. Due to this asymmetry, it is essential to compute both $C(A,B)$ and $C(B,A)$. If $C(A,B) > C(B,A)$, then front A is considered better than B.

1.8.3.2 Inverted Generational Distance (IGD)

In this measure introduced in [69, 70], the IGD value for a given front A is determined by defining a reference set S^*, which can be an optimal or a near-optimal set. Subsequently, the average distance between a solution in S^* and its closest solution in A is computed using the following formula:

$$\mathrm{IGD}(A) = \frac{1}{|S^*|} \sum_{y \in S^*} \min_{x \in A} \{d_{xy}\} \tag{1.2}$$

The value of IGD is determined by calculating the Euclidean distance, denoted as d_{xy}, between a solution x in A and a solution y in the reference set S^*, in a normalized objective space. In this normalized space, the maximum and minimum values of each objective, with respect to the reference set, are equal to 100 and 0, respectively. This measure assesses both the distribution of solutions in A and the proximity of A to S^*. A smaller value of IGD indicates a better quality for the front A, meaning that it is closer to the reference set S^* and exhibits a more favorable distribution of solutions.

1.8.3.3 Hypervolume Indicator (HV)

The hypervolume indicator HV, also proposed by Zitzler & Thiele [68], is a hybrid measure that combines aspects of convergence and diversity to evaluate the quality of a given nondominated front independently of other fronts. The HV associated with a front A is determined by the total volume of the objective space covered by A. To calculate this volume, a reference point that represents an upper bound over all the objectives must be specified which is often defined as the Nadir point (Fig. 1.2).

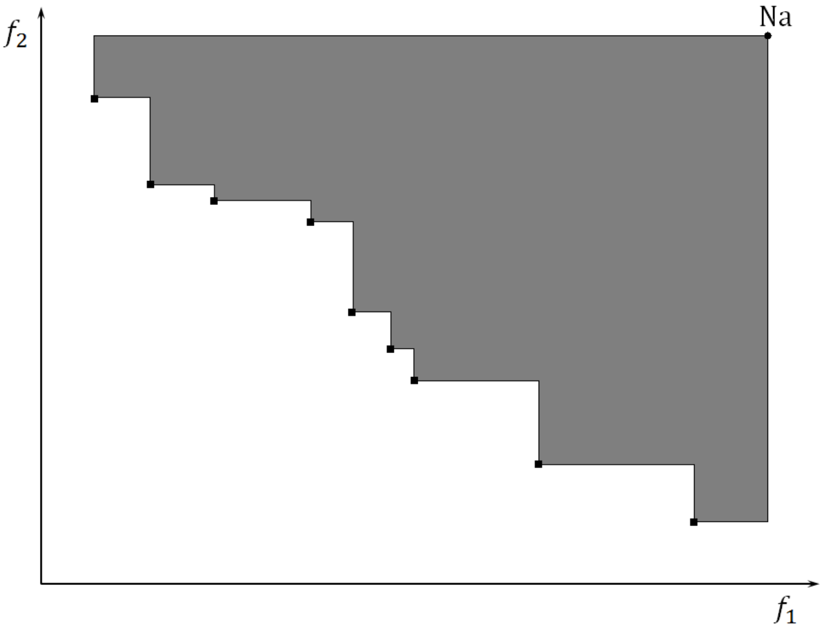

Fig. 1.2 Hypervolume indicator of a nondominated set of nine solutions in 2D-space.

1.8.3.4 Spacing (SP)

The Spacing (SP) metric is intended to assess the evenness of the distribution of solutions within an approximation set [67]. A Spacing value of 0 signifies that all members of the approximation set are equally spaced. The spacing metric computes the standard deviation of the Euclidean distances d_i between solution i and its closest neighbor in a nondominated front A and is computed as follows:

$$\mathrm{SP}(A) = \sqrt{\frac{1}{|A|-1}\sum_{i=1}^{|A|}\left(\bar{d}-d_i\right)^2} \tag{1.3}$$

where $\bar{d}$ is the average distance between solutions and their closest neighbor.

1.8.3.5 Overall Spread (OS)

This metric measures the extent of the Pareto front covered by the approximation [71]. A larger value is indicative of better performance. It is determined by the following formula:

$$\mathrm{OS}(A) = \prod_{i=1}^{m} \frac{\left|\max\limits_{x \in A} f_i(x) - \min\limits_{x \in A} f_i(x)\right|}{|f_i(\mathrm{Na}) - f_i(\mathrm{Id})|} \tag{1.4}$$

where Na is the Nadir point or the point with maximum value for each objective value and Id is the ideal point or the point with minimum objective value for each objective value.

1.8.4 Genetic Algorithms (GA)

Genetic Algorithms draw inspiration from the evolutionary dynamics observed in natural species, harnessing the principles of variation and selection inherent in biological evolution [72]. Organism evolution in nature involves a sequence of improvements, enabling a species to better adapt to its environment or the evolving conditions it meets. This adaptation primarily relies on two fundamental mechanisms: natural selection and reproduction. Genetic algorithms, belonging to the broader family of evolutionary algorithms, are iterative and stochastic optimization methods designed to tackle challenging NP-hard optimization problems.

In the realm of scheduling, Genetic algorithms treat sequences or schedules as individual entities within a population. Each individual is characterized by its fitness, determined by the associated objective function value. This process unfolds iteratively, with each iteration referred to as a generation. A generation comprises survivors from the previous one, alongside newly generated schedules i.e., the offspring or children of the previous generation. Typically, the population size remains consistent from one generation to the next. The offspring is created through the reproduction and mutation of individuals from the prior generation, often referred to as parents or chromosomes. In a problem where the solution contains multiple parts pertaining to different decisions, a chromosome may encompass sub-chromosomes, each carrying information about the specific part of the solution. Moreover, each part can have its own encoding scheme (permutation, integer and continuous values …etc.). A mutation in a parent chromosome may correspond to an adjacent pairwise interchange within the relevant sequence. Within each generation, the fittest individuals reproduce, while the less fit individuals are removed. The following is a pseudocode for a basic genetic algorithm:

Genetic Algorithm for a typical scheduling problem

---

1. Initialize a population P of size N with random schedules.
2. Evaluate the fitness of each schedule in the population by calculating the objective function.

Repeat for G generations or until convergence criteria are met:
1. Select parents for the next generation:
   Utilize a selection method (e.g., roulette wheel, tournament, or rank-based) based on the fitness values of the job permutations.

4. Apply crossover (recombination) and mutation operators to create a new population:
   Crossover operator (e.g., order-based crossover):
      Select two parents from the current population.
      Choose a crossover point to split the parent permutations.
      Create two children by recombining the segments of the parents.
   Mutation operator (e.g., swap, insertion, or specific mutations):
      Apply a mutation to selected individuals with a low probability Pm.

5. Evaluate the fitness of each schedule in the offspring population by computing the objective function.

6. Update the old population P by introducing the offspring population.
End of Iteration Loop.

7. Return the best schedule found in the final population, which corresponds to the minimum objective value.

1.8.4.1 Creating the Initial Population

A crucial phase in the operation of the genetic algorithm process is the generation of the initial population. This step significantly impacts the final solution's quality and the number of generations required to achieve it [62]. The degree of diversity within the population plays a vital role in preventing the algorithm from becoming trapped in local optima. Typically, the initial population is established through random selection, heuristic approaches, or problem-specific techniques. Furthermore, it is essential to generate the population's size optimally to ensure effective exploration of the search space and optimize the algorithm's overall performance.

1.8.4.2 Selection

This phase revolves around determining which individuals from the current population will act as parents and be replicated in the new population. Typically, individuals with superior fitness are more likely to become parents and are suitable for crossover, potentially driving the population towards improved solutions. Nevertheless, individuals with lower fitness still maintain a slight chance of being selected. This can contribute to an effective enhancement process and keep the search open to low quality solutions that may help escape local optima. Many selection methods can be applied. Some of the prominent ones used are the following:

- **Random:** As the name suggests, this method assigns an equal probability to all individuals and doesn't consider the fitness function, making it more suitable during exploration.
- **Roulette Wheel:** Here, each individual is assigned a selection probability proportional to their performance, giving those best suited to the problem a higher chance of being chosen. However, this strategy can encounter issues when there is a wide variance in chromosome fitness, potentially resulting in some chromosomes having very low selection probabilities, making the search biased towards areas with locally high objective values.
- **Rank:** Rank selection involves ordering the individuals based on their fitness in descending order. This approach favors individuals with higher ranks, removing the element of chance from the selection process. Unlike the Roulette wheel method, rank selection considers the performance ranking of a group of individuals, ensuring that all chromosomes have a fair chance of being selected. This method can be extended to multi-objective problems where individuals may have the same quality if they belong to the same nondominated front and have a higher rank than individuals whom they dominate.
- **Tournament:** Tournament selection begins by randomly selecting a group of T individuals and then conducting a tournament among them to determine the one with the best performance. This process is repeated p times to generate the p individuals in the new population.

1.8.4.3 Crossover

The crossover operator is a probabilistic operation that is used to create new chromosomes. It involves selecting two individuals from the pool of potential parents using one of the selection methods. Subsequently, portions of their genetic sequences are exchanged based on a predetermined crossover probability, usually referred to as Pc. This process yields new offspring, who inherit specific characteristics from both parents, thereby enriching the population. In the literature, several crossover techniques have been identified. We'll discuss the two most commonly used methods:

- **One-Point Crossover:** This approach the genetic sequences of the two parents are divided at a single, randomly chosen point. The segments to the right of this chosen point are then exchanged between the two parents, resulting in two children.
- **Two-Point Crossover:** In two-point crossover, two random cut points are selected to divide each parent into three segments. The outer segments of the first parent are assigned to the first child, while the remaining portion of the first child is filled with genes from the second parent. This process is repeated for the second parent, generating the second child.

1.8.4.4 Mutation

Following the genetic crossover, the generated offspring undergo a mutation process that affects a single individual with an extremely low probability known as the mutation probability, often referred to as Pm. This process gives rise to a novel individual that did not previously exist. The mutation operator serves the crucial role of preventing the genetic algorithm from converging solely towards local optima. It achieves this by introducing new information and diversity into the population over successive generations and by expanding the search towards a broader frame in the search space. The following is a detailed overview of different types of mutation operators commonly used for scheduling problems:

- **Swap Mutation:** Also known as pairwise interchange, it is one of the simplest mutation operators. It selects two random elements (e.g., job positions or machine assignments) within an individual's schedule and swaps them.
- **Insertion Mutation:** Insertion mutation involves taking a job from one position in a schedule and inserting it at another randomly chosen position.
- **Scramble Mutation:** This method shuffles a random subset of jobs within a schedule.
- **Inversion Mutation:** This method selects a segment of a schedule and reverses the order of the jobs within that segment.
- **Non-Uniform Mutation:** Non-uniform mutation involves randomly changing the values of genes (job assignments, Speed settings ...etc.) within an individual's schedule. The magnitude of these changes is often controlled by a probability distribution, which can lead to both small and large perturbations.

The choice of mutation operator depends on the specific scheduling problem and its characteristics. Researchers and practitioners may experiment with different mutation operators or even combine multiple types to strike the right balance between exploration and exploitation of the solution space. The effectiveness of a mutation operator can vary based on the problem's complexity, and it often requires fine-tuning through empirical testing.

1.8.5 Particle Swarm Optimization (PSO)

PSO draws inspiration from swarm intelligence and the social behavior observed in birds flocking or fish schooling [73]. It has shown remarkable success in solving single-objective, continuous optimization problems, thanks to several advantages it offers, including ease of implementation, preservation of population diversity, and computational efficiency.

The fundamental model of PSO comprises a swarm of particles traversing the decision space. Each particle is defined by its position in the space, representing a candidate solution to the problem, and a velocity vector (gradient), indicating its direction of movement. To achieve global optimization, cooperation among the particles in the swarm is leveraged. The behavior of each particle, denoted as i, is influenced by its own record of the best position visited, known as the personal best ($pbest_i$), and the best solution visited by the entire swarm, referred to as the global best ($gbest_i$). In each iteration $t+1$, the velocity v_i^{t+1} of a particle i at position x_i^t is updated using the following crossover rule:

$$v_i^{t+1} = Wv_i^t + C_1R_1(pbest_i^t - x_i^t) + C_2R_2(gbest_i^t - x_i^t) \tag{1.5}$$

where W is the particle's inertia weight, and determines how much the previous velocity influences the current one. A higher value of W promotes exploration of the decision space, while a lower value encourages local exploitation. C_1 represents the particle's cognitive attraction factor towards its own best performance, while C_2 represents the social attraction factor towards the best performance of its neighbors. Two independent random samples, R_1 and R_2, are typically generated from a uniform distribution in the range of [0, 1]. After updating the velocity, the particle's position is then updated as follows:

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{1.6}$$

The following is a pseudocode for a basic Particle Swarm Optimization algorithm:

Particle Swarm Optimization for Scheduling

Initialize a population of particles (solutions) with random positions and velocities. Each particle represents a potential schedule.

Initialize best-known position (*pbest*) for each particle to be its current position
Evaluate the fitness of each particle based on the scheduling problem's objective (e.g., makespan, total completion time).

Repeat for a specified number of iterations or until convergence is achieved:
    Calculate the swarm's best-known position (*gbest*), which is the best solution among all particles.

    For each particle in the swarm:
        Update the particle's velocity using equation (1.5)
        Update the particle's position using equation (1.6)
        Evaluate the fitness of the particle's new position based on the scheduling objective
        If current postion is better than particle's *pbest*
            Update the particle's best-known position (*pbest*) to be its current position.
        End if
    End For
End of Iteration Loop.

Return the best schedule found, corresponding to the *gbest*.

1.8.6 Ant Colony Optimization (ACO)

Ant Colony Optimization is a widely recognized metaheuristic initially developed by Dorigo et al. [74]. The ACO concept draws inspiration from the foraging behavior of ant colonies. In the quest for food, ants initially wander randomly within their colony. As they explore their environment, ants deposit pheromone substances to mark their paths, serving as a signal for other ants to follow. Pheromone is a volatile chemical substance that gradually evaporates over time. Once ants locate a food source, subsequent groups of ants follow the trails left by their predecessors and also deposit pheromones on their respective paths. Eventually, the probability of discovering the shortest path increases, leading most ants to converge on the shortest route to reach the food source. In the realm of ACO algorithms, an artificial ant colony iteratively constructs solutions to a given problem using artificial pheromone trails.

These trails are connected to previously discovered solutions and heuristic information. Communication among ants occurs indirectly, solely through adjustments to the pheromone levels they leave on their trails during the algorithm's execution. Since the solutions created by the ants may not be locally optimal, many ACO algorithms permit the ants to enhance their solutions through local search procedures. As a result, ACO algorithms integrate various techniques, including local search methods and dispatching rules, within a unified framework. The fundamental ACO framework encompasses four key steps:

Step 1. Set parameters and initialize pheromone trails.
Step 2. Generate a set of solutions using a blend of pheromone trails and dispatching rules.
Step 3. Apply a local search procedure to improve each solution in the set generated in Step 2.
Step 4. (Update Settings)

If the best solution among those generated in Step 3 surpasses the best solution obtained thus far,

replace the current best solution with this superior one and record its corresponding objective function value.

End If

Update pheromone trail values and return to Step 2 to initiate the next iteration.

Describing the application of a basic ACO algorithm to a standard flow shop scheduling problem can be outlined as follows: In each iteration (denoted as t), when an ant is constructing a job sequence, it begins with an empty sequence and progressively appends a job from the remaining unscheduled jobs to the sequence in formation. There are two potential actions for the ant, each with specific probabilities: With a probability q, the ant selects the unscheduled job j that maximizes the product of two factors: the pheromone trail $\tau_{jk}(t)$ associated with assigning job j to position k in the current sequence, and the heuristic desirability η_{jk}^{β} of placing job j in position k. The term t indicates the dependence of the pheromone trail on the current iteration count. The parameter η_{jk} signifies the heuristic desirability for assigning job j to position k, and β represents a parameter that influences the weight of this heuristic desirability. With a probability of $1-q$, the ant chooses job j randomly, utilizing a probability distribution:

$$P_{jk} = \frac{\tau_{jk}(t)\eta_{jk}^{\beta}}{\sum_{i \in J} \tau_{ik}(t)\eta_{ik}^{\beta}}$$

In this context, J denotes the collection of unscheduled jobs. The heuristic desirability, represented as η_{jk}, can be established using a particular scheduling rule of choice.

There are two approaches for updating the pheromone trails: immediate updating and delayed updating. Immediate updating, a form of local updating, is performed each time an ant adds a new job to its partial sequence. Delayed updating, in contrast, represents a global updating method performed at the conclusion of each iteration. At the end of iteration t, the delayed updating procedure begins by evaporating the pheromone for each combination of j and k. Subsequently, it infuses additional pheromone, albeit selectively, into specific combinations of j and k. Specifically, this reinforcement is only applied to combinations that correspond to the best solution obtained in iteration t. In the event that the best solution produced during iteration t involves placing job $j$in position k, then

$$\tau_{jk}(t+1) = (1-\rho)\tau_{jk}(t) + \rho\Delta\tau_{jk}(t)$$

where ρ, $0 < \rho \leq 1$, is a parameter that represents the pheromone evaporation rate, and the term $\rho\Delta\tau_{jk}(t)$ represents the amount of deposited pheromone at iteration t which is computed based on the quality of the obtained best solution.

1.9 Conclusion

This chapter introduces scheduling problems within production systems. It covers fundamental definitions and terminologies related to scheduling problems under various environments and specifications. It explores various classifications of scheduling problems and provides an overview of scheduling problem structures. Fundamental concepts related to maintenance and transportation within manufacturing facilities are also broadly reviewed in order to give the reader a better understanding of the upcoming modeling methodologies and detailed computations presented in chapters 3 and 4. The chapter then gives a brief review of computational complexity of optimization problems and discussed the resolution methodologies appropriate to each class ranging from exact methods such as linear programming and Johnson's algorithm specifically used for the solvable 2-machine flow shop problem with makespan minimization, to heuristic and approximation methods such as local search procedures, sequencing rules and NEH heuristic specifically used for the NP-hard flow shop scheduling problem. After that, the chapter delves deeply into the definition of single and multi-objective optimization problems, the classification of the various optimization strategies and the implementation and analysis of metaheuristic algorithms which constitutes a substantial part of the content presented in the subsequent chapters.

Chapter 2

Literature Review

This chapter discusses the relevant main concepts and previous work on the production scheduling problem under shared renewable and nonrenewable resource constraints. Section 2.1 focuses on related work concerning the flow shop and maintenance scheduling problem with a shared nonrenewable resource constraint. Section 2.2 presents a literature review on the energy-efficient flow shop scheduling problem with blocking and collision-free transportation constraints. Finally, Section 2.3 presents concluding remarks on the current state of the literature.

2.1 Related Work on FMSP-SNR

Since the 1950s, there has been a notable surge of interest in flow shops and scheduling theory. Johnson's algorithm, introduced in 1954, sparked extensive research in this field. While Johnson's rule is efficient, it is limited to 2-machine flow shops and a special three-machine case. Problems with more than 2 machines remain challenging because they necessitate the enumeration of all possible job sequences, either explicitly or implicitly. Branch and Bound algorithms with accurate bounds on the objective function can be employed for small problems but become impractical for larger ones. As a result, many researchers have shifted their focus toward approximation methods and dispatching rules that can rapidly generate "good" solutions and are easily implementable on the shop floor. Noteworthy contributions in this direction include the works of Palmer [76], Campbell et al. [77], and Nawaz et al. [60].

In classical literature on production scheduling, there is a common assumption that machines are consistently available throughout the planning period. However, practical situations often involve machines experiencing both planned and unplanned periods of unavailability due to maintenance actions or the lack of required consumable resources. In terms of maintenance, two significant tasks are typically practiced: (1) Preventive Maintenance (PM) which involves quick interventions such as inspections or periodic replacement of aging components to ensure machine reliability and (2) Corrective Maintenance (CM) which is carried out after an unexpected failure and focuses on restoring the machine to a functional state. It is generally assumed that CM requires more time and incurs higher costs compared to PM [78].

Over the past few decades, numerous research articles have been published regarding maintenance requirements while another branch of research focused on the scheduling of scares resources. These articles can be categorized into three main classes which will be discussed in the next sections: (1) Production scheduling with machine availability constraints; (2) Joint production scheduling and maintenance planning and (3) Production scheduling with resource constraints. The analysis identifies research gaps and categorizes the pertinent literature in Table 2.1 based on several criteria, including the scheduling environment, additional constraints imposed on the problem, optimization objectives, and the modeling and solution methods employed.

2.1.1 Production Scheduling with Machine Availability Constraints

Models adopting this approach consider maintenance intervals as constraints in production scheduling problems. In these models, the fixed-time period between consecutive maintenance interventions serves as a constraint on production operations. The following are notable contributions in this area: Lee [79] proved that the problem of minimizing makespan in a 2-machine problem with unavailability constraints is NP-Hard. The author provided dynamic programming algorithms and heuristics for both preemptive and non-preemptive cases with one availability period on each machine separately. Schmidt [80] conducted a comprehensive review of problems with availability constraints in various machine environments. Aggoune [81] studied fixed availability constraints and time window constraints, highlighting that permutation flow shops may not yield optimal solutions. Kubzin et al. [82] presented approximation schemes for the 2-machine flow shop scheduling problem considering both preemptive and semi-preemptive (where a portion of the process time needs to be reprocessed) cases. Hadda et al. [83] proposed an improved heuristic with a lower-bound analysis based on Lee's [79] work. Choi et al. [84] investigated the complexity properties of special cases (ordered and proportionate flow shops) under fixed availability constraints. Cui et al. [85] studied non-permutation flow shops with predefined maintenance plans and flexible maintenance plans. Both problems were formulated as mixed integer programs.

These contributions have shed light on the challenges and potential approaches for addressing production scheduling problems considering unavailability constraints and maintenance planning.

The interdependence between production and maintenance, where both activities act on the same resources, is evident. However, it is observed that many practitioners often overlook this relationship and tend to plan production and maintenance activities separately. This approach, particularly in production scheduling with machine unavailability constraints, often prioritizes maintenance over production. As a result, the production scheduler is compelled to adjust their plan to accommodate the fixed predetermined maintenance plan, leading to potential disruptions and inefficiencies [86].

2.1.2 Joint Production Scheduling and Maintenance Planning

In the joint production scheduling and maintenance planning approach it is assumed that the number of maintenance tasks and their frequency are unknown and should be jointly determined as decision variables with production job sequences. Joint production scheduling and maintenance planning have been approached by various researchers. Kaabi et al. [87] proposed four heuristics for scheduling maintenance and production on a single machine, aiming to minimize total tardiness in production and the earliness and tardiness of maintenance tasks. They considered release dates, due dates, and idle times as factors influencing machine age. In a subsequent study [88], the authors extended their approach to a flow shop with different flexible maintenance intervals for each machine. Benbouzid-Sitayeb et al. [89] developed heuristics and meta-heuristics to optimize maintenance costs by assigning different slopes for earliness and tardiness costs based on a non-linear maintenance cost approximation model. Cassady & Kutanoglu [90] integrated corrective repairs into a joint production and maintenance model using reliability and the Weibull distribution, aiming to minimize total tardiness. While their study highlighted the benefits of integrating production and maintenance, they did not consider the costs associated with the plan. Later on in [91], they extended the model to allow for multiple interruptions of a job, aiming to minimize the sum of completion times. Sortrakul et al. [92] addressed computational limitations by employing a genetic algorithm for larger problems. Wang & Liu [93] applied a similar approach to a 2-machine flow shop with minimization of makespan as the objective. Ruiz et al. [94] presented a review of different PM policies and used a conservative approach for planning PM actions, ensuring they occur before the optimal PM time period. On the other hand, Berrichi et al. [86] and Berrichi & Yalaoui [95] employed a rational strategy in parallel machine problems, using bi-objective algorithms to minimize makespan and system unavailability through reliability models. Wang & Liu [96] studied a two-stage hybrid flow shop with preventive maintenance, sequence-dependent setup times, and the same objectives. Khatami & Zegordi [97] introduced a single-server maintenance setting to the previous model, while Naderi et al. [98, 99] conducted similar studies to Ruiz et al. [94], investigating different PM policies and criteria in flexible flow shops and flexible flow lines. Allaoui et al. [100] studied the problem of minimizing $Cmax$ by simultaneously scheduling a set of jobs and one PM action after a defined time period on a 2-machine flexible flow shop. Pan et al. [101] considered expected machine breakdowns and variable maintenance time influenced by machine degradation to minimize the weighted sum of tardiness in a single machine problem. Hadidi & Al-Turki [102] studied a single machine integrated model that considered the costs of corrective maintenance actions, aiming to minimize total maintenance costs and total holding costs. Later on the authors presented a review of integrated production maintenance and quality models [103].

In recent publications, the integration of maintenance and production has been further explored. Bajestani and Beck [104] proposed a new approach in a flow shop problem where the processing times of jobs are influenced by discrete machine deterioration over time. Building on the idea presented by Berrichi et al. [86], Mokhtari et al. [105] extended the concept to a parallel machine problem with multiple maintenance scenarios, allowing for partial repairs of machines. Salmasnia & Mirabadi-Dastjerd [106] considered different levels of preventive maintenance, each level restoring the machine to a certain extent. The highest level, referred to as "as good as new" (AGAN), aimed to minimize the sum of completion time and maintenance cost in a single machine problem. These studies have contributed to the advancement of integrated maintenance and production approaches in various scheduling problems.

2.1.3 Production Scheduling with Resource Constraints

Limited research has been dedicated to scheduling with resource constraints compared to the extensive research on joint production and maintenance planning. Resource availability and types vary in practice, with resources being either discrete or continuous. Their availability patterns differ based on the system's characteristics. Resources may be continuously available but in limited quantities, requiring allocation to different operations or machines. Alternatively, they may have irregular availability at specific time intervals. Daniels & Mazzola [107] addressed the flexible resource flow shop problem using a decomposition-based taboo search. This approach considered different processing modes for each resource allocation. Daniels & Mazzola [108] further developed a branch and bound algorithm and a heuristic solution for the flexible resource flow shop. In this problem, resources are workers assigned to operations, considering their availability at any given time. Gupta et al. [109] studied heuristics for hybrid flow shops with linearly controllable process times and assignable due dates. Mokhtari et al. [110] focused on the flow shop problem with resource-dependent processing times. They proposed a decomposition-based solution approach that allocates resources to operations first, followed by a sequencing algorithm. Behnamian & Fatemi Ghomi [111] employed a modified version of the genetic algorithm to solve a bi-objective hybrid flow shop problem. This problem involved setup times and resource allocations with machines of different speeds and resources capable of linearly reducing processing times. Notably, maintenance was not considered in their model, and the objectives were to minimize makespan and the total amount of consumed resources. Edis & Oguz [112] investigated parallel machine problems under resource constraints with flexible processing times. They also provided a review of various mathematical programs proposed for different scenarios. Wong et al. [113] studied the parallel machine problem and preventive maintenance scheduling, taking into account maximum machine ages. Their model considered PM actions not only for machines but also for resources (molds) requiring maintenance. Multiple maintenance tasks with varying durations were considered. Nguyen et al. [114] examined parallel machines with a single consumable resource. The processing rate was a linear function of the resource consumption, and two sets of constraints were considered: varying total available resources over time and a constraint on the changing resource consumption rate between two intervals.

Research on joint production and maintenance scheduling incorporating resource constraints has been relatively limited, with only a few papers published. One notable contribution is by Belkaid et al. [115], who proposed an integrated model for a parallel machine environment, considering a staircase cumulative resource function.

Table 2.1 Literature overview on the FMSP-SNR

Reference	Scheduling environment	Additional constraints and characteristics					Optimization objectives			Solving methods			
		Machine constraints	Resource type	Resource supply maint.	Resource supply control	Resource holding cost	$Cmax$	Total Production cost.	Other	Meta-heuristic	Heuristic	Math. Model	Other
Lee [79]	Flow shop	Availability					✓				✓		✓
Aggoune [81]	Flow shop	Avail. / Maint.					✓			✓			
Kubzin et al. [82]	Flow shop	Availability					✓						✓
Hadda et al. [83]	Flow shop	Availability					✓				✓		
Choi et al. [84]	Flow shop	Availability					✓		✓		✓		✓
Cui et al. [85]	Flow shop	Avail. / Maint.					✓			✓		✓	
Kaabi et al. [87]	Single Mach.	Maintenance							✓		✓		
Kaabi et al. [88]	Flow shop	Maintenance							✓	✓	✓		
Benbouzid-Sitayeb et al. [89]	Flow shop	Maintenance							✓	✓	✓		
Cassady et al. [90]	Single Mach.	Maintenance							✓				✓
Cassady et al. [91]	Single Mach.	Maintenance							✓		✓		✓
Sortrakul et al. [92]	Single Mach.	Maintenance							✓	✓			✓
Wang et al. [93]	Flow shop	Maintenance					✓			✓			✓
Ruiz et al. [94]	Flow shop	Maintenance					✓			✓	✓		
Berrichi et al. [86]	Parallel	Maintenance					✓		✓	✓			
Berrichi et al. [95]	Parallel	Maintenance							✓	✓	✓		
Wang et al. [96]	Hyb.Flow shop	Maintenance					✓		✓	✓			
Nadiri et al. [98]	Flow line	Maintenance					✓			✓			
Pan et al. [101]	Single Mach.	Maintenance							✓			✓	
Hadidi et al. [102]	Single Mach.	Maintenance						✓				✓	
Bajestani et al. [104]	Flow shop	Maintenance							✓		✓	✓	✓
Daniels et al. [107]	Flow shop		Non-Renew.				✓			✓		✓	
Daniels et al. [108]	Flow shop		Renewable				✓				✓		✓
Mokhtari et al. [110]	Flow shop		Non-Renew.				✓	✓		✓			
Behnamian et al. [111]	Hyb. Flow shop		Non-Renew.				✓	✓		✓		✓	
Edis et al. [112]	Parallel		Non-Renew.				✓					✓	✓
Wong et al. [113]	Parallel	Maintenance	Renewable	✓			✓			✓		✓	
Nguyen et al. [114]	Parallel		Renewable						✓		✓	✓	
Belkaid et al. [115]	Parallel		Non-Renew				✓			✓		✓	
This work	Flow shop	Maintenance	Non-Renew	✓	✓	✓	✓	✓		✓	✓	✓	

2.2 Related Work on EFSP-BCFT

In this section, an overview of the literature relevant to the EFSP-BCFT is provided. The primary areas covered include: (1) AGV dispatching and routing, and (2) Energy-efficient scheduling and transportation. The analysis identifies research gaps and categorizes the pertinent literature in Table 2.2 based on several criteria, including the scheduling environment, additional constraints imposed on the problem, optimization objectives, and the modeling and solution methods employed.

2.2.1 AGV Dispatching and Routing

Initially, the AGV scheduling and routing problem may appear akin to the well-known vehicle routing problem (VRP). However, a fundamental distinction arises when we consider the assumption of infinite path load capacity in the classic VRP. Issues such as collisions and deadlocks are typically not considered in the VRP. Regrettably, in enclosed environments like production facilities, transportation pathways often have limited capacity, allowing for just one vehicle in most real-world scenarios. Concerns like AGV collisions and undesirable congestion primarily manifest in multi-AGV systems. Addressing congestion and deadlock concerns can be achieved through the implementation of a straightforward AGV-guide path layout. Unidirectional loop network layouts (ULNL) are widely employed in automated manufacturing settings. These networks establish connections between all workstations through a path that traverses each workstation exactly once. Notably, such layouts demand lower initial investment costs compared to their counterparts [116]. Furthermore, they prove highly effective in mass production systems like flow shops, where material flow strictly follows a unidirectional path from the input device to the output device, crossing every workstation just once. The simplest form of unidirectional guide path design is the single loop configuration. Prior simulation studies have indicated that the throughput achieved with a single loop flow path can be nearly identical to that of a conventional multi-loop flow path, particularly under light to average shop workloads. Moreover, the performance of a single loop only exhibits a marginal decline, typically less than 6%, when subjected to heavy workloads [117].

Once the guide path layout is established, the next critical aspect is to resolve the challenge of collision-free routing for AGVs. Several methods can be employed for this purpose:

1. **Static methods:** In this approach, an allocated path segment remains occupied until a vehicle completes its entire assigned journey.
2. **Time-window-based methods:** These methods allow a path segment to be used by different vehicles, but they ensure that these usages occur during non-overlapping time-windows.
3. **Dynamic methods:** In this category, the utilization of any path segment is dynamically determined during routing, typically in an online fashion [118].

The choice of the appropriate collision-free routing method depends on the specific guide path layout adopted. Additionally, the number of different speed settings available to AGVs plays a crucial role. Consider a scenario involving a unidirectional single-loop guide path with multiple AGVs. In this case, if AGVs can operate at varying speeds, the potential for collisions arises, particularly when a faster AGV approaches a slower one in the loop. Addressing such speed-related collisions becomes necessary.

In the literature, both online and offline dispatching and routing methodologies have been extensively explored. These studies encompass a wide spectrum of cases, ranging from straightforward unidirectional guide paths to more intricate network paths, as well as scenarios involving free-ranging and waterborne AGVs. For example, Ulusoy et al. [119] tackled the challenge of offline simultaneous scheduling of machines and multiple identical AGVs within a Flexible Manufacturing System (FMS) with the objective of minimizing makespan. They employed a genetic algorithm to solve the problem and observed that the GA outperformed the time-window-based approach in 59% of the test problems, with the reverse being true for 6%. Sabuncuoglu et al. [120] introduced a beam search-based algorithm, demonstrating that an offline algorithm can effectively schedule both machines and vehicles in an AGV-served FMS, often yielding superior results compared to real-time AGV dispatching rules. Behnamian et al. [121] utilized a GA and proposed a MILP model along with a heuristic algorithm based on Johnson's algorithm to minimize makespan in a three-machine flow shop scheduling problem, where transportation between machines was considered. Elmi et al. [122] applied simulated annealing to address a blocking hybrid flow shop scheduling problem involving robotic cells and sequencing of robot movements. Zabihzadeh et al. [123] investigated a similar problem that included blocking restrictions, aiming to minimize makespan and determine the closest feasible number of robots. They developed an Ant Colony Optimization algorithm with double pheromone and a genetic algorithm to tackle the problem. Ahmadi-javid et al. [124] focused on permutation flow shop scheduling with a finite number of transportation vehicles. They formulated the problem as a MILP and developed two anarchic society algorithms to minimize makespan. Additionally, they presented methods for establishing lower bounds for the problem and conducted a sensitivity analysis of system performance concerning transportation times and the number of vehicles. In a more recent study, Amirteimoori et al. [125] proposed a MILP model and a Parallel Hybrid PSO-GA Algorithm to address the simultaneous scheduling of jobs and AGVs within a flexible job shop system.

Given the intricate nature of the problem, the studies mentioned thus far have generally refrained from integrating the AGV-related sub-problems of task assignment and path planning with considerations for collision avoidance or transportation speed control. They typically assumed that traffic-related issues could either be disregarded or left for real-time control during schedule execution. Additionally, energy-related concerns, such as battery capacity, total energy consumption minimization, and setup energy, were typically not taken into account. For instance, Nishi et al. [126] introduced a bi-level decomposition algorithm based on mixed-integer formulation and Lagrangian relaxation to address a job scheduling and conflict-free routing problem of AGVs in a flexible flow shop served by general path

AGVs. The overarching objective here was to minimize the total weighted tardiness of the job set. Li et al. [127] examined the scheduling of hybrid flow shops and multiple robots sharing the same linear track without the ability to cross paths. To mitigate conflicts among robots, the track was divided into segments, with each segment exclusively occupied by one hoist. The primary aim was to maximize throughput. Saidi-Mehrabad et al. [128] tackled the job shop scheduling and conflict-free routing of AGVs, utilizing a MILP model and an ant colony algorithm to minimize makespan as the primary objective. The study also explored determining the optimal number of AGVs. Lyu et al. [129] concentrated on the machine and AGV scheduling problem within a FMS, incorporating AGV path planning and conflict-free routing. They proposed a combination of a GA with the Dijkstra algorithm to minimize makespan. Shouwen et al. [130] investigated the scheduling of quay cranes, AGVs, and automated stacking cranes, considering conflict-free path planning for AGVs within a general automated container terminal (ACT) layout. They introduced a bi-level programming model, aiming to minimize the maximum makespan of all containers while also minimizing AGV travel distance. Two GA-based methods were developed to solve the problem. Murakami [131] addressed the dispatching and conflict-free routing problem in a capacitated AGV FMS system, taking into account AGV and machine buffer capacities. A MILP model, based on a time-space network, was utilized to formulate and solve the problem. Zou et al. [132] tackled an AGV dispatching problem within the material handling process of an FMS, focusing on minimizing transportation costs, including travel costs, penalty costs for schedule violations, and AGV-related expenses. Their approach included formulating a MILP model and presenting a discrete artificial bee colony algorithm. However, it's important to note that none of these papers incorporated considerations for sequence-dependent setup times (SDST) or energy consumption and battery management into their respective models and solutions.

2.2.2 Energy-efficient scheduling and transportation

In recent years, the consideration of energy consumption as an objective to be optimized has gained significant attention, particularly with a focus on the transportation energy component, which constitutes a substantial portion of energy consumption in manufacturing plants and ACTs. Furthermore, various types of material handling equipment have been taken into account, including overhead hoists, AGVs, and transmission belts. For example, Xin et al. [133] explored the energy-efficient control of a quay crane, multiple AGVs, and multiple automated stacking cranes within the framework of flow shop scheduling for ACTs. They formulated a bi-level optimization problem to determine time windows that maximize space for energy efficiency while minimizing the overall completion time. Lu et al. [134] investigated energy-efficient flow shop scheduling with SDSTs and controllable transportation speeds, where transportation occurred via a transmission belt connecting the machines. They employed a hybrid multi-objective backtracking search algorithm with a genetic algorithm technique to address the problem. Jiang et al. [135] delved into a similar problem and achieved improved results by utilizing a multi-objective evolutionary algorithm based on decomposition. Xin et al. [136] extended the previous two models by considering multiple independent belts between consecutive machines that could be controlled to transport jobs at different speeds. They employed a discrete whale swarm optimization algorithm with various enhancements. Dai et al. [137] addressed an energy-efficient flexible job shop scheduling problem, with transportation tasks handled by a multi-AGV system. They developed a multi-objective MILP model and an enhanced genetic algorithm to optimize both total energy consumption, including transportation energy, and makespan objectives. Li et al. [138] conducted a similar study, incorporating the use of an imperialist competitive algorithm and introducing total tardiness as a third objective. Zhang et al. [139] investigated an energy-efficient hybrid flow shop scheduling problem with assumptions and objectives similar to Dai et al. [137]. They designed a three-stage approach based on decomposition to address the problem. Tan et al. [140] employed an enhanced PSO to tackle a carbon-efficient flexible open shop scheduling problem while optimizing AGV fleet size. More recently, He et al. [141] introduced a MILP and an enhanced multi-objective evolutionary algorithm with an opposition-based learning strategy (EMOEA/OBL) to solve an AGV-served energy-efficient job shop problem, taking into account SDSTs and controllable AGV speeds. The authors also conducted a study on the impact of the number of AGVs on the system's performance.

The studies discussed in this subsection have thus far overlooked the inclusion of blocking and collision-free transportation constraints, as well as the consideration of AGV battery management. As previously noted, there is limited literature on AGV battery management, especially in the context of integrating production scheduling with AGV battery swap/charging planning. Most existing research primarily focuses on distinctions between various battery charging strategies. For instance, McHaney [142] differentiates between four types of charging schemes: (1) Opportunity charging, where an AGV recharges during idle time; (2) Automatic charging, in which an AGV continues operating until its battery level drops below a certain threshold, (3) a combination of the previous two approaches, and (4) Rail-based charging. Kabir et al. [143] employed simulation models to demonstrate that partial battery charging can have a significant positive impact on short-term production capacity, particularly during high-demand periods. In terms of battery life and battery energy monitoring, Kawakami et al. [91] investigated the management of valve-regulated lead-acid batteries' life cycles. They analyzed the effect of AGV operation modes, such as movement, loading, and unloading, on battery life and deterioration through simulations. They proposed a battery management strategy aimed at minimizing battery-related costs. Meißner et al. [22] monitored various movement modules, including translatory movement, rotary movement, and lifting and lowering of the load-carrying platform of AGVs, and then modeled these movements using physical laws. Their results indicated that the energy requirements for translational and rotational movement increase in relation to the weight of the carried load. Another study conducted by Yonggou et al. [145] revealed that motion power accounted for slightly less than half of the total power demand in AGVs. Other components, such as sensors, microcontrollers, and embedded computers, could consume a nontrivial amount of energy, which should be minimized along with motion energy.

In the context of production scheduling and AGV battery management, Kabir et al. [19] conducted experiments to assess the impact of four different routing heuristics for battery management on manufacturing facility productivity. These heuristics determined the routing of AGVs to one of the battery stations when the battery charge dropped below 20%. Their findings indicated that more frequent battery swapping decision-making led to increased system productivity. However, it's worth noting that the study did not consider scenarios involving extended blocking times due to high congestion, which could result in significant battery discharge before a battery swap operation. Mousavi et al. [146] developed a hybrid algorithm combining GA and PSO to handle task scheduling for AGVs within a FMS. Their objectives encompassed minimizing both makespan and the number of AGVs while taking into account AGV battery charge. Barak et al. [147] explored the energy-efficient multi-objective scheduling of machines and fuel-powered AGVs within an FMS. They modeled AGV energy consumption as a linear function based on load and distance. The primary objectives were to minimize total fuel consumption and total operational costs. However, their assumptions included AGVs being capable of refueling anywhere within the manufacturing plant without needing to return to a refueling station, and the assumption that AGVs only consumed fuel during movement, with no energy consumption during idle states. Abderrahim et al. [148] employed a variable neighborhood search technique to address a job shop scheduling problem involving Battery-powered AGVs responsible for transportation. Their goal was to minimize makespan while ensuring that each AGV maintained a battery discharge level above a specified threshold to preserve battery life.

2.3 Conclusion

In summary, the literature review reveals that there has been no previous research conducted on the integration of flow shop scheduling problems with simultaneous consideration of maintenance planning and nonrenewable resource supply control. Moreover, research that considered maintenance planning do not extend to cover the planning of maintenance activities on resource supply equipment. The consideration of resource holding and inventory costs are also neglected in most previous works. To the best of our knowledge, the integration of maintenance and resource constraints in the flow shop problem remains partially explored in the existing literature.

As for the flow shop scheduling problem with transportation resources, the literature review reveals that there has been no previous research conducted on the integration of flow shop scheduling problems with simultaneous considerations for transportation speed control, AGV collision-free scheduling, and AGV battery management. Furthermore, no mixed-integer linear programming models have been proposed within this research domain. An additional significant concern frequently overlooked is the substantial idle energy consumption by AGVs during unforeseen extended periods of inactivity resulting from congestion and blockages within the system.

Table 2.2 Literature overview on the EFSP-BCFT

Reference	Scheduling environment	Additional constraints and characteristics							Optimization objectives			Solving methods		
		Transport. speed control	Conflict-free constraints	Battery management	Fleet sizing	Guide path layout	SDST	Blocking constraints	$Cmax$	Total Energy Consum.	Other	Meta-heuristic	MILP	Other
Ulusoy et al. [119]	FMS					Network			✓			✓		
Sabuncuoglu et al. [120]	FMS					Network		✓	✓		✓			✓
Behnamian et al. [121]	Flow shop					Linear			✓			✓	✓	✓
Elmi et al. [122]	Flow shop				✓			✓	✓			✓	✓	
Ahmadi-Javid et al. [124]	Flow shop				✓	Linear			✓			✓	✓	
Amirteimoori et al. [125]	Job shop						✓		✓			✓	✓	
Nishi et al. [126]	Flow shop		✓			Network					✓			✓
Li et al. [127]	Flow shop		✓			Linear		✓			✓		✓	
Saidi-Mehrabad et al. [128]	Job shop		✓		✓	Network			✓			✓	✓	
Lyu et al. [129]	FMS		✓		✓	Network			✓			✓		
Shouwen et al. [130]	ACT		✓			Network			✓		✓	✓	✓	
Murakami [131]	FMS		✓			Network					✓		✓	
Zou et al. [132]	FMS		✓		✓	Network					✓	✓	✓	
Xin et al. [133]	Flow shop	✓				Free range			✓	✓				✓
Xin et al. [136]	Flow shop	✓					✓		✓	✓		✓	✓	
Dai et al. [137]	Job shop								✓	✓		✓	✓	
Li et al. [138]	Job shop						✓		✓	✓	✓	✓		
Zhang et al. [139]	Flow shop						✓		✓	✓		✓	✓	
Tan et al. [140]	Open shop				✓				✓		✓	✓		
He et al. [141]	Job shop	✓			✓	Network	✓		✓	✓	✓	✓	✓	
Kabir et al. [19]	FMS		✓	✓	✓	Network					✓			✓
Mousavi et al. [146]	FMS			✓	✓				✓			✓		
Barak et al. [147]	FMS	✓		✓						✓	✓	✓		
Abderrahim et al. [148]	Job shop			✓		Network			✓			✓		
This work	Flow shop	✓	✓	✓	✓	Single loop	✓	✓	✓	✓		✓	✓	

Chapter 3

Flow shop and Maintenance Scheduling Problem with a Shared Nonrenewable Resource constraint

This chapter addresses the flow shop and maintenance scheduling problem under a shared nonrenewable resource (FMSP-SNR) in a permutation flow shop environment, considering constrained and nonrenewable resource allocation. The objective is to find a schedule and resource allocation that achieves a balanced trade-off between two key performance measures: the expected makespan of the schedule ($Cmax$) and the total production cost (TPC). The TPC consists of two main components: the total maintenance cost, encompassing both preventive maintenance and corrective maintenance costs, and the total resource consumption and holding costs.

The chapter is organized as follows: Section 3.1 presents a detailed description of the problem. Section 3.2 provides details about the model assumptions and context of the problem setup. In section 3.3, A Mixed Integer Non-Linear Programming (MINLP) formulation for the FMSP-SNR is detailed. A numerical example is given in section 3.4 where a complete solution is evaluated step by step. Section 3.5 describes the metaheuristic algorithms developed in order to solve the FMSP-SNR where the proposed BORLS algorithm is first presented in depth in section 3.5.1. Following this, section 3.5.2 delves into the NSGA-II metaheuristic and elaborates on the evolutionary operators incorporated into the algorithm. Then, section 3.5.3 focuses on the BOPSO algorithm and outlines the rules governing search procedure. Section 3.6 provides details about the computational tests and presents experimental results for the FMSP-SNR. Finally, concluding remarks are presented in section 3.7.

3.1 Problem Description

The problem can be described as follows: We have a set of n jobs, denoted as $J_1, J_2, \ldots, J_n$, that need to be processed on m machines, denoted as $M_1, M_2, \ldots, M_m$, in a specific flow sequence. This results in a total of mn operations, denoted as O_{ij} (where i belongs to $\{1, \ldots, n\}$ and j belongs to $\{1, \ldots, m\}$), that need to be scheduled. Each operation has a fixed processing time, denoted as P_{ij}, and requires a normal quantity of resource, denoted as Q_{ij}, as well as an expected downtime quantity of resource, denoted as $\overline{Q}_{ij}$, in cases where processing on a particular machine is interrupted due to corrective tasks. It is important to note that a job cannot commence unless the required quantity of resource, denoted as AQ_{ij}, is available in the stock where:

$$AQ_{ij} = Q_{ij} + \overline{Q}_{ij} \quad i \in \{1, .., n\}\ j \in \{1, .., m\} \tag{3.1}$$

Meaning that for each operation to begin, it is necessary to have the required normal quantity of resource available. Furthermore, an additional reserve of that resource quantity, estimated based on the machine's state, should also be accessible to account for unexpected breakdowns. Moreover, machines can experience multiple failures during the processing of a job. Each machine is treated as a single component, and we assume that the time to failure of a specific machine M_j follows a Weibull distribution with scale parameter θ_j and shape parameter β_j, where $\beta_j > 1$ for all j belonging to $\{1, \ldots, m\}$. This implies that machines have an increasing failure rate over time unless they are partially repaired. An age-based preventive maintenance policy is implemented to reduce unforeseen machine breakdowns. When a failure occurs, minimal repairs are performed, meaning that corrective maintenance actions restore the machine to a functional state without impacting its age. In contrast, preventive maintenance brings the machine back to AGAN state. The expected durations of preventive maintenance and corrective maintenance are denoted as PT_j and CT_j, respectively. Similarly, the expected costs of preventive maintenance and corrective maintenance are represented as PC_j and CC_j, respectively. It is assumed that $CT_j > PT_j$ and $CC_j > PC_j$.

By considering a continuous production flow instead of discrete jobs, we can derive the failure rate and expected number of failures within a defined period of working time. Consider a PM interval denoted as T, where a preventive maintenance action is performed on a machine at the end of each T time units. When the machine is in AGAN state following each PM interval, the production and maintenance operations can be represented as a renewal process. Moreover, as minimal repairs are conducted when a machine experiences a breakdown, the occurrence of these breakdowns during each PM interval can be modeled using a non-homogeneous Poisson process.

Define $N(T)$ as the count of failures that transpired within the duration of T time units between successive PM actions. The expected number of such failures is given by:

$$m(T) = \int_{t=0}^{t=T} z(t)dt \tag{3.2}$$

where $z(t)$ is the machine's hazard function under the Weibull distribution of the time to failure and it is given by:

$$z(t) = \frac{\beta}{\eta^{\beta}} t^{\beta-1} \tag{3.3}$$

Therefore, we get

$$m(T) = \left(\frac{T}{\eta}\right)^{\beta} \tag{3.4}$$

After determining the PM period for each machine, the rational strategy proposed by Berrichi et al. [86] is employed to insert the PM tasks.

In practice, multiple product batches are often manufactured using the same equipment. Consequently, it becomes crucial to acknowledge the possibility that subsequent batches might demand processing on the same set of machines. To address this, we consider the condition of the equipment (both production and supply machines) after completing all jobs. As a result, we assume that an extra preventive maintenance cost is accrued based on the final state of the machines from the last PM intervention until the conclusion of the last job, as indicated in equation (3.60).

Apart from the maintenance of production machines, the system faces additional constraints due to resource requirements, which introduce more challenges and complexities to the underlying job scheduling problem, especially when the consumable resource is shared among all machines in the system. Numerous studies in the literature have addressed resource-constrained scheduling, offering complexity analyses of scheduling problems under nonrenewable resource constraints. Some of these studies focus on single-machine problems [149 – 155]. Meanwhile, others tackle parallel machines problems [156 – 159].

In unconstrained flow shop scheduling problems, the main focus is on determining the job sequence. However, when a shared resource exists within the system, scheduling the operations becomes an additional concern after fixing the job sequence. In other words, for a particular sequence of jobs on machines, it becomes necessary to prioritize the operations concerning resource consumption.

The resource supplement allocated to the cell, which contains a set of machines, is provided at a constant rate RR_o, while the supplier machine is operational. Here, R_o represents the normal production rate of the supplier machine, and the value R can be adjusted to control the machine's speed. It's worth noting that the supplier machine is susceptible to unforeseen breakdowns and thus requires maintenance. To model the failure process of the supplier machine, we employ an availability model that considers the impact of the production rate and treats the machine as a single component. The availability of a machine, denoted as M, is defined as the probability that the machine is functioning properly at any given point in time, performing its required function.

$$A(t) = Pr(M \text{ is operational at time } t)$$

In this problem, availability is furthermore used to model the expected cumulative resource supply. The used point availability expression of the supply machine is as follows:

$$A(t) = \frac{1}{\sqrt{R^{\omega}\lambda t + 1}} \tag{3.5}$$

where ω is a shape parameter that reflects the sensitivity of the machine to the production rate and λ is a scale parameter that expresses the decrease in availability when a normal rate $(R = 1)$ is used. Additionally, we assume that PM actions can restore the supply machine to its AGAN condition. When the parameter ω is large, employing a rate higher than the normal one might become too risky, leading to a higher probability of premature failure. Conversely, when ω is small, it becomes advantageous to leverage the machine's robustness and utilize higher supply rates. Fig. 3.1 illustrates the evolution of the machine's availability under various rates.

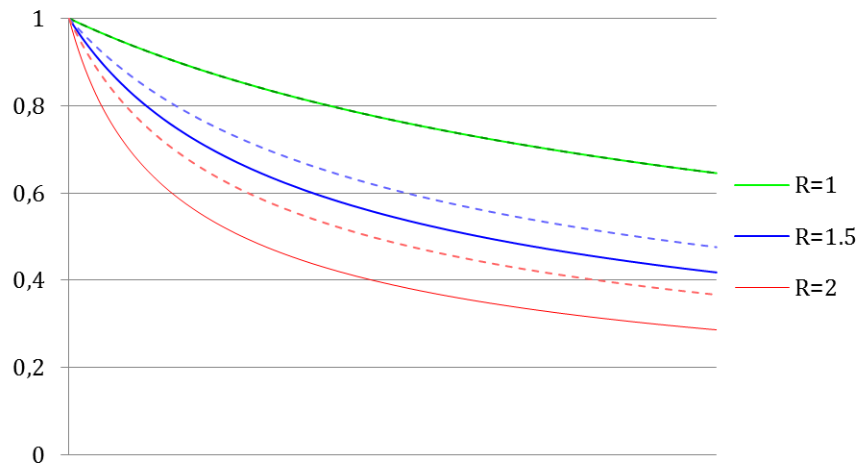

Fig. 3.1 Supply machine availability for different production rates. Normal lines represent availability for $\omega = 3$ and dashed lines for $\omega = 2.2$

The total expected uptime of the machine to a given date T excluding downtime due to PM actions is:

$$U(T) = \int_0^T A(t)dt = \frac{2}{R^{\omega}\lambda}\left(\sqrt{R^{\omega}\lambda T + 1} - 1\right) \tag{3.6}$$

The expected cumulative supply at T is:

$$E[CS(T)] = RR_o U(T) \tag{3.7}$$

In order to compute the earliest date $t(r)$ to start an operation with r cumulative resource consumption units, $E[CS(t)]$ need to be inverted, and we get:

$$t(r) = \frac{r}{RR_o}\left(\frac{R^{\omega-1}\lambda}{4R_o}r + 1\right) \tag{3.8}$$

If PRT (*resp*. PMP) are the PM duration (*resp*. period), we refer to the renewal cycle time as CT, where:

$$CT = PRT + PMP \tag{3.9}$$

Therefore, completion time of the first PM action is CT and nCT for the n^{th} one. Since we assume that PM restores the machine to AGAN we got:

$$A(nCT) = 1 \quad n = 1,2,\ldots \tag{3.10}$$

Of course, while a PM action is being performed, the machine is shut down and becomes unavailable. We can express the availability at any given point in time as:

$$A(t) = \begin{cases} \dfrac{1}{\sqrt{R^{\omega}\lambda(t - nCT) + 1}} & nCT \leq t < nCT + PMP \\ 0 \quad nCT + PMP \leq t < (n+1)CT \quad n = 1,2,\ldots \end{cases} \tag{3.11}$$

From this expression we can deduce the expected cumulative supply at any given point in time:

$$E[CS(t)] = \begin{cases} \dfrac{2R_o}{R^{\omega-1}\lambda}\left(\sqrt{R^{\omega}\lambda(t - nCT) + 1} - 1\right) + nE[CS(PMP)] \\ \qquad\qquad\qquad\qquad nCT \leq t < nCT + PMP \\ nE[CS(PMP)] \quad nCT + PMP \leq t < (n+1)CT \quad n = 1,2,\ldots \end{cases} \tag{3.12}$$

Fig. 3.2 and Fig. 3.3 depict the supply machine's availability trend and the expected cumulative supply, respectively, while considering the impact of preventive maintenance activities.

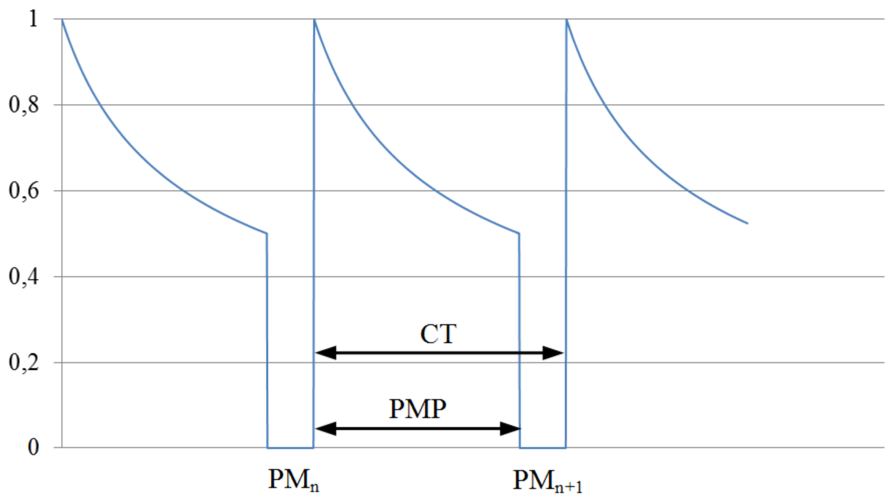

Fig. 3.2 Supply machine availability with PM consideration

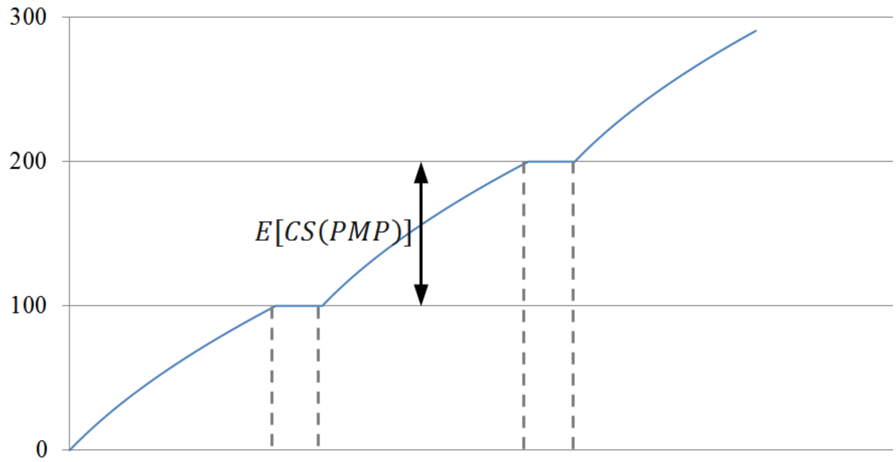

Fig. 3.3 Cumulative Supply with PM consideration

The earliest date to start an operation with r cumulative resource consumption units and after n renewal cycles in this case is written as follows:

$$t(r) = \frac{r - nE[CS(PMP)]}{RR_o}\left(\frac{R^{\omega-1}\lambda}{4R_o}(r - nE[CS(PMP)]) + 1\right) + nCT$$
$$nE[CS(PMP)] \leq r < (n+1)E[CS(PMP)] \tag{3.13}$$

The resource quantity generated by the supplier is stored in the "global stock" of the cell, where it remains until all jobs are completed. When an operation O_{ij} starts, the necessary quantity is drawn from the global stock and moved to the "local stock" specific to machine j. During the operation's progress, the resource is gradually consumed from the local stock without incurring any additional holding cost. We assume that the time required to pull the required resource quantity is negligible and can be disregarded.

From a decision-making perspective, when a set of operations (up to m) must be initiated simultaneously, and there is an insufficient resource quantity available, two priority rules are applied to determine which operation to start first once enough resource becomes available:

1. Priority is given to the operations based on their resource quantity requirements. Operations with smaller quantity needs are selected first.
2. If certain operations require an equal amount of resource, they are further ranked according to their machine level, with operations in the later stages of the process being selected first.

When calculating the resource holding cost, the average inventory between two consecutive operation start dates, multiplied by the time interval between them, serves as a straightforward approach to approximate the total expenses incurred in maintaining the stock.

3.2 Model Assumptions and Problem Setup

The general form of the problem is governed by the following assumptions:

1. Preemption of jobs is not allowed, meaning each operation must be completed without interruption from another.
2. The problem follows a permutation flow shop, where the sequence of jobs remains unchanged from one machine to another.
3. Each machine can process one job at a time, and each job can be processed on only one machine at a time.
4. Intermediate buffers between consecutive machines have unlimited capacity.
5. Resources are managed through a dedicated supply line with a limited capacity (i.e., limited supply rate RR_o), a resource container (both global and local) with unlimited storage capacity, and an initially empty stock dedicated to all machines (refer to Fig. 3.4).
6. At any given point in time, cumulative resource consumption never exceeds cumulative resource supply.
7. No resource is required while a job is waiting in the buffer between two consecutive machines.
8. Any residual resource remaining after completing all operations is not stored for future tasks and is treated as waste.
9. All machines are in a "good as new" condition at time zero.

The problem setting, as described above, is depicted in Fig. 3.4.

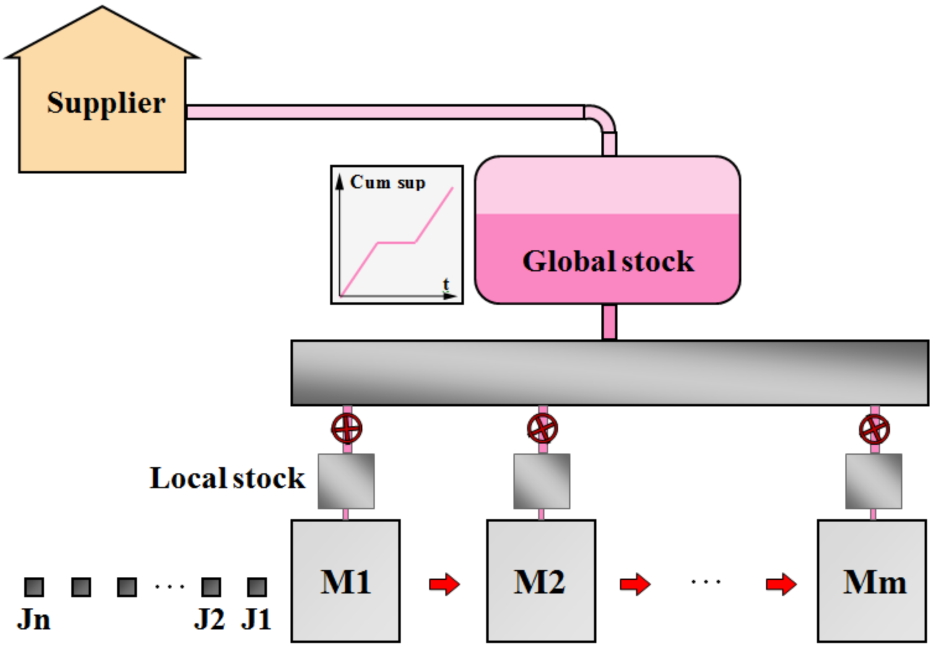

Fig. 3.4 Problem setting.

This problem commonly arises in chemical and pharmaceutical setups, where raw chemical materials necessitate additional consumable ingredients during the processing. In such scenarios, process interruptions caused by unexpected equipment breakdowns or excessive ingredient usage may require the entire job to be reprocessed. A study by von Hoyningen-Huene & Kiesmüller [160] examined similar circumstances in a parallel machine problem with stochastic failures and non-resumable jobs. They evaluated the expected makespan when jobs had to be restarted due to breakdowns occurring before processing was fully completed, considering that a job could be reprocessed at most once. In our situation, we focus on the resumable case, where an extra quantity of substances is added to preserve the chemical process activity and prevent irreversible damage to the compound while the machine undergoes repair. As mentioned earlier, besides the regular quantity, an additional reserve is required before commencing any operation. This extra amount acts as a safeguard against sudden job destruction and allows the maintenance team time to prepare equipment and investigate the cause of the problem in case of a failure. In real-life scenarios, if a corrective maintenance intervention takes longer than anticipated and the entire local stock is consumed, operators may resort to directly consuming from the global stock (as depicted in Fig. 3.5 for job 2 on machine M2). If this is not feasible, the job is restarted. Furthermore, any remaining resource after the completion of an operation can be used by the subsequent operation on the same machine.

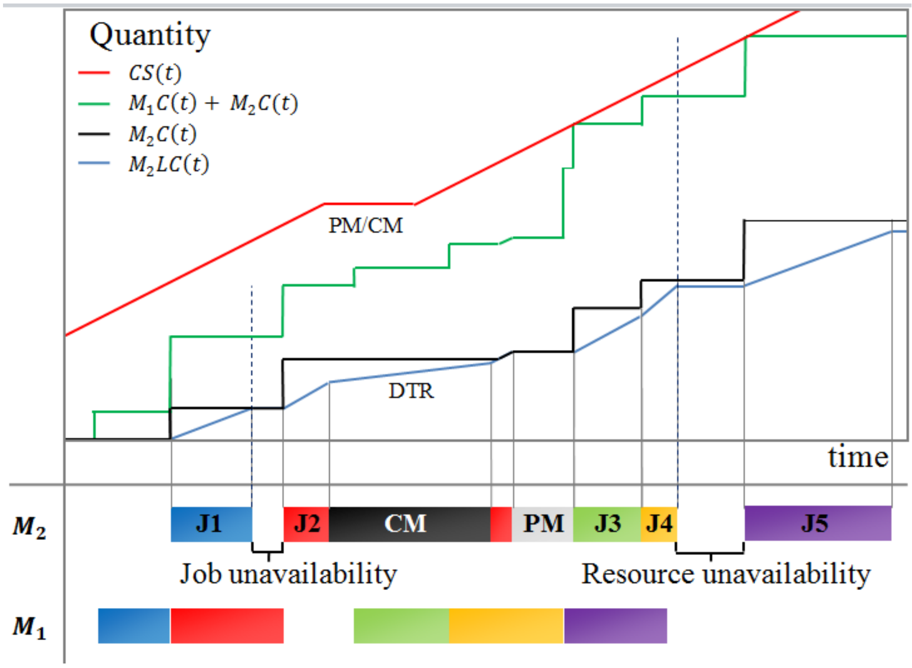

Fig. 3.5 Interaction between the three aspects of the problem; where:

$CS(t)$: *Cumulative supply at time* t*;*

$M_1C(t) + M_2C(t)$: *Cumulative consumption at time* t*;*

$M_2C(t)$: *Partial cumulative consumption by* M_2 *at time* t*;*

$M_2LC(t)$: *Local cumulative consumption by* M_2 *at time* t*;*

Given the sensitivity and potential hazards of the substances involved, specific measures are taken both at the end of and before each batch of jobs. These measures may include reprocessing damaged jobs, cleaning, machine restoration, and inspections. However, these setup procedures between batches can be time-consuming, lasting for hours or even an entire day. During this period, any remaining resources might become outdated and unsuitable for further use. To address this, a process is implemented to discharge the residual additives. To eliminate these residuals effectively, the total estimated quantity of resources required is used to determine the appropriate moment to cut off the supply, thus avoiding overproduction. For instance, the supplier machine is shut down when the total produced resource reaches the total expected consumed resource. This stopping date is referred to as ST in our context.

Considering the long production times in such industries, resource-holding costs become a significant factor that should be considered. As a result, we have introduced the inventory cost as another important element that impacts the overall performance of the plant.

Fig. 3.5 illustrates the intricate interaction between the three key aspects of this problem: production, maintenance, and resource supply.

3.3 A Mixed Integer Non-Linear Programming Formulation for the FMSP-SNR

This section presents a detailed proposal for a mixed-integer nonlinear programming (MINLP) formulation of the problem. The MINLP model is utilized to identify a representative sample from the Pareto optimal set, which allows to evaluate the performance of the developed solving methods (refer to section 3.6.2).

3.3.1 Indices

- i Jobs $i = 1 \dots n$
- j, p Machines $j = 1 \dots m$
- k, l Position $k = 1 \dots n$

3.3.2 Parameters

- P_{ij} Processing time of job i on machine j
- Q_{ij} Uptime quantity of resource required by job i on machine j
- PT_j Preventive maintenance time on machine j
- CT_j Corrective maintenance time on machine j
- PC_j Preventive maintenance cost on machine j
- CC_j Corrective maintenance cost on machine j
- θ_j Scale parameter of machine j
- β_j Shape parameter of machine j
- PRT Preventive maintenance time on supply machine
- CRT Corrective maintenance time on supply machine
- PRC Preventive maintenance cost supply machine
- CRC Corrective maintenance cost supply machine
- λ Scale parameter of supply machine
- ω Rate sensitivity factor of supply machine
- R_o Normal resource rate
- R_{max} Maximum allowed resource arrival rate
- R_{min} Minimum allowed resource arrival rate
- q Resource unit cost
- h Resource holding cost per unit of time
- $\alpha \in (0, 1)$ Downtime consumption rate factor
- M A large positive number
- $\varepsilon_1, \varepsilon_2$ Small positive numbers

3.3.3 Decision Variables

- X_{ki} Binary that takes value 1 if job i is assigned position k in the sequence

- Y_{kj} Binary that takes value 1 if a PM task is assigned following the k^{th} job on j^{th} machine
- φ_{kj} Integer that indicates how many PM tasks on supply machine have been performed before k^{th} operation on j^{th} machine
- σ_{kj} Integer used to compute the resource stock level right before performing k^{th} operation on j^{th} machine
- ψ_{kj} Real used to compute the resource stock level right before performing k^{th} operation on j^{th} machine
- π_{kj} Binary used to compute the resource stock level right before performing k^{th} operation on j^{th} machine
- e_{kj} Binary used to construct unconstrained schedule (σ_3)
- I_{kj}^{lp} Binary that takes value 1 if k^{th} operation on j^{th} machine is performed any time after l^{th} operation on p^{th} machine
- Z_{kj}^{lp} Binary that takes value 1 if k^{th} operation on j^{th} machine is performed right after l^{th} operation on p^{th} machine
- ρ_{kj} Binary that takes value 1 if the k^{th} operation on j^{th} machine is started after the resource supply stopping date
- PMP_j Preventive maintenance period on machine *j*
- PRP Preventive maintenance period on supply machine
- R Resource supply rate

3.3.4 Intermediate Variables

- P_{kj} Processing time of the k^{th} job on machine *j*
- Q_{kj} Uptime quantity of resource required the k^{th} job on machine *j*
- A_{kj} Age of machine *j* after processing the k^{th} job
- B_{kj} Age of machine *j* before processing the k^{th} job
- N_{kj} Expected number of j^{th} machine failures from last PM intervention until the completion of the k^{th} job
- η_{kj} Expected number of j^{th} machine failures while processing the k^{th} job
- μ Number of preventive maintenance actions on supply machine
- ν Expected number of Corrective maintenance actions on supply machine
- U Total uptime of supply machine
- D Total downtime of supply machine due to corrective maintenance
- CDT_{kj} Expected downtime of j^{th} machine while processing the k^{th} job due to corrective maintenance
- PDT_{kj} Preventive maintenance downtime of j^{th} machine immediately after processing the k^{th} job
- MC_{kj} Expected maintenance cost immediately after processing the k^{th} job on the j^{th} machine
- FMC Average maintenance cost after completing all the jobs

- S'_{kj} Start time of the k^{th} operation on the j^{th} machine in unconstrained schedule (σ_3)
- C'_{kj} Completion time of the k^{th} operation on the j^{th} machine in unconstrained schedule (σ_3)
- S_{kj} Start time of the k^{th} operation on the j^{th} machine in final schedule (σ_4)
- C_{kj} Completion time of the k^{th} operation on the j^{th} machine
- DTR_{kj} Downtime resource consumption rate of j^{th} machine while processing the k^{th} job due to corrective maintenance
- AQ_{kj} Expected actual quantity of resource consumed by job k on machine j
- r_{kj} Cumulative resource demand of operation O_{kj}
- RL_{kj} Resource stock level right before starting operation O_{kj}
- INV Total inventory cost
- TRC Total resource costs
- ξ_{kj} Resource consumed since the previous PM task on the supply machine
- t_{kj} Earliest allowed start time of the k^{th} operation on the j^{th} machine
- ST Stop date of resource supply
- CT Renewal cycle time of supply machine

3.3.5 Mathematical Model

Objectives:

$$Minimize\ C_{max} = C_{nm} \tag{3.14}$$

$$Minimize\ TPC = TMC + TRC \tag{3.15}$$

Subject to:

$$\sum_{i=1}^{n} X_{ki} = 1 \quad \forall k \in \{1,..,n\} \tag{3.16}$$

$$\sum_{k=1}^{n} X_{ki} = 1 \quad \forall i \in \{1,..,n\} \tag{3.17}$$

$$P_{kj} = \sum_{i=1}^{n} X_{ki} P_{ij} \quad \forall k \in \{1,..,n\}\ \forall j \in \{1,..,m\} \tag{3.18}$$

$$Q_{kj} = \sum_{i=1}^{n} X_{ki} Q_{ij} \quad \forall k \in \{1,..,n\}\ \forall j \in \{1,..,m\} \tag{3.19}$$

$$PMP_j \leq \sum_{i=1}^{n} P_{ij} \quad \forall j \in \{1,..,m\} \tag{3.20}$$

$$PMP_j \geq \max_{1 \leq i \leq n} \{P_{ij}\} \quad \forall j \in \{1,..,m\} \tag{3.21}$$

$$Y_{kj} \leq \frac{2A_{kj} + P_{k+1,j}}{2PMP_j} \quad \forall k \in \{1,..,n-1\} \;\; \forall j \in \{1,..,m\} \tag{3.22}$$

$$Y_{kj} \geq \frac{2A_{kj} + P_{k+1,j}}{2PMP_j} - 1 \quad \forall k \in \{1,..,n-1\} \;\; \forall j \in \{1,..,m\} \tag{3.23}$$

$$A_{kj} = B_{kj} + P_{kj} \quad \forall k \in \{1,..,n\} \, \forall j \in \{1,..,m\} \tag{3.24}$$

$$B_{k+1,j} = A_{kj}(1 - Y_{kj}) \quad \forall k \in \{1,..,n-1\} \, \forall j \in \{1,..,m\} \tag{3.25}$$

$$N_{kj} = \left(\frac{A_{kj}}{\theta_j}\right)^{\beta_j} \quad \forall k \in \{1,..,n\} \, \forall j \in \{1,..,m\} \tag{3.26}$$

$$\eta_{1j} = N_{1j} \quad \forall j \in \{1,..,m\} \tag{3.27}$$

$$\eta_{kj} = N_{kj} - N_{k-1,j}\big(1 - Y_{k-1,j}\big) \quad \forall k \in \{2,..,n\} \, \forall j \in \{1,..,m\} \tag{3.28}$$

$$CDT_{kj} = CT_j \eta_{kj} \quad \forall k \in \{1,..,n\} \, \forall j \in \{1,..,m\} \tag{3.29}$$

$$PDT_{kj} = PT_j Y_{kj} \quad \forall k \in \{1,..,n\} \, \forall j \in \{1,..,m\} \tag{3.30}$$

$$MC_{kj} = PC_j Y_{kj} + CC_j \eta_{kj} \quad \forall k \in \{1,..,n\} \, \forall j \in \{1,..,m\} \tag{3.31}$$

$$DTR_{kj} = \alpha \frac{Q_{kj}}{P_{kj}} \quad \forall k \in \{1,..,n\} \, \forall j \in \{1,..,m\} \tag{3.32}$$

$$\overline{Q}_{kj} = DTR_{kj} CDT_{kj} \quad \forall k \in \{1,..,n\} \, \forall j \in \{1,..,m\} \tag{3.33}$$

$$AQ_{kj} = Q_{kj} + \overline{Q}_{kj} \quad \forall k \in \{1,..,n\} \, \forall j \in \{1,..,m\} \tag{3.34}$$

$$ST = t_{nm} \tag{3.35}$$

$$CT = PRP + PRT \tag{3.36}$$

$$\mu = \varphi_{nm} \tag{3.37}$$

$$U = \frac{\sum_{k=1}^{n} \sum_{j=1}^{m} AQ_{kj}}{RR_o} \tag{3.38}$$

$$D = ST - U - PRT\mu \tag{3.39}$$

$$\nu = \frac{D}{CRT} \tag{3.40}$$

$$\begin{gathered} \big(S'_{lp} + \varepsilon_1 Q_{lp} + \varepsilon_2 p\big) - \big(S'_{kj} + \varepsilon_1 Q_{kj} + \varepsilon_2 j\big) + MI_{kj}^{lp} \geq 0 \\ \forall l,k \in \{1,..,n\} \, \forall p,j \in \{1,..,m\} \end{gathered} \tag{3.41}$$

$$I_{kj}^{lp} + I_{lp}^{kj} = 1 \quad \forall l, k \in \{1, .., n\} \, \forall p, j \in \{1, .., m\} \, \& \, kj \neq lp \tag{3.42}$$

$$r_{kj} = \sum_{l=1}^{n} \sum_{p=1}^{m} I_{kj}^{lp} AQ_{lp} + AQ_{kj} \qquad \forall k \in \{1, .., n\} \, \forall j \in \{1, .., m\} \tag{3.43}$$

$$\varphi_{kj} \geq \frac{r_{kj}}{E[CS(PRP)]} - 1 \quad \forall k \in \{1, .., n\} \, \forall j \in \{1, .., m\} \tag{3.44}$$

$$\varphi_{kj} \leq \frac{r_{kj}}{E[CS(PRP)]} \quad \forall k \in \{1, .., n\} \, \forall j \in \{1, .., m\} \tag{3.45}$$

$$\xi_{kj} = r_{kj} - E[CS(PRP)]\varphi_{kj} \quad \forall k \in \{1, .., n\} \, \forall j \in \{1, .., m\} \tag{3.46}$$

$$t_{kj} = \frac{\xi_{kj}}{RR_o} \left(\frac{R^{\omega-1}\lambda}{4R_o} \xi_{kj} + 1 \right) + CT\varphi_{kj} \quad \forall k \in \{1, .., n\} \, \forall j \in \{1, .., m\} \tag{3.47}$$

$$\sigma_{kj} \geq \frac{S_{kj}}{CT} - 1 \quad \forall k \in \{1, .., n\} \, \forall j \in \{1, .., m\} \tag{3.48}$$

$$\sigma_{kj} \leq \frac{S_{kj}}{CT} \quad \forall k \in \{1, .., n\} \, \forall j \in \{1, .., m\} \tag{3.49}$$

$$\pi_{kj} \geq \frac{S_{kj} - \sigma_{kj}CT}{PRP} - 1 \quad \forall k \in \{1, .., n\} \, \forall j \in \{1, .., m\} \tag{3.50}$$

$$\pi_{kj} \leq \frac{S_{kj} - \sigma_{kj}CT}{PRP} \quad \forall k \in \{1, .., n\} \, \forall j \in \{1, .., m\} \tag{3.51}$$

$$\psi_{kj} = \left(S_{kj} - \sigma_{kj}CT\right)\left(1 - \pi_{kj}\right) + PRP\pi_{kj} \quad \forall k \in \{1, .., n\} \, \forall j \in \{1, .., m\} \tag{3.52}$$

$$\rho_{kj} \geq \frac{S_{kj}}{ST} - 1 \quad \forall k \in \{1, .., n\} \, \forall j \in \{1, .., m\} \tag{3.53}$$

$$\rho_{kj} \leq \frac{S_{kj}}{ST} \quad \forall k \in \{1, .., n\} \, \forall j \in \{1, .., m\} \tag{3.54}$$

$$RL_{kj} = \left(\frac{2R_o}{R^{\omega-1}\lambda} \left(\sqrt{R^{\omega}\lambda\psi_{kj} + 1} - 1 \right) + E[CS(PRP)]\sigma_{kj} \right) \left(1 - \rho_{kj}\right) + r_{nm}\rho_{kj} - \left(r_{kj} - AQ_{kj}\right) \qquad \forall k \in \{1, .., n\} \, \forall j \in \{1, .., m\} \tag{3.55}$$

$$Z_{kj}^{lp} \geq \sum_{v=1}^{n} \sum_{w=1}^{m} I_{lp}^{vw} - \sum_{v=1}^{n} \sum_{w=1}^{m} I_{kj}^{vw} - M\left(1 - I_{kj}^{lp}\right) + 2 \qquad \forall k, l \in \{1, .., n\} \, \forall j, p \in \{1, .., m\} \tag{3.56}$$

$$\sum_{l=1}^{n} \sum_{p=1}^{m} Z_{11}^{lp} = 0 \tag{3.57}$$

$$\sum_{l=1}^{n} \sum_{p=1}^{m} Z_{kj}^{lp} = 1 \quad \forall k \in \{2, .., n\} \, \forall j \in \{1, .., m\} \tag{3.58}$$

$$INV = \frac{1}{2}\left(\sum_{k=2}^{n}\sum_{j=1}^{m}\sum_{l=1}^{n}\sum_{p=1}^{m}\left(S_{kj} - S_{lp}\right)\left(RL_{kj} + RL_{lp} - AQ_{lp}\right)Z_{kj}^{lp} + S_{11}RL_{11}\right) \quad (3.59)$$

$$FMC = \sum_{j=1}^{m} PC_j \frac{A_{nj}}{\sum_{i=1}^{n} P_{ij}} + PRC\frac{ST - (PRP + PRT)\mu}{ST} \quad (3.60)$$

$$TMC = \sum_{k=1}^{n}\sum_{j=1}^{m} MC_{kj} + PRC\mu + CRC\nu + FMC \quad (3.61)$$

$$TRC = q\sum_{k=1}^{n}\sum_{j=1}^{m}\overline{Q}_{kj} + hINV \quad (3.62)$$

$$S'_{k,j+1} \geq C'_{kj} \quad \forall k \in \{1,..,n\}\, \forall j \in \{1,..,m-1\} \quad (3.63)$$

$$S'_{k+1,j} \geq C'_{kj} + PDT_{kj} \quad \forall k \in \{1,..,n-1\}\, \forall j \in \{1,..,m\} \quad (3.64)$$

$$S'_{kj} = e_{kj}\left(C'_{k-1,j} + PDT_{k-1,j}\right) + \left(1 - e_{kj}\right)\left(C'_{k,j-1}\right) \quad \forall k \in \{2,..,n\}\, \forall j \in \{2,..,m\} \quad (3.65)$$

$$S'_{11} = 0 \quad (3.66)$$

$$S'_{k+1,1} = C'_{k,1} \quad \forall k \in \{1,..,n-1\} \quad (3.67)$$

$$S'_{1,j+1} = C'_{1,j} \quad \forall j \in \{1,..,m-1\} \quad (3.68)$$

$$C'_{kj} = S'_{kj} + P_{kj} + CDT_{kj} \quad \forall k \in \{1,..,n\}\, \forall j \in \{1,..,m\} \quad (3.69)$$

$$S_{kj} \geq t_{kj} \quad \forall k \in \{1,..,n\}\, \forall j \in \{1,..,m\} \quad (3.70)$$

$$S_{k,j+1} \geq C_{kj} \quad \forall k \in \{1,..,n\}\, \forall j \in \{1,..,m-1\} \quad (3.71)$$

$$S_{k+1,j} \geq C_{kj} + PDT_{kj} \quad \forall k \in \{1,..,n-1\}\, \forall j \in \{1,..,m\} \quad (3.72)$$

$$C_{kj} = S_{kj} + P_{kj} + CDT_{kj} \quad \forall k \in \{1,..,n\}\, \forall j \in \{1,..,m\} \quad (3.73)$$

$$R_{min} \leq R \leq R_{max} \quad (3.74)$$

$$T_{min} \leq PRP \leq T_{max} \quad (3.75)$$

The MINLP formulation tackles the problem of simultaneously determining a sequence of n jobs, developing a PM plan for each production and supply machine, and setting the supply rate to minimize both objectives ($Cmax$ and TPC). These objectives are nonconvex functions and exhibit discontinuous behavior in relation to certain decision variables. For instance, the objective functions yield different constant values for specific ranges of preventive maintenance intervals on production machines. Similarly, the objective functions are nonconvex and discontinuous with respect to the preventive maintenance interval for the resource supply machine. The model comprises a mix of linear and nonlinear constraints. The job sequencing constraints (3.16), (3.17) ensure that each position in the sequence corresponds to a single job, and each job is assigned to a unique position in the sequence. Additionally, linear constraints (3.18), (3.19) are used to calculate the processing time and normal resource requirement for the kth job in the sequence on machine j. The PM intervals for the production machines are selected from a discrete set of integer values within the range from the maximum processing time to the sum of processing times. This results in no PM intervention during the entire planning horizon, as indicated by linear constraints (3.20), (3.21). To establish the lower bound on PMP_j, the assumption is made that any machine can handle the job with the maximum processing time without requiring intervention throughout its processing.

The nonlinear constraints concerning Y_{kj} (3.22); (3.23) capture the conditions for implementing a preventive maintenance action based on a rational strategy. They involve deciding whether to perform the PM action earlier or later compared to the chosen PMP_j and the current age of machine j. In linear constraint (3.24), the age of machine j after processing job k is computed as the sum of its age before processing the job and the processing time of job k. The nonlinear constraint (3.25) sets the age of the machine before processing job $k+1$ to zero if a PM action is carried out after completing job k. This ensures that PM is performed immediately after job k, resetting the age of the machine. Constraint (3.26) is derived from (3.4) and estimates the expected number of machine failures between two consecutive PM actions based on the current age of the machine. Constraints (3.27); (3.28) are used to determine the expected number of failures during the processing of a job. All constraints (3.26) – (3.28) are nonlinear since they are derived from (3.4), which represents a convex and monotone function of the operational time of the machine since the last PM intervention.

Nonlinear (3.29) and linear (3.30) constraints introduce the corrective and preventive maintenance times into the schedule. CM times CDT_{kj} will be added to the raw processing times P_{kj}. i.e., job k on machine j will not be completed until $P_{kj} + CDT_{kj}$ time units are elapsed. PM times PDT_{kj} will be added after the processing job k. i.e., machine j will not be free to process job $k+1$ only after PDT_{kj} time units after the completion of job k. (3.31) computes the partial maintenance cost of operation O_{kj}.

The calculation of the downtime resource consumption rate for operation O_{kj} during a CM repair is determined by equation (3.32). It is assumed that all jobs share the same downtime resource factor α. Equation (3.33) represents the unexpected resource quantity consumed by operation O_{kj} during the downtime of the machine. The total actual quantity of resource consumed is presented in equation (3.34). In equation (3.35), the stopping date of the resource supply is computed using equation (3.8), which is a convex and monotone function of the resource parameter r. In this context, r refers to the total expected quantity of resource consumed by all the jobs. In other words, the supplier will not halt the machine until the total normal resource consumption, along with the estimated downtime quantities, are produced.

The number of PM actions performed on the supply machine is determined in equations (3.36); (3.37) by the value of φ_{nm}. This is equivalent to taking the floor of ST divided by the PM cycle time. Equation (3.38) expresses the total uptime of the supply machine, which is calculated as its total expected produced quantity of resources divided by its production rate. The total downtime due to corrective repairs is introduced in equation (3.39) by subtracting the total uptime and the total preventive maintenance time from the stopping date (assuming the planning horizon starts at instant 0). In equation (3.40), the expected number of CM tasks is computed. All of these computations are nonlinear in nature.

In linear constraint (3.41), the priority of each operation is established based on the previously described priority rules. The values of ε_1 and ε_2 are set to ensure that the correct order of tie-breaking is followed. Linear constraint (3.42) ensures that either operation O_{kj} precedes O_{lp} or vice versa. In (3.43), the prioritization is utilized to compute the cumulative consumed resource up to operation O_{kj} j. Constraints (3.44); (3.45) are used to calculate the number of renewal cycles for the supply machine before the execution of each operation. In (3.46); (3.47), the earliest date to start any operation is determined based on the cumulative resource consumption and the number of renewal cycles. Constraints (3.48) – (3.54) are modeling constraints used to set up the values of σ_{kj}, π_{kj}, ψ_{kj} and ρ_{kj} for computing the inventory cost of the schedule. In (3.55), the resource level right before the start of each operation is computed. All constraints from (3.43) – (3.55) are nonlinear with respect to the involved decision variables, while (3.56) – (3.58) are linear constraints for Z_{kj}^{lp}. Constraint (3.56) ensures that if two operations are consecutive in their priorities, the variable Z_{kj}^{lp} takes the value 1. (3.57) ensures that the first operation is not preceded by any other operation, and constraint (3.58) ensures that any operation, other than the first, is consecutively preceded by a single operation. Constraints (3.59) – (3.62) involve nonlinear computations for total production cost. In (3.59), the total inventory cost is presented. For (3.60), it is assumed that production machine j receives a PM intervention after the completion of the last job with a probability of $A_{nj}/\sum_{i=1}^{n} P_{ij}$ and the supplier machine with a probability of $[ST-(PRP+PRT)\mu]/ST$. This equation is established under the assumption that if no PM intervention is made during the planning horizon, then an obligatory one is performed at the end (as a setup for the next batch). The two components of the production cost are introduced in (3.61); (3.62).

Constraints (3.63) – (3.69) are responsible for constructing the unconstrained schedule, upon which the operations prioritization is based. The various time limits that restrict the starting date of each operation in the final schedule are described in (3.70) – (3.72). To determine the starting date for each operation, the maximum of these limits is taken, as shown in the equation (3.73). It is important to note that since the completion times for schedules σ_3 and σ_4 are nonlinear, all constraints involving the makespan computations are also nonlinear.

Constraint (3.74) imposes upper and lower bounds on the supply rate, ensuring that it remains within a specific range. This constraint is linear. Similarly, constraint (3.75) sets upper and lower bounds on the PM interval for the supply machine, restricting it to a certain range. This constraint is also linear.

Developing a linear model becomes challenging due to the significant cost associated with linearizing most of the nonlinear functions and constraints. This cost arises from the introduction of a large number of new variables, both binary and real-valued. For instance, linearizing equations (3.7) and (3.8) used to model the cumulative supply and earliest start dates necessitates dividing the range of values that the resource supply rate R and the cumulative supply r can take into multiple parts. Each part requires defining a linear function, and in the case of Eq. (3.7), adding binary variables equal to the number of blocks to select the correct block and evaluate the linearized function due to its concave nature. Another illustration of the complexity in developing a linear model is found in constraints (3.22); (3.23), where the separation of the two decision variables Y_{kj} and PMP_j demands linearization. This linearization entails discretizing the range of possible values that the variable PMP_j can take for each machine and introducing a binary variable for each point. Even for relatively small problems, the number of added binary variables can reach hundreds, further increasing the model's complexity and computational requirements.

3.4 Numerical Example

Consider an illustrative example of a flow shop problem with two stages. The provided data is as follows:

		J1	J2	J3	J4	J5	J6	J7	J8
P_{ij}	M1	4	5	8	10	6	3	7	6
	M2	8	7	7	6	9	6	4	3
Q_{ij}	M1	4	3	8	2	5	5	6	3
	M2	4	2	5	5	1	8	10	10

	PT	CT	PC	CC	θ	β
M1	4	10	6	13	30	2
M2	2	8	4	29	38	3,2

PRT	CRT	PRC	CRC	λ	ω	R_o	q	h
3	15	7	15	0.1	3	1.7	2.5	0.05

We assume α = 0.2. Now, suppose we have to evaluate the following schedule:

Job permutation: J1- J2- J3- J4- J5- J6- J7- J8

PM periods:

M1	M2	Supply
18	30	25

Fig. 3.6 depicts the schedule, which we refer to as σ_1, obtained by disregarding machine aging effects and resource constraints.

Let σ_2 represent the schedule constructed by considering machine deterioration and CM activities, but disregarding PM tasks and resource constraints. The corresponding Gantt chart is illustrated in Fig. 3.7.

σ_3 illustrated in Fig. 3.8 is the schedule obtained by incorporating the PM actions into σ_2. σ_3 serves as the basis for generating σ_4 by incorporating resource availability constraints and applying the previously mentioned priority rules to prioritize production operations when simultaneous start is infeasible due to resource insufficiency. For example, attempting to start J2 on M1 and J1 on M2 simultaneously (as shown in σ_3) is not feasible since, at $t = 6.69$, the expected global stock level is at 5.89, which is only sufficient to start one of the operations. Consequently, we prioritize launching J2 on M1 based on the first rule, considering that O_{21} would consume a smaller quantity compared to O_{12}. The corresponding Gantt chart is displayed in Fig. 3.9.

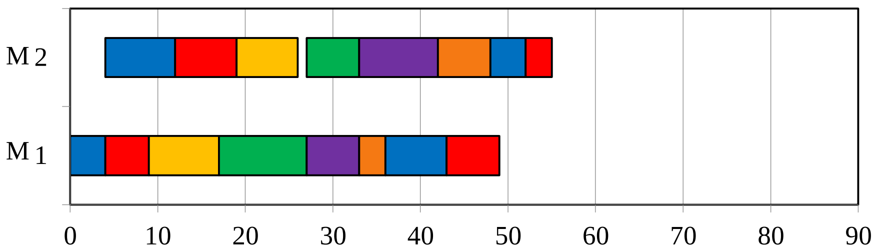

Fig. 3.6 Gantt chart σ_1 schedule with $C_{max}(\sigma_1) = 55$

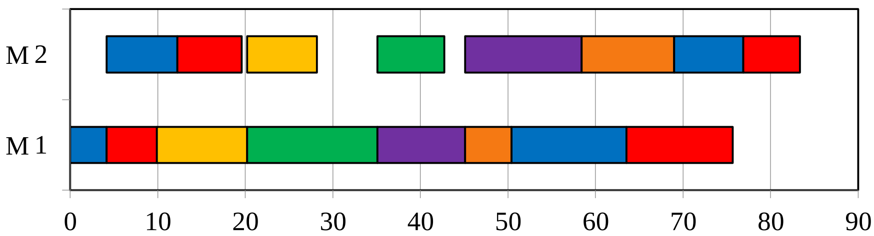

Fig. 3.7 Gantt chart σ_2 schedule with $E[C_{max}(\sigma_2)] = 83.34$

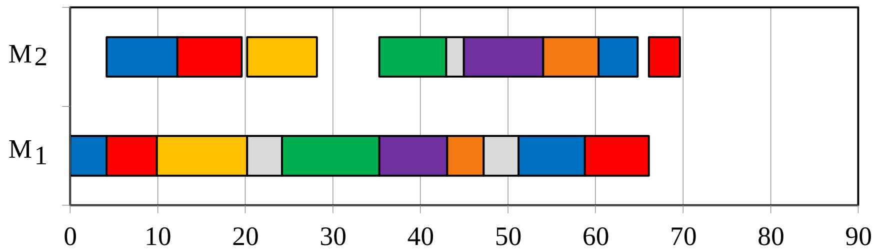

Fig. 3.8 Gantt chart σ_3 schedule with $E[C_{max}(\sigma_3)] = 69.62$

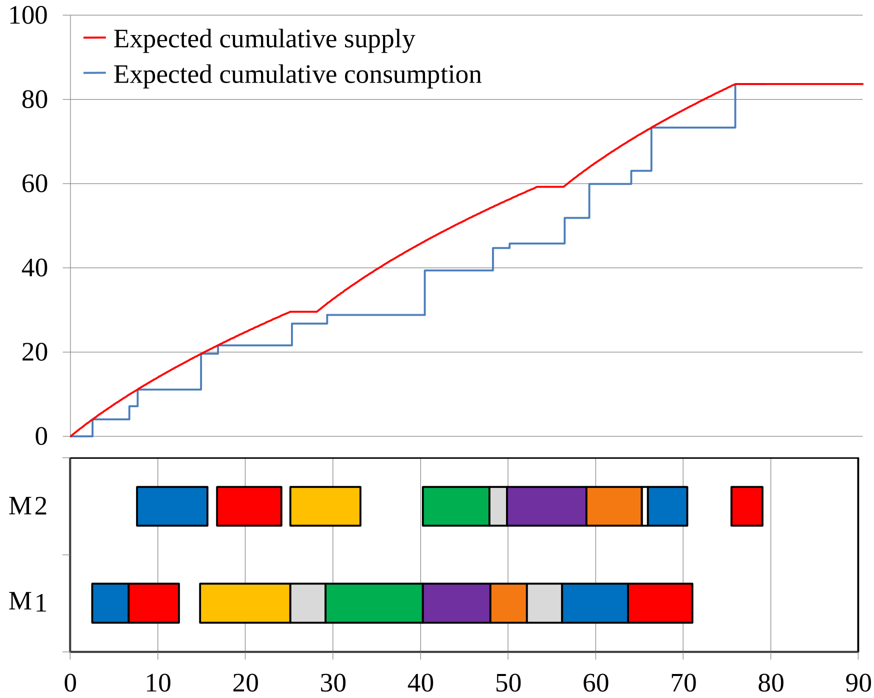

Fig. 3.9 Gantt chart σ_4 schedule with $E[C_{max}(\sigma_4)] = 79.05$

Table 3.1 presents the calculated values of the three performance measures for the three partial schedules and the final integrated schedule.

Table 3.1 The resulting values of the makespan, total maintenance and resource consumption costs for the three partial schedules and for the final one

Schedule	C_{max}	TMC	TRC
σ_1	55		
σ_2	83.34	104,47	
σ_3	69.62	43.79	
σ_4	79.05	83.28	7.03

The problem's objectives exhibit clear antagonism. On one hand, to minimize the makespan, it is necessary to insert PM actions without considering the optimal cost-wise PM interval. This approach could lead to either excessive or insufficient PM and CM actions, resulting in increased maintenance costs. On the other hand, if the goal is to reduce production costs, maintenance tasks must be strategically introduced to achieve an optimal balance between PM and CM costs. The production schedule should satisfy the constraints imposed by maintenance, meaning that jobs need to be grouped efficiently between PM tasks to minimize the earliness and tardiness of PM actions in relation to their optimal dates. Additionally, jobs with high downtime resource consumption rates should be released for processing when machines are in good operating condition and the probability of failure is low to reduce expected additional resource consumption. From the supplier's perspective, the decision would be to decrease the supply rate to reduce the number of maintenance interventions and adjust inventory levels. However, this would result in longer waiting times for resources to be available, potentially impacting the makespan. In summary, the optimization of the production plan involves carefully balancing conflicting objectives related to makespan minimization, production cost reduction, and efficient maintenance planning. The final decisions must take into account these antagonistic aspects to achieve an effective and balanced production schedule.

3.5 Metaheuristic Algorithms for the FMSP-SNR

In this section we describe the various algorithms used to solve FMSP-SNR. First, we describe the bi-objective randomized local search heuristic (BORLS) specifically developed to find a set of near-optimal nondominated solutions to the problem and present the different algorithms used in the solution decoding and evaluation. Then describe the nondominated sorting genetic algorithm and bi-objective particle swarm optimization algorithm and detail the how the solutions are represented and how the various operators and search mechanisms are implemented.

3.5.1 Bi-Objective Randomized Local Search Heuristic (BORLS)

The study introduces a two-phase, bi-objective randomized local search heuristic called BORLS to efficiently solve the problem and obtain a high-quality approximation of the Pareto optimal front within a reasonable computational time. The BORLS heuristic first identifies a high-quality solution for the makespan objective by determining the best resource rate and devising a maintenance plan for supply and production machines. It then utilizes the well-known NEH algorithm to find the best job sequence based on the specified supply and maintenance settings. After obtaining the initial solution, BORLS applies a randomized local search technique using insertion moves to generate neighboring solutions and guide the search towards finding the minimum total production cost solution while trying to remain as close as possible to the Pareto front. The progression of the search depends on the user's preferences regarding the maximum number of generated solutions and the intensity of the local search. These preferences are translated into two parameters that impact the speed and quality of the heuristic.

The first phase of the heuristic commences by establishing the maintenance plan for the production machine. The model assumes that machines can experience multiple failures during job processing, and the time to failure for each machine j follows a Weibull distribution with shape parameter $\beta_j > 1$ and scale parameter η_j. For each machine j, there are two types of maintenance: preventive maintenance and corrective maintenance. Preventive maintenance takes PT_j time and restores the machine to as good as new condition. On the other hand, corrective maintenance takes CT_j time, and it is assumed that the repair is minimal, leaving the machine's age unchanged after any corrective task. By considering continuous machine operation and disregarding scheduling requirements (i.e., the discretization of production into jobs), and treating each machine independently, the aforementioned machine parameters can be used to derive an optimal preventive maintenance planning for each machine. The objective of this planning is to maximize the average availability of each machine.

Equation (3.4) calculates the number of failures $N(T)$ that could occur during a specified operating time T, which represents the PM period for the machine. Now, the objective is to determine the optimal value of T that maximizes the average machine availability. The average renewal cycle consists of T time units of production, followed by $N(T)CT_j$ time units for corrective maintenance and PT_j time units for the preventive maintenance task. After the completion of the PM action, the machine is restored to its original state, and a new cycle begins. The resulting steady-state availability $A_j(T)$ for a PM interval T_j of the machine j is then given by:

$$A(T) = \frac{T}{T + N(T)CT_j + PT_j} \tag{3.76}$$

Using the expression provided, we can derive the optimal PM period T_j^{Cmax} that aims to minimize the makespan by maximizing $A(T)$ as follows:

$$T_j^{Cmax} = \eta_j \left[\frac{PT_j}{CT_j(\beta_j - 1)}\right]^{\frac{1}{\beta_j}} \tag{3.77}$$

Although the derivation of the optimal PM period T_j^{Cmax} ignores the assumptions regarding the discontinuous nature of production and the interdependency between machines, it can still serve as a heuristic method to plan the PM actions for minimizing $Cmax$. After calculating T_j^{Cmax}, the value is adjusted by restricting it to the permissible interval, as specified by constraints (3.20); (3.21).

By pursuing a slightly different approach, we can derive the optimal PM period that minimizes the maintenance cost. This involves replacing PT_j with the preventive maintenance cost PC_j and replacing CT_j with a modified corrective maintenance cost. The modified corrective maintenance cost is a combination of the actual corrective maintenance cost CC_j and the downtime resource cost incurred during the repair time of the machine. The modified corrective maintenance cost MCC_j is expressed as follows:

$$MCC_j = CC_j + aCT_jq \tag{3.78}$$

Therefore, the optimal PM period that targets a minimum total production cost TPC by minimizing the maintenance cost for production machines is:

$$T_j^{TPC} = \eta_j \left[\frac{PC_j}{MCC_j(\beta_j - 1)}\right]^{\frac{1}{\beta_j}} \tag{3.79}$$

Similar to the derivation of T_j^{Cmax}, the derivation of T_j^{TPC} also disregards the assumptions related to the discontinuous nature of production and the interdependency between machines.

Nonetheless, it can serve as a heuristic approach to plan the PM actions for minimizing TPC. The next step involves setting up the resource supply and the PM plan for the supply machine to minimize the makespan. To achieve a minimum makespan, it is intuitive to ensure that the resource supply is sufficient to cope with the high consumption rate. Thus, the goal is to maximize the average resource supply rate. This rate can be quantified by dividing the total expected quantity of consumed resource by the time required to fully supply that quantity, denoted as the stop date of resource supply ST. Since the minimum makespan schedule has not been constructed yet, the total expected quantity of consumed resource cannot be directly computed. As an alternative, we use the total uptime resource consumption quantity TUQ as a lower bound, given by $TUQ = \sum_{i=1}^{n} \sum_{j=1}^{m} Q_{ij}$. With a fixed resource quantity to produce, the objective is to minimize the time required to fully deliver it, i.e., pushing the stop date of resource supply as early as possible by minimizing ST. The equation for ST can be obtained from equation (3.13) by replacing the resource variable r with TUQ. However, this equation is nonlinear and discontinuous, making it challenging to obtain an optimal solution that minimizes ST directly. As a result, a different approach is adopted to quickly obtain a good enough solution. The problem is decomposed into two sub-problems:

1. Finding the optimal PM period that maximizes the steady-state availability of the supply machine for a fixed supply rate.
2. Finding the optimal supply rate that minimizes the stop date of resource supply based on the optimal PM period obtained in the first sub-problem.

The steady-state availability A_{PRP} for a PM interval PRP of the supply machine is given by:

$$A_{PRP} = \frac{\int_0^{PRP} A(t)\,dt}{CT} = \frac{\int_0^{PRP} A(t)dt}{PRP + PRT} \tag{3.80}$$

From equation (3.6) we get:

$$A_{PRP} = \frac{2\left(\sqrt{R^{\omega}\lambda PRP + 1} - 1\right)}{R^{\omega}\lambda(PRP + PRT)} \tag{3.81}$$

For a fixed value of R, differentiation and algebraic analysis results in an optimal PM interval PRP_R^* of:

$$PRP_R^* = \frac{2\sqrt{R^{\omega}\lambda PRT} + R^{\omega}\lambda PRT}{R^{\omega}\lambda} \tag{3.82}$$

Therefore, the problem of determining the optimal value of R, which minimizes the stop date of resource supply, can be reformulated as a continuous optimization problem with a single variable. This problem can be easily solved by selecting an appropriate step size and evaluating the stop date of resource supply for each value of R within the permissible interval. The optimal value of the supply rate is denoted as R^*.

After determining the maintenance planning and the supply rate, the NEH heuristic is employed to discover the optimal job schedule that minimizes the makespan. The evaluation of each subsequence in NEH takes into account the resource constraints, PM planning, and CM interruptions. The resulting schedule represents the first generated solution and serves as the initial point for the local search process. In the second phase, the local search algorithm explores the neighborhood of the initial solution by perturbing it. The neighborhood search solely involves insertion moves, where a randomly selected job is inserted into a randomly chosen position in the sequence. As previously mentioned, the heuristic relies on two parameters: the number of generated solutions and the search intensity, which determines the number of insertion moves between two consecutive generated solutions. The primary objective of the local search algorithm is to find the optimal schedule for the total production cost while systematically exploring the search path for potential nondominated solutions. To achieve this, the trajectory of the local search must always be directed towards the optimal Pareto front. This trajectory is achieved by transforming the bi-objective problem into a single objective problem through the aggregation of the two objectives, using variable weights to guide the algorithm. For each new solution, the weights are updated, and the search proceeds from the last generated solution. Algorithm 3.1 provides the pseudocode outlining the local search algorithm.

Algorithm 3.1 Bi_objective_randomized_local_search

Inputs:	
nb, I values	Input nb the number of solutions and I the local search intensity
for each machine $j = 1{:}m$	
$PMP_j = T_j^{Cmax}$	
Optimize PRP and R initial values	
Apply NEH algorithm to generate the initial solution	
$PRP_0 = PRP$	Memorize PRP
for $i = 1{:}nb$	For each generated solution
$w = (nb - i)/nb$	Update aggregation weights
$R = R^{*}w + R_{min}(1 - w)$	Update supply rate
for each machine $j = 1{:}m$	
$PMP_j = wT_j^{Cmax} + (1 - w)T_j^{TPC}$	Adjust the PM intervals for production machines
Correct PMP_j	Constraints (3.20); (3.21)
$PRP = PRP_R^{*}$	
if $wCmax + (1 - w)TPC$ is less for this solution	
Adjust PRP: $PRP = (PRP_R^{*} + PRP_0)/2$	
$PRP_0 = PRP$	Memorize PRP
else $PRP = PRP_0$	Reject PRP adjustment
$z = wCmax + (1 - w)TPC$	
for $j = 1{:}I$	

Apply a random insertion move to the sequence
if $wCmax + (1 - w)TPC \leq z$ for this sequence
Accept the move — New state of the LS
else Reject the move — Restore old state of the LS
Insert the final solution in the obtained solution set
Disregard dominated solutions

Fig. 3.10 illustrates a typical progression of the BORLS heuristic and elucidates the various steps involved in constructing the nondominated set of solutions.

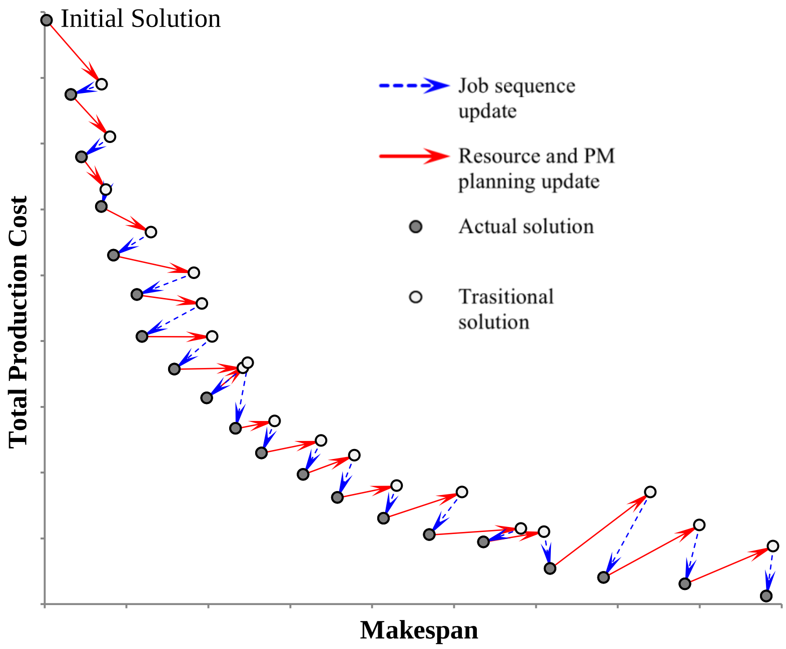

Fig. 3.10 Typical Progression of the BORLS heuristic

Algorithm 3.2 is responsible for evaluating a specific solution and returning its fitness values, which are then used by metaheuristics to search for the optimal Pareto frontier. The algorithm comprises three main steps. The first step (lines 1 – 6) involves constructing the unconstrained schedule σ_3. This is achieved by determining the start times of the various jobs on the machines, denoted as S'_{kj}, and calculating partial values of intermediate variables (e.g., AQ_{kj} and MC_{kj}) that remain invariant with respect to the final schedule σ_4. In the second main part (lines 7 – 13), a list $\mathcal{L}$ is constructed using the Operations-priority-list procedure (Algorithm 3.3), which is then utilized to sort the operations based on their priorities. To enable the comparison between any two operations, the required information is stored in Z. The sorting procedure is designed to leverage the inherent relationships of operation start dates in a permutation flow shop. There are two relationships to be exploited:

$$S_{kj} \leq S_{k+1,j} \tag{3.83}$$

$$S_{kj} \leq S_{k,j+1} \tag{3.84}$$

By leveraging these two relationships, it becomes possible to minimize the number of comparisons required and achieve an efficient solution evaluation process.

Algorithm 3.2 Schedule_evaluation

Inputs:

Problem's Parameters
Job permutation
PM periods
Resource supply rate

$TMC = TRC = 0$ Initialize total costs
for $j = 1{:}m$
  for $k = 1{:}n$
    Compute $\left(Q_{kj} \quad S'_{kj} \quad \overline{Q}_{kj} \quad AQ_{kj} \quad MC_{kj}\right)$
    $TMC = TMC + MC_{kj}$
    $TRC = TRC + q\overline{Q}_{kj}$
$\mathcal{L} =$ Operations_priority_list$\left(Q_{kj} \quad S'_{kj} \quad AQ_{kj}\right)$
$r = 0$ Initialize cumulative consumption count
for $i = 1{:}|\mathcal{L}|$ where $|\mathcal{L}| = nm$
    $r = r + \mathcal{L}[i]_{AQ}$ Update r
    $k = \mathcal{L}[i]_k$ Extract job position index
    $j = \mathcal{L}[i]_j$ Extract machine index
    $r_{kj} = r$
    Compute $t_{kj} = t(r_{kj})$
  for $j = 1{:}m$
    for $k = 1{:}n$
      Compute $\left(S_{kj} \quad C_{max}\right)$
  Compute INV
  Complete TMC by adding supply maintenance cost
  $TRC = TRC + hINV$

In the third step, the final schedule is constructed, and the makespan is computed. To evaluate the resource inventory cost, the same sorting procedure (Algorithm 3.3) is utilized again, but this time, it is employed solely to sort the operations' starting dates. This sorting allows for the calculation of the average resource stock between each consecutive starting date.

Algorithm 3.3 Operations_priority_list

Inputs:

$Q_{kj} \quad S'_{kj} \quad AQ_{kj}$ values

$\mathcal{L} = \emptyset$ Initialize $\mathcal{L}$

$i = y = 1$	Initialize Z counter and memo
$Z = (Q_{k1} \;\; S'_{k1} \;\; AQ_{k1} \;\; k \;\; 1)$	
for $k = 1{:}n$	For machine 1's operations
$\mathcal{L}.add\{Z\}$	Add Z without comparisons
for $j = 2{:}m$	
for $k = 1{:}n$	
$Z = \left(Q_{kj} \;\; S'_{kj} \;\; AQ_{kj} \;\; k \;\; j\right)$	
if $y \neq j$ **then**	If j has been incremented
$i = j$	Start comparing from j^{th} rank
if Rule$(\mathcal{L}[i],\ Z) = "true"$ **then**	
$\mathcal{L}.insert(i,\ Z)$	Insert Z at position i of the list
$i = i + 1$	Increment Z counter
else	
$c = "false"$	Initialize loop stopper
while $c = "false"$	
$i = i + 1$	Check the next element in $\mathcal{L}$
if $i = \lvert\mathcal{L}\rvert$ **then**	
$c = "true"$	$\mathcal{L}$ is fully exhausted
$\mathcal{L}.add\{Z\}$	
else if Rule$(\mathcal{L}[i],\ Z) = "true"$ **then**	
$c = "true"$	Stopping rank reached
$\mathcal{L}.insert(i,\ Z)$	
$i = i + 1$	
$y = j$	Memorize machine counter

3.5.2 Nondominated Sorting Genetic Algorithm (NSGA-II)

NSGA-II is a highly effective multi-objective evolutionary algorithm developed by Deb et al. [161]. Over the past decade, it has gained significant success due to its robustness and efficiency, as evidenced by studies such as [162 – 164]. The algorithm employs concepts of dominance, crowding, and elitism to guide the population towards the Pareto optimal surface and generate multiple diverse solutions in a single run. NSGA-II decomposes the population of solutions into several fronts. The nondominated solutions are assigned rank 1 and form the first front, $\mathcal{F}_1$. Solutions that are only dominated by solutions in the previous fronts, except for solutions in $\mathcal{F}_1$, receive rank 2 and form the second front, $\mathcal{F}_1$. This process continues, and in general, a solution receives rank k if it is only dominated by individuals belonging to the union of fronts $\mathcal{F}_1 \cup \mathcal{F}_2 \cup \cdots \cup \mathcal{F}_{k-1}$. The solutions within each front are then sorted based on their crowding distance. The crowding distance of a solution is determined by the circumference of the hypercube defined by its neighbors and is set to infinity if it is a boundary point. Solutions with higher crowding distance are favored as they introduce more diversity into the population. In each iteration of the algorithm, an offspring population of size N is generated. During the replacement phase, the old and offspring populations are combined and ranked using the two criteria: Nondominance and Crowding. The better half of the combined population forms the new population. A binary tournament selection is employed, wherein two solutions are randomly selected from the population. The nondominance ranking is used to initially choose the best individuals. If both selected solutions belong to the same front, the crowding criterion is then used to select the individual with the largest crowding distance. The following are the different pseudo-codes used in NSGA-II and their description:

The nondominated sorting algorithm is utilized to take a population $\mathcal{P}$ of solutions and break it down into a sorted list of fronts $\mathcal{F} = \{\mathcal{F}_1\ \mathcal{F}_2\ \cdots\}$ where $|\mathcal{F}| \leq |\mathcal{P}|$. However, since there are only two objectives involved, implementing the full version of the nondominated sorting algorithm primarily designed for problems with three or more objectives would not be optimal [86, 106]. Instead, it is more suitable to implement an efficient bi-objective version of the sorting algorithm with a worst-case performance of $O(N^2)$. Basseur et al. [165], described in Algorithm 3.4 below. The parameter ns is used to count the total number of solutions included in the set of fronts $\mathcal{F}$ and to halt the sorting procedure once the number of solutions equals or exceeds half of the population's size after forming and adding a particular front $\mathcal{F}_i$, which will then represent the last front where crowding distance assignment will be applied (Algorithm 3.5).

Algorithm 3.4 Nondominated_sorting

Inputs:
$\mathcal{P}$ population

$N = \lvert\mathcal{P}\rvert$	
$\mathcal{P}.sort\langle f_1, f_2\rangle$	Sort $\mathcal{P}$ according to f_1 then according to f_2 if two solutions have the same f_1 value
$ns = 0$	Initialize total added solutions
$i = 1$	Initialize front counter
while $ns \leq N/2$	
$p = \mathcal{P}.pull[1]$	Pull out highest rank solution
$\mathcal{F}_i.add\{p\}$	
$ns = ns + 1$	Increment ns by one
for each $q \in \mathcal{P}$	
if $(p \nprec q) \Vert (p = q)$ **then**	
$p = \mathcal{P}.pull\{q\}$	Pull out q from $\mathcal{P}$
$\mathcal{F}_i.add\{p\}$	q belongs to the i^{th} front
$ns = ns + 1$	Increment ns by one
$i = i + 1$	Increment front counter by one

Algorithm 3.5 is responsible for assigning a crowding distance value to each solution in a specific population $\mathcal{H}$, which represents a nondominated front. Due to the bi-objective nature of the problem, sorting the population according to each objective (line 3) can be omitted. Thanks to the sorting algorithm, population $\mathcal{H}$ is already sorted in non-decreasing order of f_1 and non-increasing order of f_1. As a result, the crowding distance can be directly assigned to each solution in $\mathcal{H}$ in one pass. To ensure fairness in the comparison of solutions with

different objective scales, normalization is performed using the maximum and minimum values of the objective functions. This normalization is especially crucial when the objectives are on vastly different scales. To address this issue, a uniform dynamic scaling of the objectives is implemented, as proposed by Pedersen & Golberg [166]. The pseudocode for the the NSGA-II is presented in Algorithm 3.6.

Algorithm 3.5 Crowding_distance_computation

Inputs:

$\mathcal{H}$ population

$l = |\mathcal{H}|$
for each objective v
  $\mathcal{H}.sort\langle v \rangle$ — Sort $\mathcal{H}$ according to v value
  $\mathcal{H}[1]_{dist} = \infty$ — Boundary solutions are always selected
  $\mathcal{H}[l]_{dist} = \infty$
  if $l > 2$ **then** — For non-boundary points
    for $i = 2 : (l - 1)$
      $\mathcal{H}[i]_{dist} = \mathcal{H}[i]_{dist} + (\mathcal{H}[i+1].v - \mathcal{H}[i-1].v)/\left(f_v^{max} - f_v^{min}\right)$

Algorithm 3.6 The_procedure_of_NSGA_II

$\mathcal{P}_0.initialize$ — Generate random population
$N = |\mathcal{P}_0|$
$\mathcal{Q}_0 = offspring(\mathcal{P}_0)$
$t = 0$ — Initialize generation counter
while Stopping criterion not satisfied
  $\mathcal{R}_t = \mathcal{P}_t \cup \mathcal{Q}_t$ — Combine parents and offspring
  $\mathcal{F}$ = Nondominated_sorting($\mathcal{R}_t$)
  $\mathcal{P}_{t+1} = \emptyset$ — Initialize next population
  $i = 1$ — Initialize front counter
    while $|\mathcal{P}_{t+1}| + |\mathcal{F}_i| \le N$
      $\mathcal{F}_i$.Crowding_distance_computation
      $\mathcal{P}_{t+1}.addAll\{\mathcal{F}_i\}$ — Add the entire front
      $i = i + 1$
    $\mathcal{F}_i.sort\langle dist \rangle$ — Sort $\mathcal{F}_i$ according to $dist$ value
    $\mathcal{P}_{t+1}.add\{\mathcal{F}_i[1:(N - |\mathcal{P}_{t+1}|)]\}$
    $\mathcal{Q}_{t+1} = offspring(\mathcal{P}_{t+1})$
    $t = t + 1$ — Increment generation counter

In the FMS-SNR, a specific solution is represented by a chromosome with three fields, each encoding a different aspect of the solution: production, maintenance, and supply rate. The production field contains the sequence of job indexes, determining the order in which jobs are processed. The maintenance field represents the preventive maintenance time intervals for both production machines and the supply machine. Since there is only one type of resource considered, the PM period for the supplier machine is encoded in the rightmost gene of this field.

The production field of the chromosome utilizes a two-point permutation crossover operator. To create the first offspring chromosome, the extreme genes from the first parent are copied, and the middle genes are filled with the missing jobs according to their positions (from left to right) from the second parent. Similarly, for the second offspring chromosome, the extreme genes from the second parent are copied, and the middle genes are filled with the missing jobs based on their positions (from left to right) from the first parent. This process is illustrated in Fig. 3.11.

Parent 1	7	6	2	5	1	4	3	8
Parent 2	5	1	4	2	7	8	3	6
Offspring 1	7	5	1	4	2	6	3	8
Offspring 2	5	7	2	1	4	8	3	6

Fig. 3.11 Two point permutation crossover

An insertion mutation operator is applied, which involves removing a randomly chosen gene from the chromosome and inserting it into another randomly selected position. As a result of this mutation, the remaining genes are shifted accordingly to accommodate the inserted gene, as depicted in Fig. 3.12.

Offspring	4	6	5	1	2	3	8	7
***μ* Offspring**	4	6	3	5	1	2	8	7

Fig. 3.12 Insertion mutation

Regarding the maintenance and supply rate fields of the chromosome, a uniform crossover is selected. This crossover method involves combining the alleles of one parent with the corresponding alleles of the other parent. The resulting combination is then rounded to integer values for the maintenance periods, as illustrated in Fig. 3.13.

Parent 1	49	19	33	89
random 1	0.26	0.26	0.49	0.82
Parent 2	28	24	54	71
random 2	0.40	0.46	0.68	0.42
Offspring 1	33	23	44	86
Offspring 2	41	21	47	81

Fig. 3.13 Uniform crossover

3.5.3 Bi-Objective Particle Swarm Optimization (BOPSO)

The second meta-heuristic algorithm used to tackle the problem is the Particle Swarm Optimization algorithm, originally developed by Kennedy and Eberhart in 1995. In contrast to Genetic Algorithms or other meta-heuristics like Simulated Annealing and Ant Colony Optimization, PSO is conventionally applied to single-objective, continuous optimization problems. However, when tackling multi-objective scheduling problems with PSO, certain adaptations and modifications to the fundamental model become essential.

Before implementing a PSO algorithm for the FMS-SNR described in this chapter, three issues need to be addressed. The first issue involves adapting the continuous variables (velocity and position vectors) of the algorithm to represent the job permutation. To achieve this, the authors employ the Smallest Position Value rule (SPV) proposed by Tasgetiren et al. [167], and the solution representation according to this rule is elaborated upon later in the paper. The second and third issues are related to adapting the algorithm for MOPs since these problems lack an absolute global optimum definition. Therefore, new selection and update procedures for $pbest$ and $gbest$ are required to enhance convergence towards better solutions. To preserve the diversity of the population and obtain a good approximation of the Pareto frontier, elitism is applied. The elitist approach involves maintaining a separate domination-free archive containing global best solutions that serve as guides for the rest of the swarm. In each iteration, the entire population of particles is evaluated, and the archive of global best solutions is updated. This updating process involves checking the inclusion of candidate members from the current population and removing dominated members from the archive. Additionally, a maximum size is allotted to the archive, and therefore, the crowding distance mechanism described for the case of NSGA-II is implemented to ensure proper management and maintenance of the archive.

In this study, Algorithm presents the pseudocode for the BOPSO algorithm.

Algorithm The_procedure_of_BOPSO	
$\mathcal{P}.initialize$	Generate random population
$\mathcal{A} = \emptyset$	Initialize archive
$\mathcal{F}$ = Nondominated_sorting($\mathcal{P}$)	
$\mathcal{A} = \mathcal{F}_1$	
$t = 0$	Initialize generation counter
while Stopping criterion not satisfied	
for each $q \in \mathcal{P}$	
assign $gbest_q(\mathcal{A})$	
assign $pbest_q$	
update $v_q,\ x_q$	Update velocity and position
$\mathcal{F}$ = Nondominated_sorting($\mathcal{P}$)	
$\mathcal{A} = Update(\mathcal{A},\ \mathcal{F}_1)$	

As previously mentioned, a particle is characterized by its position in the decision space and its velocity. To represent the particle's position, the maintenance and supply rate values directly indicate the possible PM intervals and supply rates. However, the challenge arises when representing the job-permutation decision. To address this challenge, n continuous position values are employed to indirectly represent the sequence of jobs. The SPV rule is then applied to establish a direct link between the particle's position and the processing sequence. Fig. 3.14 provides an example of the SPV procedure used to construct the corresponding processing sequence. In the example, the smallest position value is 0.85, corresponding to the second job index, thus making job 2 the first job in the sequence. The second smallest position value is 2.26, corresponding to the sixth index, making J6 the second job in the sequence, and so on. In essence, the job indices list is sorted according to the corresponding position values of the particle, resulting in the complete job permutation.

Job indices	J1	J2	J3	J4	J5	J6
Position values	2.47	0.85	3.55	5.57	5.06	2.26
SPV Sequence	J2	J6	J1	J3	J5	J4

Fig. 3.14 Job permutation obtained by SPV rule.

Several methods have been proposed to adapt the PSO algorithm for multi-objective problems. Some of these methods include a dynamic neighborhood strategy suggested by Hu & Eberhart [168], the utilization of crowding distance for global best selection as in Sierra & Coello [169], and ranking methods employed by authors such as Di Pierro et al. [170], Garza-Fabre et al. [171] and Wang & Yang [172]. Additionally, a decomposition-based approach was developed by Zhang & Li [173] for selecting *gbest* and *pbest* solutions. In our study, we have introduced a novel selection method specifically designed for bi-objective optimization problems. Fig. 3.15 illustrates the assignments of *gbest* and *pbest*. The population is initially decomposed using Algorithm 3.1, generating a list of fronts where solutions in each front are sorted based on objective 1. Consequently, each front's first and last solutions represent its two extreme points. For each solution, a global best solution is assigned from the archive solutions. This assignment process is carried out in a simple manner to ensure that the algorithm's complexity and computational effort to select global best solutions do not significantly increase while maintaining population diversity throughout the search.

The selection of the *gbest* and *pbest* for a particle q_{ij} of index j in a front i is done using the following rule:

$$gbest_{ij} \leftarrow q_{\mathcal{A}k} \quad where \quad k = \left\lceil \frac{|\mathcal{A}|}{|\mathcal{F}_i|} j \right\rceil \quad \forall i \tag{3.85}$$

If $pbest_{ij} \prec q_{i-1,k}$

$$pbest_{ij} \leftarrow q_{i-1,k} \quad where \quad k = \left\lceil \frac{|\mathcal{F}_{i-1}|}{|\mathcal{F}_i|} j \right\rceil \quad \forall i > 1 \tag{3.86}$$

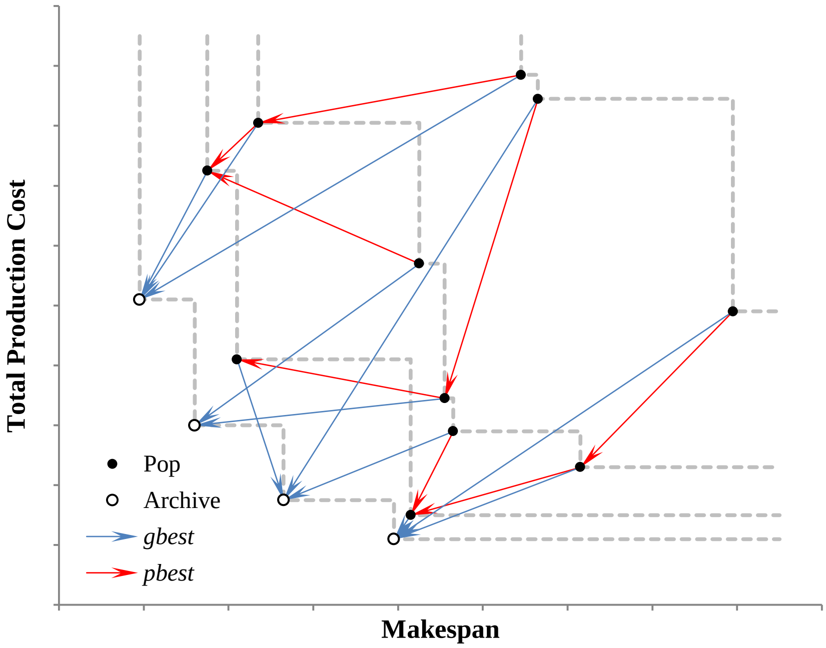

Fig. 3.15 *gbest* and *pbest* update rule

In the first nondominated front, each particle maintains a record of its personal best position. If the current position of the particle dominates its personal best solution, then the new position becomes the particle's *pbest*. By applying this rule, the convergence towards the non-dominated frontier can be improved. Furthermore, solutions in a particular front are influenced by their neighboring points in the subsequent lower ranked nondominated front, both as personal best solutions and global best solutions from the archive. For instance, extreme solutions are attracted to the corresponding extreme solutions in the archive, while the same principle applies to middle points, preventing the loss of good solutions during the process and preserving diversity.

3.6 Experimental Results for the FMSP-SNR

Computational experiments were carried out to assess the performance of the three developed algorithms. All meta-heuristics were implemented in JAVA, and all computations were performed on a PC with an i3 processor clocked at 1.70 GHz and equipped with 4 GB of RAM.

To evaluate the quality of solutions provided by the two solving methods, five performance measures were used: the C metric, hypervolume indicator HV, the inverted generational distance measure IGD, the number of obtained solutions, and the computational time.

To evaluate the significance of the difference between the results obtained by the two algorithms, the Mann Whitney U test was employed. This statistical significance test is nonparametric and does not require any assumptions about the distribution of the samples. It allows us to determine, with a specified confidence level, whether two independent samples were selected from the same probability distribution (null hypothesis) or from different distributions (alternative hypothesis). When using the U test, two general assumptions are made: the observations within each sample are independent, and the observations are ordinal.

3.6.1 Test Problems and Algorithm Setup

The computational experiment involves generating eight (n, m) permutation flow shop problems, where n represents the number of jobs, and m represents the number of machines. The proposed problem set is as follows: (15, 3), (20, 5), (30, 5), (40, 7), (50, 10), (60, 10), (80, 20), and (100, 20). The data parameters' values were generated using the formulas mentioned in Table 3.2 below.

Table 3.2 Benchmark used for problem data generation

Parameter	Generation formula
P_{ij}	U [1, 100]
Q_{ij}	U [1, 100]
PT_j	U [1, 100]
CT_j	$PT_j + U$ [1, 400]
PC_j	U [1, 200]
CC_j	$PC_j + U$ [1, 800]
θ_j	U [1000, 2000]
β_j	U (1, 4.5)
PRT	U [1, 100]
CRT	$PRT + U$ [1, 400]
PRC	U [1, 200]
CRC	$PRC + U$ [1, 800]
λ	U (5E-5, 1.5E-4)
ω	U (2, 4.5)
q	U (2, 10)
h	U (0.05, 0.1)

The values of the supply rate are constrained within the range of 1 to 2. Regarding the preventive maintenance interval for the supply machine, the maximum (and minimum) values are determined by maximizing the availability when a minimum (and maximum) rate is used. Formulae for these limits are given in the equations below:

$$T_{max} = \frac{2\sqrt{R_{min}^{\omega}\lambda PRT} + R_{min}^{\omega}\lambda PRT}{R_{min}^{\omega}\lambda} \tag{3.87}$$

$$T_{min} = \frac{2\sqrt{R_{max}^{\omega}\lambda PRT} + R_{max}^{\omega}\lambda PRT}{R_{max}^{\omega}\lambda} \tag{3.88}$$

The stopping criterion for both algorithms is the maximum number of iterations, which remains the same. Additionally, to ensure a fair comparison in terms of computational time, the same population size was used.

Since the performance of a meta-heuristic is greatly influenced by its parameter settings, pilot experiments were initially conducted to identify potential values of these parameters where the algorithms exhibit promising performance. Subsequently, an extensive exploration of different combinations of parameter values was carried out across various problem sets.

To compare the different parameter combinations, the hypervolume indicator, which combines both convergence and diversity aspects, was adopted as the performance criterion. The reference point was obtained by selecting the maximum values for both objectives (Nadir point) over all trials. The values of HV were averaged over all trials, and then they were averaged over all test problems, following the approach used in [95]. The combination of parameter values that yielded the highest HV value was selected as the final constant parameter settings to be used in all experiments. The potential parameter values and the best values of each parameter (highlighted in bold type) are listed below:

NSGA-II
Crossover probability: 0.5, 0.7, 0.9 **(0.9)**
Mutation probability: 0.05, 0.2, 0.35 **(0.2)**

BOPSO
Inertia weight: 0.45, 0.7, 0.95 **(0.7)**
Cognitive factor: 0.4, 0.7, 1 **(0.4)**
Social factor: 0.2, 0.6,1 **(0.6)**
Mutation probability: 0.05, 0.15, 0.3 **(0.05)**

3.6.2 Computational Results from the MINLP

Before conducting computational experiments to assess the performance difference between the NSGA-II and BOPSO algorithms, the developed MINLP model was utilized to generate optimal solutions for some instances using the Lingo solver. However, due to the high complexity and nonlinearity of the model, the solver was unable to obtain solutions in a reasonable time, even for small instances with only 3 jobs and 2 machines, unless the supply rate and the supply machine's maintenance period decisions were fixed. As a result, the solver was used to compare the performance of the developed metaheuristics against Pareto optimal solutions found by Lingo solver for a specific instance with 3 jobs and 2 machines, while keeping the supply rate fixed to a constant value and fixing the maintenance period for the supply machine. Despite the lengthy computational time required by the solver to find a single Pareto optimal solution (averaging 1.5 hours per solution), it successfully generated the complete Pareto optimal frontier consisting of 7 solutions. The two metaheuristic algorithms (NSGA-II and BOPSO) were then run 20 times each on the same test instance. The non-dominated sets obtained from both algorithms were reported in less than 0.15 seconds each.

The results are presented in Fig. 3.16, depicting the tradeoff frontier between optimal makespan and total production cost, along with the solutions obtained from the two algorithms, including their percentage of occurrence in the reported nondominated fronts.

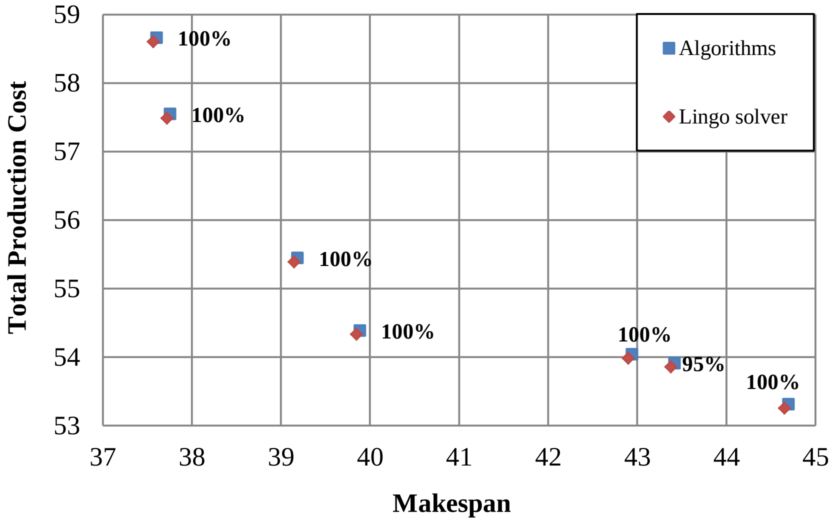

Fig. 3.16 Solution sets obtained from Lingo solver and both NSGA-II and BOPSO algorithms

The figure clearly shows that the algorithms consistently found 6 out of the Pareto optimal solutions in all cases (100% success rate), and in 95% of the cases, they also discovered one additional Pareto optimal solution. Moreover, all the generated nondominated sets exclusively comprised optimal Pareto solutions. In essence, the algorithms never reported any suboptimal solutions in any of the runs.

To gain insights into the mathematical model's capabilities and to provide a more comprehensive assessment of the developed metaheuristics' performance, the runtime of the Lingo solver was extended to 8 hours per solution, which is equivalent to one shift in the industry. In this setting, solutions within 1% of the lower bound were considered as optimal solutions, and all decision variables were allowed to be optimized by the Lingo solver. In contrast to the previous experiment where the supply rate and preventive maintenance interval for the supply machine were fixed, in this scenario, the optimal Pareto frontier becomes infinite due to the continuous variables R and PRP. Consequently, for each problem in this experiment, the solver was used to generate only 10 Pareto sufficient solutions. The process began by finding the minimum makespan solution and the minimum total production cost solution. Subsequently, the ε-constraint method was employed to generate the other 8 solutions, gradually relaxing an additional constraint on the TPC and minimizing the $Cmax$ at each step. Similar to the previous example experiment, the two metaheuristic algorithms (NSGA-II and BOPSO) were executed 20 times each on each instance. The nondominated sets obtained from both NSGA-II and BOPSO were then combined, and the best 10 solutions from the overall fronts, determined based on the crowding distance measure, were compared against the optimal front obtained by Lingo.

Table 3.3 displays the results for the C and IGD metrics, which compare the overall fronts obtained by combining the pairs of fronts from both metaheuristic algorithms, with the optimal fronts obtained from Lingo.

Table 3.3 Minimum, average and maximum values of the C and IGD metrics comparing the combined metaheuristic fronts against the optimal fronts from Lingo.

	C (OPT, Algorithms)			IGD (Algorithms)		
(n, m)	Min	Avr.	Max	Min	Avr.	Max
(3, 2)	0	0.01	0.05	0	0.01	1.259
(3, 2)	0	0.005	0.05	0	0.007	1.48
(3, 2)	0	0.015	0.05	0	0.045	3.177
(4, 2)	0	0.005	0.05	0	0.009	1.708
(4, 2)	0	0.005	0.05	0	0.01	2.043
(4, 2)	0	0	0	0	0	0
(5, 2)	0	0.015	0.05	0	0.033	3.338
(5, 2)	0	0.025	0.05	0	0.057	4.26
(5, 2)	0	0.02	0.05	0	0.066	4.275

The Lingo solver successfully generated optimal solutions for instances with up to 5 jobs and 2 machines. However, instances with 3 machines or 6 jobs with 2 machines were not solvable within the specified runtime. Similar to the previous single instance example where R and PRP variables were fixed, the metaheuristic algorithms exhibited excellent performance. In the second instance with 5 jobs, where the algorithms performed relatively worse compared to other instances, one solution from the overall front was dominated by the optimal front in 5 different occasions. Despite this, the average IGD values for all instances are very low, indicating a satisfactory distribution of the solutions.

3.6.3 Computational Results from NSGA-II and BOPSO

During the computational experiment, both algorithms were executed 30 times on each of the eight test problems. The same set of parameter values was used for each trial across all the test problems. For every trial, a new instance of problem data was generated to ensure diversity in the experiment.

Table 3.4 presents the minimum, average, and maximum values of the C metric obtained for each test problem. The maximum values indicate that the BOPSO algorithm was able to entirely dominate NSGA-II at least once over the 30 trials for all problems, while NSGA-II was able to entirely dominate BOPSO only once, specifically in the case of the (40, 7) test problem. The p-values of the U test suggest that neither of the two methods significantly outperforms the other with a 99% confidence level for the first four problems. However, for the second half of the problem set, BOPSO overwhelmingly outperforms NSGA-II, especially when the problem size increases.

Table 3.4 Minimum, average and maximum values of the two C metric couples along with the associated p-values for each test problem.

	C (NSGA-II, BOPSO)			C (BOPSO, NSGA-II)			
(n, m)	Min	Avr.	Max	Min	Avr.	Max	p-val
(15, 3)	0	0.28	0.79	0	0.44	1	0.01
(20, 5)	0	0.35	0.84	0	0.3	1	0.19
(30, 5)	0	0.45	0.94	0	0.3	1	0.02
(40, 7)	0	0.36	1	0	0.37	1	0.45
(50, 10)	0	0.21	0.84	0	0.68	1	<0.01
(60, 15)	0	0.05	0.31	0.29	0.85	1	<0.01
(80, 20)	0	0.01	0.15	0.66	0.98	1	<0.01
(100, 20)	0	0.01	0.14	0.88	0.99	1	<0.01

Table 3.5 provides a summary of the results obtained for the IGD measure. It is evident that NSGA-II yields significantly better results, in terms of average values, for the first four problems. This can be attributed to the large number of solutions produced by the NSGA-II algorithm, as shown in Table 3.5. The abundance of solutions in NSGA-II's set suggests that it is better distributed and closer to the reference set. However, it is important to note that most of the reference set is composed of solutions from NSGA-II for this set of problems, as NSGA-II generates more solutions and the C metric values are relatively similar. On the other hand, BOPSO consistently achieves the minimum value of 0 (indicating full domination) at least once in every test problem, while NSGA-II reaches that value only in problem (40, 7), aligning with the findings of the C metric. Additionally, BOPSO delivers significantly better (lower) values for the IGD measure, particularly for large-sized problems.

Table 3.5 Minimum, average and maximum values of the IGD measure of the two algorithms along with the associated p-values for each test problem.

	IGD (NSGA-II)			IGD (BOPSO)			
(n, m)	Min	Avr.	Max	Min	Avr.	Max	p-val
(15, 3)	0.07	4.12	37.5	0	8.2	30.82	<0.01
(20, 5)	0.18	9.05	149	0	11.47	34.6	<0.01
(30, 5)	0.03	4.33	51.9	0	9.14	41.8	<0.01
(40, 7)	0	3.12	15.8	0	5.34	18.97	<0.01
(50, 10)	0.18	7.72	47.5	0	3.21	14.94	0.01
(60, 15)	0.82	12.9	54.2	0	1.2	7.74	<0.01
(80, 20)	7.91	31.7	116	0	0.2	2.48	<0.01
(100, 20)	8.93	28.8	100	0	0.14	1.66	<0.01

Table 3.6 presents the minimum, average, and maximum number of obtained solutions for both algorithms in each test problem. A notable difference between the two algorithms can be observed regarding this criterion. NSGA-II is able to evolve a large proportion of its

population to form a nondominated front compared to the BOPSO algorithm. This can be attributed to the genetic selection mechanism in NSGA-II, which favors low rank solutions for the next iteration, enabling it to produce a high number of solutions. However, this approach may lead to a slower convergence rate and difficulties in escaping local optima. On the other hand, BOPSO employs a few solutions stored in the archive to guide the swarm towards global best solutions through a social learning process. The inertial weight allows dominated particles to explore around the archived solutions they are attracted to, thereby incrementally improving the quality of solutions in that vicinity before potentially escaping local optima. The cognitive weight then decelerates the particle's speed, leading it back towards its global and personal best solutions, and the process repeats. Essentially, BOPSO utilizes the entire swarm to enhance the quality of a select few solutions. This strategy aids in achieving fast convergence while avoiding local optima. The evidence of this behavior is evident in the progression of the average number of solutions with the size of the problem. BOPSO's approach demonstrates its ability to efficiently explore and exploit the solution space, leading to improved performance for larger problem sizes.

The values obtained by NSGA-II show a decreasing trend as the number of jobs and machines increase, indicating a lack of convergence. On the other hand, the values obtained by BOPSO remain stable, with an average of around 18 obtained solutions, showcasing its ability to maintain a consistent performance across different problem sizes.

Table 3.6 Minimum, average and maximum values of the number of obtained solutions by the two algorithms for each test problem.

	NSGA-II			BOPSO		
(*n*, *m*)	Min	Avr.	Max	Min	Avr.	Max
(15, 3)	85	99.4	100	10	21.8	34
(20, 5)	49	97.2	100	6	16.67	33
(30, 5)	40	96	100	7	16.17	26
(40, 7)	34	95.3	100	10	16.47	26
(50, 10)	28	84.1	100	9	16.7	25
(60, 15)	26	55.5	100	13	18.1	26
(80, 20)	17	38.5	75	14	19.63	27
(100, 20)	20	51.9	121	13	23.6	30

Table 3.7 presents the average computational time in seconds required by both algorithms to reach the stopping criterion. The results show that NSGA-II requires slightly less computational time compared to BOPSO, with BOPSO taking approximately 21% more time on average. However, it is worth noting that this percentage tends to decrease as the problem size increases.

Table 3.7 Average values of the computational time in seconds spent by the two algorithms for each test problem.

(*n*, *m*)	NSGA-II	BOPSO

(15, 3)	0.67	0.83
(20, 5)	1.37	1.67
(30, 5)	2.08	2.59
(40, 7)	4.06	5.02
(50, 10)	7.72	9.42
(60, 15)	16.08	19.14
(80, 20)	33.06	38.68
(100, 20)	96.84	112.22

Table 3.8 displays the average values of the hypervolume indicator HV for the two solution sets obtained by the NSGA-II and BOPSO algorithms for each test problem. In this case, the Wilcoxon signed-rank test was utilized to assess the significance of the difference between the two populations of HV values, as the samples depend on the problem data specification and are treated as matched samples. The Wilcoxon signed-rank test is a nonparametric statistical hypothesis test that helps determine whether two dependent samples are drawn from populations with the same probability distribution. The average hypervolume values and the p-values of the Wilcoxon signed-rank test support the earlier findings and confirm that BOPSO consistently outperforms NSGA-II, particularly as the problem size increases.

Table 3.8 Average values of the hypervolume covered by the two solution sets obtained by the two algorithms along with the associated *p*-values for each test problem.

	HV (NSGA-II)	HV (BOPSO)	
(*n*, *m*)	Avr.	Avr.	*p*-val
(15, 3)	1.84E+05	1.86E+05	0.3
(20, 5)	3.79E+05	3.62E+05	0.13
(30, 5)	9.52E+05	8.99E+05	<0.01
(40, 7)	2.99E+06	3.00E+06	0.5
(50, 10)	6.15E+06	6.68E+06	<0.01
(60, 15)	1.97E+07	2.33E+07	<0.01
(80, 20)	6.65E+07	9.56E+07	<0.01
(100, 20)	1.16E+08	1.51E+08	<0.01

3.6.4 Computational Results from BORLS

In this set of experiments, we aimed to assess the effectiveness of the proposed BORLS in improving the performance of both the NSGA-II and BOPSO algorithms. The goal was to compare the enhanced algorithms, where 5% of the population was generated using the BORLS heuristic and the rest (95%) was generated randomly, against the algorithms with purely random populations. To ensure fairness, each algorithm was allotted the same computational time and population size. To determine the proportion of the heuristic-generated population and the local search intensity, we conducted pilot experiments to find the optimal combinations that yielded the best results with minimal computational overhead. In the enhanced algorithms, the local search intensity allowed for $3n$ possible moves per solution generation. Next, we present the results comparing the performances of the algorithms with 5% heuristically generated initial populations and those with purely random populations. This comparison will help us understand the impact of using the BORLS to enhance the initial population and guide the search process.

Table 3.9 and Table 3.10 present the results of the computational experiments, including the computational time T in seconds, the number of obtained solutions Nb from the enhanced algorithms A_H and the standard algorithms A_R, the C metric values, HV, the IGD measure, and the statistical significance test results for each performance measure. The *p*-values for samples where the test indicates a significant difference between the means with a 99% confidence level are shown in bold type. The comparison results clearly indicate that the algorithms with 5% heuristically generated initial populations outperform the standard algorithms with 100% randomly generated initial populations. This performance difference becomes more significant as the problem size increases. For instance, the enhanced NSGA-II completely dominates the standard one in problems (80, 20) and (100, 20). Moreover, the number of obtained solutions is greater in the case of the enhanced NSGA-II algorithm, especially for large problems, where the size of the nondominated set obtained from the enhanced algorithm is 37% larger for problems with 100 jobs and 20 machines. Conversely, the number of generated solutions for BOPSO remains more or less the same across different problem sizes.

In the second set of experiments, we aimed to assess the effectiveness of the enhanced algorithms and the BORLS as a separate search algorithm. Similar to the first set of experiments, each algorithm was given an equal runtime for fair comparison. The results in Table 3.11 compare the performances of the BORLS heuristic (L), the enhanced NSGA-II (E1), and the enhanced BOPSO algorithm (E2). The pairwise statistical significance tests results used to compare the performances of each pair of algorithms for each performance measure are summarized in Table 3.12. Red bold type indicates that the first algorithm is significantly superior, while blue bold type indicates that the second algorithm is significantly superior. The comparisons between the two enhanced algorithms (E1 and E2) show that NSGA-II performs better than BOPSO in small- and medium-sized problems, particularly in terms of the IGD and C measures. This is mainly due to NSGA-II's ability to generate a large number of solutions, which helps it cover a wider area of the search space. However, this capability also slows down the convergence of NSGA-II and makes it more likely to get stuck in local optima, as observed in the earlier experiments with purely random initial populations. With the enhancement of the initial population using the BORLS heuristic, the algorithms start closer to the optimal front and achieve lower values for IGD and C performance measures. This advantage, however, becomes less evident as the size of the problem increases (for instance, in problems 80, 20 and 100, 20) and the initial distance from the optimal front becomes greater. Overall, the enhanced algorithms show promising results, and the BORLS heuristic proves to be an effective approach for improving the performance of the algorithms in multi-objective optimization problems.

Table 3.9 Results from enhanced NSGA-II and random population NSGA-II algorithms.

Problem		T(s)	$Nb(A_H)$	$Nb(A_R)$	$C(A_H, A_R)$	$C(A_R, A_H)$	$HV(A_H)$	$HV(A_R)$	$IGD(A_H)$	$IGD(A_R)$	pv(C)	pv(HV)	pv(IGD)
	Max.		100	100	1	1	5.03E+06	4.92E+06	17.21	215.87			
(15, 3)	Avrg.	0.72	97.20	98.27	0.53	0.30	1.13E+06	1.08E+06	4.80	16.75	**0.0031**	**0.0015**	**0.0085**
	Min.		20	54	0	0	4.00E+04	3.31E+04	0	0			
	Max.		100	100	1	1	3.22E+07	3.21E+07	19.92	52.40			
(20, 5)	Avrg.	1.49	96.60	92.47	0.52	0.38	6.21E+06	6.11E+06	4.15	11.16	0.0597	0.0578	0.022
	Min.		56	15	0	0	2.43E+05	2.20E+05	0	0			
	Max.		100	100	1	0.92	9.36E+07	9.24E+07	18.37	34.58			
(30, 5)	Avrg.	2.36	98.33	93.43	0.78	0.13	2.22E+07	2.06E+07	1.53	9.05	**2E-09**	**1.94E-06**	**9E-09**
	Min.		80	10	0.03	0	1.09E+06	1.21E+06	0	0			
	Max.		100	100	1	0.88	3.81E+08	3.51E+08	4.83	45.83			
(40, 7)	Avrg.	4.73	96.70	91.57	0.85	0.10	6.52E+07	5.99E+07	0.57	13.68	**5E-11**	**9.86E-06**	**2E-11**
	Min.		37	22	0.07	0	4.76E+06	3.14E+06	0	1.49			
	Max.		100	100	1	0.99	9.68E+08	9.67E+08	2.72	71.33			
(50, 10)	Avrg.	9.25	91.30	90.73	0.82	0.14	2.38E+08	2.17E+08	0.43	14.18	**2E-10**	**1.59E-06**	**5E-11**
	Min.		33	41	0.03	0	6.58E+07	4.80E+07	0	0.01			
	Max.		100	100	1	0.68	2.50E+09	2.29E+09	1.62	51.99			
(60, 15)	Avrg.	19.07	87.80	79.37	0.93	0.05	9.50E+08	8.38E+08	0.12	12.67	**7E-13**	**8.67E-07**	**3E-12**
	Min.		44	26	0	0	1.57E+08	1.21E+08	0	1.60			
	Max.		100	100	1	0	1.52E+10	1.31E+10	0	36.49			
(80, 20)	Avrg.	42.15	94.97	85.23	1	0	7.83E+09	6.65E+09	0	13.16	**8E-15**	**8.67E-07**	**6E-13**
	Min.		59	30	1	0	2.79E+09	1.74E+09	0	3.18			
	Max.		150	150	1	0	2.75E+10	2.25E+10	0	24.25			
(100, 20)	Avrg.	118.56	140.57	102.30	1	0	1.18E+10	9.74E+09	0	14.16	**8E-15**	**8.67E-07**	**6E-13**
	Min.		85	37	1	0	3.50E+09	2.58E+09	0	6.28			

Table 3.10 Results from enhanced BOPSO and random population BOPSO algorithms.

problem		T(s)	$Nb(A_H)$	$Nb(A_R)$	$C(A_H, A_R)$	$C(A_R, A_H)$	$HV(A_H)$	$HV(A_R)$	$IGD(A_H)$	$IGD(A_R)$	pv(C)	pv(HV)	pv(IGD)
	Max.		100	100	1	1	4.55E+06	4.53E+06	15.72	166.73			
15,3	Avrg.	0.92	48.47	42.03	0.59	0.24	1.53E+06	1.49E+06	3.61	14.68	**0.0001**	0.0204	**0.0017**
	Min.		12	14	0	0	1.82E+05	1.82E+05	0	0			
	Max.		56	85	1	1	1.28E+07	1.29E+07	18.28	62.53			
20,5	Avrg.	1.86	29.27	27.43	0.40	0.47	5.37E+06	5.45E+06	3.93	6.44	0.1913	0.1105	0.2166
	Min.		5	9	0	0	4.97E+05	4.20E+05	0	0			
	Max.		55	42	1	1	9.75E+07	9.74E+07	19.57	23.70			
30,5	Avrg.	2.94	24.93	24.27	0.54	0.38	2.15E+07	2.14E+07	3.47	4.96	0.0526	0.1401	0.0425
	Min.		10	12	0	0	3.68E+06	3.60E+06	0	0			
	Max.		47	47	1	0.95	1.67E+08	1.70E+08	6.21	13.66			
40,7	Avrg.	5.91	28.20	30.50	0.56	0.34	7.21E+07	7.05E+07	1.82	4.40	**0.0038**	**1.58E-03**	**0.0006**
	Min.		14	16	0.03	0	1.86E+07	1.76E+07	0	0.42			
	Max.		51	48	1	0.86	1.67E+09	1.69E+09	5.32	13.17			
50,10	Avrg.	11.03	31.87	32.60	0.68	0.24	2.95E+08	2.90E+08	1.20	5.06	**2E-06**	**2.50E-03**	**1E-07**
	Min.		22	19	0	0	4.45E+07	3.96E+07	0	0.85			
	Max.		63	59	1	1.00	5.35E+09	5.59E+09	7.03	18.47			
60,15	Avrg.	22.36	44.60	45.97	0.78	0.18	1.57E+09	1.54E+09	0.66	4.79	**6E-08**	**3.58E-04**	**1E-08**
	Min.		22	32	0	0	4.20E+08	4.04E+08	0	0.00			
	Max.		93	75	1	0.53	1.85E+10	1.71E+10	1.31	24.16			
80,20	Avrg.	48.03	58.60	56	0.88	0.07	7.51E+09	7.18E+09	0.18	6.14	**8E-12**	**1.44E-06**	**8E-12**
	Min.		35	35	0.18	0	1.95E+09	1.82E+09	0	1.66			
	Max.		119	101	1	0.53	3.34E+10	3.20E+10	1.99	33.42			
100,20	Avrg.	131.03	66	70.30	0.93	0.05	1.25E+10	1.20E+10	0.13	6.91	**3E-12**	**9.60E-07**	**8E-12**
	Min.		34	51	0.17	0	1.47E+09	1.30E+09	0	0.73			

Table 3.11 Results from BORLS, enhanced NSGA-II and enhanced BOPSO algorithms.

Problem		T(s)	Nb(L)	Nb(E1)	Nb(E2)	HV(L)	HV(E1)	HV(E2)	IGD(L)	IGD(E1)	IGD(E2)	C(LS, E1)	C(E1, L)	C(LS, E2)	C(E2, L)	C(E1, E2)	C(E2, E1)
	Max.		33	100	27	4.20E+06	5.67E+06	5.73E+06	318.06	36.26	46.56	1	1	0.74	1	1	0.97
(15, 3)	Avrg.	0.28	17.17	79.53	16.50	1.28E+06	1.45E+06	1.45E+06	27.23	8.01	14.67	0.18	0.65	0.24	0.38	0.35	0.31
	Min.		4	16	3	1.32E+05	1.41E+05	1.56E+05	4.17	0.25	3.07	0	0	0	0	0	0
	Max.		32	100	34	2.76E+07	3.27E+07	3.20E+07	142.47	47.59	19.17	0.98	1	0.77	1	1	0.99
(20, 5)	Avrg.	0.76	18.40	86.63	19.93	6.83E+06	7.51E+06	7.30E+06	20.62	6.51	8.44	0.19	0.57	0.34	0.36	0.56	0.23
	Min.		6	27	8	7.34E+05	8.67E+05	1.10E+06	3.31	0.20	1.21	0	0	0	0.06	0	0
	Max.		30	100	38	5.26E+07	5.30E+07	5.27E+07	23.22	11.59	22.34	0.98	1	0.76	0.47	1	0.93
(30, 5)	Avrg.	1.82	19.43	91.33	22.73	1.72E+07	1.76E+07	1.72E+07	7.83	4.37	8.44	0.28	0.46	0.46	0.12	0.54	0.18
	Min.		7	29	11	2.64E+06	3.21E+06	2.35E+06	3.04	0.85	3.01	0	0	0.09	0	0	0
	Max.		37	100	44	4.28E+08	4.45E+08	4.41E+08	17.09	12.25	23.32	0.85	1	0.86	0.69	1	0.64
(40, 7)	Avrg.	4.60	23.37	92.37	27.27	9.01E+07	9.40E+07	9.24E+07	6.95	3.20	6.69	0.24	0.54	0.49	0.17	0.64	0.16
	Min.		12	18	13	9.26E+06	8.61E+06	9.27E+06	1.87	0.44	2.32	0	0	0.18	0	0.07	0
	Max.		47	100	51	1.01E+09	1.06E+09	1.04E+09	14.94	5.37	8.27	0.84	0.89	0.81	0.52	1	0.67
(50, 10)	Avrg.	11.29	30.70	96.07	37.10	3.83E+08	4.02E+08	4.00E+08	5.93	1.99	4.09	0.27	0.46	0.57	0.17	0.74	0.14
	Min.		12	52	24	3.29E+07	3.31E+07	3.12E+07	1.77	0.26	1.66	0	0.04	0.22	0	0.36	0
	Max.		49	100	67	2.81E+09	2.83E+09	2.73E+09	8.55	6.80	7.80	0.91	0.775	0.94	0.66	1	1
(60, 15)	Avrg.	26.94	36.67	96.30	44.50	1.07E+09	1.11E+09	1.11E+09	4.04	2.78	3.37	0.43	0.37	0.57	0.23	0.59	0.32
	Min.		25	36	28	2.28E+08	2.17E+08	2.15E+08	0.69	0.29	1.24	0.01	0	0.14	0	0	0
	Max.		64	100	82	1.81E+10	1.81E+10	1.71E+10	6.62	6.25	11.20	0.99	0.83	1	0.85	1	1
(80, 20)	Avrg.	76.25	49.33	99.83	61.47	6.80E+09	6.95E+09	6.92E+09	2.97	2.54	2.98	0.57	0.28	0.60	0.29	0.41	0.52
	Min.		33	96	40	3.12E+09	3.12E+09	3.05E+09	0.31	0.21	0.23	0	0	0.08	0	0	0
	Max.		62	150	82	3.00E+10	3.12E+10	3.08E+10	5.20	7.56	9.08	0.96	0.70	1	0.78	0.99	1
(100, 20)	Avrg.	123.43	49.00	137.07	59.83	1.11E+10	1.13E+10	1.13E+10	2.62	1.96	2.92	0.58	0.29	0.67	0.24	0.53	0.39
	Min.		33	45	39	1.06E+09	1.04E+09	9.85E+08	0.14	0.21	0.29	0.14	0	0.17	0	0	0

Table 3.12 Pairwise statistical significance test for BORLS, enhanced NSGA-II and enhanced BOPSO algorithms.

Problem	HV(L, E1)	HV(L, E2)	HV(E1, E2)	IGD(L, E1)	IGD(L, E2)	IGD(E1, E2)	C(L, E1)	C(L, E2)	C(E1, E2)
(15, 3)	6.3E-05	5.2E-04	0.4468	7.7E-05	0.0992	4.4E-04	7.3E-07	0.0382	0.2388
(20, 5)	3.8E-05	0.0062	0.0184	4.3E-06	0.0149	1.7E-04	5.1E-06	0.3865	3.3E-04
(30, 5)	3.6E-04	0.3906	0.0052	2.3E-04	0.3894	1.7E-04	0.0041	1.4E-09	4.9E-07
(40, 7)	8.7E-05	0.0036	0.0011	6.9E-06	0.0801	3.7E-05	2.7E-04	1.1E-08	3.3E-09
(50, 10)	4.2E-06	1.4E-04	0.1029	1.9E-08	0.0110	1.7E-06	0.0023	2.2E-10	7.1E-11
(60, 15)	4.9E-05	4.8E-04	0.4387	0.0023	0.0345	0.0917	0.2529	2.1E-07	0.0038
(80, 20)	7.4E-05	0.0150	0.4795	0.0917	0.2722	0.2436	8.1E-05	3.8E-05	0.1534
(100, 20)	0.0018	0.0225	0.3366	0.0143	0.4823	0.0114	3.3E-05	3.7E-08	0.0519

Finally, the performance of the heuristic algorithm is compared with the two enhanced metaheuristics. Regarding the pair (L, E1), which represents the comparison between BORLS and NSGA-II, the results show that the latter algorithm outperforms BORLS in terms of hypervolume for all test problems and in terms of IGD and C measures for small- and medium-sized problems. However, interestingly, for problems (80, 20) and (100, 20), the fronts obtained from BORLS significantly dominate those obtained from NSGA-II despite the lower number of generated solutions. It's important to note that the IGD and HV measures are more sensitive to the number of solutions in the front than the C metric. Having more solutions in the nondominated front results in a lower IGD value due to the denser distribution of solutions and a higher covered hypervolume due to the higher resolution of the front. Therefore, the difference in the results given by the three performance measures can be explained by the fact that NSGA-II is capable of generating more nondominated solutions, which leads to its better performance in the IGD and HV measures. Overall, these results highlight the strengths and weaknesses of each algorithm, and they demonstrate that the BORLS heuristic can be a valuable addition to metaheuristic algorithms, particularly in situations where it effectively guides the search towards optimal solutions despite generating fewer solutions compared to other algorithms. The comparison between the BORLS and the enhanced BOPSO algorithm, represented by the pair (L, E2), reveals that the BORLS has an advantage in small- and medium-sized problems. However, for large problems, BORLS maintains domination over BOPSO in terms of the C metric. To provide a clearer understanding of the differences in performances between the algorithms, a few samples of the nondominated fronts obtained from each algorithm are displayed for problems (60, 15), (80, 20), (100, 20), and two additional problems (150, 35) and (200, 40) in Figures 3.17, 3.18, 3.19, 3.20, and 3.21, respectively. The total production cost (TPC) axis is represented using a logarithmic scale with base 2. Fronts A1 and A2 represent solutions obtained from the standard NSGA-II and BOPSO, respectively. From the figures, it is evident that the main difference between the BORLS fronts and the fronts obtained from the two enhanced metaheuristics lies in the fact that the BORLS is capable of obtaining lower values for TPC, while the enhanced metaheuristics are capable of obtaining lower values for $Cmax$. The reason for this difference is that the enhanced algorithms use the minimum makespan initial solution generated by the BORLS in their initial populations. This particular solution is the only one generated by the BORLS that remains unaffected by the local search procedure. The metaheuristics then improve upon this initial point and obtain better solutions, especially when $Cmax$ is heavily emphasized over TPC. However, since the initial populations consist of fewer heuristically generated solutions compared to when BORLS is used as an independent search algorithm, the search uses much less exploitation power and gradually loses proximity to the optimal front as it proceeds to minimize TPC. As a result, the metaheuristics start with relatively poorer initial solutions when TPC is largely emphasized over $Cmax$.

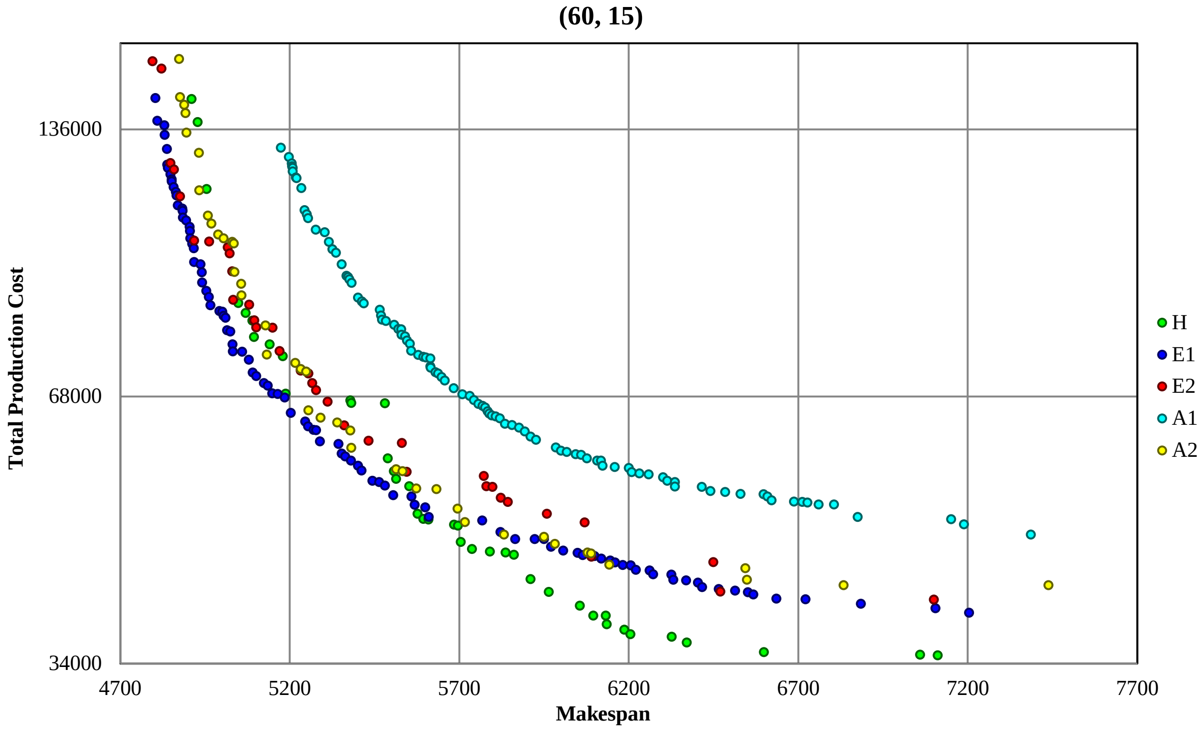

Fig. 3.17 Nondominated fronts for problem (60, 15)

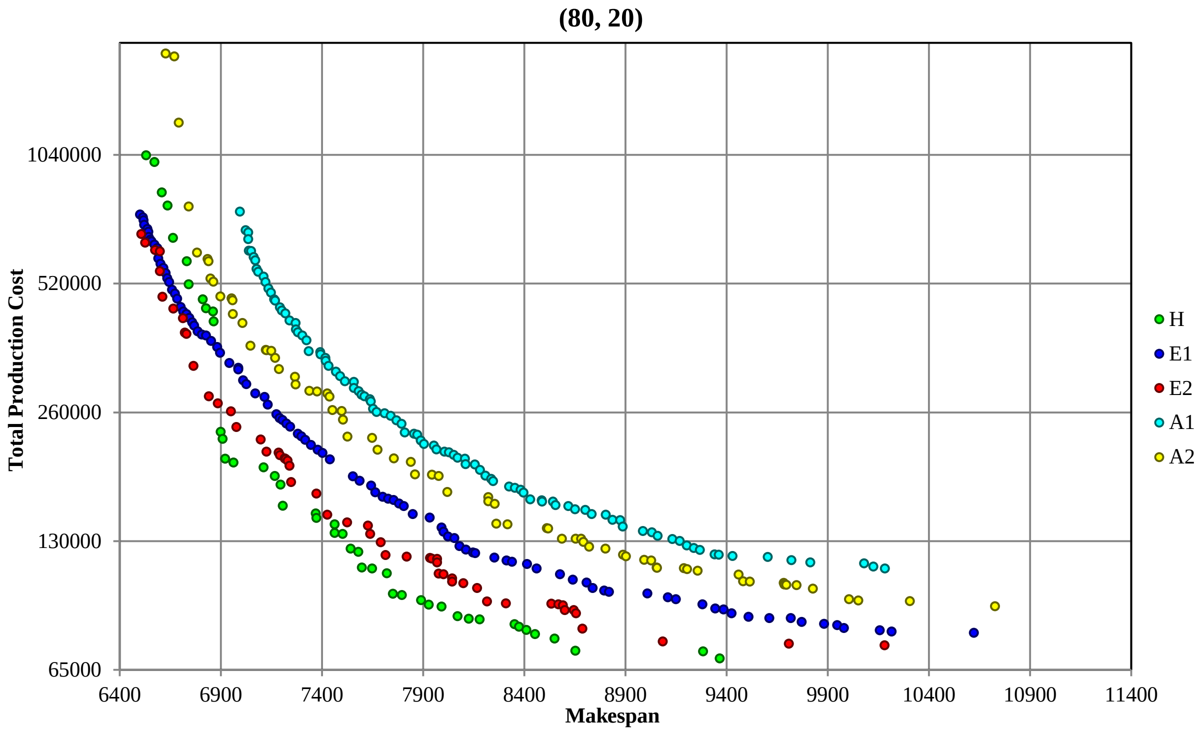

Fig. 3.18 Nondominated fronts for problem (80, 20)

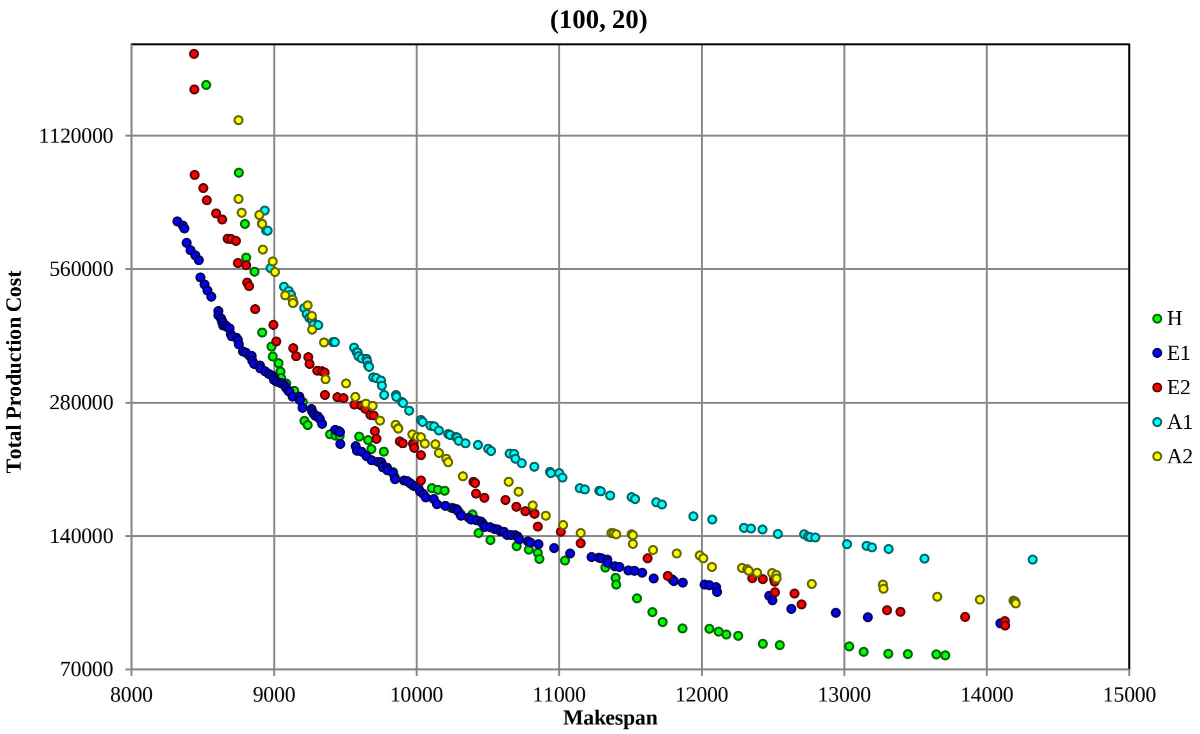

Fig. 3.19 Nondominated fronts for problem (100, 20)

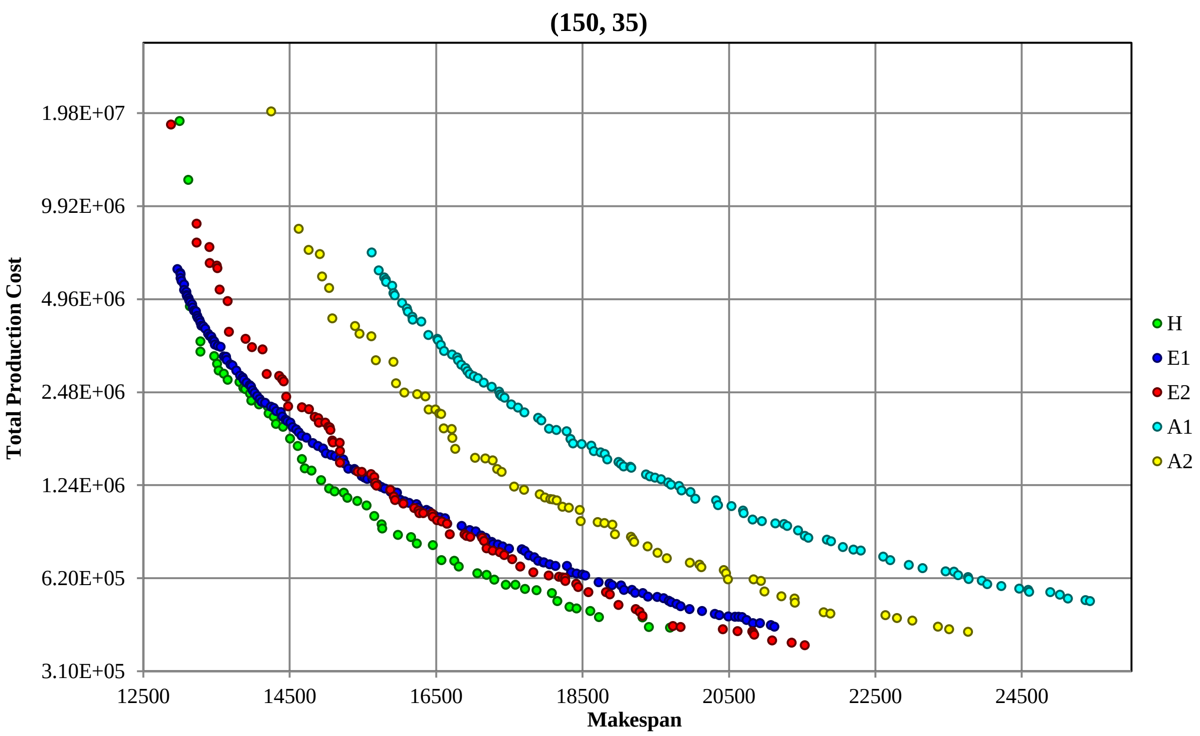

Fig. 3.20 Nondominated fronts for problem (150, 35)

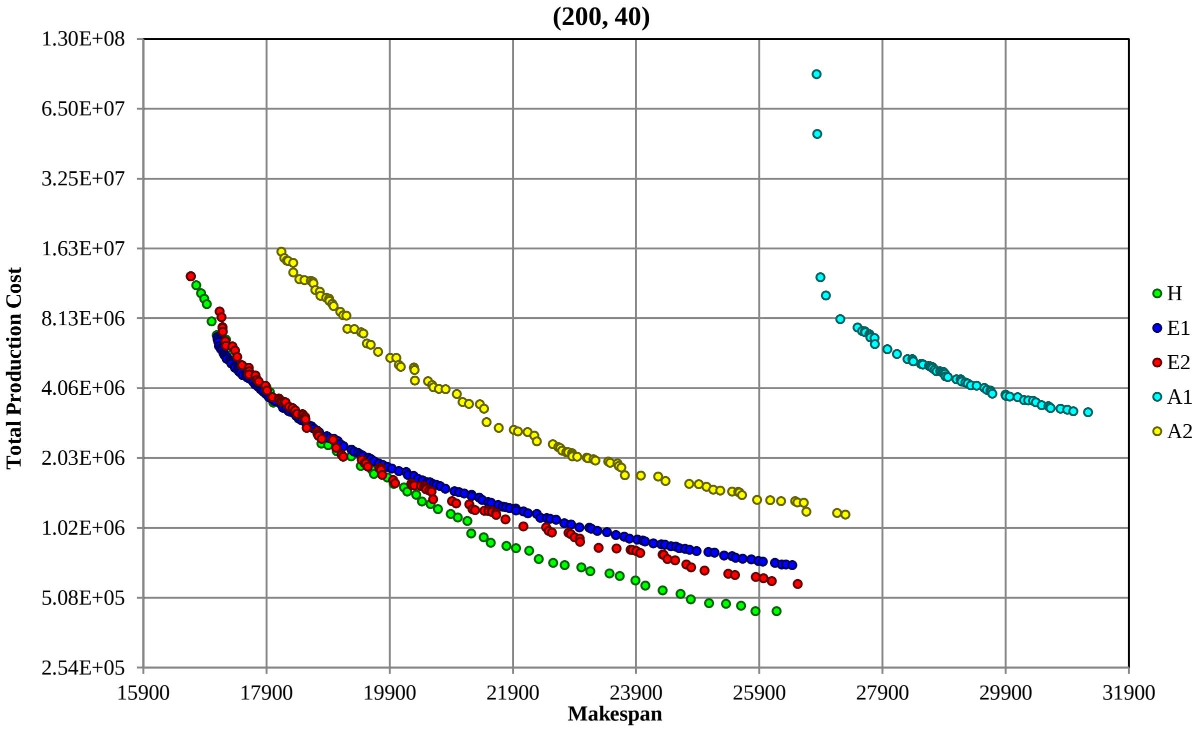

Fig. 3.21 Nondominated fronts for problem (200, 40)

3.7 Conclusion

In this chapter, we propose an integrated model to address the flow shop and maintenance scheduling problem with a shared nonrenewable resource (FMSP-SNR) in a permutation flow shop environment. The resource supply machine also undergoes degradation and requires preventive maintenance interventions. Our objective is to optimize both the makespan and the total production cost, which includes total maintenance cost, total consumed resource cost, and resource inventory cost. The model involves making three types of decisions simultaneously: determining the job sequence through the system, specifying preventive maintenance periods for each production machine and the resource supply machine, and setting the supply rate. To tackle this complex problem, we adopt a systematic approach. Firstly, we create an unconstrained schedule of production and maintenance tasks, considering only the production machines. Then, we employ a fast computation method to account for the resource constraints and determine the actual starting and completion dates of the tasks. For solving the problem, we develop and test two multi-objective metaheuristic algorithms: the second version of the nondominated sorting genetic algorithm (NSGA-II) and a bi-objective particle swarm optimization algorithm (BOPSO). Additionally, we introduce a novel bi-objective randomized local search (BORLS) heuristic capable of generating multiple non-dominated solutions. The BORLS is used as an initial population generation procedure for both metaheuristics and as an independent search algorithm. Computational experiments on purely random initial generation algorithms demonstrate that BOPSO outperforms NSGA-II, particularly for large problems. BOPSO utilizes an external domination-free archive to store global best solutions, which guides the swarm of particles toward the optimal frontier. To enhance the convergence of the algorithm while maintaining solution diversity, we adopt an assignment strategy for personal and global best solutions. The swarm is decomposed into nondominated fronts using the nondominated sorting algorithm, and for each particle in fronts with a rank greater than one, a personal best solution is assigned from the next lower rank front, while the global best solution is selected from the repository. A simple computation is carried out to determine these global and personal guides, considering the bi-objective nature of the problem.

In the subsequent experiments involving the metaheuristics with enhanced initial populations and the BORLS heuristic, we found that NSGA-II emerged as the most suitable algorithm for small- and medium-sized problems, closely followed by BOPSO and BORLS, which exhibited similar performance levels. However, for large and very large problems, all three algorithms performed equally well, with a slight advantage for BORLS when the emphasis was on minimizing the total production cost.

Chapter 4

Energy-efficient Flow shop Scheduling Problem with Blocking and Collision-Free Transportation constraints

This chapter provides an in-depth investigation of the Energy-Efficient Flow Shop Scheduling Problem with Blocking and Collision-Free Transportation Constraints (EFSP-BCFT), considering Sequence Dependent Setup Times (SDST). The chapter focuses on the scheduling of Automated Guided Vehicles (AGV), encompassing aspects of transportation speed control and battery management. It also presents the fundamental modeling assumptions and introduces a multi-objective Mixed Integer Linear Programming (MILP) formulation. To tackle this complex problem, we introduce an Enhanced Multi-Objective Ant Colony Algorithm (EMOACA) designed to simultaneously minimize makespan and total energy consumption.

The chapter is structured as follows: Section 4.1 presents a description of the problem and lists the underlying assumptions governing the mathematical model. Section 4.2 provides a mixed integer linear programming formulation for the problem. Section 4.3 presents an Enhanced multi-objective ant colony optimization algorithm (EMOACA) as a proposed metaheuristic to solve the EFSP-BCFT, providing a comprehensive understanding of its mechanisms through subsections 4.3.1 – 4.3.12. Experimental results for the EFSP-BCFT are provided in section 4.4 . Finally, the chapter is concluded through section 4.5.

4.1 Problem Description and underlying assumptions

The Energy-Efficient Flow Shop Scheduling Problem with Blocking and Collision-Free Transportation Constraints (EFSP-BCFT) involves the scheduling of a set of n jobs that must be processed by a set of m machines in a flow shop sequence. These jobs are transported between machines using R AGVs via a single-loop, unidirectional guide path, as depicted in Fig. 4.1. It's important to note that the job sequence remains fixed throughout the process, and there are zero intermediate buffers between machines. This means that blocking constraints prevent a job from being processed on a machine until the previous job in the sequence has been completed, except for the last machine, where jobs are promptly collected and transferred to the output device by a collector robot as soon as they finish processing. Each job-machine pair, denoted as operation OP_{ij}, involves a specific job j being processed on a particular machine i. There are a total of nm such operations, each with a predetermined processing time P_{ij}. Additionally, there are sequence-dependent setup time ST_{ijp} and setup energy SE_{ijp} associated with preparing machine i to process job p after job j. Importantly, these setups are non-anticipatory, meaning they can only be performed after AGVs have unloaded job j from machine i, and they do not require the presence of job p at that moment.

To facilitate travel between machines, AGVs necessitate a certain amount of transportation time and energy, which depends on several factors:

1. Distance Between Machines denoted as D_{ki} and represents the travel distance from machine k to machine i.
2. AGV Speed Settings: AGVs can be operated at various discrete speed settings, which can be selected from a predefined set of speeds denoted as S.
3. Load Carried by AGVs: The energy required for travel also depends on the load carried by the AGV during transportation.

Additionally, AGVs deplete the energy stored in their onboard batteries, and these batteries have a limited energy capacity denoted as C. To ensure AGV functionality, a battery must be changed and recharged once its charge level falls below a critical threshold. To facilitate this, a battery station is situated in the same area as the input to the shop floor.

The energy consumption of AGVs can be categorized into three distinct rates:

1. Travel Energy Consumption Rate TE_v: This rate varies depending on the selected AGV speed setting, v, which belongs to the set of available speeds, $S = \{1, 2, \dots, v_{max}\}$.
2. AGV's Robotic Arm Energy AE: This energy is required for loading or unloading a job.
3. AGV's Idle Power IE: This is the electric power used by each AGV throughout the scheduling horizon.

If an AGV's current battery charge is insufficient to complete its next transportation task, it is required to travel to the input/battery (I/B) station. There, its depleted battery is swapped with a fully charged one within a fixed change time denoted as BCT. It is important to note that all AGVs share a single unit capacity track, and therefore, they must not collide or overlap during the entire scheduling horizon. However, for simplification in modeling, it is assumed that AGVs are treated as zero-dimensional points, and multiple AGVs can occupy the same point on the track simultaneously. Safety distances between consecutive AGVs are managed during real-time control.

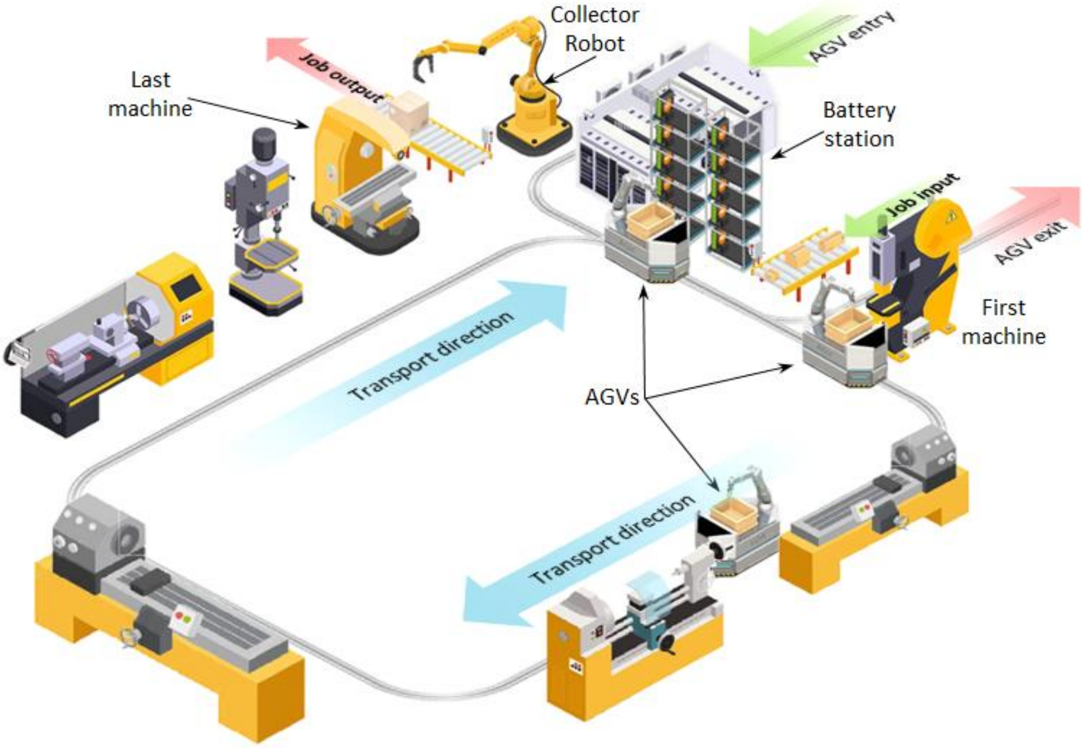

Fig. 4.1 Depiction of an EFSP-BCFT setup featuring seven machines and three AGVs. The placement of the battery station is adjacent to the job input conveyor, creating an integrated Input/Battery (I/B) station. Simultaneously, a collector robot is positioned next to the job output conveyor, tasked with collecting completed jobs from the last machine. The AGV guide path follows a unidirectional route, comprising a single entry point and one exit point, allowing for the adjustment of the number of AGVs utilized.

The following are additional assumptions required for the EFSP-BCFT modelling:

1. All jobs are independent, possess the same load, become available at time zero, and have no precedence relationships among them.
2. Each machine has a capacity to process only one job at a time, and no job passing or preemption is permitted.
3. The Input/Battery (I/B) station is treated as a dummy machine for modeling purposes.
4. While AGVs are in the I/B station for battery swapping, they can obstruct the track, and overtaking by other AGVs is prohibited.

5. Adequate fully charged replacement batteries are constantly available at the I/B station, ready to be installed on the AGVs at any given moment.
6. The I/B station has the capability to perform multiple battery swaps concurrently for multiple AGVs.
7. All machines and AGVs are operational from time zero onwards, without any breakdowns or interruptions.
8. All AGVs are identical, utilize the same battery type, and are initially stationed at the I/B station.
9. AGVs can transport only one job at a time and have exclusive control over job loading and unloading, except for the unloading of the last machine.
10. Loading and unloading times for jobs are considered negligible; however, the energy expended during loading and unloading is taken into account.

4.2 Mixed Integer Linear Programming Formulation

In this section, we present a MILP formulation for addressing the problem. At first, the various sets, indices, parameters, and decision variables used in the formulation are presented. Then, the two objectives and the model's constraints are described.

4.2.1 Sets

- The set of machines: $M = \{1, 2, \dots, m\}$
- The set of jobs: $J = \{1, 2, \dots, n\}$
- The set of AGVs: $V = \{1, 2, \dots, R\}$
- The set of AGV speeds: $S = \{1, 2, \dots, v_{max}\}$
- The set of AGV-travel's time points: TP

4.2.2 Indices

- *Machines*: $i, k, f \in M$ where $|M| = m$
- *Jobs*: $j, p, l \in J$ where $|J| = n$
- *AGVs*: $r, a \in V$ where $|V| = R$
- *Speeds*: $v \in S$ where
- *Time points' indices:* To represent the various moves made by AGVs, we divide each operation into six distinct moments. Each of these moments corresponds to a specific time point and total distance traveled by the AGVs. the time points are indexed as $h, e \in TP = \{BS, SU, CU, SL, CL, LO\}$ where BS indicates the AGV's arrival to the battery charging station if a battery change is required. SU is the start of the AGV unloaded move. CU indicates the the completion of the unloaded move. SL is the start of the loaded move (or job transfer operation) to the next station. CL is the

completion of the loaded move and LO indicates the moment the AGV loads the transferred job on the machine and it marks the start of the job processing.

4.2.3 Parameters

- P_{ij}: Processing time of job j on machine i
- D_{ik}: Transportation distance between machine i and machine k where $i\ or\ k = 0$ represents the I/B station.
- L: Total length of the AGVs' track where $L = D_{ik} + D_{ki} \quad \forall i, k \in M \quad i \neq k$
- W: Load energy factor
- TE_v: Travel energy consumption rate at speed setting v
- AE_h: AGV's robotic arm energy required when loading or unloading jobs. i.e., in time points $h = SL, LO$, where $AE_{SL} = AE_{LO}$. The value of AE_h is zero for the rest of the time points
- IE: AGV's idle power
- U_h: Binary parameter used to force AGVs to keep being loaded for the entire loaded transfer of jobs. The values of U_h are represented by the array [1 1 1 0 0 1]. The zeros correspond to the time points $h = SL, CL$
- BCT: Battery change time
- C: Maximum battery consumption level (Battery capacity)
- ST_{ijp}: Setup time between job j and job p on machine i
- SE_{ijp}: Setup energy between job j and job p on machine i
- G: A big number

4.2.4 Variables

Binary variables:

- $Z_{jp} = 1$ if job j immediately precedes job p in the job sequence. A dummy initial/final job referred to as j_0 is added to complete the model
- $X_{ijkpr} = 1$ if transfer operation OP_{ij} immediately precedes operation OP_{kp} by AGV r. A dummy initial/final operation referred to as operation $OP_{0,0}$ is added in order to complete the model yielding additional $X_{0,0,ijr}$ and $X_{ij,0,0,r}$ variables $\forall i \in M, \forall j \in J, \forall r \in V$
- $B_{ij} = 1$ if there is a battery change before starting operation OP_{ij}
- $S_{hijrv} = 1$ if speed setting v is chosen for AGV r's travel speed from time point h of operation OP_{ij} to time point immediately after it.
- $Y_{hijekp} = 1$ if time point h of operation OP_{ij} immediately precedes time point e of operation OP_{kp}. A dummy initial/final time point referred to as time point tp_0 is added to complete the model yielding additional $Y_{0,0,0,hij}$ and $Y_{hij,0,0,0}$ variables $\forall i \in M, \forall j \in J, \forall h \in TP$.
- $F_{hijr} = 1$ if AGV r is loaded in time point h of operation OP_{ij}

Integer variables:

- LP_r: Number of laps performed by AGV r at the end of the schedule

Continuous variables:

- $Cmax$: Maximum completion time
- T_{hij}: Time point h of operation OP_{ij}
- TD_{hijr}: Total distance traveled by AGV r at time point h for operation OP_{ij}
- BL_{hijr}: Battery consumption level of AGV r at time point h
- PE_{hijr}: Partial energy consumed during AGV r's travel at time point h of operation OP_{ij} from previous time point. The dummy initial/final time point tp_0 is added in order to complete the model yielding additional $PE_{0,0,0,r}$ variables $\forall r \in V$
- TR_{hij}: Order of time point h of operation OP_{ij}

4.2.5 Mathematical Model

Objectives:

$$\text{min.}\, Cmax \tag{4.1}$$

$$\text{min.}\, TEC = \sum_{i\in M}\sum_{j\in J}\left(\sum_{p\in J}\left(SE_{ijp}Z_{jp}\right) + \sum_{r\in V}\sum_{h\in TP} PE_{hijr}\right) + \sum_{r\in V} PE_{0,0,0,r} + 2nmAE_{LO} + R\cdot IE\cdot Cmax \tag{4.2}$$

Constraints:

$$\sum_{j\in\{J,j_0\}} Z_{pj} = 1 \qquad \forall p \in \{J, j_0\} \tag{4.3}$$

$$\sum_{j\in\{J,j_0\}} Z_{pj} = \sum_{j\in\{J,j_0\}} Z_{jp} \qquad \forall p \in \{J, j_0\} \tag{4.4}$$

$$\sum_{j\in\{J,j_0\}} Z_{jj} = 0 \tag{4.5}$$

$$T_{LO,ip} \geq T_{SL,i+1,j} + ST_{ijp}Z_{jp} - G\left(1 - Z_{jp}\right) \qquad \forall i \in M - \{m\}, \forall j,p \in J \quad j \neq p \tag{4.6}$$

$$T_{LO,m,p} \geq T_{LO,m,j} + P_{m,j} + ST_{m,jp}Z_{jp} - G\left(1 - Z_{jp}\right) \qquad \forall j,p \in J \quad j \neq p \tag{4.7}$$

$$Cmax \geq T_{LO,m,j} + P_{m,j} \qquad \forall j \in J \tag{4.8}$$

$$\sum_{i\in M}\sum_{j\in J} X_{ij,0,0,r} = 1 \qquad \forall r \in V \qquad (4.9)$$

$$\sum_{i\in M}\sum_{j\in J} X_{0,0,ijr} = 1 \qquad \forall r \in V \qquad (4.10)$$

$$\sum_{i\in M}\sum_{j\in J} X_{ijijr} + X_{0,0,0,0,r} = 0 \qquad \forall r \in V \qquad (4.11)$$

$$\sum_{k\in M}\sum_{p\in J} X_{ijkpr} + X_{ij,0,0,r} = \sum_{k\in M}\sum_{p\in J} X_{kpijr} + X_{0,0,ijr} \qquad \forall i \in M, \forall j \in J, \forall r \in V \qquad (4.12)$$

$$\sum_{k\in M}\sum_{p\in J}\sum_{r\in V} X_{kpijr} + X_{0,0,ijr} = 1 \qquad \forall i \in M, \forall j \in J \qquad (4.13)$$

$$\sum_{v\in S} S_{hijrv} = 1 \qquad \forall i \in M, \forall j \in J, \forall r \in V, \forall h \in TP \qquad (4.14)$$

$$T_{SU,ij} \geq T_{BS,ij} + BCT \cdot B_{ij} \qquad \forall i \in M, \forall j \in J \qquad (4.15)$$

$$T_{CU,ij} \geq T_{SU,ij} + vD_{0,i-1}S_{SU,ijrv} - G\left(2 - B_{ij} - \sum_{k\in M}\sum_{p\in J} X_{kpijr} + X_{0,0,ijr}\right) \qquad \forall i \in M, \forall j \in J, \forall v \in S \qquad (4.16)$$

$$T_{CU,ij} \geq T_{SU,ij} + vD_{k,i-1}S_{SU,ijrv} - G\left(1 + B_{ij} - \sum_{p\in J}\sum_{a\in V} X_{kpija}\right) \qquad \forall i,k \in M, \forall j \in J, \forall r \in V, \forall v \in S \qquad (4.17)$$

$$T_{SL,ij} \geq T_{CU,ij} \qquad \forall i \in M, \forall j \in J \qquad (4.18)$$

$$T_{SL,ij} \geq T_{LO,i-1,j} + P_{i-1,j} \qquad \forall i \in M - \{1\}, \forall j \in J \qquad (4.19)$$

$$T_{CL,ij} \geq T_{SL,ij} + vD_{i-1,i}S_{SL,ijrv} - G\left(1 - \sum_{k\in M}\sum_{p\in J} X_{kpijr} + X_{0,0,ijr}\right) \qquad \forall i \in M, \forall j \in J, \forall r \in V, \forall v \in S \qquad (4.20)$$

$$T_{LO,ij} \geq T_{CL,ij} \qquad \forall i \in M, \forall j \in J \qquad (4.21)$$

$$T_{SU,ij} \geq T_{LO,i+1,p} - G\left[3 - Z_{pj} - \left(\sum_{k\in M}\sum_{l\in J} X_{klijr} + X_{0,0,ijr}\right) - \left(\sum_{k\in M}\sum_{l\in J} X_{kl,i+1,pr} + X_{0,0,i+1,pr}\right)\right] \quad \forall i \in M - \{m\}, \forall j,p \in J, r \in V \tag{4.22}$$

$$T_{BS,ij} \geq T_{LO,kp} - G\left(1 + B_{ij} - \sum_{r\in V} X_{kpijr}\right) \quad \forall i,k \in M, \forall j,p \in J \tag{4.23}$$

$$T_{BS,ij} \geq T_{LO,kp} + vD_{k,0}S_{LO,kprv} - G\left(2 - B_{ij} - X_{kpijr}\right) \quad \forall i,k \in M, \forall j,p \in J, r \in V, \forall v \in S \tag{4.24}$$

$$TD_{BS,ijr} \leq G\left(1 - X_{0,0,ijr}\right) \quad \forall i \in M, \forall j \in J, r \in V \tag{4.25}$$

$$TD_{h+1,ijr} \geq TD_{hijr} \quad \forall i \in M, \forall j \in J, r \in V, h \in TP - \{LO\} \tag{4.26}$$

$$TD_{SU,ijr} \leq TD_{BS,ijr} + G\left(1 - \sum_{k\in M}\sum_{p\in J} X_{kpijr} + X_{0,0,ijr}\right) \quad \forall i \in M, \forall j \in J, r \in V \tag{4.27}$$

$$TD_{CU,ijr} \geq TD_{SU,ijr} + D_{0,i-1}B_{ij} + D_{k,i-1}\left(1 - B_{ij}\right) - G\left(1 - \sum_{p\in J} X_{kpijr}\right) \quad \forall i,k \in M, \forall j \in J, r \in V \tag{4.28}$$

$$TD_{CU,ijr} \leq TD_{SU,ijr} + D_{0,i-1}B_{ij} + D_{k,i-1}\left(1 - B_{ij}\right) + G\left(1 - \sum_{p\in J} X_{kpijr}\right) \quad \forall i,k \in M, \forall j \in J, r \in V \tag{4.29}$$

$$TD_{CU,ijr} \geq TD_{SU,ijr} + D_{0,i-1} - G\left(1 - X_{0,0,ijr}\right) \quad \forall i \in M, \forall j \in J, r \in V \tag{4.30}$$

$$TD_{CU,ijr} \leq TD_{SU,ijr} + D_{0,i-1} + G\left(1 - X_{0,0,ijr}\right) \quad \forall i \in M, \forall j \in J, r \in V \tag{4.31}$$

$$TD_{SL,ijr} \leq TD_{CU,ijr} + G\left(1 - \sum_{k\in M}\sum_{p\in J} X_{kpijr} - X_{0,0,ijr}\right) \quad \forall i \in M, \forall j \in J, r \in V \tag{4.32}$$

$$TD_{CL,ijr} \geq TD_{SL,ijr} + D_{i-1,i}\left(\sum_{k\in M}\sum_{p\in J} X_{kpijr} + X_{0,0,ijr}\right) \quad \forall i \in M, \forall j \in J, r \in V \tag{4.33}$$

$$TD_{CL,ijr} \leq TD_{SL,ijr} + D_{i-1,i} + G\left(1 - \sum_{k\in M}\sum_{p\in J} X_{kpijr} + X_{0,0,ijr}\right) \quad \forall i \in M, \forall j \in J, r \in V \tag{4.34}$$

$$TD_{LO,ijr} \leq TD_{CL,ijr} + G\left(1 - \sum_{k\in M}\sum_{p\in J} X_{kpijr} - X_{0,0,ijr}\right) \quad \forall i \in M, \forall j \in J, r \in V \tag{4.35}$$

$$TD_{BS,ijr} \geq TD_{LO,kpr} + D_{k,0}B_{ij} - G\left(1 - X_{kpijr}\right) \quad \forall i,k \in M, \forall j,p \in J, r \in V \tag{4.36}$$

$$TD_{BS,ijr} \leq TD_{LO,kpr} + D_{k,0}B_{ij} + G\left(1 - X_{kpijr}\right) \quad \forall i,k \in M, \forall j,p \in J, r \in V \tag{4.37}$$

$$TD_{LO,ijr} \leq TD_{LO,kpr} + D_{k,0} + G\left(1 - X_{kp,0,0,r}\right) \quad \forall i,k \in M, \forall j,p \in J, r \in V \tag{4.38}$$

$$L \cdot LP_r \geq TD_{LO,ijr} \quad \forall i \in M, \forall j \in J, r \in V \tag{4.39}$$

$$\sum_{e\in TP}\sum_{k\in M}\sum_{p\in J} Y_{hijekp} + Y_{hij,0,0,0} = 1 \quad \forall i \in M, \forall j \in J, h \in \{TP, tp_0\} \tag{4.40}$$

$$\sum_{e\in TP}\sum_{k\in M}\sum_{p\in J} Y_{ekphij} + Y_{0,0,0,hij} = 1 \quad \forall i \in M, \forall j \in J, h \in \{TP, tp_0\} \tag{4.41}$$

$$\sum_{h\in TP}\sum_{i\in M}\sum_{j\in J} Y_{hijhij} + Y_{0,0,0,0,0,0} = 0 \tag{4.42}$$

$$TR_{hij} \geq TR_{ekp} + 1 - G\left(1 - Y_{hijekp}\right) \quad \forall i,k \in M, \forall j,p \in J, \forall h,e \in TP, hij \neq ekp \tag{4.43}$$

$$TR_{h+1,ij} \geq TR_{hij} + 1 \quad \forall i \in M, \forall j \in J, \forall h \in TP - \{BS\} \tag{4.44}$$

$$T_{ekp} \geq T_{hij} - G\left(1 - Y_{hijekp}\right) \quad \forall i,k \in M, \forall j,p \in J, \forall h,e \in TP, hij \neq ekp \tag{4.45}$$

$$TD_{ekpr} \geq TD_{hijr} - G\left(1 - Y_{hijekp}\right) \quad \forall i,k \in M, \forall j,p \in J, r \in V, \forall h,e \in TP, hij \neq ekp \tag{4.46}$$

$$T_{ekp} - T_{hij} \geq v\left(TD_{ekpr} - TD_{hijr}\right) - G\left(2 - S_{hijrv} - Y_{hijekp}\right) \quad \forall i,k \in M, \forall j,p \in J, r \in V, \forall v \in S, \forall h,e \in TP, hij \neq ekp \tag{4.47}$$

$$TD_{hijr} \geq TD_{hij,r+1} \quad \forall i \in M, \forall j \in J, r \in V - \{R\}, \forall h \in TP \tag{4.48}$$

$$TD_{hij,1} \leq TD_{hij,R} + L \quad \forall i \in M, \forall j \in J, \forall h \in TP \tag{4.49}$$

$$F_{SL,ijr} \geq \sum_{f \in M} \sum_{l \in J} X_{flijr} + X_{0,0,ijr} \quad \forall i \in M, \forall j \in J, \forall r \in V \tag{4.50}$$

$$F_{ekpr} \geq F_{hijr} - 1 + Y_{hijekp} - U_e\left(\sum_{f \in M} \sum_{l \in J} X_{flkpr} + X_{0,0,kpr}\right) \quad \forall i,k \in M, \forall j,p \in J, r \in V, \forall h,e \in TP \quad hij \neq ekp \tag{4.51}$$

$$BL_{ekpr} \geq BL_{hijr} + TE_v\left(TD_{ekpr} - TD_{hijr}\right) + IE\left(T_{ekp} - T_{hij}\right) + AE_e\left(\sum_{f \in M} \sum_{l \in J} X_{flkpr} + X_{0,0,kpr}\right) - G\left(2 - S_{hijrv} - Y_{hijekp}\right) \quad \forall i,k \in M, \forall j,p \in J, r \in V, \forall v \in S \forall h \in TP \quad \forall e \in TP - \{SU\} \quad hij \neq ekp \tag{4.52}$$

$$BL_{ekpr} \geq BL_{hijr} + TE_v W\left(TD_{ekpr} - TD_{hijr}\right) + IE\left(T_{ekp} - T_{hij}\right) + AE_e\left(\sum_{f \in M} \sum_{l \in J} X_{flkpr} + X_{0,0,kpr}\right) - G\left(3 - S_{hijrv} - Y_{hijekp} - F_{hijr}\right) \quad \forall i,k \in M, \forall j,p \in J, r \in V, \forall v \in S \forall h \in TP \quad \forall e \in TP - \{SU\} \quad hij \neq ekp \tag{4.53}$$

$$BL_{SU,kpr} \geq BL_{hijr} + TE_v\left(TD_{SU,kpr} - TD_{hijr}\right) + IE\left(T_{SU,kp} - T_{hij}\right) - G\left(3 - B_{kp} - S_{hijrv} - Y_{hij,SU,kp} + \sum_{f \in M}\sum_{l \in J} X_{flkpr} + X_{0,0,kpr}\right) \quad \forall i,k \in M, \forall j,p \in J, r \in V, \forall v \in S, \forall h \in TP \quad hij \neq SU,kp \tag{4.54}$$

$$BL_{SU,kpr} \geq BL_{hijr} + TE_v\left(TD_{SU,kpr} - TD_{hijr}\right) + IE\left(T_{SU,kp} - T_{hij}\right) - G\left(2 + B_{kp} - S_{hijrv} - Y_{hij,SU,kp}\right) \quad \forall i,k \in M, \forall j,p \in J, r \in V, \forall v \in S, \forall h \in TP \quad hij \neq SU,kp \tag{4.55}$$

$$BL_{BS,ijr} \leq C \quad \forall i \in M, \forall j \in J, r \in V \tag{4.56}$$

$$BL_{LO,ijr} + (TE_v + IEv)\left(L \cdot LP_r - TD_{LO,ijr}\right) - G\left(2 - S_{LO,ijrv} - Y_{LO,ij,0,0,0}\right) \leq C \quad \forall i \in M, \forall j \in J, r \in V, \forall v \in S \tag{4.57}$$

$$PE_{ekpr} \geq TE_v\left(TD_{ekpr} - TD_{hijr}\right) - G\left(2 - S_{hijrv} - Y_{hijekp}\right) \quad \forall i,k \in M, \forall j,p \in J, r \in V, \forall v \in S, \forall h,e \in TP \quad hij \neq ekp \tag{4.58}$$

$$PE_{ekpr} \geq TE_v W\left(TD_{ekpr} - TD_{hijr}\right) - G\left(3 - S_{hijrv} - Y_{hijekp} - F_{hijr}\right) \quad \forall i,k \in M, \forall j,p \in J, r \in V, \forall v \in S, \forall h,e \in TP \quad hij \neq ekp \tag{4.59}$$

$$PE_{0,0,0,r} \geq TE_v\left(L \cdot LP_r - TD_{LO,ijr}\right) - G\left(2 - S_{LO,ijrv} - Y_{LO,ij,0,0,0}\right) \quad \forall i \in M, \forall j \in J, r \in V, \forall v \in S \tag{4.60}$$

$$Cmax \geq 0 \tag{4.61}$$

$$T_{hij}, TR_{hij} \geq 0 \quad \forall h \in TP, \forall i \in M, \forall j \in J \tag{4.62}$$

$$TD_{hijr}, BL_{hijr}, PE_{hijr} \geq 0 \quad \forall h \in TP, \forall i \in M, \forall j \in J, r \in V \tag{4.63}$$

$$LP_r, PE_{0,0,0,r} \geq 0 \quad r \in V \tag{4.64}$$

The objective functions (4.1) and (4.2) aim to minimize two aspects: the maximum completion time and the total energy consumption throughout the schedule. The total energy consumption comprises four components: (1) the total setup energy; (2) the total transportation energy; (3) the total loading/unloading energy; and (4) the total idle energy for AGVs. The energy consumed during processing and by the collector is considered constant and thus excluded from the total energy consumption. Constraints (4.3) – (4.13) pertain to the sequencing of jobs and AGV operations. Constraints (4.3) – (4.5) ensure that each job must be preceded and followed by a single different job. Constraints (4.6) – (4.8) establish the fundamental start and completion time conditions for jobs. In (4.6), it is stipulated that the processing of job j on machines $1, 2, \dots, m-1$ can only commence after the preceding job p has been unloaded from the machine and the requisite setup has been executed. This constraint effectively functions as a blocking constraint when job j's loading and job p's unloading are executed by separate AGVs. Conversely, in (4.7), job j can initiate as soon as the setup is finalized after the completion of the preceding job. Finally, (4.8) calculates the makespan. Constraints (4.9); (4.10) dictate that each AGV should initiate and conclude its route exclusively at the dummy operation, indicating travel to and from the Input/Battery (I/B) station. In (4.11) ; (4.12), the orderly progression of AGVs through their designated operations is guaranteed, while (4.13) enforces that each operation is allocated to a single AGV. Constraint (4.14) serves to select one speed setting for each time point for each AGV.

Constraints (4.15) – (4.25) establish fundamental relationships between the time points of AGV travel for individual operations and between consecutive operations. (4.15) represents the time spent at the battery station if operation OP_{ij} necessitates a battery change. (4.16) calculates the completion time of the unloaded move $T_{CU,ij}$ to machine $i-1$ with speed setting v, starting from the battery station when a battery change is required. (4.17) computes $T_{CU,ij}$ in scenarios where no battery change is needed. (4.18); (4.19) ensure that a loaded transfer of job j from machine $i-1$ to i cannot commence unless the assigned AGV has completed its unloaded travel and job j has finished processing on machine $i-1$. (4.20) signifies that the assigned AGV r completes the loaded transfer of job j from machine $i-1$ to i after traveling the distance $D_{i-1,i}$ at speed setting v. (4.21) mandates that the assigned AGV must complete its loaded move before job j can undergo processing. (4.22) enforces a blocking constraint to prevent the same AGV from both loading job j on machine i and unloading job p from machine i to machine $i+1$ simultaneously. Constraints (4.23); (4.24) establish connections between consecutive AGV operations, both in cases where a battery change is not required and in cases where the AGV must visit the battery station after loading job p on machine k before performing transportation tasks for OP_{ij}.

Constraints (4.25) – (4.38) establish essential relationships between AGVs' total traveled distances at each time point for individual operations and between consecutive operations. (4.25) initializes the total traveled distance for each AGV to 0. (4.26) enforces that the total traveled distance cannot decrease between consecutive time points of each operation, adhering to the unidirectional movement constraint. (4.27) requires the assigned AGV to wait at the battery station when a battery change is necessary. Constraints (4.28); (4.29) calculate the accurate total traveled distance upon completion of the unloaded move. The distance is computed by adding the distance from the battery station when a battery change occurred, or the distance from the previous AGV position if no battery change was needed. Constraints (4.30); (3.31) handle scenarios involving the first AGV assigned to an operation. (4.32) compels the assigned AGV to wait for job j's completion at machine $i-1$ before unloading it and transferring it to machine i. Constraints (4.33); (4.34) compute the total traveled distance for the assigned AGV once the loaded move is finished. (4.35) forces the AGV to wait at machine i until the preceding job is completed and the necessary machine setup is performed. Constraints (4.36); (4.37) establish links between consecutive AGV operations where the assigned AGV must travel back to the battery station (if a battery change is needed) before conducting transportation tasks for operation OP_{ij}. (4.38) ensures that AGVs that have completed all their assigned transportation tasks remain at the battery station once they reach it. For all these constraints, if an AGV is not assigned to the respective operation OP_{ij}, it can move freely within its allowed range of possible speed settings. Constraint (4.39) calculates the number of laps completed by each AGV.

Constraints (4.40) – (4.49) encompass collision prevention and AGV speed-related constraints. Constraints (4.40) – (4.45) are utilized to arrange all the time points of operations using the variable Y_{hijekp}. Constraint (4.46) updates the total traveled distance of each AGV for each unselected operation. (4.47) guarantees that the speed of each AGV between consecutive time points h of operation OP_{ij} and e of operation OP_{kp} adheres to the speed setting chosen for time point h of OP_{ij}. Constraints (4.48); (4.49) are in place to prevent AGVs from overlapping. AGV $r+1$ is not permitted to overtake AGV r, and AGV 1 should not lap AGV R, ensuring a collision-free schedule.

Constraints (4.50); (4.51) mandate that if AGV r is assigned to operation OP_{ij}, it must be loaded during the transportation of the job between time points SL and CL of that operation. Constraints (4.52) – (4.57) pertain to the update of energy consumption levels and battery capacity limitations. Constraints (4.52); (4.53) increase the battery energy consumption level based on the distance traveled, time elapsed between consecutive time points, and the speed setting used by AGV r, except when a battery change is possible. Constraints (4.54); (4.55) allow the battery consumption level to drop to 0 after a battery change occurs at time point SU. Constraint (4.56) caps the battery energy consumption level at the battery capacity whenever AGVs reach time point BS. Constraint (4.57) ensures that each AGV possesses sufficient battery energy to reach the battery station after completing its last assigned transportation task. Constraints (4.58) – (4.60) compute the partial transportation energy consumed for every AGV movement based on the traveled distance and the selected speed setting. Finally, Constraints (4.61) – (4.64) establish the lower bounds for all continuous variables in the mathematical model, setting them to a minimum of 0.

4.3 Enhanced Multi-Objective Ant Colony optimization Algorithm

This section delves deeply into the EMOACA metaheuristic, providing a comprehensive grasp of its mechanisms. It explains how the traditional single-objective ACO framework is adapted to address multi-objective problems through a decomposition approach. The section also covers various aspects, including the representation of solutions, the utilization of a three-stage transition probability model for solution construction, the solution decoding process, and the updating of both pheromone matrix groups and the external archive of nondominated solutions. Furthermore, this section explores several innovative enhancement techniques aimed at enhancing the algorithm's performance. These enhancements encompass an effective low-high resolution search strategy, the creation of a novel type of heuristic information for AGV dispatching, and an energy reduction improvement procedure based on the identification of critical paths in the schedule.

4.3.1 Overview of the Ant Colony Optimization Framework

To seek near-optimal or, ideally, optimal solutions for the EFSP-BCFT, an enhanced metaheuristic algorithm is devised based on ant colony optimization. ACO is a robust population-based metaheuristic known for its ability to swiftly generate favorable solutions for problems such as flow shop scheduling [174, 175] and various other combinatorial problems [176]. However, like many metaheuristic algorithms, it can sometimes become stuck in suboptimal solutions due to premature convergence. Addressing this issue necessitates careful parameter tuning, particularly focusing on the evaporation rate and the quantity of pheromone deposited by the ants. One approach to prevent premature convergence is to set constraints on the maximum and minimum pheromone amounts, ensuring that the algorithm remains open to exploring locally less favorable options, which may guide it away from local optima. However, while mitigating premature convergence, there's a risk of the algorithm losing its ability to exploit promising regions efficiently, hindering the generation of high-quality solutions within a reasonable timeframe. To overcome this challenge, this study introduces multiple enhancement techniques into a multi-objective ant colony optimization framework to tackle the EFSP-BCFT. Specifically, it leverages heuristic information for both objectives to adapt the EMOACA for bi-objective optimization. Additionally, a local search algorithm is developed to enhance obtained solutions by reducing the total energy of the schedule without extending the makespan. Furthermore, this research introduces several search improvement strategies to augment the exploration capabilities of EMOACA.

4.3.2 Overview of the EMOACA Procedure for the EFSP-BCFT

To effectively address the bi-objective nature of the problem and generate a diverse range of solutions, a decomposition and information-sharing strategy is proposed. The problem is partitioned into N single-objective sub-problems, with each sub-problem tackled by one ant, except for the minimum $Cmax$ and the minimum TEC subproblems, which have A additional ants assigned to each subproblem. Consequently, the colony comprises $N + 2A$ ants, with N being the basic population size and A representing the number of additional ants employed to balance and reinforce the search around the two extremes of the Pareto front (PF). In traditional ACO for single-objective minimization problems, transition probability models typically utilize a single pheromone and heuristic information matrix. Furthermore, these models typically update the pheromone matrix using only the global-best solution. However, EMOACA aims to approximate the entire optimal PF for the problem. Unfortunately, a single pheromone matrix cannot encompass all the information pertaining to the entire colony and guide each ant simultaneously toward its predefined goal within the optimal PF [177]. Therefore, EMOACA employs Ω sets of pheromone matrices denoted as $\tau^0, \tau^1, \dots, \tau^{\Omega-1}$, where $1 < \Omega < N$. Each set is initialized with the maximum pheromone quantity τ_{max} and bounded by the minimum quantity τ_{min}. Each set comprises three pheromone matrices, corresponding to the three primary aspects of a solution in the EFSP-BCFT: (1) Job sequence; (2) AGV speed; and (3) AGV dispatching. Battery switching decisions are heuristically made during the solution decoding and evaluation process. Each ant i ($i = -A, -A+1, \dots, N + A - 1$) shares information about its individual progress by depositing a fraction γ of its pheromone on group τ^{g_1} and $(1-\gamma)$ on group τ^{g_2}. The value of γ is obtained by the following formula:

$$\begin{aligned} &\gamma = 1 && \text{if } i \leq 0 \text{ or } \mathrm{i} \geq \mathrm{N}-1 \\ &\gamma = 1 - \left|\frac{i}{N-1} - \frac{g_1}{\Omega - 1}\right| (\Omega - 1) && \text{otherwise} \end{aligned} \tag{4.65}$$

where g_1 and g_2 are obtained based on the index of ant i as follows:

$$\begin{aligned} &g_1 = g_2 = 0 && \text{if } i \leq 0 \\ &g_1 = g_2 = \Omega - 1 && \text{if } i \geq N-1 \\ &g_1 = \left\lfloor \frac{(\Omega-1)i}{N-1} \right\rfloor, \quad g_2 = g_1 + 1 && \text{otherwise} \end{aligned} \tag{4.66}$$

The number of additional ants for each PF extreme point is calculated according to the following equation:

$$A = \left\lfloor \frac{N + 1 - \Omega}{2(\Omega - 1)} \right\rfloor \tag{4.67}$$

It's worth noting from equations (4.65); (4.66) that these additional A ants are exclusively associated with a single pheromone matrix group, either τ^0 or $\tau^{\Omega-1}$, depending on the subproblem being addressed (min. $Cmax$ or min. TEC, respectively). To illustrate this, let's consider an example where $N = 6$ and $\Omega = 3$. In this case, $A = 1$, signifying that one ant is introduced to tackle subproblem 0 and another to address subproblem 5 at each end of the PF. For instance, both added ant -1 and ant 0 are exclusively linked to matrix group τ^0, while ant

3, for example, is associated with matrix groups τ^1 and τ^2. The contribution of pheromone is divided into a fraction of $\gamma = 0.8$ to group τ^1 and a fraction of 0.2 to group τ^2.

In each iteration, each ant i creates a novel solution x^i by evaluating pheromone trails on the same two matrix groups, as defined by the proposed transition probability model. Subsequently, a local evaporation process is conducted on these two pheromone matrix groups. This entails reducing the pheromone levels along the path used by the ant to construct the solution, with a reduction fraction of $\alpha\gamma$ for group τ^{g_1} and $\alpha(1-\gamma)$ for group τ^{g_2}, where α represents the local evaporation parameter. This local evaporation step is essential for preventing subsequent ants associated with the same matrix groups from following the same path as earlier ants, thus promoting diversity in the search process. Once an ant has assembled a complete solution, comprising a job sequence, AGV speed settings for each operation, and a feasible sequence of operations, this solution is decoded to yield a feasible, conflict-free schedule, along with a battery switching plan. This obtained solution is then further refined to minimize TEC while maintaining $Cmax$ at its current value. The generated solution serves to update an external archive, which exclusively contains nondominated solutions encountered during the search process. The external archive ultimately represents the final output of the EMOACO. However, in this study, the archive takes on an active role beyond just storing nondominated solutions; it is also instrumental in generating a reference point for each subproblem. These reference points are instrumental in guiding each ant toward its specific objective in the objective space. Within the proposed EMOACA framework, each subproblem i is associated with a reference point RP_i in the objective space. The aim of each ant belonging to subproblem i is to generate solutions with objective values as close as possible to the designated reference point RP_i. Among these solutions, only the closest one, denoted as x^i_{best}, is considered the best solution for subproblem i and is utilized to update the relevant pheromone matrices. It's important to note that the best solution for a given subproblem doesn't necessarily have to be a member of the nondominated archive. Fig. 4.2 provides an illustration of the problem decomposition, ant assignments, reference points, and matrix groups, along with their corresponding pheromone contribution factors.

Once all ants have finished generating their solutions, the pheromone matrices go through a global pheromone update. This update involves two steps: first, an evaporation process is applied, which is governed by the global evaporation rate parameter (β). Subsequently, a reinforcement step occurs, focusing exclusively on the entries utilized to construct the best solutions for each sub-problem. Following these updates, a new iteration of the algorithm is executed until the maximum number of iterations ($maxiter$) is attained.

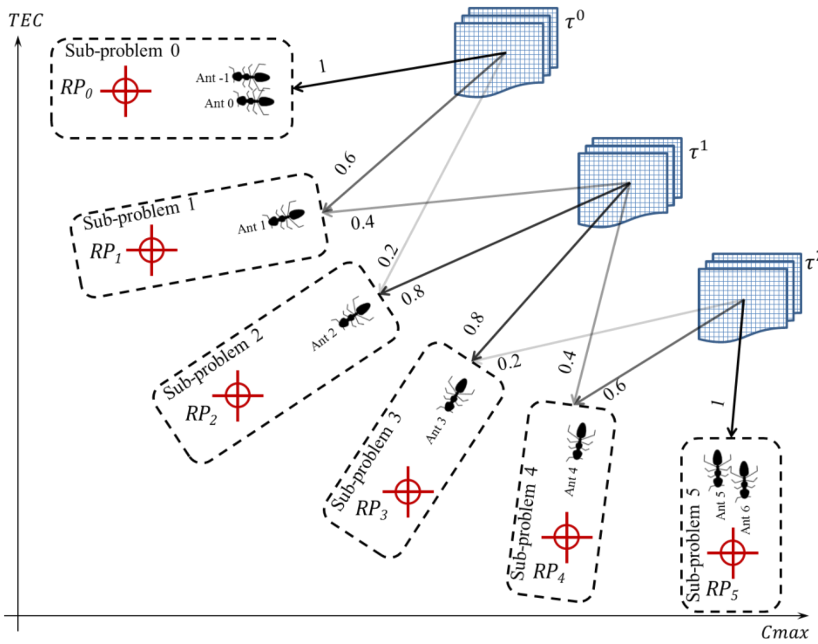

Fig. 4.2 An illustration of the problem's decomposition into six subproblems along with the assignment of ants to each subproblem. Additionally, the pheromone fractions that link each ant to the three matrix groups are depicted.

Sections 4.3.3, 4.3.4, and 4.3.5 delve into the technical details associated with updating the external archive and the best solutions for each subproblem, elucidating how the moving reference points for each sub-problem are established in each iteration. Additionally, we introduce a range of search enhancement strategies, including a novel Low-High Resolution Search (LHRS) strategy specifically tailored for the AGV speed setting aspect of solution construction. Following this, in 4.3.6, a comprehensive account of the solution representation and the transition probability model employed for constructing each component of a feasible solution is provided. 4.3.7 outlines the procedures for updating the pheromone matrix groups, while 4.3.8 delves into the design of the corresponding heuristic information matrix groups. Subsequently, 4.3.9 illustrates the collision detection and resolution procedure used to adjust the solution to adhere to the conflict-free constraint. Then, 4.3.10 expounds on how the constructed solution is decoded to yield a conflict-free schedule along with an AGV battery switching plan. 4.3.11 provides an exhaustive description of a solution improvement algorithm. Lastly, in 4.3.12, the pseudocode for the proposed EMOACA is presented.

4.3.3 Update of the External Nondominated Solutions Archive and Best Solutions

Following the construction and energy-reduction enhancement of each new solution, denoted as x^i, by ant i, its objective values are meticulously compared with those of the nondominated solutions stored in the external archive. If none of the solutions within the archive dominates x^i, then x^i is incorporated into the archive, and any solutions that were formerly dominated by x^i are subsequently removed. These archived solutions collectively form the definitive PF approximation produced by the EMOACA. For the purpose of updating the best solution, denoted as x_{best}^{i}, generated by ant i, the algorithm calculates the Goal Attainment Error $GE_i(t)$. This metric quantifies how closely the best solution, x_{best}^{i}, aligns with the reference point $RP_i(t)$ that is specified at iteration t. If the best solution x_{best}^{i} has its objective vector F_i^*, the value of $GE_i(t)$ its calculated as follows:

$$GE_i(t) = |F_i^* - RP_i(t)|^2 \tag{4.68}$$

If the Goal Attainment Error of a newly generated solution, x^i, is lower than that of the current best solution, x_{best}^{i}, then x_{best}^{i} is updated to be equal to x^i.

4.3.4 Reference Point Setting

The reference points are derived from a uniform sampling of an image front, which is shaped based on the convex front formed by solutions within the archive. The convex front represents the lower-left boundary of the convex hull that encompasses the solutions in the objective space. In contrast, the image front is a collection of line segments from which the reference points will be uniformly sampled. To create this image front, an image of the convex front is generated using a parameter called θ, where θ lies in the open interval (0, 1). The process begins by selecting a subset of solutions from the archive that forms a convex and nondominated front. Then, an image set of points is computed from this selected convex subset. Let F_i represent the objective vector of a point within the convex archive, with F_i^1 and F_i^2 denoting its respective objective values. Let n be the number of selected convex points, where n is greater than or equal to 2. The coordinates IM_i^1 and IM_i^2 of the corresponding image point of convex point i in the objective space are computed as:

$$\begin{cases} IM_1^1 = F_1^1(1-\theta) \\ \\ IM_1^2 = F_1^2 \end{cases} \qquad \begin{cases} IM_i^1 = F_i^1\left[1-\theta\left(1-\dfrac{\sum_{j=2}^{i}\left|F_j-F_{j-1}\right|}{\sum_{j=2}^{n}\left|F_j-F_{j-1}\right|}\right)\right] & i = 2,3,\dots,n \\ \\ IM_i^2 = F_i^2\left(1-\theta\dfrac{\sum_{j=2}^{i}\left|F_j-F_{j-1}\right|}{\sum_{j=2}^{n}\left|F_j-F_{j-1}\right|}\right) & i = 2,3,\dots,n \end{cases}$$

After calculating the image points, the image front is created by connecting each consecutive pair of image points with line segments. Subsequently, the reference points are uniformly selected from the image front to ensure the diversity and even distribution of solutions. Fig. 4.3 provides an illustration of how the reference points are generated from the archive solutions, assuming a basic population size of 16 ants and an image front parameter value of 0.5. The image parameter θ plays a crucial role in determining the distance of the reference points from the best solutions stored in the archive. When θ is set close to 0, the image front closely aligns with the convex front and nearly overlays it. If θ is increased, the image front will extend towards the bottom-left, resulting in reference points being placed further away from the archived solutions, until the two extreme points reach the two axes of the graph if θ

is set equal to 1. Each time a new solution is added to the archive, the reference points are updated accordingly.

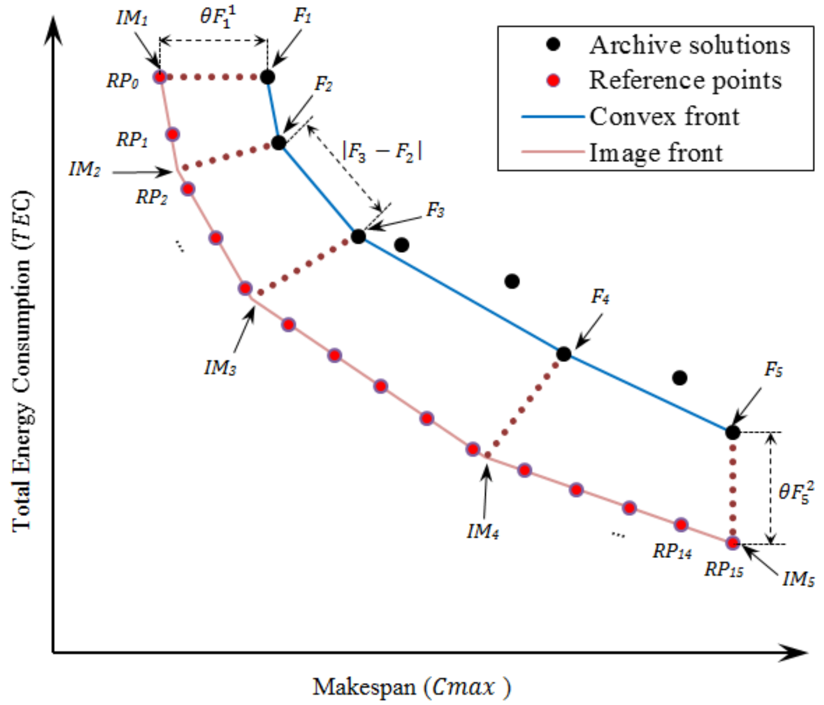

Fig. 4.3 Image front and reference point setting for a basic population size $N = 16$ and image factor $\theta = 0.5$.

4.3.5 Search Improvement Strategies

To enhance the search capability of the EMOACA, several techniques have been devised to bolster the probability transition model and augment the exploration process. These techniques are elaborated upon in the subsequent sections.

4.3.5.1 Pairwise Exchange Mutation (PEM) for Job Sequence

As numerous ants can simultaneously contribute to updating the same pheromone matrices, a substantial amount of pheromone can accumulate on the same edges, potentially leading to rapid saturation of these matrices. This is particularly prominent for later positions in the job sequence, where the number of alternative choices is limited compared to earlier positions. To address this issue, the PEM incorporates randomness into the generated job sequence by selecting two random jobs and exchanging their positions. The likelihood of applying the PEM is directly regulated by the exploration parameter ε.

4.3.5.2 Low-High Resolution Search (LHRS) Strategy for AGV Speed Setting

The Low-High Resolution Search (LHRS) strategy aims to explore the AGV speed settings space in a low-resolution manner during the first half of the search process. This is achieved by modifying the transition model to choose a single speed setting for all operations in the schedule. Individual operation speed settings can be fine-tuned later through collision correction or energy reduction algorithms, if necessary. However, for updating the best solutions and pheromone matrices, only the originally selected speed setting is considered. In the second half of the search, the transition model shifts to a high-resolution mode where it selects a specific speed setting for each operation, allowing for a more accurate approximation of the optimal solution after the algorithm has had enough time to learn the general structure of the optimal speed settings.

4.3.5.3 Periodic Pheromone Matrix Reset (PPMR)

To prevent premature convergence and escape from local optima, a periodic pheromone reset technique has been devised. This method revitalizes the algorithm's exploration behavior by resetting all pheromone matrices to their maximum pheromone level τ_{max} three times throughout the search process.

4.3.6 Three-Stage Solution Construction

To create a new solution for the EFSP-BCFT using the EMOACA, a problem-specific transition probability model has been devised, taking into account the problem's unique features. The model employs two sets of pheromone matrices and heuristic information matrices to guide each ant in constructing a solution. This construction process occurs in three stages:

- **Stage 1.** Job Sequence Specification where the model determines the sequence of jobs;
- **Stage 2.** Transportation Speed Selection where a transportation speed setting is selected for each operation in the schedule;
- **Stage 3.** Operation Sequence Assignment where the model assigns a sequence of operations to each AGV based on the decisions made in the preceding stages.

In the subsequent sections, it is assumed that ant i is associated with matrices indexed as g_1 and g_2. Ant i generates a new solution utilizing a pseudo-random transition probability model, which takes into account three key factors: the pheromone information matrices τ^{g_1} and τ^{g_2}, the heuristic information matrices η^{g_1} and η^{g_2}, and the current solution x^i.

4.3.6.1 Transition Rule for Job Sequence Stage

The probability of selecting job j in the k^{th} position in the sequence (P_{jk}) is given by:

$$P_{jk} = \begin{cases} 1 \quad \text{if} \quad j = \arg\max\limits_{j \in CL_k} \left\{ \gamma \left(\eta_{pj}^{g_1}\right)^{\lambda} \left(\tau_{pj}^{g_1}\right)^{\text{phj}(k)} + (1-\gamma) \left(\eta_{pj}^{g_2}\right)^{\lambda} \left(\tau_{pj}^{g_2}\right)^{\text{phj}(k)} \right\} & \& \quad rand \leq q \\ \dfrac{\gamma \left(\eta_{pj}^{g_1}\right)^{\lambda} \left(\tau_{pj}^{g_1}\right)^{\text{phj}(k)} + (1-\gamma) \left(\eta_{pj}^{g_2}\right)^{\lambda} \left(\tau_{pj}^{g_2}\right)^{\text{phj}(k)}}{\sum_{l \in CL_k} \gamma \left(\eta_{pl}^{g_1}\right)^{\lambda} \left(\tau_{pl}^{g_1}\right)^{\text{phj}(k)} + (1-\gamma) \left(\eta_{pl}^{g_2}\right)^{\lambda} \left(\tau_{pl}^{g_2}\right)^{\text{phj}(k)}} & \text{if} \quad rand > q \end{cases}$$

Here, p represents the job at the $(k-1)^{\text{th}}$ position in the sequence, and CL_k is the candidate list comprising all unscheduled jobs up to position k. phj(k) and λ denote the pheromone and heuristic information power factors, while $rand$ is a random number generated within the range of 0 to 1. Additionally, q represents the parameter for selecting the best edge. It's important to note that the transition model incorporates a pheromone power factor for each position k, which is an innovative addition to the conventional transition model. This is done for the same reasons as for why the Pairwise Exchange Mutation was introduced.

4.3.6.2 Transition Rule for AGV Speed Selection Stage

As delineated in section 4.3.5 , the transition rule for AGV speed selection allows for two modes. In the low resolution search (LRS) mode the probability P_v of selecting AGV speed setting v where $v \in S$, for all transfer operations is given by:

$$P_v = \begin{cases} 1 \quad \text{if} \;\; v = \arg\max_{v \in S} \left\{ \gamma \left(\eta_v^{g_1}\right)^{\lambda} \left(\tau_v^{g_1}\right)^{\text{phs}} + (1-\gamma)\left(\eta_v^{g_2}\right)^{\lambda} \left(\tau_v^{g_2}\right)^{\text{phs}} \right\} & \& \;\; rand \le q \\ \dfrac{\gamma \left(\eta_v^{g_1}\right)^{\lambda} \left(\tau_v^{g_1}\right)^{\text{phs}} + (1-\gamma)\left(\eta_v^{g_2}\right)^{\lambda} \left(\tau_v^{g_2}\right)^{\text{phs}}}{\sum_{w \in S} \gamma \left(\eta_w^{g_1}\right)^{\lambda} \left(\tau_w^{g_1}\right)^{\text{phs}} + (1-\gamma)\left(\eta_w^{g_2}\right)^{\lambda} \left(\tau_w^{g_2}\right)^{\text{phs}}} & \text{if} \;\; rand > q \end{cases}$$

In the high resolution search (HRS) mode, the probability P_{kjv} of selecting AGV speed setting v where $v \in S$, for transfer of the k^{th} job to machine j is given by:

$$P_{kjv} = \begin{cases} 1 \quad \text{if} \;\; v = \arg\max_{v \in S} \left\{ \gamma \left(\eta_{kjv}^{g_1}\right)^{\lambda} \left(\tau_{kjv}^{g_1}\right)^{\text{phs}} + (1-\gamma)\left(\eta_{kjv}^{g_2}\right)^{\lambda} \left(\tau_{kjv}^{g_2}\right)^{\text{phs}} \right\} & \& \;\; rand \le q \\ \dfrac{\gamma \left(\eta_{kjv}^{g_1}\right)^{\lambda} \left(\tau_{kjv}^{g_1}\right)^{\text{phs}} + (1-\gamma)\left(\eta_{kjv}^{g_2}\right)^{\lambda} \left(\tau_{kjv}^{g_2}\right)^{\text{phs}}}{\sum_{w \in S} \gamma \left(\eta_{kjw}^{g_1}\right)^{\lambda} \left(\tau_{kjw}^{g_1}\right)^{\text{phs}} + (1-\gamma)\left(\eta_{kjw}^{g_2}\right)^{\lambda} \left(\tau_{kjw}^{g_2}\right)^{\text{phs}}} & \text{if} \;\; rand > q \end{cases}$$

where phs is the pheromone power factor used for the speed selection stage.

4.3.6.3 Transition Rule for AGV Dispatching Stage

The probability P_{pd} of selecting operation d right after operation p such that $d \in OCL$, where OCL is the candidate list of operations eligible to be scheduled in a given position of the AGV dispatching sequence is given by:

$$P_{pd} = \begin{cases} 1 \;\; \text{if} \;\; v = \arg\max_{p \in OCL} \left\{ \gamma \left(\eta_{pd}^{g_1}\right)^{\lambda} \left(\tau_{pd}^{g_1}\right)^{\text{pho}} + (1-\gamma)\left(\eta_{pd}^{g_2}\right)^{\lambda} \left(\tau_{pd}^{g_2}\right)^{\text{pho}} \right\} & \& \;\; rand \le q \\ \dfrac{\gamma \left(\eta_{pd}^{g_1}\right)^{\lambda} \left(\tau_{pd}^{g_1}\right)^{\text{pho}} + (1-\gamma)\left(\eta_{pd}^{g_2}\right)^{\lambda} \left(\tau_{pd}^{g_2}\right)^{\text{pho}}}{\sum_{u \in OCL} \gamma \left(\eta_{pu}^{g_1}\right)^{\lambda} \left(\tau_{pu}^{g_1}\right)^{\text{pho}} + (1-\gamma)\left(\eta_{pu}^{g_2}\right)^{\lambda} \left(\tau_{pu}^{g_2}\right)^{\text{pho}}} & \text{if} \;\; rand > q \end{cases}$$

where pho is the pheromone power factor used in the AGV dispatching stage. Initially, the OCL contains the operation of the first job on the first machine. Whenever a new operation is selected, it is removed from OCL, and a new set of schedulable operations is added to the candidate list based on the previously scheduled operations. This ensures that the transition rule consistently produces feasible solutions that adhere to both flow shop constraints and blocking constraints. To clarify, an operation $OP_{i[j]}$ of a job in position j on machine i is added to OCL if $OP_{i-1,[j]}$, $OP_{i,[j-1]}$ and $OP_{i+1,[j-1]}$ were all previously scheduled. It's important to note that the total number of the pheromone matrix entries for stage 3 is $\Omega(mn)^2$, which can be substantial for large-size problems. Additionally, a significant portion of these matrices will never be utilized for constructing feasible solutions. To mitigate the space requirements of EMOACA, we propose a novel representation for the pheromone matrices in stage 3 using two registers: an Operations Register for each matrix g, containing any operation d selected after each operation p at any point during the search denoted OR_{gpd}; and a pheromone register, equivalent to the pheromone matrix τ_{pd}^{g}, for each matrix g, holding the pheromone values of all edges connecting any operation d selected after each operation p at any point during the search. If a new edge $p \to d$ (i.e., selecting operation d immediately after operation p) is chosen by ant i, with corresponding matrix groups g_1 and g_2 at iteration t, and this edge has not been selected since the beginning of the search or the last Periodic Pheromone Matrix Reset (PPMR), then the pheromone levels $\tau_{pd}^{g_1}(t)$ and $\tau_{pd}^{g_2}(t)$ at iteration t can be straightforwardly set to $\max\{\tau_{min}, \tau_{max}(1-\beta)^{t'}\}$, where t' counts the number of iterations since the last PPMR. These values can then be stored in the pheromone registers of g_1 and g_2, along with the corresponding edge $p \to d$ stored in $OR_{g_1,pd}$ and $OR_{g_2,pd}$.

Fig. 4.4 illustrates the three-stage solution construction and AGV assignment process. In the first stage, the job sequence is established as J2 - J1 - J4 - J3. Moving to the second stage, if the current number of iterations is less than half of $maxiter$, the Low-Resolution Search (LRS) mode is employed for AGV speed setting. This entails selecting setting 3 for all operations. Conversely, if the current number of iterations exceeds half of $maxiter$, the High-Resolution Search (HRS) mode is initiated, where a specific setting is chosen for each operation. In the third and final stage, the AGV operation sequence is created, and AGVs are assigned to operations. To assign an AGV to transport a job from station $i-1$ to station i, the AGV dispatching algorithm scans anticlockwise from station $i-1$ until it locates the closest available AGV. If multiple AGVs are found at the same position, the algorithm selects the earliest AGV assigned to that position to prevent collisions. For example, to transport job 1 from machine 2 to machine 3, AGV A1 is assigned since it is stationed at station 1 after its previous operation, while A2 is stationed at station 4. However, for the task of transporting job 4 from machine 1 to machine 2, A2 is chosen because both AGVs were initially at station 4, but A2 is ahead of A1 in that location.

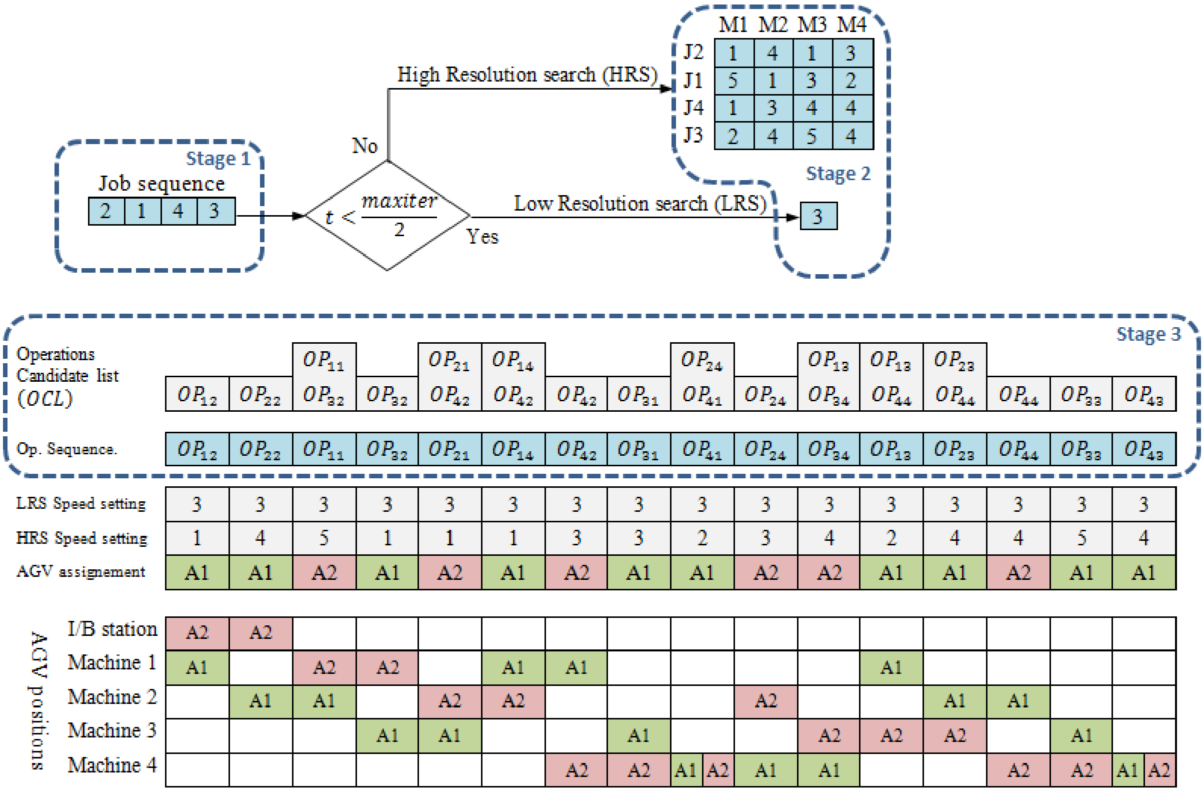

Fig. 4.4 An example of the solution representation and construction for an instance with 4 jobs, 4 machines, 2 AGVs and 5 different speed settings.

To enhance the user-friendliness of the algorithm and control its exploration tendency with more ease, it is proposed to establish a relationship between all the pheromone power factors phj(k) for $k = 1, 2, \ldots, n$, phs, and pho with a common exploration parameter ε. This way, by adjusting the value of ε, users can regulate the power factors. Higher ε values correspond to lower power factor values, resulting in more evenly distributed transition probabilities. The specific linkage between power factors and ε will be elaborated upon in the experimental section of this chapter (refer to section 4.4.2).

4.3.7 Pheromone Trail Update

The pheromone trail update involves two key steps: a local update that occurs while constructing a solution and a global update that takes place at the end of each iteration. Let's assume that ant i is associated with matrices indexed as g_1 and g_2, with pheromone-dropping fractions of γ and $1-\gamma$. The updated pheromone trail, after selecting job j immediately following job p, is determined as follows:

$$\tau_{pj}^{g_1} \leftarrow \tau_{pj}^{g_1}(1-\alpha\gamma) \qquad \tau_{pj}^{g_2} \leftarrow \tau_{pj}^{g_2}\big(1-\alpha(1-\gamma)\big)$$

After all ants have constructed their solutions, the algorithm proceeds to the second step, where the original pheromone matrix, before the local update, is utilized for the global update. In this step, both evaporation and reinforcement are applied to the pheromone matrix. The global pheromone evaporation for the job sequence matrix τ^g at iteration t is calculated as follows:

$$\tau_{pj}^g(t+1) \leftarrow (1-\beta)\tau_{pj}^g(t) \quad \forall p, j \in J, \;\; g = 0, 1, \dots, \Omega - 1$$

Next, each ant i updates its corresponding pheromone matrices τ^{g_1} and τ^{g_2} by depositing a quantity of pheromone that is proportional to its contribution fraction to each matrix and also a function of its best performance so far (x_{best}^i).

$$\tau_{pj}^{g_1}(t+1) \leftarrow \tau_{pj}^{g_1}(t) + \gamma z_i(t) \qquad \text{if } \; p \to j \in x_{best}^i$$

$$\tau_{pj}^{g_2}(t+1) \leftarrow \tau_{pj}^{g_2}(t) + (1-\gamma) z_i(t) \quad \text{if } \; p \to j \in x_{best}^i$$

where $z_i(t)$ represents the amount of pheromone deposited by ant i after iteration t. This pheromone is added to the entry (p, j) of the matrices only if job j is scheduled immediately after job p in the sequence derived from ant i's best solution x_{best}^i. The value of $z_i(t)$ is determined by the following equation:

$$z_i(t) = \min\left\{1, \frac{c \cdot t}{maxiter}\right\}$$

where c represents the pheromone intensity. The value of $z_i(t)$ increases gradually as the search progresses toward the optimal Pareto front, reaching a maximum value of 1. Unlike traditional single-objective minimization problems, the pheromone quantities in this study are not directly tied to objective function values.

The updated pheromone trail for AGV speed setting v in both LRS and HRS modes is calculated as follows:

LRS: $\tau_v^{g_1} \leftarrow \tau_v^{g_1}(1-\alpha\gamma)$ $\qquad$ $\tau_v^{g_2} \leftarrow \tau_v^{g_2}\big(1-\alpha(1-\gamma)\big)$

HRS: $\tau_{kjv}^{g_1} \leftarrow \tau_{kjv}^{g_1}(1-\alpha\gamma)$ $\qquad$ $\tau_{kjv}^{g_2} \leftarrow \tau_{kjv}^{g_2}\big(1-\alpha(1-\gamma)\big)$

The global pheromone evaporation for the speed setting matrix τ^g at iteration t is defined as follows:

LRS: $\tau_v^g(t+1) \leftarrow (1-\beta)\tau_v^g(t) \qquad \forall v \in S, \;\; g = 0, 1, \dots, \Omega - 1$

HRS: $\tau_{kjv}^g(t+1) \leftarrow (1-\beta)\tau_{kjv}^g(t) \quad \forall k \in J, \;\; \forall j \in M, \;\; \forall v \in S, \;\; g = 0, 1, \dots, \Omega - 1$

The global pheromone reinforcement for the AGV speed setting matrix, considering ant i, its personal best solution x_{best}^i, and the associated matrices g_1 and g_2 after iteration t, can be described as follows:

$$\text{LRS}\begin{cases} \tau_v^{g_1}(t+1) \leftarrow \tau_v^{g_1}(t) + \gamma z_i(t) & \text{if } \; v \in x_{best}^i \\ \tau_v^{g_2}(t+1) \leftarrow \tau_v^{g_2}(t) + (1-\gamma) z_i(t) & \text{if } \; v \in x_{best}^i \end{cases}$$

$$\text{HRS}\begin{cases}\tau_{kjv}^{g_1}(t+1) \leftarrow \tau_{kjv}^{g_1}(t) + \gamma z_i(t) & \text{if} \ \ (k,j,v) \in x_{best}^i \\ \\ \tau_{kjv}^{g_2}(t+1) \leftarrow \tau_{kjv}^{g_2}(t) + (1-\gamma) z_i(t) & \text{if} \ \ (k,j,v) \in x_{best}^i \end{cases}$$

This amount of pheromone $z_i(t)$ is deposited on the entry (k, j, v) of the pheromone matrices only if the AGV speed setting v is selected for operation of job k on machine j in ant i's best solution x_{best}^i. The new local pheromone trail after scheduling operation d right after operation p in the AGV dispatching sequence is given by the following new assignments:

$$\tau_{pd}^{g_1} \leftarrow \tau_{pd}^{g_1}(1-\alpha\gamma) \qquad\qquad \tau_{pd}^{g_2} \leftarrow \tau_{pd}^{g_2}\big(1-\alpha(1-\gamma)\big)$$

The global pheromone evaporation for the operation sequence matrix τ^g at iteration t is determined by the following formula:

$$\tau_{pd}^{g}(t+1) \leftarrow (1-\beta)\tau_{pd}^{g}(t) \quad \forall p, d, \quad g = 0, 1, \dots, \Omega - 1$$

The global pheromone reinforcement of the AGV dispatching matrix, associated with ant i's personal best solution x_{best}^i and the relevant matrices indexed as g_1 and g_2, is carried out after iteration t using the following procedure:

$$\tau_{pd}^{g_1}(t+1) \leftarrow \tau_{pd}^{g_1}(t) + \gamma z_i(t) \qquad \text{if} \ \ (p \to d) \in x_{best}^i$$

$$\tau_{pd}^{g_2}(t+1) \leftarrow \tau_{pd}^{g_2}(t) + (1-\gamma) z_i(t) \ \ \text{if} \ \ (p \to d) \in x_{best}^i$$

The quantity of pheromone, denoted as $z_i(t)$, is applied to the entry (p, d) of the pheromone matrices solely if operation d is scheduled immediately after operation p within the AGV dispatching sequence derived from ant i's optimal solution, x_{best}^i.

4.3.8 Heuristic Information

In the upcoming subsections, w_g is introduced as the weight linked to matrix index g, which is determined using the following equation:

$$w_g = \frac{g}{\Omega - 1}$$

4.3.8.1 Heuristic Information for Job Sequence Stage

The g^{th} heuristic information η_{jk}^{g}, used to select job k immediately following job j in the sequence, incorporates two elements: the time distance δ_{jk}^{T} and energy distance δ_{jk}^{E} between these two jobs. δ_{jk}^{T} signifies the makespan of the partial sequence formed by scheduling job k after job j, disregarding transportation constraints, while δ_{jk}^{E} accounts for the total setup energy between k and j. The time distance can be calculated through the recursive relations presented in (4.69).

$$\begin{cases} c_1 = P_{1j} + ST_{1jk} + P_{1k} \\ c_i = \max\left\{c_{i-1}, \sum_{l=1}^{i} P_{lj} + ST_{ijk}\right\} + P_{ik} \end{cases} \tag{4.69}$$

$$\delta_{jk}^{T} = c_m, \quad \delta_{jk}^{E} = \sum_{i=1}^{m} SE_{ijk}$$

$$\delta_{jk}^{g} = \left(1 - w_g\right)\overline{\delta_{jk}^{T}} + w_g\overline{\delta_{jk}^{E}} \tag{4.70}$$

$$\overline{\delta_{jk}^{T}} = \frac{\delta_{jk}^{T} - \min_{l,p\in J}\{\delta_{lp}^{T}\}}{\max_{l,p\in J}\{\delta_{lp}^{T}\} - \min_{l,p\in J}\{\delta_{lp}^{T}\}}$$

$$\overline{\delta_{jk}^{E}} = \frac{\delta_{jk}^{E} - \min_{l,p\in J}\{\delta_{lp}^{E}\}}{\max_{l,p\in J}\{\delta_{lp}^{E}\} - \min_{l,p\in J}\{\delta_{lp}^{E}\}}$$

Subsequently, the values of δ_{jk}^{T} and δ_{jk}^{E} are normalized within the range of 0 to 1 to derive $\overline{\delta_{jk}^{T}}$ and $\overline{\delta_{jk}^{E}}$. Following this, the normalized distance δ_{jk}^{g} between j and k, designated for matrix g, is calculated as illustrated in (4.70).

Ultimately, the g^{th} heuristic information η_{jk}^{g}, utilized for choosing job k immediately following job j in the sequence, is determined through the following calculation:

$$\eta_{jk}^{g} = 1 - \sigma\frac{\delta_{jk}^{g} - \min_{l,p\in J}\{\delta_{lp}^{g}\}}{\max_{l,p\in J}\left\{\delta_{lp}^{g}\right\} - \min_{l,p\in J}\left\{\delta_{lp}^{g}\right\}} \tag{4.71}$$

where $\sigma \in (0, 1)$ is the heuristic information shape parameter.

4.3.8.2 Heuristic Information for AGV Speed Stage

The g^{th} heuristic information η_{v}^{g}, used for selecting speed setting v for all transfer operations in LRS mode, is defined as follows:

$$\eta_{v}^{g} = 1 - \sigma\frac{\left|v - w_g v_{max}\right|}{v_{max}} \tag{4.72}$$

The g^{th} heuristic information η_{ijv}^{g}, used for selecting setting v for the transfer of the i^{th} job on machine j in HRS mode, is defined as follows:

$$\eta_{ijv}^{g} = 1 - \sigma\frac{\left|v - w_g v_{max}\right|}{v_{max}} \tag{4.73}$$

The heuristic information is biased towards higher speeds if $Cmax$ is emphasized and lower speeds if TEC is emphasized.

4.3.8.3 Heuristic Information for Dispatching Stage

In contrast to the previous stages where heuristic information matrices are precomputed and fixed before the search begins, the AGV dispatching heuristic information is algorithmically computed during the construction of the AGV dispatching sequence. Let $d \in OCL$ represent a candidate operation to be scheduled after the previously selected operation p, and let d denote the operation of the j^{th} job in the constructed job sequence on machine i. Additionally, let u be the actual distance from the assigned AGV's position to station $i-1$, and s_{ij} be the actual AGV speed when selecting setting v for operation d. Similar to the job sequencing heuristic information, we define two types of distances between operation p and d: a time distance δ^T_{pd} and an energy distance δ^E_{pd}, which are computed using the equations (4.74); (4.75).

$$\delta^T_{pd} = \begin{cases} \dfrac{D_{i-1,i}}{s_{ij}} + P_{i-1,j} & \text{if } i > 1 \;\&\; u = 0 \\ \dfrac{u + D_{i-1,i}}{s_{ij}} & \text{otherwise} \end{cases} \tag{4.74}$$

$$\delta^E_{pd} = \delta^T_{pd} IE + TE_v(u + D_{i-1,i}W) \tag{4.75}$$

$$\delta^g_{pd} = (1 - w_g)\overline{\delta^T_{pd}} + w_g\overline{\delta^E_{pd}} \tag{4.76}$$

$$\overline{\delta^T_{pd}} = \frac{\delta^T_{pd} - \min\limits_{h\in OCL}\{\delta^T_{ph}\}}{\max\limits_{h\in OCL}\{\delta^T_{ph}\} - \min\limits_{h\in OCL}\{\delta^T_{ph}\}}$$

$$\overline{\delta^E_{jk}} = \frac{\delta^E_{jk} - \min\limits_{h\in OCL}\{\delta^E_{ph}\}}{\max\limits_{h\in OCL}\{\delta^E_{ph}\} - \min\limits_{h\in OCL}\{\delta^E_{ph}\}}$$

δ^T_{pd} estimates the time required for the assigned AGV to travel to machine $i-1$ and transfer the job to machine i, while δ^E_{pd} estimates the amount of energy required for this process, both assuming no battery switching or intermediary waiting. As before, these values are normalized to obtain δ^T_{pd} and δ^E_{pd}. The heuristic information η^g_{pd} is then computed similarly, as shown in the following equation:

$$\eta^g_{pd} = 1 - \sigma\frac{\delta^g_{pd} - \min\limits_{h\in OCL}\{\delta^g_{ph}\}}{\max\limits_{h\in OCL}\{\delta^g_{ph}\} - \min\limits_{h\in OCL}\{\delta^g_{ph}\}} \tag{4.77}$$

In a way, the heuristic information assesses the desirability of selecting an operation where the assigned AGV waits for $P_{i-1,j}$ time to transfer the same job it previously transported, versus selecting an operation where the AGV performs an unloaded move. After applying the transition model and selecting an operation from the candidate list, the assigned AGV's position is updated, and the process is reiterated until all operations are scheduled. Algorithm 4.1 outlines the probability transition rule and AGV assignment procedure for the AGV dispatching stage.

Algorithm 4.1 Probability_transition_and_AGV_assignment_procedure_for_AGV_dispatching_stage

Inputs: Candidate list of schedulable operations (OCL); Previously selected operation (p); Operations register for each matrix g containing all operations d selected after operation p (OR_{gpd}); Pheromone register for each matrix g containing pheromone values of all edges connecting operations d selected after operation p (τ^g_{pd}); EMOACA's current iteration (t)

Outputs: Selected operation of j^{th} job in machine i (OP_{ij}); Index of the selected operation in candidate list (k^*); Assigned AGV (r)

Define CLP_k as probability to select k^{th} operation in OCL where $|CLP| = |OCL|$ and $CLP_k = 0$ for all entries k;
$k^* = 1$; // Initialize selected operation index
$CLP_{max} = 0$; // Initialize best edge with maximum probability
For $k = 1$ to $|OCL|$

$d = OCL_k$;
Identify closest AGV r and compute actual distance u;
Compute normalized time and energy distances $(\overline{\delta_{pd}^T}\ ;\ \overline{\delta_{pd}^E})$;
Compute heuristic information for related matrices g_1 and g_2 where $\eta_{pd}^g = (1 - w_g)\overline{\delta_{pd}^T} + w_g \overline{\delta_{pd}^E}$;
$a = 1$; // a is used to iterate through the selected operations in the operations register
While $a \leq |OR_{g_1,p}|$ **do** // Search for pheromone level of edge $p \rightarrow d$ in operations register
If d is in $OR_{g_1,p}$ **then** // If entry $OR_{g_1,p,d}$ already exists
$a = |OR_{g_1,p}| + 1$; // To break from while loop
$CLP_k = CLP_k + \gamma(\eta_{pd}^{g_1})^\lambda (\tau_{pd}^{g_1})^{\text{pho}}$;
End if
$a = a + 1$;
End while
If $a = |OR_{g_1,p}| + 1$ **then** // If edge $p \rightarrow d$ was not selected before t
$CLP_k = CLP_k + \gamma(\eta_{pd}^{g_1})^\lambda (\max\{\tau_{min}, \tau_{max}(1 - \beta)^t\})^{\text{pho}}$;
Add d to $OR_{g_1,p}$ and record pheromone value $\max\{\tau_{min}, \tau_{max}(1 - \beta)^t\}$ as $\tau_{pd}^{g_1}$ in pheromone register
End if
Repeat previous 12 lines for g_2 with $1 - \gamma$ fraction;
If $CLP_k > CLP_{max}$ **then** // Update best edge index
$CLP_{max} = CLP_k$;
$k^* = k$;
End if
End for
If $rand > q$ **then**
Perform roulette wheel to select an operation index k^* from OCL according to transition rule;
End if
Output selected operation $OP_{ij} = OCL_{k^*}$ with index k^* and closest AGV r as assigned AGV;

4.3.9 Collision Detection and Resolution

Even though AGVs are assigned to operations in a manner that prevents overtaking and preserves their order in the single-loop guide path, collisions can still occur due to variations in AGV movement, including differences in speed and stops between unloaded and loaded moves as AGVs wait for job completion. Consequently, a collision detection and resolution method has been incorporated to rectify AGV movement. This method involves adjusting AGV speeds and, when required, adding mandatory idle times to AGV movement planning to entirely prevent collisions among AGVs.

In the MILP model, collision detection is primarily based on the cumulative distance traveled by AGVs. The AGVs are numbered from 1 to R, and each AGV $r + 1$ is not allowed to exceed the total distance traveled by the previous AGV r, $\forall r = 1 \ldots R - 1$. Furthermore, AGV 1 is not permitted to overtake AGV R during the scheduling process. In the proposed EMOACA model, if a collision is detected after scheduling an operation, it is resolved using two distinct methods. In the first method, the algorithm iteratively reduces the AGV's speed, one setting at a time, for the portion of the operation where the collision occurred. This reduction continues until the partial schedule becomes feasible without collisions. However, if further speed reduction doesn't eliminate the collision, the second method is employed. This second approach involves imposing a blocking wait time at the location where the collision was detected. In this case, the AGV waits until the track is clear before resuming its transportation operation at the originally selected speed setting. To illustrate this collision detection and resolution procedure, consider the example depicted in Fig. 4.5 below. Suppose that the operation for AGV r (shown in blue) has already been scheduled, and when scheduling the operation for AGV $r+1$ (represented by the light red dashed line) with the chosen speed setting v, a collision occurs during AGV $r+1$'s unloaded travel portion of the operation. Initially, the first step of collision correction reduces the speed of the unloaded move to the minimum (indicated by the red dashed line). However, a collision still persists. To resolve this, the unloaded move is split into two segments: the first part employs the minimum speed, while the second part employs the originally selected speed. A blocking time interval is inserted between these two segments, ensuring a collision-free motion plan for AGV $r+1$'s operation (illustrated by the solid red line).

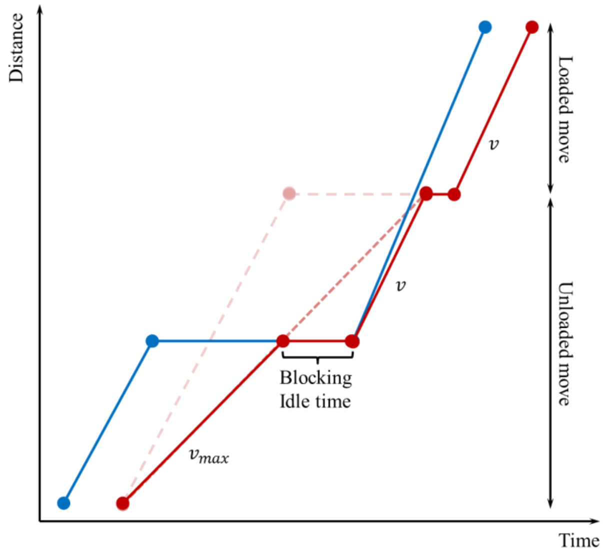

Fig. 4.5 Collision detection and resolution procedure

4.3.10 Solution Decoding, Schedule Construction and Battery Switch Planning

Once the solution has been represented in terms of job sequence, AGV speeds, and AGV operation sequence, the next step in the algorithm is to decode this representation and generate a feasible schedule. The schedule construction is carried out in an iterative manner, one operation at a time, following the order specified by the AGV operation sequence. First, a detailed explanation of the rules governing AGV planning for battery switch stops is provided. Following that, the methods used for avoiding collisions between consecutive AGVs to ensure a feasible solution for the EFSP-BCFT are addressed.

Since AGVs are only able to switch batteries at the input station, and the AGV track follows a unidirectional path, battery switch decisions must be made before commencing a new lap. Additionally, the maximum battery capacity should be sufficient to complete at least one lap. Given that AGVs consume energy even during idle states, accurately predicting whether the current battery energy level will allow for a new lap without requiring a battery swap at the I/B station is rather difficult. To address this, an estimate of idle energy consumption is calculated by creating a relaxed version of the actual schedule. This relaxed schedule assumes that as soon as jobs finish processing on machines, they are immediately unloaded and then transported by an AGV, operating at the selected speed setting v, to the next machine. In this estimation, AGV battery capacity is treated as infinite, collisions are disregarded, however, machine blocking constraints and SDSTs are considered when computing the relaxed start time for any job j^{th} at position i in the constructed sequence. This relaxed start time is denoted as S'_{ij}. The calculation proceeds as follows:

$$\begin{aligned}
&S'_{1,1} = ST_{1,0,1} && \\
&S'_{i,1} = \max\{S'_{i-1,1} + P_{i-1,1} + vD_{i-1,i}, ST_{i,0,1}\} && \forall i = 2, \dots, m \\
&S'_{1,j} = S'_{1,j-1} + P_{1,j-1} + ST_{1,j-1,j} && \forall j = 1, \dots, R \\
&S'_{1,j} = \max\{S'_{1,j-1} + P_{1,j-1} + ST_{1,j-1,j}, S'_{m,j-R} + vD_{m,1}\} && \forall j = R+1, \dots, n \\
&S'_{ij} = \max\{S'_{i,j-1} + P_{i,j-1} + ST_{i,j-1,j}, S'_{i-1,j} + P_{i-1,j} + vD_{i-1,j}\} && \forall i = 2, \dots, m, \forall j = 2, \dots, n
\end{aligned} \tag{4.78}$$

Using this relaxed schedule, an energy prediction is made each time the current AGV is about to pass the I/B station for its next operation. The prediction estimates whether the AGV will have sufficient battery charge to complete the new lap. It considers both the forecasted battery charge level at the end of the final transfer operation on this next lap and the energy required for the AGV's return trip to the I/B station. If this estimate falls below the 20% threshold, it signals that the AGV must undergo a battery switch before proceeding with the next operation. Algorithm 4.2 details the energy forecast and battery switch planning procedure.

Algorithm 4.2 AGV_energy_consumption_forecast_procedure

Inputs: AGV index (r); Previous operation ($OP_{i'j'}$); Next operation (OP_{ij}); AGV speed setting for next operation (v); AGV r's current battery consumption level (BL_r); AGV r's machine, job and speed setting sequences (MS_r, JS_r, SS_r), AGV r's next operation's index in sequence (o); Relaxed start times (S'); Battery capacity (C)

Outputs: AGV r's energy forecast (EF); Decision of whether to perform a battery switch before next operation OP_{ij}

If ($i < i'$) **then** // If AGV r needs to cross the I/B station to perform transfer operation OP_{ij}

  $EF = TE_v(D_{i',i-1} + D_{i-1,i}W) + 2AE$;

  $p = i$; $k = o + 1$;

  While ($MS_{rk} > p$) **do** // While the next operation in AGV r's operations sequence is within the same lap

    $x = MS_{rk}$; $v = SS_{rk}$;

    $EF = EF + TE_v(D_{p,x-1} + D_{x-1,x}W) + 2AE$;

    $p = x$; $k = k + 1$;

  End while

  $v = SS_{rk}$; $x = MS_{rk}$; $y = JS_{rk}$;

  $EF = EF + TE_vD_{p,0} + IE(S'_{xy} - S'_{i'j'})$; // Add travel energy to battery station and estimated total idle energy

  If ($BL_r + EF > 0.8C$) **then**

    Insert a battery switch operation before operation OP_{ij}

  End if

End if

4.3.11 Solution Improvement

An energy reduction algorithm has been developed to enhance the generated solutions, aiming to achieve improved performance by adjusting AGV speeds while maintaining the same makespan. This improvement algorithm focuses on identifying non-critical operations, which are operations that can tolerate some delay in their start times without increasing the schedule's maximum completion time. The algorithm leverages the available delay margins to reduce AGV speeds for these non-critical operations, consequently reducing AGV energy consumption while adhering to collision constraints throughout the process. The improvement algorithm begins by designating the last operation in the schedule as a critical operation, denoted as cop. Critical operations are defined as those with zero margin for delay and are therefore not affected by the algorithm. Each identified critical operation is added to a queue of critical operations, denoted as COP. The algorithm then selects the leading operation from this COP queue, denoted as cop^*, to initiate a *backward branching* process. Backward branching involves examining the operations that impose time constraints on the current critical operation, cop^*. The algorithm assesses whether each of these constraints is binding, in which case the corresponding operation is also considered critical and should be included in the COP queue. Alternatively, if the constraint is non-binding, the corresponding operation is labeled as non-critical, denoted as $ncop$, and is assigned a positive time margin. These non-critical operations are then placed in a separate queue denoted as $NCOP$ and are organized in descending order of their start times.

For each operation, denoted as OP_{ij}, corresponding to the j^{th} job in the sequence on machine i, up to four backward time constraints can be defined. Each constraint is associated with an operation resulting from the backward branching procedure:

1. The start time of the operation previously assigned to the same AGV, considering the travel time to station $i-1$ and then the transportation time from station $i-1$ to station i, if OP_{ij} is not the first operation allocated to the AGV.
2. The completion time of the j^{th} job on station $i-1$, provided that $i>1$, along with the AGV travel time from station $i-1$ to station i.
3. The completion time of $OP_{i,j-1}$ plus the setup time of machine i, if $j>1$.
4. The unloading time of the $(j-1)^{th}$ job from machine i, assuming $i<m$, in addition to machine i's setup time if $j>1$.

After the backward branching procedure is completed, the algorithm proceeds by exploring the $NCOP$ queue to identify opportunities for energy reduction. First of all, The algorithm starts by selecting the peak operation $ncop^*$ from the queue. If the assigned AGV's speed is already at its minimum, no further improvement can be made, and the algorithm proceeds to the next peak element in the $NCOP$ queue. However, if there's room for reducing the AGV's speed, the algorithm proceeds with a *forward/lateral branching* of $ncop^*$ to calculate the maximum AGV speed reduction achievable. The forward/lateral branching process involves identifying operations directly affected by the AGV speed reduction for the current noncritical operation and computing their time margins concerning the time constraints they impose on their immediate succeeding operations. For each operation OP_{ij} corresponding to the j^{th} job in the sequence on machine i, up to four forward operations and one lateral operation can be defined. The start time of each of these operations can be directly influenced by the speed setting for OP_{ij}. The start times are denoted and described as follows:

Forward operations:

1) S_1^F The start time of the next operation assigned to the same AGV if OP_{ij} is not the AGV's last operation
2) S_2^F The start time of $OP_{i+1,j}$ if $i < m$
3) S_3^F The start time of $OP_{i,j+1}$ if $j < n$
4) S_4^F The unloading time of the next job assigned to the same AGV plus the setup time in order to start a new job on the freed machine if it is not the input station and if OP_{ij} is not the last operation assigned to the AGV

Lateral operation:

5) S^L The start time of $OP_{i-1,j+1}$ after unloading the j^{th} job from machine $i-1$ plus the machine's setup time if $i > 0$

Fig. 4.6 shows the start times affected by the decrease of AGV speed on operation OP_{ij} and explains how the forward time margins are derived.

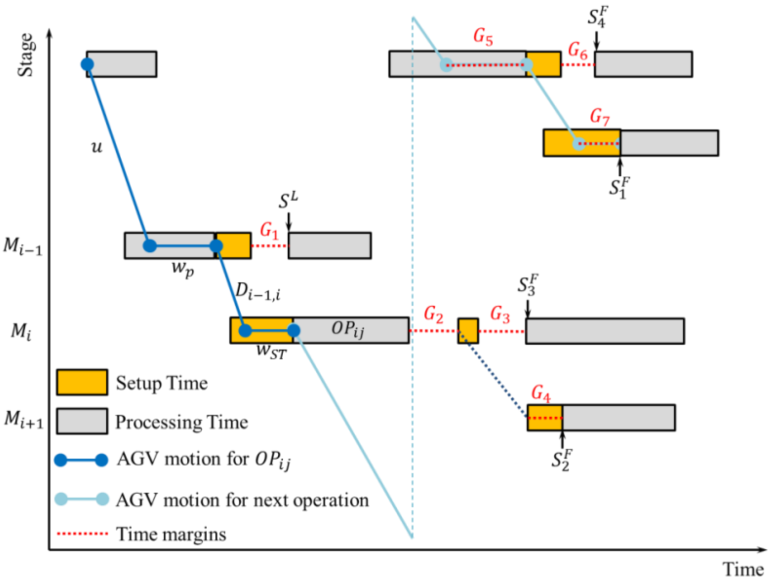

Fig. 4.6 Forward/lateral operations and all possible time margins relative to operation OP_{ij}.

The forward/lateral branching procedure generates child operations, and these operations are organized into a separate queue called $TREE$, which represents the exploration tree generated during the branching process. The $TREE$ queue is sorted in ascending order based on the operations' ΔS values. The ΔS value for each operation in the $TREE$ queue quantifies the maximum allowable speed reduction; taking into account the cumulative backward margin from that operation all the way to $ncop^*$. This allowable reduction depends on whether the operation originated from a forward or lateral branch of $ncop^*$. The calculation of ΔS values is as follows:

Forward branch operations:

$$\Delta S = \min\left\{\frac{CM + w_{ST}}{D_{i-1,i}T}, \frac{CM + w_P + w_{ST}}{(u + D_{i-1,i})T}\right\} \tag{4.79}$$

Lateral branch operations:

$$\Delta S = \frac{CM + w_P}{Tu} \tag{4.80}$$

where CM is the cumulative margin from the parent non-critical operation to the child operation. For the example shown in Fig. 4.6, CM values for each affected start time is computed as follows:

1) $S_1^F : G_5 + G_7$
2) $S_2^F : G_2 + G_4$
3) $S_3^F : G_2 + G_3$
4) $S_4^F : G_5 + G_6$
5) $S^L : G_1$

w_P represents the idle time of the AGV while the transferred job is being processed on machine M_{i-1}. w_{ST} represents the idle time of the AGV during the setup time of machine M_i. u denotes the distance traveled by the AGV when it is unloaded and heading to the loading site M_i, while T represents the increase in travel time per unit of distance when the AGV's speed is reduced by one setting.

Once the initial forward/lateral branching process is finished, and the first set of operations, referred to as the *Original tree*, is generated and placed in the $TREE$ queue, the algorithm initiates the exploration of this original tree. It identifies the maximum allowable speed reduction, ensuring no critical operation is delayed. In this new procedure, referred to *Forward deep branching*, the algorithm selects the highest-priority operation from the queue, denoted as $leaf^*$, and conducts the previously explained forward branching process to create new child operations. These new operations are then organized in ascending order of their ΔS values, which are computed based on their cumulative margin concerning $ncop^*$. The algorithm continues the forward deep branching until it reaches the "closest" critical operation, which eventually becomes the peak element of the $TREE$ queue. The forward deep branching subprogram determines the maximum achievable speed reduction ΔS^* for $ncop^*$. Assuming that the initial AGV speed setting for $ncop^*$ is v, the new speed setting v' is calculated as follows:

$$v' = \min\{v + \lfloor \Delta S^* \rfloor, v_{max}\}$$

After implementing the new AGV speed setting for $ncop^*$, the schedule is updated. In case a collision is detected, the algorithm makes corrections by incrementally increasing the speed setting by one unit at a time until the collision is resolved. Once the final speed setting is determined and accepted, all the newly identified critical operations during the procedure are added to COP. The algorithm then proceeds to select the next noncritical operation from $NCOP$ for a forward/lateral branching until all the operations in the queue have been explored. Subsequently, the algorithm conducts a backward branching on the following critical operation in COP to identify additional noncritical operations, continuing this process until all critical operations have been explored. Finally, the improvement algorithm terminates and provides the newly enhanced solution, characterized by reduced total energy consumption while maintaining the same maximum completion time. Fig. 4.7 illustrates an example of the three stages of the branching procedure, and Algorithm 4.3 outlines the pseudocode of the improvement algorithm as implemented in EMOACA.

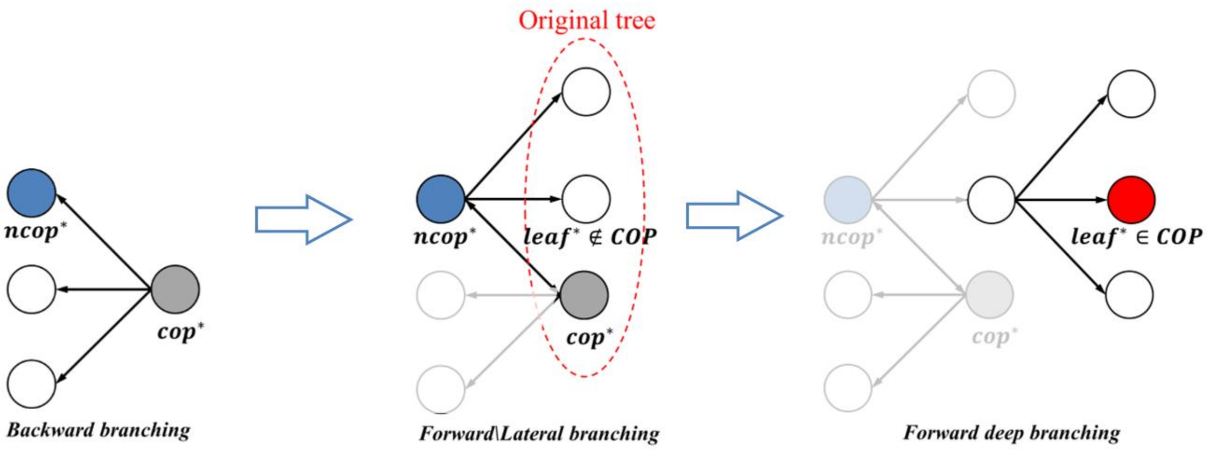

Fig. 4.7 An illustration of the three stages involved in the branching procedure

---

Algorithm 4.3 Energy_reduction_algorithm

Input: Complete feasible schedule (Sched) with start times, backward, forward and lateral time margins for each operation and $Cmax$ and TEC objective values;

Priority queues: Critical operations (COP) sorted in decreasing order of start times; Non-critical operations ($NCOP$) sorted in decreasing order of start times then decreasing order of stages; Tree of explored operations ($TREE$) sorted in increasing order of ΔS then in decreasing order of start times

Initialization

Insert the last operation of the Sched in COP;

Algorithm iteration

While ($|COP| > 0$) **do**

  Extract the peak critical operation from COP, call it cop^*;

  Perform backward branching on critical operation to obtain a new set $\mathcal{BB}(cop^*)$;

  For ($i = 1\ to\ |\mathcal{BB}(cop^*)|$) **do**

    If Margin$(\mathcal{BB}(cop^*)_i \rightarrow cop^*) = 0$ **then**

      Insert $\mathcal{BB}(cop^*)_i$ in COP;

    Else

      Insert $\mathcal{BB}(cop^*)_i$ in $NCOP$;

    End if

End for

While $(|NCOP| > 0)$ **do**

Extract the peak non-critical operation from $NCOP$, call it $ncop^*$;

If $(v < v_{max})$ **then** // where v is the current speed setting for operation $ncop^*$

Clear $TREE$;

Perform forward/lateral branching on $ncop^*$ to obtain a new set $\mathcal{FB}(ncop^*) \cup \mathcal{LB}(ncop^*)$;

For $(i = 1\ to\ |\mathcal{FB}(ncop^*) \cup \mathcal{LB}(ncop^*)|)$ **do**

// Compute cumulative margin of $\mathcal{FB}(ncop^*) \cup \mathcal{LB}(ncop^*)_i$ and deduce its ΔS value

$CM\left(\mathcal{FB}(ncop^*) \cup \mathcal{LB}(ncop^*)_i\right) = \text{Margin}\left(ncop^* \rightarrow \mathcal{FB}(ncop^*) \cup \mathcal{LB}(ncop^*)_i\right)$;

Compute ΔS for $\mathcal{FB}(ncop^*) \cup \mathcal{LB}(ncop^*)_i$ according to (4.79); (4.80);

Insert $\mathcal{FB}(ncop^*) \cup \mathcal{LB}(ncop^*)_i$ in $TREE$ with its cumulative margin value;

End for

// Exploring original tree (Deep forward branching subprogram)

While $(leaf^* \notin COP)$ **do** // where $leaf^*$ is the peak operation in $TREE$ queue

Extract peak operation $leaf^*$ from $TREE$ priority queue;

Perform forward branching on $leaf^*$ to obtain a new set $\mathcal{FB}(leaf^*)$;

For $(i = 1\ to\ |\mathcal{FB}(leaf^*)|)$ **do**

// Compute cumulative margin of $\mathcal{FB}(leaf^*)_i$ and deduce its ΔS value

Compute $CM(\mathcal{FB}(leaf^*)_i) = CM(leaf^*) + \text{Margin}(leaf^* \rightarrow \mathcal{FB}(leaf^*)_i)$

Compute ΔS for $\mathcal{FB}(leaf^*)_i$ according to (4.79); (4.80);

Insert $\mathcal{FB}(leaf^*)_i$ in $TREE$;

End for

End while

Denote maximum possible speed reduction of $leaf^*$ as ΔS^*;

Compute new speed setting $v' = \min\{v + \lfloor \Delta S^* \rfloor, v_{max}\}$;

Apply new speed setting and update the schedule (Sched^*);

While $(\text{Collision}(\text{Sched}^*) = \text{TRUE})$ **do**

$v' = v' - 1$;

Apply new speed setting and update the schedule (Sched^*);

End while

Check for new critical operations and include them in COP

End if

End while

End while

Output: New complete feasible schedule Sched^* with lower TEC and identical $Cmax$;

4.3.12 EMOACA Procedure for the EFSP-BCFT

The comprehensive pseudocode for the proposed EMOACA algorithm designed for EFSP-BCFT is outlined in Algorithm 4.4, and a simplified flowchart is depicted in

Algorithm 4.4 The_procedure_of_EMOACA

Inputs

Parameters: Number of jobs (n); Number machines (m); Number of AGV's (r); Number of AGV speed levels (V);

Data: Processing times (P); Setup times (ST); Setup energy (SE); Transportation distances (D); Battery capacity (C); Battery change time (BC);

Initialization

Parameters: Initialize: Basic population size (N); Maximum number of iteration $(maxiter)$; Number of matrices (Ω); Local and global pheromone evaporation rates (α, β); Pheromone deposition intensity (c); Maximum and minimum pheromone levels (τ_{max}, τ_{min}); Best edge selection pressure (q); Pheromone power factors $(\text{phj}; \text{phs}; \text{pho})$; Heuristic information power factor (λ); Heuristic information shape (σ); Image front factor (θ);

Heuristic information: Initialize: Job sequence and AGV speeds heuristic information matrices $\left(\eta^{g}_{pj};\ \eta^{g}_{kjv}\right)$ according to (4.71) – (4.73);

Pheromone trail: Initialize: Job sequence pheromone matrices $\left(\tau^{g}_{pj}\right)$, AGV speeds' pheromone matrices $\left(\tau^{g}_{kjv}\right)$ with τ_{max};

Initialize operations and pheromone registers as empty: $OR_{gpd} \leftarrow \emptyset$, $\tau^{g}_{pd} \leftarrow \emptyset$;

Initialize value of personal best goal attainment error (GE_a) and set it equal to $+\infty$ for each ant a;

Compute number of additional extremity ants (A) using (4.67);

Algorithm iteration

$t = t' = 0$; // Iteration counters

While $(t \le maxiter)$ **do**

    $t = t + 1$, $t' = t' + 1$;

    If $\left(t = \frac{maxiter}{4}\right.$ **or** $t = \frac{maxiter}{2}$ **or** $\left.t = \frac{3\,maxiter}{4}\right)$ **then** // Perform a PPMR

        $t' = 0$;

        Reset pheromone matrices for each job sequence and speed setting stages τ^{g}_{pj}, τ^{g}_{kjv} to τ_{max};

        Clear the all operations and pheromone registers $OR_{gpd} = \emptyset$, $\tau^{g}_{pd} = \emptyset$;

    End if

    $i = -A$; // Ant counter

    Randomize the set of ants for the this current iteration;

    While $(i < N + A)$ **do**

        Select a new random ant a from the remaining set of ants in current iteration t;

        Find corresponding matrices (g_1 and g_2) using (4.66) and compute fraction of pheromone γ using (4.65);

        Construct job sequence based on transition probability model for stage 1;

        If $(rand < \varepsilon)$ **then** // Perform a pairwise exchange mutation on the constructed jobs sequence

            Perform the PEM on the obtained job sequence

        End if

        If $\left(t < \frac{maxiter}{2}\right)$ **then** // Perform a low resolution search on AGV speed selection stage

            Construct AGV speeds based on low resolution transition model for stage 2;

        Else // Perform a high resolution search on AGV speed selection stage

            Construct AGV speeds based on high resolution transition model for stage 2;

        End else

        Construct the relaxed schedule and compute the relaxed start times S' according to (4.78);

        Construct operation sequence and AGV assignment based on transition probability for stage 3 using Algorithm 4.1;

        Construct and evaluate the resulting collision-free schedule with battery switch plan using Algorithm 4.2

        Proceed with the energy reduction improvement procedure (Algorithm 4.3);

        Perform local pheromone update on each pheromone matrix for each stage;

        Update the nondominated front archive;

        If $(t > 1)$ **then**

            Evaluate goal attainment error $GE_a(t)$ of the current solution (x^a) for the current iteration t using (4.68);

        End if

        Update ant a's best personal solution (x^a_{best}); // The current solution x^a is the best solution in the first iteration

$i = i + 1;$

End while

Update image front and reference points; // Setting a reference point for each ant for the next iteration

Proceed with global pheromone update (evaporation then deposition) for each original matrix for stages 1 and 3: τ_{pj}^{g}, τ_{pd}^{g};

Proceed with global pheromone evaporation for each original matrix for stage 2: τ_{kjv}^{g};

If $\left(t < \frac{maxiter}{2}\right)$ **then** // Perform first half of LHRS for pheromone reinforcement on AGV speed selection stage

Proceed with low resolution global pheromone deposition for each original matrix for stage 2: τ_{kjv}^{g};

Else // Perform second half of LHRS for pheromone reinforcement on AGV speed selection stage

Proceed with high resolution global pheromone deposition for each original matrix for stage 2: τ_{kjv}^{g};

End else

End while

Output: Nondominated front archive and solutions' analysis report;

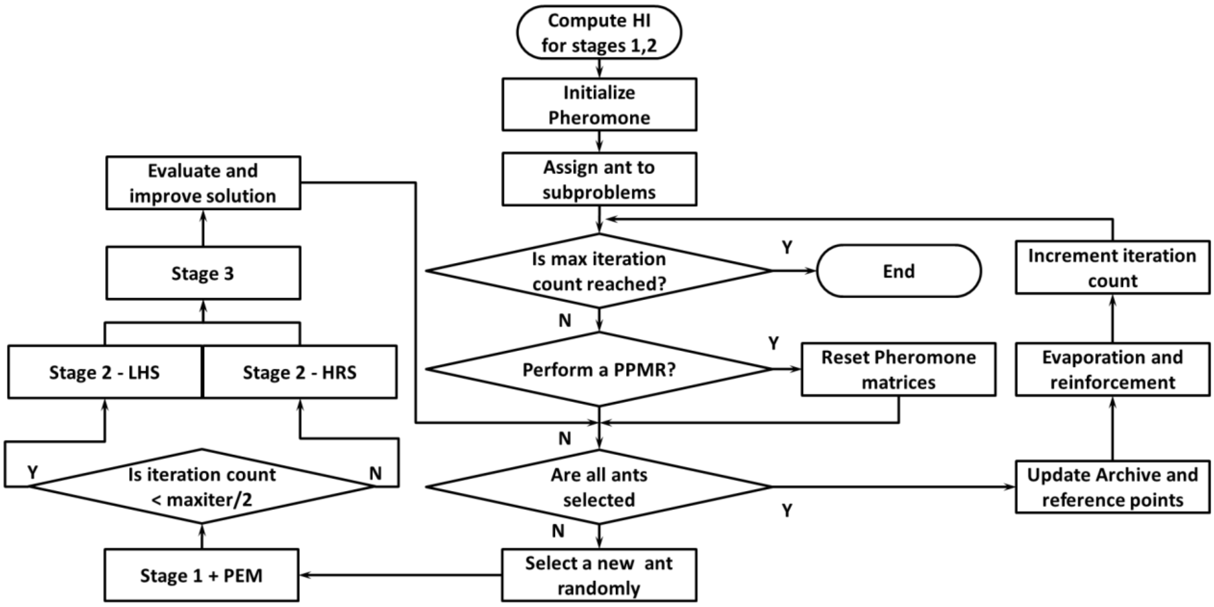

Fig. 4.8 A simplified flow chart for the EMOACA procedure.

4.4 Experimental Results for the EFSP-BCFT

In this section, the proposed EMOACA model for the EFSP-BCFT is validated through a series of experiments. Firstly, we present the case study and problem data. Secondly, we employ the EMOACA to assess the potential energy savings offered by the model. Thirdly, to evaluate the impact of the enhancement techniques integrated into the algorithm, we conduct tests on numerous variants of the EMOACA and compare their performance using multiple metrics. Finally, to showcase the effectiveness of the EMOACA, we tackle different-scale instances of the EFSP-BCFT using the proposed algorithm and compare the results with those obtained from other metaheuristics, including the nondominated sorting genetic algorithm (NSGA-II) [161], the multi-objective evolutionary algorithm with decomposition (MOEA/D) [178], as well as state-of-the-art methods like the modified multi-objective particle swarm optimization (MMOPSO) [147] and the enhanced multi-objective evolutionary algorithm with opposition-based learning strategy (EMOEA/OBL) [141]. All the algorithms and experiments were implemented in JAVA and executed on an Intel® i3 PC with 4 GB RAM and a 1.70 GHz processor.

4.4.1 Case Introduction

The case study presented here draws inspiration from previous literature [4, 19, 116]. It consists of fifteen jobs and eight workstations, each equipped with one machine. Additionally, there is a battery switching station positioned adjacent to the job input conveyor. The AGV guide path follows a unidirectional loop with a total length of 900 meters, and the distance between consecutive pickup/delivery (P/D) points at each workstation is fixed at 100 meters. The system allows for a limited number of AGVs, ranging from one to fifteen, to enter or exit the shop. This flexibility is essential for balancing transportation capacity between this shop and neighboring facilities or for conducting maintenance operations outside a busy shop. While in reality, AGV speed can be continuously adjusted within a range of 0.4 m/s to 2 m/s, we have discretized this range into five distinct speed settings according to the proposed model. The layout of the workshop is illustrated in Fig. 4.8.

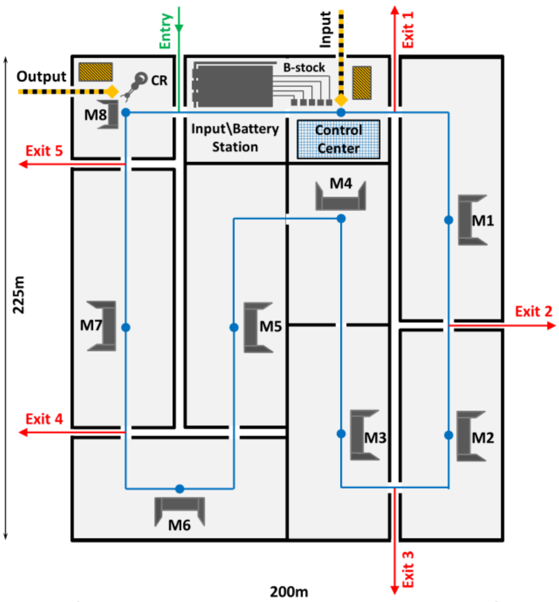

Fig. 4.9 Workshop layout in the case study

Table 4.1 provides an overview of the processing times (P_{ij}) for each job on every machine, while Table 4.2 outlines the necessary setup times (ST_{ijp}) required between consecutive jobs. It's notable that, in general, processing times exceed setup and transportation times, which aligns with common real-world scenarios as found in existing literature [134, 137, 140]. Table 4.3 presents the power consumed by each machine during setup (SE_{ijp}). To calculate setup energy values (SE_{ijp}), the setup power figures are multiplied by the corresponding setup times. As for AGV specifications, such as power requirements for each module, battery capacity, and initial charges, these have been abstracted from existing models in the literature to suit the requirements of this case study, simulating a real-world situation. The initial battery charges range randomly from 20% to 100% and are displayed in

Table 4.4. Each AGV is equipped with batteries of identical nominal capacity ($C = 1.44$ kWh). The battery switching process takes 5 minutes, or 300 seconds [19]. Table 4.5 provides estimates of the AGVs' empty travel power, categorized by speed setting [22]. The corresponding TE_v values, employed in the proposed model, are computed by dividing the travel power by the speed values. When AGVs are loaded, their travel power increases by a factor of $W = 1.6$ due to the job weight [142]. The robotic arm's average power demand for loading/unloading a job is 1.08 kW [22]. The arm's movements encompass lifting/lowering the job, translating it, and the empty arm movement from the AGV load platform to the machine's load position and vice versa. Nevertheless, this entire process lasts for less than 10 seconds and can be disregarded. However, AGVs still consume 3 Wh of robotic arm energy (AE) for each loading or unloading operation. Additional electrical components, such as the embedded PC, microcontroller, and sensors, maintain a nominal idle power (IE) of 22 W [145] throughout the AGV's active period.

Table 4.1 The processing times (seconds) of each job on each machine

Machine	J1	J2	J3	J4	J5	J6	J7	J8	J9	J10	J11	J12	J13	J14	J15
1	180	598	778	988	806	938	834	324	332	552	960	718	518	198	766
2	570	908	466	666	954	416	406	358	622	492	630	534	268	620	708
3	922	668	458	540	548	380	860	976	778	958	420	490	454	176	182
4	394	352	670	536	566	172	830	528	854	736	1018	982	186	584	272
5	368	754	322	978	502	940	780	596	534	788	456	386	588	348	224
6	698	210	458	954	750	382	848	402	630	828	242	918	922	786	702
7	762	834	500	986	442	578	956	368	434	348	872	906	228	186	686
8	406	420	386	742	254	1004	860	526	806	352	270	820	380	842	714

Table 4.2 The setup times (seconds) between consecutive jobs

	J1	J2	J3	J4	J5	J6	J7	J8	J9	J10	J11	J12	J13	J14	J15
J0	91	177	73	146	272	46	111	136	58	87	205	37	154	174	219
J1	-	274	140	163	139	197	72	66	192	212	37	96	84	104	141
J2	274	-	129	119	123	104	235	224	80	59	293	208	220	161	161
J3	121	150	-	165	77	105	59	112	70	125	134	144	46	78	196
J4	153	120	177	-	83	166	141	93	114	65	170	81	192	104	48
J5	149	121	92	91	-	167	74	92	117	48	170	77	94	9	102
J6	205	103	96	174	159	-	150	221	26	93	226	196	155	159	223
J7	57	222	55	150	76	146	-	39	125	158	77	126	6	33	169
J8	55	222	89	89	83	192	44	-	145	144	59	72	77	53	134
J9	188	90	81	120	100	36	108	146	-	45	189	133	124	92	179
J10	227	66	121	54	66	89	142	166	44	-	217	136	141	86	111
J11	43	303	138	163	165	223	54	42	205	227	-	104	93	124	146
J12	87	191	137	60	77	205	130	95	142	114	113	-	157	76	32
J13	92	207	64	192	87	146	10	71	131	143	92	149	-	76	193
J14	103	160	75	95	32	163	45	49	100	74	137	85	64	-	129
J15	126	159	182	39	87	240	165	111	204	118	162	37	208	107	-

Table 4.3 The setup power (kW) for each machine

M1	M2	M3	M4	M5	M6	M7	M8
1.22	1.03	1.14	1.01	1.12	1.21	1.10	0.90

Table 4.4 The Initial battery charge level for each AGV

AGV	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Charge	83%	38%	75%	52%	63%	25%	98%	20%	55%	100%	35%	96%	70%	89%	41%

Table 4.5 Travel power consumed by AGVs during empty travel with respect to speed setting

Setting	1	2	3	4	5
Speed (m/s)	2	1	0.67	0.5	0.4
Power consumption (W)	405.8	191.2	119.2	82.8	60.4
Corresponding TE_v values	202.9	191.2	178.8	165.6	151

4.4.2 Algorithm Parameter Setting

Algorithm performance is notably affected by its parameters. Preliminary experiments were conducted on various versions of the case study to determine approximate values for the fixed parameters used in subsequent computational experiments. Other parameters, specifically the pheromone power factors for job sequence construction, speed selection, and AGV operation sequencing, are analytically derived based on the estimated setting of the exploration parameter ε. The reader can find these parameter values in Table 4.6 . The pheromone power factors have been derived in a manner that ensures uniform exploration rates across each stage of the solution construction process. To elaborate, the exploration rate for the job sequence construction stage is defined as $\mu_{\mathrm{j}} = 1 - P_{\mathrm{j}}^{n}$, where P_{j} represents the maximum probability of selecting an unscheduled job for any position k, assuming that the heuristic information is equal for all jobs. The value of P_{j} is:

$$P_{\mathrm{j}} = \frac{\tau_{max}^{\mathrm{phj}(k)}}{\tau_{max}^{\mathrm{phj}(k)} + (n-k)\tau_{min}^{\mathrm{phj}(k)}} \quad \forall k = 1, \dots, n-1$$

The pheromone power factor for selecting an unscheduled job for position k can be readily derived as follows:

$$\mathrm{phj}(k) = \frac{\log\left(\frac{P_{\mathrm{j}}(n-k)}{1-P_{\mathrm{j}}}\right)}{\log\left(\frac{\tau_{max}}{\tau_{min}}\right)}$$

In a similar fashion, we establish the exploration rate for the speed selection stage as $\mu_s = 1 - P_{\mathrm{s}}^{mn}$, where P_{s} is the maximum probability to select a speed setting for any operation if the heuristic information is equal for all other settings. Consequently, the pheromone power factor can be derived as:

$$\mathrm{phs} = \frac{\log\left(\frac{P_{\mathrm{s}}(v_{max}-1)}{1-P_{\mathrm{s}}}\right)}{\log\left(\frac{\tau_{max}}{\tau_{min}}\right)}$$

For the AGV operation sequence, the transition model operates within constraints defined by a variable candidate list in each step of the construction process. It can be demonstrated that, for the EFSP-BCFT, the maximum size of the candidate list for AGV operations is:

$$|OCL|_{max} = \left\lfloor \frac{m+1}{2} \right\rfloor$$

As the size of the candidate list ranges from 1 to $|OCL|_{max}$, we define the exploration rate for the AGV operation sequencing stage as follows:

$$\mu_{\mathrm{o}} = \frac{1}{|OCL|_{max}} \sum_{|OCL|=1}^{|OCL|_{max}} (1 - P_{\mathrm{o}}^{mn})$$

where P_{o} is the maximum probability to select an unscheduled operation for any position p if the heuristic information is equal for all operations. Finally, to relate the individual exploration rates with the experimentally estimated ε we define:

$$\varepsilon = 1 - (1 - \mu_{\mathrm{j}})(1 - \mu_{\mathrm{s}})(1 - \mu_{\mathrm{o}})$$

For instance, when setting $\varepsilon =$ 0.075 with the given case data, the resulting values are phj(1) = 1.94, phj(14) = 1.37, phs = 2.13, and pho = 1.92.

Table 4.6 Parameter setting of EMOACA

Parameter	Value
Number of pheromone matrices (Ω)	20
Basic Population size (N)	100
Number of additional ants (A)	Analytically derived in (4.67)
Maximum number of iteration $(maxiter)$	2000
Local and global pheromone evaporation rates $(\alpha;\ \beta)$	0.05; 0.01
Pheromone deposition intensity (c)	2
Maximum and minimum pheromone levels $(\tau_{max};\ \tau_{min})$	10; 0.1
Best edge selection pressure (q)	0.2
Exploration rate (ε)	0.075
Pheromone power factors (phj; phs; pho)	Analytically derived
Heuristic information power factor (λ)	1
Heuristic information shape (σ)	0.67
Image front factor (θ)	0.05

4.4.3 Evaluation of the Energy-efficient Scheduling Optimization Model

To assess the energy-saving potential of the proposed model, the EMOACA is utilized to search for optimal or near-optimal solutions to the problem. The experimental results, showcasing the maximum completion time and the total energy consumption achieved by the proposed algorithm, are displayed in Fig. 4.10. An inherent trade-off between the two objectives is evident. The makespan varies between 284 and 472 minutes, with corresponding total energy values ranging from 6.59 kWh to 3.95 kWh. Minimizing the makespan tends to increase energy consumption, while prioritizing energy efficiency can lead to a reduction in energy consumption by a factor of approximately 1.67. Conversely, favoring productivity over energy efficiency can result in an increase in makespan by roughly the same ratio of 1.67. The PF consists of 57 solutions, with points of the same color indicating identical job sequences but differing in other solution characteristics like AGV speed and dispatching. Within these 57 solutions, there are 13 distinct job sequences, signifying that more than 77% of the solutions were obtained by modifying AGV speed and/or dispatching strategies. This outcome underscores the significance of considering integrated machine and AGV scheduling in the EFSP-BCFT.

To delve deeper into the analysis, we have selected three solutions from the nondominated set for further examination. These solutions encompass the two extremes and one that strikes a balance, satisfying a 50%-50% compromise for each objective value using the analytic hierarchy process [179]. The 50%-50% compromise solution is highlighted in Fig. 4.10, with a makespan of 332 minutes and energy consumption of 4.14 kWh. The results of this analysis are presented in Fig. 4.10 below.

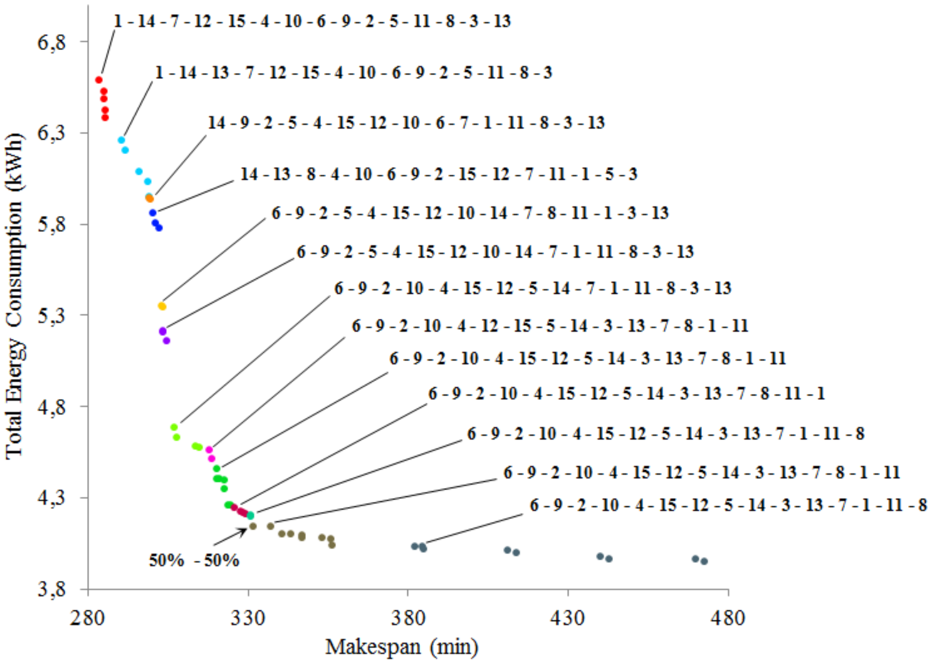

Fig. 4.10 Pareto front obtained by the EMOACA for the basic case.

The statistics presented in Table 4.7 indicate that AGV energy constitutes the largest portion of the total energy consumption within the entire Pareto Front. Additionally, it is evident that AGV energy usage is closely tied to the maximum travel distance, with unloaded distance being the primary contributor to the extra travel when aiming for lower $Cmax$ values. Furthermore, the average selected speed in the solution with the minimum $Cmax$ is slightly lower than that of the 50%-50% compromise solution. This implies that, to enhance productivity, it is more advisable for AGVs to execute more unloaded moves rather than utilizing higher speeds

Table 4.7 General statistics of the chosen solutions of the basic case.

Indicator	Solutions		
	Minimum $Cmax$	50%-50% compr.	Minimum TEC
Total AGV energy consumption (kWh) (% of TEC)	3.74 (56.6%)	2.42 (58.5%)	2.22 (56.5%)
Total setup energy (kWh) (% of TEC)	2.86 (43.4%)	1.72 (41.5%)	1.72 (43.5%)
Maximum traveled distance (m)	10800	4500	3600
Average unloaded distance (m) (% of total distance)	7025 (67.9%)	900 (22.2%)	250 (7.4%)
Number of unloaded moves	95	29	10
Average number of transferred jobs per AGV	14.75	8.75	3.75
Average selected speed (m/s)	1.5	1.59	0.4
Number of battery switches	4	2	1
Minimum recorded battery charge (%)	22.8	24.5	19.3

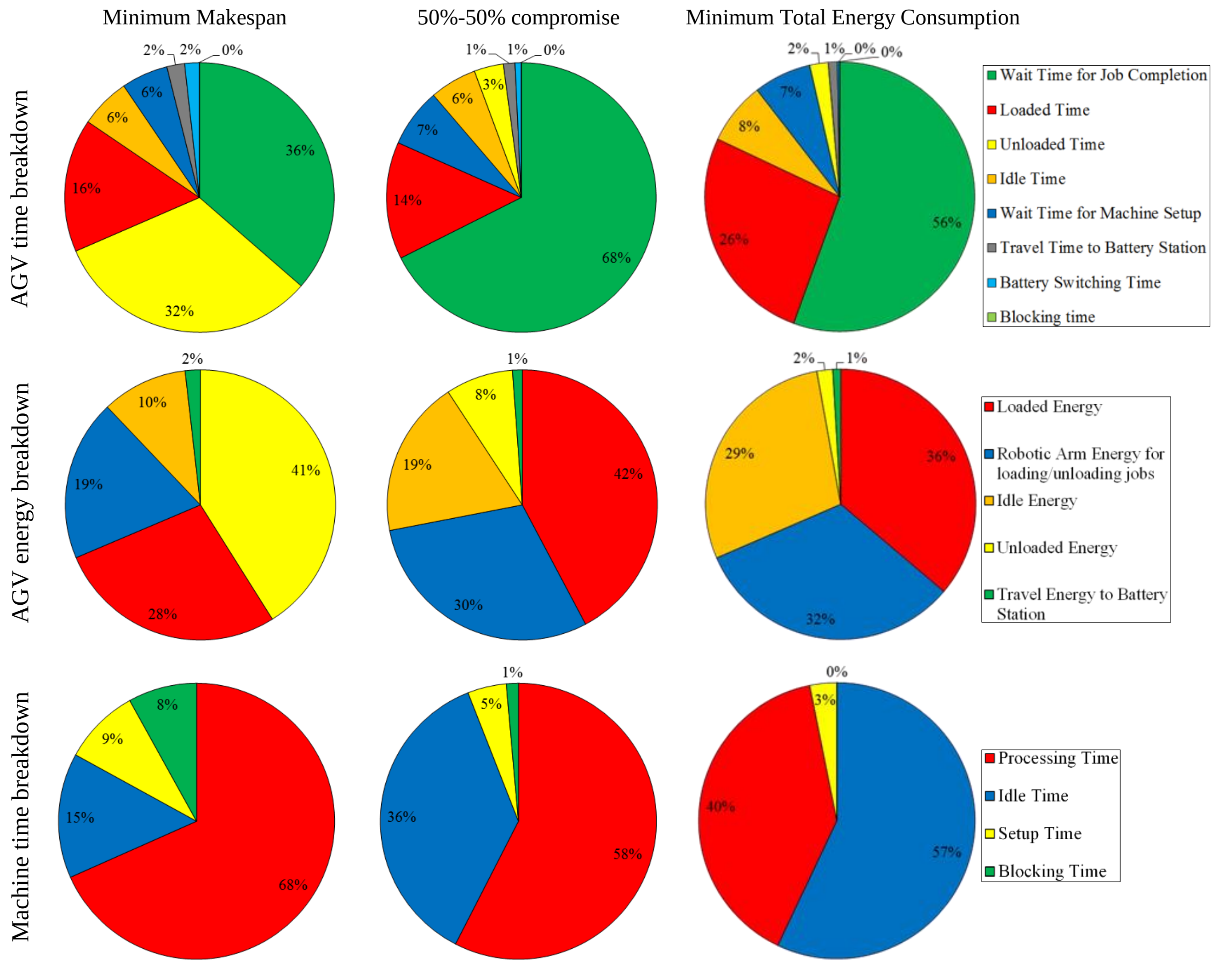

Fig. 4.11 Time and energy consumption breakdown of AGV and machines operations for the three selected solutions.

The breakdown of various time and energy components for AGV and machine states in each of the selected solutions is illustrated in Table 4.10. Notably, the largest AGV time component is the wait time for job completion. This represents the time AGVs spend at pickup/delivery points within busy workstations while waiting for job completion before they can transfer it to the next workstation. The proportion of this wait time tends to decrease when a lower makespan is favored, primarily due to the increased time spent on unloaded moves, as well as when a lower TEC is favored due to the reduction in AGV speed. In terms of energy breakdown, it's evident that the most variable energy consumption component is the unloaded energy, which ranges from 2% for minimum TEC to 41% for minimum $Cmax$. This increase in unloaded energy largely replaces the amount of idle state energy. The machine time breakdown indicates that the percentage of blocking time for the minimum $Cmax$ solution is 8%. This occurs because machines must wait for an AGV coming from earlier stages to unload the job, in contrast to the minimum TEC scenario where AGVs preferentially transport the same job from one machine to the next without considering unloaded moves to pick up jobs from subsequent machines.

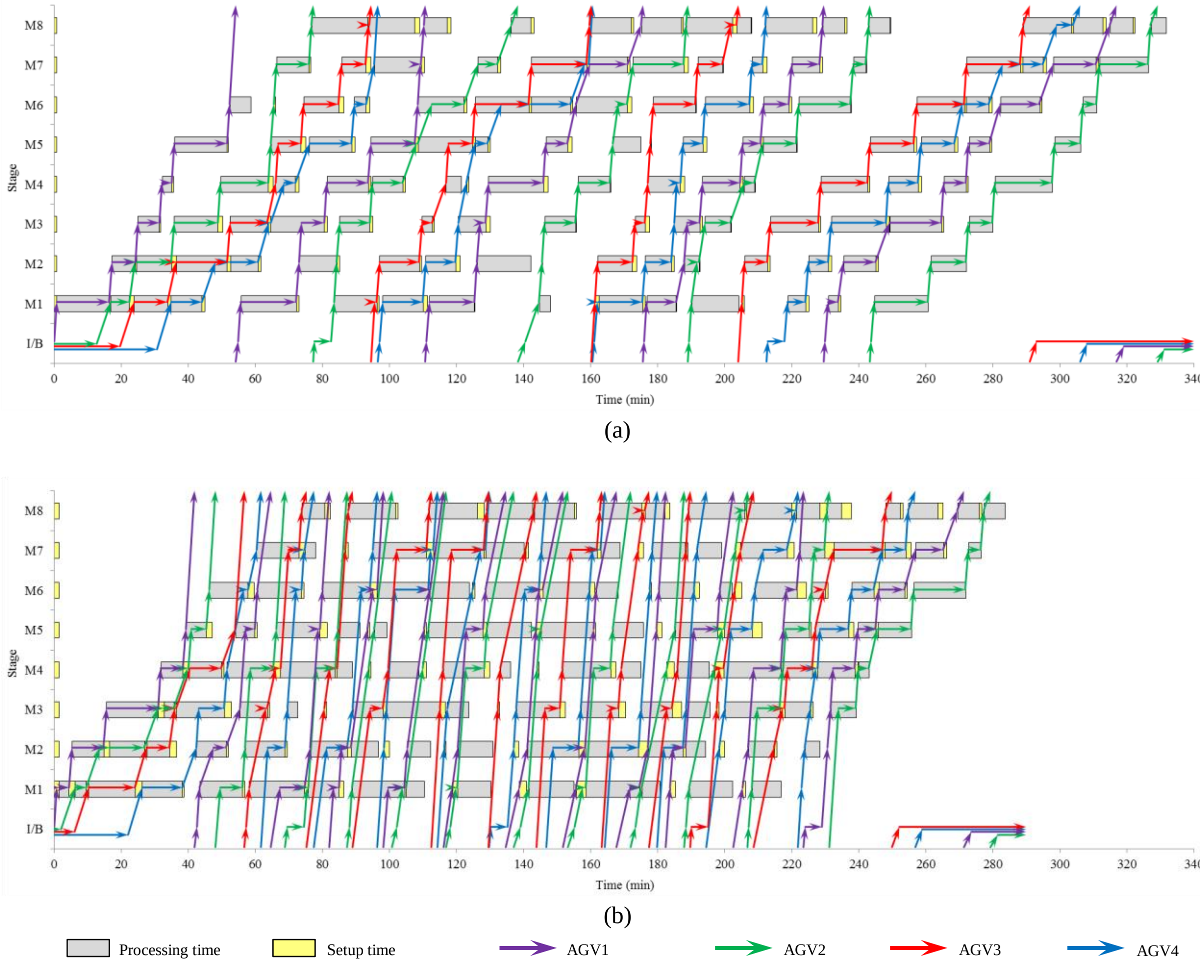

Fig. 4.12 Gantt chart and AGV movement schedule. (a) 50%-50% compromise solution; (b) Minimum makespan solution.

Fig. 4.12 displays the machine and AGV movement schedules for both the 50%-50% compromise and minimum $Cmax$ solutions, illustrating the impact of performing more unloaded moves on the makespan. In the 50%-50% compromise scenario, there are 29 unloaded moves, averaging 1.64 moves per lap per AGV, with an average move distance of 124 meters. In contrast, the minimum $Cmax$ solution involves 95 unloaded moves, averaging

2.19 moves per lap, and an average move distance of 296 meters. The figure also depicts the schedule for battery replacements. For instance, in schedule (a), AGV2 undergoes a battery switch after completing one lap with 7 job transfers at the 78-minute mark, while in schedule (b), AGV2 performs a switch after completing two laps, also with 7 job transfers, at the 70-minute mark. Additionally, in schedule (b), M7 remains blocked for over 5 minutes after processing the 7^{th} job until AGV3 unloads the job and machine setup can proceed.

4.4.4 Effect of the Number of AGVs

To investigate the impact of varying the number of AGVs on system performance, the EMOACA algorithm was utilized to obtain optimal or near-optimal solutions for the case study by adjusting the number of AGVs from two to fifteen. The baseline scenario involved four AGVs, specifically AGV1, AGV2, AGV3, and AGV4, positioned in the workshop loop. Additional AGVs were introduced into the workshop from entry 1, lining up behind the initial AGV sequence. If fewer than four AGVs were employed, AGVs exited the loop from Exit 1 in accordance with their order. For instance, if three AGVs were employed, AGV1 would be the first to exit, and if only two AGVs were used, AGV2 would also exit the loop. The optimization results are presented in Fig. 4.13, with the PFs for cases involving nine to fourteen AGVs omitted for clarity, and the makespan range truncated at 490 minutes. The results clearly indicate that the number of AGVs has a substantial positive influence on both productivity and energy consumption. However, increasing the number of AGVs beyond seven has only a marginal impact. Furthermore, it appears that the minimum achievable TEC is approximately 4 kWh and cannot be further reduced with the inclusion of additional AGVs.

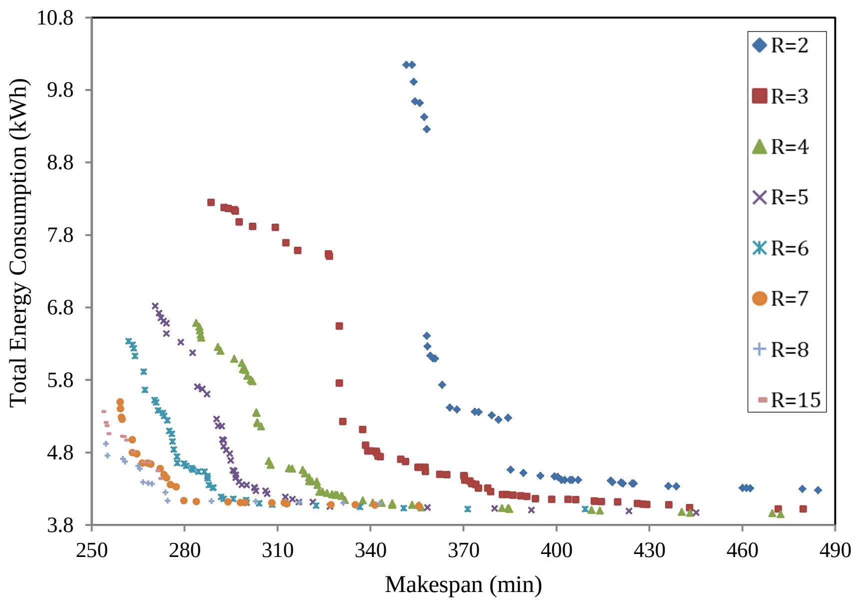

Fig. 4.13 Pareto fronts obtained by the EMOACA for multiple number of AGVs for scenario A.

To examine the impact of the relative scales of processing and transportation times on the EFSP-BCFT with varying numbers of AGVs, two additional scenarios, referred to as scenario

B and C, were introduced. In scenario B, processing and setup times were reduced by 75% to align more closely with the scale of transportation times. This adjustment resulted in processing times ranging from 43 to 255 seconds, which approximately matched the transportation time range of 50 to 250 seconds. In scenario C, processing and setup times were further reduced by 83%, leading to transportation times that were greater in scale. Consequently, the new range for processing times in scenario C fell between 7 to 41 seconds.

Resource efficiency is a critical consideration in AGV scheduling, balancing the need for optimal AGV usage without overworking or underutilizing them. In practical scenarios, solutions with moderate AGV usage are generally preferred, as both excessive and inadequate AGV usage can lead to inefficiencies or resource shortages. In this study, AGV usage is defined as the product of the proportion of time spent in loaded and unloaded states, multiplied by the ratio of the average selected speed to the maximum speed. Essentially, higher AGV usage signifies that AGVs are actively involved in transporting jobs or performing unloaded moves for more extended periods, especially when operating at higher speeds.

Fig. 4.14 visually represents the correlation between the number of AGVs and AGV usage across three different objective preferences within scenario A. When prioritizing lower $Cmax$, AGV usage significantly increases when the number of AGVs falls below six, reaching a peak of 76% when only two AGVs are used. Conversely, when the preferences are more balanced between objectives or emphasize lower TEC, AGV usage tends to be lower due to the larger scale of processing times. Increasing the number of AGVs beyond six has minimal impact on reducing the workload on other AGVs in such cases.

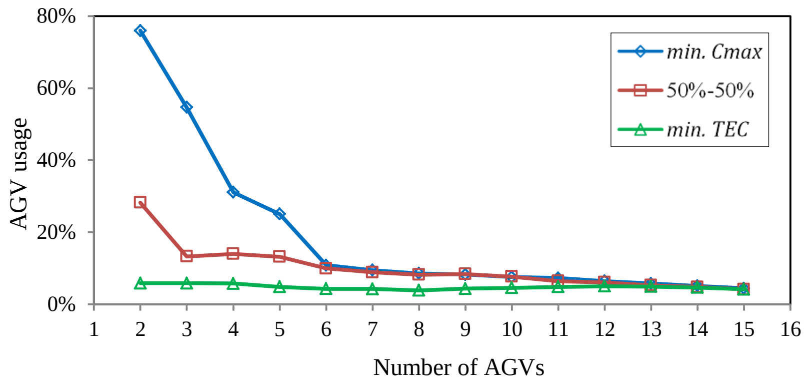

Fig. 4.14 AGV usage with respect to the number of AGVs for different objective preferences in scenario A.

To assess how the relative scales of processing times and transportation times impact AGV usage, the optimization results for different numbers of AGVs are presented in Fig. 4.15, focusing on scenario B. Compared to scenario A, scenario B exhibits higher AGV usage for five AGVs or more, primarily due to reduced wait times for job completion. However, the usage values do not increase as rapidly for the minimum $Cmax$ objective when fewer AGVs are deployed. This is because performing unloaded moves becomes less advantageous in this scenario due to the shorter processing times. Another noteworthy effect resulting from the altered scale of processing times is the wide gap between the usage values for the minimum

$Cmax$ and minimum TEC solutions even if the number of AGVs increases. This is in contrast to scenario A, where the usage values tend to converge toward the minimum usage value as the number of AGVs increases. Given that unloaded moves are unfeasible beyond six AGVs, it can be inferred that this gap primarily arises from speed differences. In scenario A, the large processing times make it more favorable to select higher transportation speeds for AGVs at the front of the AGV sequence, reducing idle energy consumption for the AGVs at the back who spend extended periods idle at the I/B station before initiating their first transfer operation. In contrast, scenario B allows AGVs to travel at the lowest speed setting (0.4 m/s) and still quickly vacate the track for the following AGV due to shorter wait times for job completion.

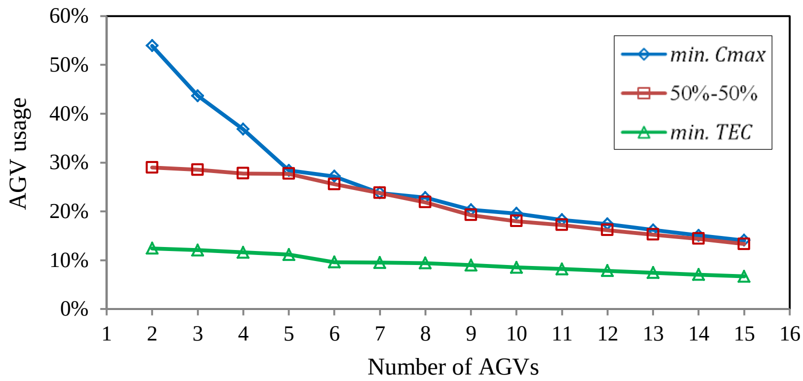

Fig. 4.15 AGV usage with respect to the number of AGV for different objective preferences in scenario B.

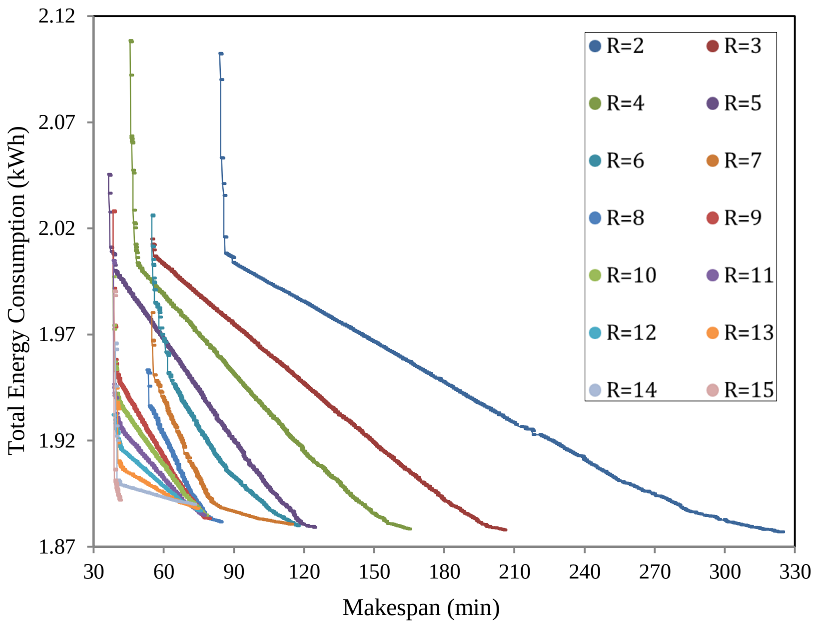

Fig. 4.16 Pareto fronts obtained by the EMOACA for multiple number of AGVs for scenario C.

In the scenario where transportation times exceed processing times, the optimization results for various numbers of AGVs are presented in Fig. 4.16. Notably, the TEC values are significantly reduced compared to scenario A. This reduction can be attributed to lower idle energy and unloaded energy consumption. However, an interesting observation is that the minimum $Cmax$ values increase when the number of AGVs goes from 5 to 6, rising from 36 to 55 minutes, and remain at this level even when two more AGVs are added. This increase is solely due to the low initial battery charge of AGV6, which is insufficient to complete at least 2 laps without requiring a battery replacement. As a result, AGV6 becomes a bottleneck resource.

The addition of AGV6, and AGV8 for similar reasons, negatively impacts performance, particularly given the short processing times, which average 24 seconds. In contrast, a single battery switch operation takes 5 minutes. This issue can be partially mitigated by adding more AGVs so that AGV6 and AGV8 only need to perform one lap each. However, even with additional resources, low-charge AGVs cannot maintain the same pace as other vehicles in order to conserve energy for completing the lap, thus remaining bottleneck resources. To address this issue, we recommend the following organizational adjustments for scenarios with relatively short processing times:

1. Request AGVs with high initial battery charge if possible or perform as many battery switches as possible before executing the shift.
2. Prearrange AGVs in decreasing order of initial battery charge so that low-charge AGVs have lighter workloads or can perform a battery swap during their wait time before their first transfer operation.
3. If feasible, perform necessary battery switches outside the workshop's loop in nearby battery stations to avoid blocking the path for following AGVs.

4.4.5 Significance of Considering Multiple Speed Settings, Battery Management and AGV Idle Power

To investigate the significance of considering multiple transportation speeds and integrating issues such as limited battery capacity and AGV idle power consumption, scenario D is introduced. In this scenario, a smaller battery capacity of 0.96 kWh is used and the battery switching time is set to 15 minutes. The new instance is then solved using both the proposed model and a reduced model that only considers a single speed setting of 2 m/s, ignoring battery capacity limitations and AGV idle power demand for comparison purposes. These models are tested on an instance involving four AGVs, and the resulting Pareto fronts are depicted in Fig. 4.17. As evident from the figure, the Pareto front obtained by the proposed full model dominates the Pareto front obtained by the reduced model. Additionally, the number of generated solutions by the proposed model is nearly 6 times greater. This demonstrates the superior capability of the proposed model to achieve better-optimized schedules for the EFSP-BCFT, especially when productivity is emphasized.

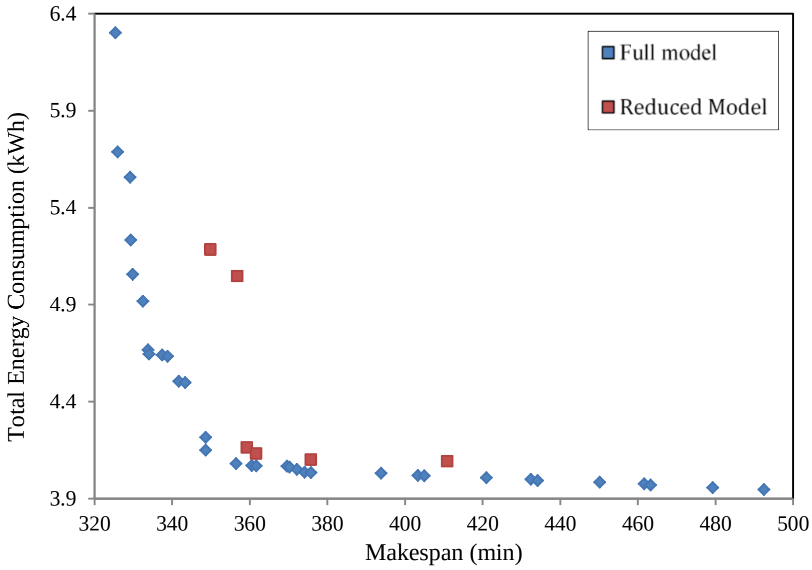

Fig. 4.17 Pareto fronts obtained by the EMOACA's full and reduced models for scenario D.

To further validate the advantages of integrating multiple AGV speeds, battery management, and idle power considerations into the optimization model, we selected the minimum makespan solutions from both Pareto fronts for an analysis of the time distribution of AGV states. Fig. 4.18 presents histograms comparing the average time spent by AGVs in each state for the selected solutions. The analysis reveals several important findings:

1. There is a significant increase in time spent on unproductive states, such as wait for job completion (+73 minutes increase) and wait for machine setup (+21 minutes).
2. In the reduced model, AGVs spend 28 minutes in standby mode due to track blocking, which is not the case in the full model. The full model adjusts AGV speeds to avoid this time component.
3. While loaded times are significantly reduced in the reduced model thanks to high AGV speeds, the lack of speed control causes AGVs to congregate in the same areas, leading to blockages. In contrast, the full model seeks to keep AGVs uniformly distributed, which is practical for safety reasons and congestion avoidance.

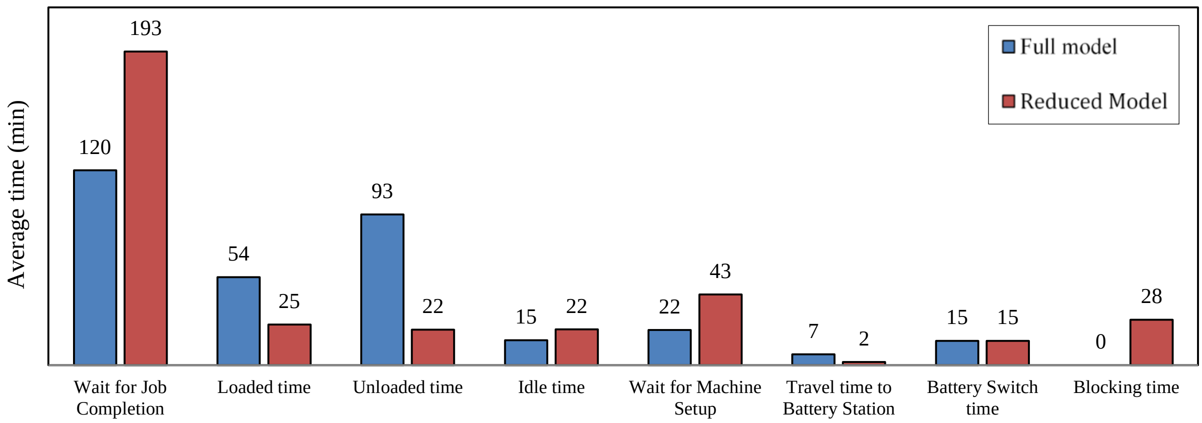

Fig. 4.18 Comparison of average time of AGV states for minimum makespan solutions obtained from scenario D.

To quantify the uniformity of AGV distribution, we computed the average standard deviation of the distance between consecutive AGVs at each moment in the time horizon, denoted as D Stdv. Lower values of D Stdv indicate more uniform AGV distribution and fewer congestion issues. Table 4.8 provides a performance comparison between the full and reduced models for various numbers of AGVs using five metrics:

1. Hypervolume indicator (HV)
2. Inverted generational distance indicator (IGD)
3. Spacing (SP)
4. Overall spread (OS)
5. Number of generated solutions (NB)

In general, higher values of HV, OS, NB, and lower values of IGD and SP indicate better performance. Since the scale of the objective values can influence the metrics, the obtained Pareto fronts were normalized between 0 and 1. The resulting values for HV, IGD, SP, and OS were multiplied by 100%. Moreover, given that IGD requires a reference PF and the optimal PF is unknown, the obtained PFs are combined into a single nondominated front which is considered as the optimal PF. It can be concluded that when using the full model, decision-makers can achieve superior results in terms of quality, the number of alternative solutions, and the distribution of AGVs.

Table 4.8 Performance of the full proposed optimization model vs. the reduced model for different number of AGVs in scenario D.

Metric	Model	$R = 2$	$R = 3$	$R = 4$	$R = 5$	$R = 6$	$R = 7$
HV	**Full**	**87.21**	**89.53**	**92.04**	**93.07**	**88.15**	**88.53**
	Reduced	85.77	85.3	77.31	83.76	82.36	71.17
IGD	**Full**	**0.002**	**0.001**	**0**	**0.01**	**0.068**	**0**
	Reduced	0.057	0.099	0.16	0.142	21.24	16.63
SP	**Full**	**0.784**	**2.857**	**4.755**	11.76	4.486	**5.011**
	Reduced	2.082	10.3	6.379	**9.109**	**4.004**	5.704
OS	**Full**	**100**	**100**	**100**	**98.68**	**100**	**91.52**
	Reduced	87.21	89.53	92.04	93.07	88.15	88.53
D Stdv. (m)	**Full**	**176.9**	**162.8**	**156**	**124.4**	**119.7**	**105.1**
	Reduced	197.9	213.3	176	138.9	129	118.7

4.4.6 Effectiveness Analysis of Enhancement Techniques

In this section, we present the results of computational tests conducted to assess the effectiveness of four enhancement techniques aimed at improving the search ability of the basic ACO algorithm for solving the EFSP-BCFT. These techniques, namely PEM, LHRS, PPMR, and the AGV dispatching heuristic information, denoted as ADHI, are individually tested against the full EMOACA algorithm, where each technique is omitted in one of the versions. The experiments are conducted on scenarios A and B, varying the number of AGVs from 2 to 7. Each algorithm runs for the same computational time as the full EMOACA. The results are summarized in **Table 4.9** which reports the mean values for each metric. A Wilcoxon signed-rank statistical test is used to identify the techniques that significantly improved the results at a 95% confidence level, highlighted in bold type.

Table 4.9 Average performance of EMOACA and its different less enhanced versions for different number of AGVs on scenarios A and B.

Metric	Algorithm	Scenario A						Scenario B						Mean
		$R=2$	$R=3$	$R=4$	$R=5$	$R=6$	$R=7$	$R=2$	$R=3$	$R=4$	$R=5$	$R=6$	$R=7$	
HV	EMOACA	94.4	84.8	90.0	86.8	88.9	86.9	91.2	89.1	86.9	87.7	86.3	87.4	88.4
	PEM	89.6	82.8	86.2	86.5	88.1	85.8	85.3	87.7	84.8	86.1	84.6	86.3	**86.2**
	LHRS	87.5	76.8	83.0	80.8	82.3	76.5	90.1	90.7	86.8	89.1	73.9	71.9	**82.5**
	ADHI	89.5	84.2	82.9	85.0	87.2		89.1	89.1	86.1	87.0	85.5		**86.6**
	PPMR	95.2	83.2	85.4	84.5	85.5	86.0	92.3	89.5	86.0	87.1	85.2	87.2	**87.3**
IGD	EMOACA	3.05	1.24	1.98	2.60	1.51	1.79	2.16	2.47	2.06	3.43	2.90	3.46	2.39
	PEM	3.73	2.49	4.78	4.55	4.10	7.90	4.14	5.75	4.81	5.51	5.68	5.16	**4.88**
	LHRS	9.56	5.59	6.70	6.88	8.05	9.88	3.76	2.78	6.36	3.07	9.20	10.86	**6.89**
	ADHI	4.67	6.16	6.29	9.79	11.70		4.12	3.10	3.00	4.83	4.18		**5.78**
	PPMR	3.45	2.87	4.05	2.74	12.18	8.42	1.17	2.19	3.96	4.39	3.60	3.53	**4.38**
SP	EMOACA	2.15	2.83	0.91	3.18	4.71	3.45	1.72	1.86	1.73	2.10	4.33	4.86	2.82
	PEM	2.63	1.46	1.81	4.33	4.03	10.21	1.81	9.91	1.11	3.02	4.97	4.47	4.15
	LHRS	6.82	5.98	11.41	4.40	9.70	5.09	9.33	7.21	10.83	2.07	15.61	12.49	**8.41**
	ADHI	2.54	1.82	5.16	7.81	15.91		2.52	4.66	5.08	3.89	3.90		**5.33**
	PPMR	6.73	1.66	2.51	4.92	15.11	9.29	1.39	1.28	2.24	2.93	3.76	3.12	4.58
OS	EMOACA	99.1	95.5	51.8	99.6	99.6	98.1	73.9	98.4	99.2	99.7	91.0	93.8	91.6
	PEM	83.8	96.1	69.6	66.8	51.9	52.7	77.7	45.2	86.5	63.6	62.4	60.2	**68.0**
	LHRS	71.4	88.6	46.2	73.4	46.2	60.3	62.5	79.6	91.8	65.7	70.8	37.8	**66.2**
	ADHI	19.3	46.2	46.5	39.3	24.7		43.3	60.3	85.7	73.5	77.9		**51.7**
	PPMR	33.0	98.7	78.2	91.2	23.7	36.7	100.0	82.0	79.4	71.6	88.8	86.8	72.5
NB	EMOACA	80.4	66.0	56.2	43.5	41.9	27.2	71.1	50.8	41.1	51.0	36.7	28.8	49.5
	PEM	68.8	58.1	46.5	38.3	38.6	15.1	73.4	28.2	21.8	40.0	20.4	30.1	**39.9**
	LHRS	92.6	84.4	58.4	49.8	50.2	32.6	89.4	58.5	49.0	59.4	38.4	36.3	58.2
	ADHI	81.6	70.4	47.8	28.1	19.8		50.5	38.2	29.8	41.3	24.2		**43.1**
	PPMR	68.8	76.4	45.4	30.4	21.3	17.1	71.2	47.9	30.2	42.3	30.6	33.0	42.9

The analysis of the results reveals the following:

1. HV and IGD values show different improvements depending on the techniques used. The most significant improvement is achieved when LHRS is employed, resulting in a 7.2% increase in HV and a decrease in IGD from an average of 6.89 to 2.39, indicating significantly better convergence.
2. The second most significant improvement is observed when using the proposed ADHI technique, which is equally effective as the PEM technique.
3. The PPMR technique is the least effective in terms of improvement.

4. In terms of spacing, LHRS and ADHI are the primary techniques that significantly enhance the algorithm's performance.
5. When LHRS is not used, the algorithm generates significantly more solutions, as expected, since the space of AGV speed settings is fully considered. However, this comes at the cost of slower convergence, reduced global exploration, and increased risk of falling into local optima.
6. Regarding the overall spread, the most effective enhancements are ADHI, followed by LHRS and PEM. The high effectiveness of ADHI in terms of OS can be attributed to the problem's structure and the shape of the optimal PF, as discussed in section 4.4.3. To achieve lower $Cmax$, AGVs need to be assigned in a way that exploits flexibility and performs more unloaded travels, leading to a rapid increase in TEC and expanding the optimal PF in this region, which cannot be reached by a blind AGV dispatching transition model.

In summary, LHRS and ADHI appear to be the most effective techniques for improving the EMOACA algorithm's performance, resulting in better convergence, exploration of the solution space, and avoidance of local optima.

4.4.7 Performance Evaluation of the EMOACA

To assess the performance of the proposed EMOACA in solving the EFSP-BCFT, we conducted a comparative analysis with several other algorithms, including NSGA-II, MOEA/D, MMOPSO, and MOEA/OBL.

Fig. 4.19 provides an illustration of the solution representation and encoding utilized in the MMOPSO for a specific instance involving 3 jobs, 3 machines, 2 AGVs, and 5 available speed settings. The encoding scheme involves several components:

1. Job Sequence Position encoding (JSP): The job sequence is represented as a real-valued vector in an n-dimensional space, where each coordinate ranges from 0 to 1. To determine the job sequence, the Smallest Position Value (SPV) rule is applied. This rule involves sorting the jobs based on their corresponding values in increasing order, allowing for the construction of the job sequence.
2. AGV Speed Position encoding (ASP): The speed setting for each AGV operation is encoded using a real value between 0 and 1, which corresponds to a coordinate in an $(n \times m)$-dimensional space. This value is then scaled to match the number of available speed settings and rounded to obtain the actual speed setting.
3. AGV Operation Sequence Position encoding (OSP): The AGV dispatching sequence is represented as a real-valued vector in an $(n \times m)$-dimensional space. Each coordinate is constrained to fall within the 0 to 1 range, and it represents the priority of a specific operation. To construct the AGV dispatching sequence, operations are ranked based on their priority values in decreasing order. This ranking guides the selection of operations from the candidate list, ensuring that the operation with the highest rank is chosen at each step.

Finally, the resultant AGV assignment follows the procedure explained in Fig. 4.4.

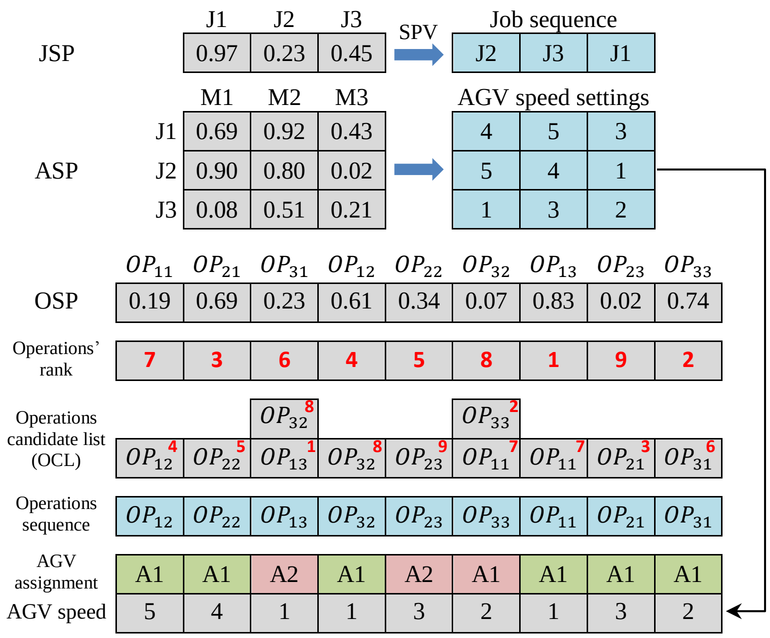

Fig. 4.19 An example of solution representation and encoding for the MMOPSO.

To represent a solution in NSGA-II and MOEA/D, chromosomes are divided into three segments:

1. Job Sequence: The first segment contains a permutation of n integer numbers, representing the job sequence.
2. AGV Speed Setting: The second segment consists of nm integer values, representing the AGV speed settings.
3. Operation Rank: The third segment contains a permutation of nm integers, representing the ranks of operations.

Similar to the concept presented in Fig. 4.19, AGV dispatching is performed based on both the job sequence and the operation ranking. Fig. 4.20 provides an example of job sequence representation, along with two-point crossover and insertion mutation operators employed in NSGA-II and MOEA/D. This example involves 8 jobs.

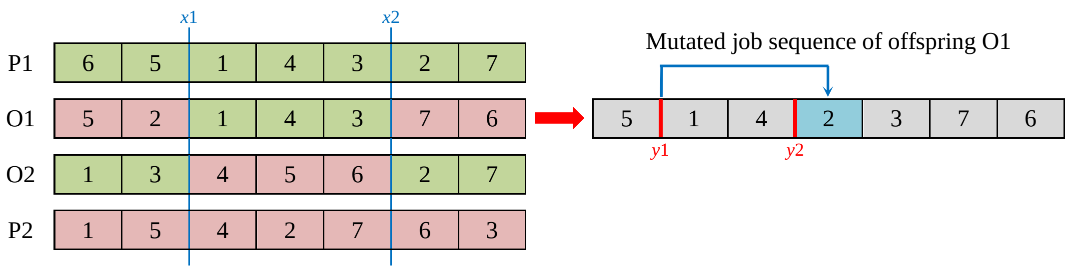

Fig. 4.20 An example of Job sequence representation, crossover and insertion mutation for NSGA-II and MOEA/D.

For the job sequence crossover operation:

- Two random parents, denoted as P1 and P2, are selected from the population.

- Two random points, $x1$ and $x2$, are chosen such that $1 \leq x1 < x2 \leq n$.
- Two new offspring, O1 and O2, are generated. To construct the job sequence for O1, the jobs located between $x1$ and $x2$ in P1 are directly copied into O1, maintaining their original positions. The missing jobs are then copied from P2 into the remaining positions in O1, preserving their order in P2.

For the mutation operation applied to O1:

- Two random points, $y1$ and $y2$, are selected such that $1 \leq y1, y2 \leq n$.
- The job at position $y1$ in the sequence is removed and reinserted at position $y2$.

These same two-point crossover and insertion mutation operators are also implemented for the AGV dispatching part of the chromosome.

Fig. 4.21 illustrates an example of AGV speed setting representation, including two-point crossover and random walk mutation operators implemented in NSGA-II and MOEA/D for an instance with 3 jobs and 3 machines. These operators work as follows:

Crossover Operator for AGV Speed Settings:

- Two random parents, P1 and P2, are selected.
- Two random points, $x1$ and $x2$, are chosen such that $1 \leq x1 < x2 \leq n$.

To generate the AGV speed settings for the first offspring, O1, the values located between $x1$ and $x2$ are directly copied into O1 at the same positions. The remaining values in O1 are copied from P2 while preserving their original positions.

Mutation Operator for AGV Speed Settings (Applied to O1):

- A speed setting is randomly selected with a predefined probability.
- The selected speed setting is randomly increased or decreased by one setting while ensuring it remains within the permissible bounds.

This process allows for the exploration of different AGV speed settings while maintaining feasibility within the defined limits.

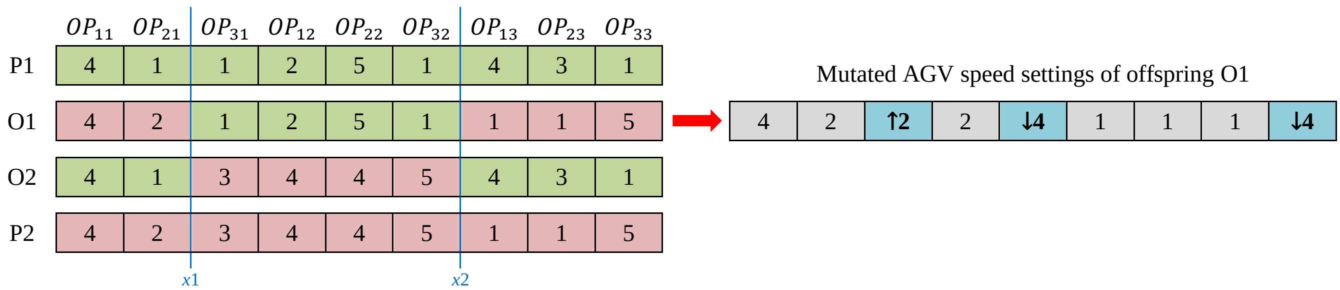

Fig. 4.21 An example of AGV speed representation, crossover and random mutation for NSGA-II and MOEA/D.

To represent a solution in MOEA/OBL, a two-layer encoding approach is devised. In the first layer, the job and AGV dispatching sequences are represented, while the second layer captures the AGV speed settings. The following describes how the first generation of chromosomes is created and how the encoding process works:

1. Generate a random array of nm real numbers for each individual, ranging between 0 and 1.
2. Rank the values in each array in ascending order to obtain a permutation of nm integers.
3. For each integer i, calculate $1 + i \bmod n$ to create a list of job indices. This list forms the job sequence by sequencing the jobs based on their first appearance order in the list.
4. Create a ranked sequence of operations by scanning the list from left to right. Generate operations with matching job indices and machine numbers corresponding to the number of occurrences of each job index. Assign a rank to each operation based on its position in the sequence.
5. This ranking is used to correct the operations sequence and ensure a feasible AGV dispatching.

Fig. 4.22 provides an example illustrating chromosome generation, operations sequence correction, and solution encoding for MOEA/OBL. The crossover and mutation operators in MOEA/OBL work on the corrected operations sequences to generate new offspring operations sequences, which are later adjusted to obtain a feasible AGV dispatching. Specifically, MOEA/OBL uses a two-point crossover operator, similar to that used in NSGA-II and MOEA/D. The mutation operator selects two random integers, $y1$ and $y2$, within the range of 1 to nm, and swaps the operations at positions $y1$ and $y2$ in the sequence. These operations help explore different solutions while maintaining feasibility.

Regarding the second layer, the representation of AGV speed settings involves nm integer values, similar to NSGA-II and MOEA/D. However, MOEA/OBL incorporates the LHRS strategy for a more equitable comparison. In the first half of the optimization process, the layer consists of a single value that represents the AGV speed setting selected for all operations. During this low-resolution phase, the crossover operator randomly chooses two speed settings from the range used by parents P1 and P2 and assigns them to each offspring, O1 and O2. Subsequently, the same random walk mutation operator is applied to the offspring. In the second half of the process, the layer is expanded to accommodate nm values by duplicating the current speed setting for each operation. This ensures that each operation has its specific speed setting. During the high-resolution phase, the crossover operator operates as follows:

1. Randomly select two integers, $x1$ and $x2$, where $1 \leq x1 < x2 \leq n$.
2. Copy all the speed settings from P1 and P2 to O1 and O2, respectively, with the exception of the transportation operations of jobs $x1$ and $x2$. For these operations, the speed settings are swapped between P1 and P2.

The mutation operator functions by selecting two random integers, $y1$ and $y2$, within the range of 1 to n, and exchanging the AGV speed settings for the transportation operations of jobs $y1$ and $y2$. This allows for exploration of different speed settings while maintaining the overall structure of the solution.

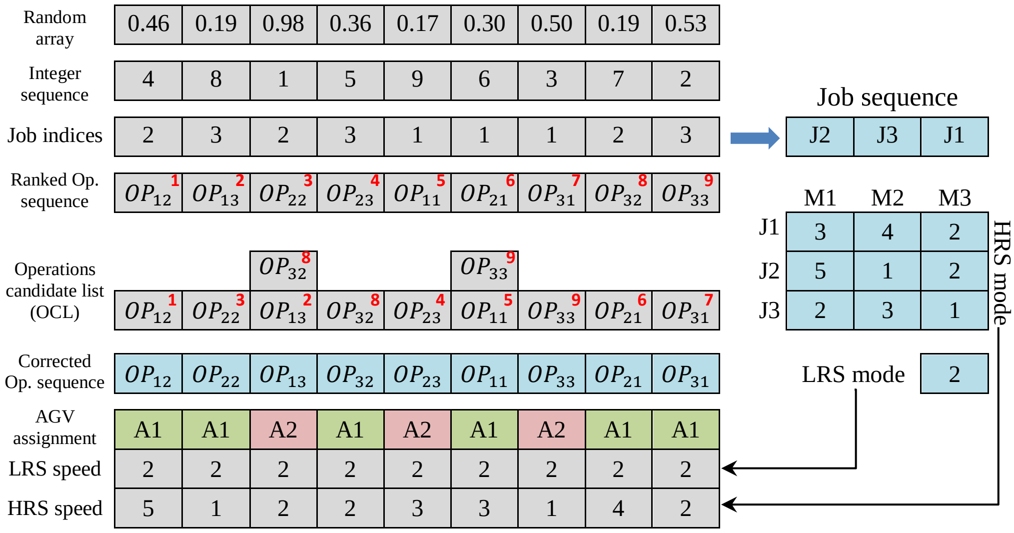

Fig. 4.22 An example of chromosome generation, operations sequence correction and solution encoding for the MOEA/OBL.

Incorporating the opposition-based learning strategy, we have followed a similar approach to He et al. [141], wherein the top 20% of the population's gene values are subjected to inversion. To illustrate, Fig. 4.23 provides an example of gene inversion for a scenario involving 3 jobs, 3 machines, and 5 AGV speed settings, as employed in MOEA/OBL's opposition learning technique.

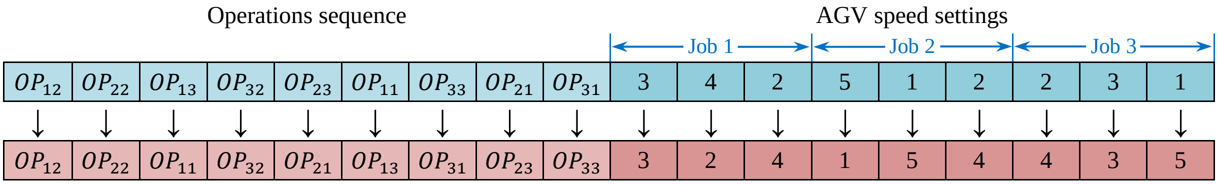

Fig. 4.23 An example of gene inversion in the MOEA/OBL.

It's essential to note that in each algorithm, the encoded solutions are decoded and evaluated using the same approach as outlined in the proposed EMOACA. Additionally, the processes of battery switching planning and collision detection and prevention are executed in a consistent manner.

In addition to the data used in Scenario A, three additional test instances were generated as follows:

1. A smaller instance with 10 jobs, 5 machines, and number of AGVs $R = \{2, 3, 4\}$.
2. A larger instance with 30 jobs, 10 machines, and $R = \{3, 5, 9\}$.
3. A very large instance with 45 jobs, 15 machines, and $R = \{4, 8, 14\}$.

A similar data structure as in Scenario A is maintained for these test problems. Processing times were randomly generated within the range of [180, 1200] seconds, setup times within [5, 300] seconds, and setup power from a continuous uniform distribution U(0.9, 1.22) kWh.

Initial battery charges were also randomly generated between 20% and 100%, while the rest of the parameters remained unchanged. Each test instance is denoted by the pair (n, m), and all algorithms were executed independently for the same computational time as the EMOACA for each instance. To categorize the problem's scale, we considered instances with 35 to 50 jobs and 10 to 20 machines as large-scale instances, instances with 15 to 34 jobs and 5 to 9 machines as medium-scale, and instances with fewer jobs and machines as small-scale. This classification aligns with previous research in the field, as seen in works such as [128, 140, 141].

Table 4.10 Parameter setting of comparative metaheuristics.

Parameter	NSGA-II	MOEA/D	MOEA/OBL	MMOPSO
Population size	100	100	100	100
Maximum number of iteration	Undetermined	Undetermined	Undetermined	Undetermined
Crossover rate (Two point crossover)	0.8	0.8	0.8	-
Mutation rate for job sequence (Insertion operator)	0.075	0.075	-	-
Mutation rate for AGV speed setting (Random walk)	0.01	0.01	0.5	-
Mutation rate for AGV operations ranks (Insertion operator)	0.1	0.1	-	-
Mutation rate for operations sequence (Swap operator)	-	-	0.5	-
Archive size	-	20	100	20
Opposition learning rate	-	-	0.2	-
Number of Neighbors	-	5	-	-
Grid size	-	-	-	16×16
Leader selection pressure	-	-	-	0.7
Inertia weight	-	-	-	0.8
Cognitive factor	-	-	-	0.15
Social factor	-	-	-	0.1

Table 4.11 Results of comparative experiments for various test instances.

Metric	Algorithm	(10, 5)			(15, 8)			(30, 10)			(45, 15)			Mean
		$R=2$	$R=3$	$R=4$	$R=2$	$R=4$	$R=7$	$R=3$	$R=5$	$R=9$	$R=4$	$R=8$	$R=14$	
HV	**EMOACA**	85.7	86.5	87.7	94.6	88.8	90.1	90.7	92.6	89.4	93.4	92.8	80.3	**90.55**
	NSGA-II	80.2	79.6	82.1	75.9	76.7	78.3	68.4	79.5	54.0	21.4	77.2	69.7	70.61
	MOEA/D	77.3	79.3	83.0	76.4	72.3	77.0	66.2	76.6	70.9	19.5	75.3	54.5	70.43
	MMOPSO	84.8	81.8	83.3	91.3	82.5	76.3	61.4	85.6	75.0	41.3	31.3	42.9	68.62
	MOEA/OBL	81.0	86.0	87.3	87.7	86.9	86.4	81.7	84.2	72.5	65.5	76.8	82.1	81.5
IGD	**EMOACA**	1.0	2.10	1.0	1.0	1.0	0.9	0.4	0.1	2.6	6.1	0.8	10.0	**0.464**
	NSGA-II	5.9	7.8	7.3	17.5	23.0	12.6	20.5	12.2	35.5	109.0	22.5	12.1	34.47
	MOEA/D	6.9	14.2	9.1	18.7	13.3	10.4	24.0	15.4	15.4	105.3	26.6	24.3	32.7
	MMOPSO	2.9	5.6	4.5	6.2	7.3	10.4	30.8	9.4	14.1	73.5	107.8	37.4	38.97
	MOEA/OBL	8.7	2.9	5.3	6.8	4.1	7.0	9.3	9.6	13.5	38.5	24.5	12.0	11.84
OS	**EMOACA**	94.2	77.7	58.9	99.3	77.8	86.5	94.3	98.8	58.6	29.2	46.7	46.9	**74.8**
	NSGA-II	52.9	62.6	64.8	6.7	13.6	27.9	4.1	4.9	8.6	3.2	34.4	38.8	27.09
	MOEA/D	51.8	39.4	46.7	4.8	53.6	55.0	3.9	6.3	15.5	13.2	28.2	67.6	30.07
	MMOPSO	71.0	92.1	94.8	50.8	55.3	98.6	16.0	21.9	64.6	4.6	5.1	2.6	49.68
	MOEA/OBL	35.6	85.8	56.0	16.4	69.3	39.4	31.5	8.6	39.2	26.6	29.8	33.5	39.32

The parameter configurations for the comparative algorithms are presented in Table 4.10. Results from the comparative experiments are provided in Table 4.11. To evaluate the significance of the difference between the results obtained by EMOACA and the other algorithms, a Wilcoxon signed rank test with a 95% confidence level was conducted, and the best-performing algorithm is highlighted in bold. The results clearly demonstrate that EMOACA outperforms all other algorithms across all instances in terms of HV and IGD metrics, except for the (45, 15) instance with 14 AGVs, where MOEA/OBL exhibits slightly better hypervolume covering. MOEA/OBL emerges as the most competitive algorithm in terms of HV and IGD while the improved MMOPSO is the most competitive in terms of OS.

To validate the superiority of the proposed EMOACA compared to the MMOPSO and MOEA/OBL algorithms, an extensive dataset based on the classical Taillard test instances for flow shop scheduling with SDST is utilized. This dataset, originally introduced by Ruiz et al. [180], is available for download at http://soa.iti.es/problem-instances. It is organized into four subsets, namely ssd10, ssd50, ssd100, and ssd125, each comprising 120 instances. These subsets correspond to different uniform distribution ranges for setup times, specifically [1, 9], [1, 49], [1, 99], and [1, 124]. Due to the inherent complexity of the problem, only the first 60 instances from each subset are considered for analysis. This results in instances with varying numbers of jobs and machines, ranging from (20, 5) to (50, 20). Notably, the original instances in this dataset provide data solely for processing times and setup times between jobs. To create a comprehensive problem representation, other parameters are generated using the following scheme:

1. The number of AGVs is randomly selected from the interval [2, m], and the number of speed settings is chosen from [2, 10], with each speed setting v $(v = 1,2, \dots v_{max})$ corresponding to a speed of $2/v$ m/s.
2. Distance between consecutive machines is generated within the range [2, 40] meters.
3. The values for TE_v corresponding to each speed setting are drawn from a continuous uniform distribution U(0.05, 2.85) Wh/m.
4. The job weight factor is generated from a continuous uniform distribution U(1, 2).
5. Nominal AGV idle power is determined from U(10, 40) W.
6. The AGV robotic arm energy is generated from U(1, 5) Wh for each job loading or unloading operation.
7. Nominal battery capacity is obtained from U(1, 2) kWh.
8. Initial battery charges are randomly set between 20% and 100%.
9. Battery switching times are determined within the range [1, 99].
10. Initial job setup times are generated based on the specific uniform distribution range of each data subset.
11. The setup power of each machine is randomly drawn from U(0.5, 2.5) kW.

In this comparative experiment, each algorithm is executed 20 times, and the average values of each metric are reported as the final results. To assess the statistical significance of the findings, the Wilcoxon signed rank test with a 95% confidence level is employed. The results for the HV, IGD, and OS metrics obtained from the computational experiments are presented in Tables 4.12, 4.13, and 4.14, respectively, with the best-performing algorithm indicated in bold type. The outcomes clearly demonstrate the superiority of the EMOACA across most of the test problems, particularly in terms of HV and OS. Regarding the IGD metric, the EMOACA consistently achieves a value of 0 in 15 instances, a value unattainable by the comparative algorithms. However, it is worth noting a slight drop in performance as the range of setup times increases. The EMOACA's enhanced performance can be attributed to several factors. Firstly, its algorithmic structure, notably the problem-specific heuristic information, enables more efficient exploration of the search space compared to the blind operators employed by the comparative algorithms. Secondly, the problem decomposition method and the LHRS enhancement contribute to diversity, while pheromone information sharing among adjacent groups of ants ensures exploitation. This approach is less disruptive to the search process compared to the random selection of parents in evolutionary algorithms, where the generated offspring may not necessarily match the quality of the selected parents due to the many constraints inherent in the problem. In summary, the EMOACA's tailored design and integrated enhancements yield superior results in solving the EFSP-BCFT across a variety of instances, making it a powerful choice for addressing this complex problem.

Table 4.12 Hypervolume results of comparative experiments for taillard_ssd test instances.

Ins.	size	ssd10			ssd50			ssd100			ssd125		
		EMOACO	MOEA/OBL	MMOPSO	EMOACO	MOEA/OBL	MMOPSO	EMOACO	MOEA/OBL	MMOPSO	EMOACO	MOEA/OBL	MMOPSO
1	(20, 5)	**80.53**	63.32	56.51	**77.07**	57.75	42.58	**80.67**	79.96	22.1	**89.93**	60.88	24.89
2	(20, 5)	**78.86**	63.71	62.55	**75.22**	54.18	36.87	**86.11**	65.87	24.07	**90.31**	72.99	20.25
3	(20, 5)	**83.82**	57.97	61.4	**87.72**	28.81	49.3	**88.19**	83.03	27.83	**88.43**	49.27	34.73
4	(20, 5)	**80.98**	63.24	58.67	**79.02**	66.57	44.04	**85**	57	12.09	**81.02**	55.27	30.61
5	(20, 5)	**77.05**	73.93	41.06	**86.86**	62.6	33.85	**84.69**	53.74	22.69	**87.77**	76.53	23.22
6	(20, 5)	**64.07**	54.03	21.31	**81.44**	68.29	19.36	79.32	72.5	7.651	**79.07**	69.77	29.18
7	(20, 5)	**57.86**	49.66	29.98	**89.1**	70.66	34.18	**79.97**	58.15	39.23	**84.17**	63.64	23.04
8	(20, 5)	**74.28**	66.78	53.38	**82.83**	76.66	41.29	**80.81**	63.26	40.2	**84.57**	49.67	11.95
9	(20, 5)	**79.14**	62.21	55.32	**84.95**	56.06	37.2	**81.5**	74.06	16.54	**88.81**	56.94	30.97
10	(20, 5)	**77.35**	60.95	46.82	**84.18**	61.4	37.9	**90.43**	50.6	26	**79.13**	76.3	32.56
11	(20, 10)	**59.95**	43.31	30.85	**79.37**	66.86	15.34	**84.39**	52.86	24.21	**83.26**	62.05	21.94
12	(20, 10)	**70.54**	50.67	30.56	**74.04**	36.88	37.27	**70.34**	33.86	21.85	**87.07**	65.65	20.79
13	(20, 10)	**59.85**	49.08	23.28	**76.86**	51.01	20.89	**78.83**	51.33	33.39	**86.73**	59.03	44.67
14	(20, 10)	**74.11**	57.13	24.62	**76.01**	47.2	40.63	**85.09**	35.56	45.02	**86.32**	38.73	35.93
15	(20, 10)	**68.08**	52.86	28.89	**81.96**	51.39	33.11	**92.35**	35.25	30.1	**79.77**	53.44	27.15
16	(20, 10)	**70.58**	51.94	24.01	**84.32**	51.46	21.21	**76.02**	40.42	19.44	**82.65**	43.11	35.38
17	(20, 10)	**62.19**	52.81	34.01	**73.94**	51.54	30.42	**85.09**	77.7	7.634	**84.78**	27.55	11.22
18	(20, 10)	**71.91**	59.99	34.45	**87.12**	46.85	34.06	**84.74**	44.11	33.27	**91.65**	47.66	27.68
19	(20, 10)	**66.12**	48.33	34.51	**75.49**	48.59	30.8	**89.5**	47.96	19.72	**78.19**	55.68	38.17
20	(20, 10)	**85.96**	50.61	45.57	**81.54**	66.93	5.882	**75.99**	38.54	20.91	**84.48**	52.23	21.55
21	(20, 20)	**66.94**	56.36	18.23	**77.69**	43.06	28.97	**73.29**	57.53	16.98	**85.89**	59.06	11.96
22	(20, 20)	**57.59**	43.08	21.66	**78.28**	55.75	13.87	**73.07**	40.1	24.65	**84.31**	44.97	35.7
23	(20, 20)	**77.65**	44.56	30.56	**79.8**	47.08	7.967	**82.74**	44.95	28.47	**77.9**	45.2	32.84
24	(20, 20)	**77.69**	33.82	19.55	**72.66**	50.09	28.67	**80.17**	52.72	28.15	**99.19**	65.64	0.095
25	(20, 20)	**60.93**	37.55	22.3	**74.39**	53.23	28.93	**84.94**	42.58	3.285	**84.09**	37.93	15.12
26	(20, 20)	**65.7**	43.71	28.29	**82.04**	77	18.79	**84.85**	48.1	13.12	**88.88**	59.5	35.25
27	(20, 20)	**83.07**	47.13	27.34	**82.1**	59.62	31.64	**79.45**	49.56	21.29	**74.97**	43.47	37.1
28	(20, 20)	**67.16**	45.67	16.59	**78.98**	35.22	23.32	**72.69**	40.73	23.9	**90.04**	40.24	27.01
29	(20, 20)	**76.26**	54.89	21.65	**73.81**	49.47	29	75.16	77.58	24.33	**75.76**	41.3	26.15
30	(20, 20)	**71.32**	51.67	19.57	**78.15**	57.45	17.45	**84.37**	72.07	30.11	63.65	**77.68**	22.77
31	(50, 5)	**82.41**	57.39	52.71	**85.83**	63.31	15.01	**89.72**	88.05	2.887	**86.75**	64.31	14.58

32	(50, 5)	**77.08**	52.97	15.04	**90.52**	67.11	13.72	**74.6**	38.61	8.19	**94.46**	51.79	2.602
33	(50, 5)	**62.34**	46.54	20.21	**78.31**	31.71	19.47	**92.03**	84.5	2.781	**97.15**	67.36	0.373
34	(50, 5)	**79.79**	55.59	50.56	**73.68**	28.5	15.2	80.44	91.82	0.01	**86.67**	73.6	7.777
35	(50, 5)	**70.47**	51.44	18.75	**84.86**	62.02	22.86	**88.75**	43.2	6.027	**88.12**	38.37	8.996
36	(50, 5)	**68.23**	52.15	35.75	**80.12**	47.08	10.02	**81.17**	44.37	10.45	**95.8**	72.56	1.703
37	(50, 5)	**83.06**	55.89	52.76	**93.75**	29.64	9.723	**89.8**	71.09	10.86	**95.63**	80.71	3.711
38	(50, 5)	**62.3**	47.49	19.08	**81.8**	70.57	10.18	**96.51**	51.94	10.56	**82.89**	49.02	6.473
39	(50, 5)	**83.86**	45.98	41.32	**87.3**	62.18	17.5	**77.2**	29.4	12.6	**90.58**	43.02	11.88
40	(50, 5)	**82.62**	56.47	60.76	**83.81**	62.59	29.24	**92.96**	89.83	1.357	**86.9**	46.04	12.93
41	(50, 10)	**77.02**	58.65	9.705	**73.89**	40.88	21.88	**79.76**	44.42	20.55	**83.63**	38.57	14.24
42	(50, 10)	**60.14**	42.09	30.24	**81.08**	44.21	23.57	**94.62**	44.31	0.968	**89.12**	49.78	15.05
43	(50, 10)	**56.57**	47.79	6.16	**68.44**	43.71	19.1	**94.92**	43.84	4.505	**85.1**	25.12	11.79
44	(50, 10)	**57.02**	44.39	25.79	**77.27**	48.33	13.54	**90.45**	36.93	8.022	**89.52**	42.14	4.391
45	(50, 10)	**54.01**	40.47	21.28	**72.5**	51.99	17.06	**94.77**	65.1	3.09	**91.04**	37.43	11.95
46	(50, 10)	**74.93**	47.37	43.3	**76.64**	48.85	22.23	**90.48**	40.29	21.57	**94.29**	40.96	9.086
47	(50, 10)	**57.97**	40.75	20.77	**75.78**	42.64	21.47	**90.18**	55.02	3.111	**96.37**	82.1	9.243
48	(50, 10)	**61.35**	38.68	7.822	**77.15**	49.24	22.44	**83.25**	43.6	19.48	**93.21**	42.41	15.24
49	(50, 10)	**77.08**	33.32	19.25	**86.51**	44.21	9.63	**77.13**	29.64	11.83	89.47	90.58	4.524
50	(50, 10)	**64.24**	48.9	11.71	**80.56**	54.92	20.11	**90.75**	34.41	15.08	**77.26**	54.48	19.07
51	(50, 20)	**59.47**	41.99	13.4	**68.6**	44.15	10.97	**84.71**	27.24	9.695	**90.05**	25.91	7.123
52	(50, 20)	**72.69**	51.89	12.77	**93.09**	76.36	0.798	**94.74**	23.83	10.41	**95.01**	32.24	6.885
53	(50, 20)	**86.33**	51.64	6.4	**79.99**	50.67	17.37	**92.11**	87.69	7.301	**92.38**	40.02	16.6
54	(50, 20)	**55.71**	38.14	10.94	**85.09**	37.69	0	**94.76**	65.63	0	**94.44**	79.32	7.57
55	(50, 20)	**65.45**	50.51	14.18	**86.01**	50.5	13.74	**86.36**	28.15	5.657	**86.18**	54.24	9.527
56	(50, 20)	**70.44**	42.81	0	**76.28**	31.43	11.97	**90.12**	55.14	2.728	**86.9**	30.68	2.253
57	(50, 20)	**55.61**	41.34	8.647	**97.41**	67.41	0.03	91.13	92.11	6.661	**88.25**	27.25	9.579
58	(50, 20)	**57.28**	38.43	15.12	**78.66**	53.28	18.24	**91.12**	54.34	8.089	**89.59**	36.7	7.546
59	(50, 20)	**59.17**	39.67	20.02	**82.28**	20.41	6.944	**94.42**	37.08	7.513	87.73	85.47	0.936
60	(50, 20)	**71.93**	49.13	9.45	**90.8**	59.4	4E-04	81.38	**92.62**	17.61	**98.21**	74.54	0

Table 4.13 Inverted generational distance results of comparative experiments for taillard_ssd test instances.

Ins.	size	ssd10			ssd50			ssd100			ssd125		
		EMOACO	MOEA/OBL	MMOPSO	EMOACO	MOEA/OBL	MMOPSO	EMOACO	MOEA/OBL	MMOPSO	EMOACO	MOEA/OBL	MMOPSO
1	(20, 5)	**0.515**	10.18	32.91	**6.536**	15.01	35.45	13.9	9.923	113.7	**2.427**	15.75	71.48
2	(20, 5)	**1.286**	8.764	27.57	**0**	20.03	39.85	**7.946**	25.56	115.7	**3.78**	22.31	195.3
3	(20, 5)	**4.735**	20.49	29.41	**3.011**	128.5	60.61	**3.159**	6.993	136.7	**3.992**	54.01	58.24
4	(20, 5)	**0.481**	13.7	24.46	**0.751**	8.332	29.88	**1.085**	31.33	150.5	**16.21**	38.09	71.62
5	(20, 5)	**1.869**	3.574	30.57	**7.964**	19	54.34	**4.607**	19.98	78.86	**6.733**	12.01	109.7
6	(20, 5)	**0.553**	7.64	42.9	**14.05**	18.08	118.1	24.71	**12.87**	180.3	**11.98**	13.01	73
7	(20, 5)	**0.875**	7.103	26.51	**0**	15.86	70.52	20.17	**18.61**	50.01	**0.786**	17.47	75.17
8	(20, 5)	**0.962**	8.743	17.05	12.68	9.826	62.07	**4.436**	17.61	45.32	**5.8**	35.6	105.6
9	(20, 5)	**0.973**	9.275	22.65	**2.459**	21.48	54.53	11.78	10.84	143	**5.191**	55.64	76.9
10	(20, 5)	**1.145**	9.999	26.47	**0**	15.96	63.88	**2.721**	54.4	77.2	9.898	9.051	56.64
11	(20, 10)	**0.353**	12.92	29.03	**3.307**	13.95	91.25	**2.717**	22.15	71.5	**4.228**	24.2	81.43
12	(20, 10)	**0.678**	14.65	47.64	**0.296**	35.22	36.53	**13.8**	24.02	38.27	**1.348**	13.94	103.1
13	(20, 10)	**1.659**	9.886	37.54	**1.131**	19.56	54.66	**0.833**	22.39	47.31	**1.414**	18.73	49.26
14	(20, 10)	**1.948**	13.4	46.29	**0.1**	20.77	33.71	**4.568**	149.6	101.7	**0**	42.02	50.54

15	(20, 10)	**0.922**	13.23	43.91	**0**	27.84	54.77	**10.35**	42.74	61.51	**4.279**	19.69	63.45
16	(20, 10)	**0.608**	16.95	47.72	**0**	37.27	94.39	**6.791**	106.7	125.5	**4.492**	27.69	41.73
17	(20, 10)	**3.119**	10.86	30.73	**0.315**	17.25	44.47	**7.384**	11.86	220.2	**7.7**	713	720.1
18	(20, 10)	**0.971**	10.22	39.78	**0.222**	25.17	61.3	**0.591**	24.45	58.19	**0**	33.05	62.86
19	(20, 10)	**0.487**	13.85	37.54	**1.854**	24.05	50.47	**1.309**	84.38	111.1	**5.143**	21.95	40.99
20	(20, 10)	**1.095**	21.4	30.3	**4.989**	15.38	104.7	**0**	29.46	53.43	**0**	28.33	74.19
21	(20, 20)	**3.798**	14.4	50.71	**0.195**	26.54	48.53	**3.93**	12.05	75.27	**3.017**	35.57	157.5
22	(20, 20)	**2.157**	15.31	38.57	**0.373**	17.95	68.91	**4.286**	27.2	53.24	**7.862**	25.8	58.22
23	(20, 20)	**1.046**	22.16	59.55	**0.926**	19.37	87.81	**0**	29.33	63.17	**4.981**	38.53	65.72
24	(20, 20)	**1.438**	155.9	116.7	**0.676**	16.09	41.37	**0.705**	19.9	47.85	**0**	185.1	899.5
25	(20, 20)	**0.085**	24.43	41.4	**2.985**	18.52	46.75	**6.196**	27.42	204.1	**5.275**	29.55	64.55
26	(20, 20)	**1.028**	16.53	39.09	**5.924**	8.808	93.01	**1.916**	30.95	81.89	**2.009**	21.96	74.39
27	(20, 20)	**0.481**	22.72	41.17	**0**	16.66	51.7	**4.231**	27.44	72.4	**7.398**	35.66	41.21
28	(20, 20)	**0.414**	19.82	48.72	**2.81**	34.39	52.25	**0.857**	24.34	49.23	**0.313**	197.1	192.3
29	(20, 20)	**2.359**	17.95	52.41	**1.823**	16.25	53.56	11.45	**7.749**	101.1	**0**	27.86	50.78
30	(20, 20)	**3.412**	17.75	51.06	**1.784**	13.74	88.74	**2.245**	13.58	62.35	16.04	**5.429**	78.35
31	(50, 5)	**1.676**	12.95	28.29	**0.576**	14.74	120.8	**11.39**	16.9	580.4	**6.288**	25.81	99.62
32	(50, 5)	**1.722**	16.42	69.5	**2.505**	15.91	170.9	**28.3**	33.67	120.2	23.2	45.93	572.2
33	(50, 5)	**0.449**	11.42	39.83	**24.02**	33.99	45.14	7.43	11.38	328.4	**7.109**	45.08	513.3
34	(50, 5)	**2.205**	18.33	27.7	**20.3**	46.95	58.99	50.6	**4.71**	539.4	**9.412**	13.17	174.3
35	(50, 5)	**0.875**	12.39	58.68	**6.71**	24.2	77.02	37.28	**29.3**	185.5	**15.51**	34.98	137.8
36	(50, 5)	**1.1**	9.529	34.03	**9.189**	22.97	89.72	29.09	**27.34**	88.81	**7.215**	46.58	840.8
37	(50, 5)	**1.627**	15.91	25.55	**12.18**	118.8	195.2	**8.725**	13.84	200.5	**9.092**	52.03	673.3
38	(50, 5)	**0.774**	12.05	40.94	**0.793**	8.953	90.42	**14.39**	31.55	231.2	18.39	23.82	216.7
39	(50, 5)	**11.26**	26.89	36.5	**0.157**	21.21	121.6	**31.35**	42.98	82.52	**13.39**	144	260.6
40	(50, 5)	**2.688**	20.36	32.59	**2.118**	16.43	64.94	**6.007**	25.02	916.8	**23.42**	26.7	167.6
41	(50, 10)	**3.825**	26.42	108.2	**0**	27.46	50.16	**6.537**	21.37	69.54	**6.344**	37.49	118.1
42	(50, 10)	**1.743**	12.63	33.15	**9.177**	23.63	52.27	**8.288**	61.78	297	**14.64**	24.49	142.3
43	(50, 10)	**3.992**	11.57	63	**19.16**	23.32	56.44	28.01	31.47	425	**16.72**	46.13	93.35
44	(50, 10)	**1.871**	11.95	32.7	**3.869**	14.5	83.1	**17.18**	57.28	240.9	**16.53**	42.31	395.4
45	(50, 10)	**1.658**	13.3	37.17	**3.22**	15.26	82.95	**7.78**	43.3	264.7	**22.38**	37.4	237.8
46	(50, 10)	**0.936**	15.55	29.39	**5.118**	18.37	83.47	**3.155**	52.55	92.02	**0.678**	62.02	309.8
47	(50, 10)	**0.907**	13.56	36.82	**1.333**	24.61	58.19	**14.33**	65.6	454.9	**2.99**	28.46	344.6
48	(50, 10)	**0.21**	18.87	71.64	**1.055**	20.64	70.36	**3.854**	41.99	74.84	**5.281**	76.05	175.5
49	(50, 10)	**1.093**	29.52	54.68	**0.733**	22.9	95.69	34.45	**26.74**	81.55	41.91	**11.19**	459.4
50	(50, 10)	**2.164**	13.33	65.87	**0.937**	16.83	90.52	**14.98**	162.9	195.2	**6.64**	26.66	84.02
51	(50, 20)	**1.636**	13.26	48.24	**1.497**	21.99	65.7	**12.74**	43.43	97.3	**25.84**	115.9	206.7
52	(50, 20)	**2.576**	15.19	82.2	**2.987**	24.6	252.6	**2.199**	169	218.3	**2.324**	161.6	324.4
53	(50, 20)	**0**	33.64	119.7	**1.56**	21.08	95.53	17.49	**14.82**	407	**15.66**	30.96	262.8
54	(50, 20)	**1.037**	15.22	45.85	**12.3**	39.32	200.1	**5.363**	3181	9120	**4.238**	43.97	435
55	(50, 20)	**3.954**	13.63	63.35	**0.503**	25.28	123.9	**18.36**	117.1	224.1	**4.664**	29.19	122.9
56	(50, 20)	**1.351**	13.65	159.6	**3.245**	42.83	91.56	**11.08**	33.14	210.9	**26.69**	46.66	195.3
57	(50, 20)	**2.673**	13.39	50.8	**2.286**	47.88	601.4	8.066	6.413	329.9	**26.67**	40.09	174.3
58	(50, 20)	**0.695**	14.77	41.76	**4.137**	17.07	101.3	**8.057**	32.85	179.3	**13.14**	109.6	541.9
59	(50, 20)	**1.479**	13.46	41.93	**14.52**	52.03	113.4	**13.24**	68.6	212.2	15.38	10.93	342.6
60	(50, 20)	**1.597**	14.22	87.29	**2.017**	22.48	325.4	46.8	**0.461**	456.2	**17.47**	53.37	1843

Table 4.14 Overall spread results of comparative experiments for taillard_ssd test instances.

Ins.	size	ssd10			ssd50			ssd100			ssd125		
		EMOACO	MOEA/OBL	MMOPSO	EMOACO	MOEA/OBL	MMOPSO	EMOACO	MOEA/OBL	MMOPSO	EMOACO	MOEA/OBL	MMOPSO
1	(20, 5)	**100**	10.02	3.056	**59.77**	33.51	3.928	**50.44**	21.94	9.283	**46.81**	10.4	9.372
2	(20, 5)	**99.31**	19.51	6.698	**87.31**	25.45	2.552	**46.01**	8.167	1.196	**40.6**	14.92	3.361
3	(20, 5)	**81.73**	13.97	0.71	**47.37**	5.098	4.514	**50.09**	26.24	9.898	**33.53**	10.46	6.427
4	(20, 5)	**100**	19.24	3.276	**92.36**	33.89	15.8	**43.73**	6.394	0.789	28.74	23.77	9.658
5	(20, 5)	**84.42**	72.18	26.28	**39.49**	25.69	5.612	**56.83**	19.85	2.055	**54.49**	21.48	4.594
6	(20, 5)	**90.07**	28.11	7.383	**41.19**	20.49	10.37	21.65	**33.92**	4.449	**43.6**	12.54	3.725
7	(20, 5)	**95.69**	22.88	19.4	**63.84**	17.5	3.517	**23.64**	9.806	1.858	**60.11**	16.06	6.73
8	(20, 5)	**100**	24.68	21.2	**63.62**	8.984	5.791	**66.3**	18.97	7.57	**43.31**	6.139	7.155
9	(20, 5)	**99.93**	26.64	3.739	**57.24**	14.77	6.565	**37.74**	17.45	6.719	**38.61**	30.04	2.563
10	(20, 5)	**85.23**	24.52	11.21	**71.05**	11.95	5.747	**48.12**	6.954	1.815	**80.09**	17.04	7.614
11	(20, 10)	**100**	15.32	4.836	**70.26**	16.7	8.02	**65.94**	23.65	9.993	**72.79**	15.82	7.614
12	(20, 10)	**83.32**	10.24	5.135	**90.54**	5.457	3.232	**59.76**	21.04	1.247	**63.45**	18.82	5.104
13	(20, 10)	**100**	18.1	5.834	**100**	9.454	1.024	**89.64**	6.219	1.874	**53.25**	20.05	8.225
14	(20, 10)	**77.02**	22.1	18.13	**94.32**	15.72	4.302	21.09	**25.77**	1.779	**70.98**	8.202	3.038
15	(20, 10)	**94.7**	24.21	6.406	**56.43**	20.76	5.835	**50.72**	14.18	2.243	**79.55**	18.11	5.901
16	(20, 10)	**92.72**	10.56	5.1	**62.95**	10.87	2.465	**46.72**	15.4	2.204	**39.49**	14.33	4.023
17	(20, 10)	**100**	12.8	0.766	**91.85**	19.41	2.938	33.27	14.86	14.28	**47.97**	4.575	4.826
18	(20, 10)	**92.48**	21.21	4.491	**66.6**	17.16	8.422	**65.11**	20.15	4.904	**86.66**	8.708	0.55
19	(20, 10)	**98.73**	23.25	0.918	**76.01**	19.59	2.179	**52.9**	12.68	5.761	**81.22**	6.332	3.159
20	(20, 10)	**95.38**	3.844	0.324	**52.99**	34.48	7.146	**82.7**	10.39	1.802	**68.37**	12.04	10.32
21	(20, 20)	**94.33**	10.35	2.317	**90.44**	11.36	2.661	**75.34**	18.64	5.144	**40**	4.574	10.32
22	(20, 20)	**100**	8.048	0.889	**65.86**	16.06	15.47	**72.4**	3.1	2.368	**56.86**	7.975	1.1
23	(20, 20)	**52.47**	20.52	0.422	**76.4**	4.047	0.672	**75.52**	6.046	6.809	**51.71**	5.557	1.484
24	(20, 20)	**71.13**	6.275	1.964	**93.06**	12.66	7.096	**93.04**	11.45	7.097	**16.55**	2.474	2.023
25	(20, 20)	**94.23**	4.401	1.254	**98**	14.9	2.388	**47.12**	7.05	1.173	**94.84**	15.05	3.312
26	(20, 20)	**100**	9.355	1.146	**53.02**	21.29	16.69	**78.57**	17.39	5.691	**77.54**	3.127	2.633
27	(20, 20)	**86.63**	0.584	2.621	**86.03**	15.74	7.239	**73.02**	5.681	2.902	**20.56**	14.08	10.1
28	(20, 20)	**98.37**	5.573	1.919	**69.71**	15.47	3.568	**99.82**	4.137	0.845	**85.46**	13.8	0.883
29	(20, 20)	**75.18**	7.562	14.1	**83.87**	14.85	0.707	**70.3**	22.09	10.25	**70.81**	14.8	3.009
30	(20, 20)	**72.96**	12.15	6.722	**66.68**	12.32	2.658	**67.43**	9.905	24.45	**76.62**	20.05	12.93
31	(50, 5)	**100**	7.688	0.101	**51.28**	5.83	0.994	3.856	7.671	4.007	**36.38**	4.671	0.812
32	(50, 5)	**82.92**	5.621	3.925	**47.6**	3.408	1.109	**12.2**	2.994	1.734	**7.536**	3.267	1.531
33	(50, 5)	**95.87**	17.62	4.447	**31.48**	1.398	1.794	**9.928**	8.939	4.356	**15.35**	3.773	0.848
34	(50, 5)	**100**	6.616	0.564	**37.14**	2.918	1.045	**6.353**	5.682	0.228	**34.07**	4.787	1.68
35	(50, 5)	**81.3**	10.59	3.359	**73.12**	4.211	1.084	**5.06**	3.238	1.261	**26.14**	4.666	1.656
36	(50, 5)	**100**	16.13	0.662	**55.41**	7.284	1.326	**13.83**	7.09	2.33	4.879	1.012	4.442
37	(50, 5)	**100**	2.862	0.231	**12.62**	5.397	5.997	**30.15**	10.3	5.372	11.9	5.451	4.378
38	(50, 5)	**85.3**	9.381	9.714	**58.37**	42.66	15.09	11.47	6.982	1.387	**5.83**	4.722	1.34
39	(50, 5)	**72.71**	3.353	0.534	**51.28**	4.15	1.406	**14.8**	0.898	1.925	**19.67**	1.681	2.365
40	(50, 5)	**97.97**	4.161	0.473	**74.16**	5.873	3.617	9.214	**17.58**	1.795	**13.32**	3.569	1.375
41	(50, 10)	**68.59**	17.62	4.985	**82.87**	12.19	0.852	**56.99**	9.745	2.539	**42.59**	3.922	1.588
42	(50, 10)	**100**	5.212	0.209	**72.23**	4.137	0.81	**14.6**	5.284	1.328	**30.84**	4.433	2.867
43	(50, 10)	**96.12**	13.85	3.102	**30.97**	11.25	5.97	3.616	5.559	3.218	**19.52**	4.074	3.383
44	(50, 10)	**98.38**	19.02	2.974	**63.96**	6.743	1.48	**10.15**	6.321	2.298	**7.627**	3.436	2.628
45	(50, 10)	**100**	11.74	0.354	**70.05**	10.66	1.364	**8.795**	5.393	6.651	**13.4**	3.728	1.043
46	(50, 10)	**100**	5.27	0.76	**64.53**	11.12	3.297	**37.73**	4.816	5.334	**20.4**	1.65	1.406

47	(50, 10)	**99.64**	6.439	2.482	**80.9**	8.514	1.704	7.294	8.549	3.927	9.098	2.746	11.17
48	(50, 10)	**86.89**	13.37	0.658	**74.39**	4.941	0.955	**65.5**	9.752	2.581	**22.18**	12.98	5.436
49	(50, 10)	**93.75**	10.19	0.946	**73.97**	3.505	4.101	**20.17**	2.796	2.085	**25.22**	2.22	6.353
50	(50, 10)	**82.52**	10.24	1.97	**58.69**	11.25	3.058	**9.486**	4.711	2.469	**42.12**	5.389	4.048
51	(50, 20)	**96.33**	6.353	3.374	**86.16**	4.722	0.989	**45.04**	3.129	4.674	4.905	0.971	5.786
52	(50, 20)	**66.79**	1.914	2.676	**14.33**	1.499	1.986	**24.14**	2.493	3.887	**16.15**	1.772	5.236
53	(50, 20)	**43.89**	3.831	8.675	**59.94**	2.406	1.669	8.197	8.775	11.3	**22.07**	2.031	8.624
54	(50, 20)	**97.94**	3.106	1.875	**21.88**	6.629	0.125	**7.881**	0.145	0.517	15.05	4.794	7.799
55	(50, 20)	**85.59**	7.205	6.853	**58.26**	3.825	2.851	**6.863**	1.883	5.851	**47.26**	2.714	3.855
56	(50, 20)	**60.85**	2.097	0	**60.82**	3.454	1.19	**16.36**	6.008	6.155	**26.94**	1.191	2.646
57	(50, 20)	**100**	5.324	1.302	**9.07**	0.035	0.193	**13.87**	10.28	10.91	**8.502**	1.584	7.571
58	(50, 20)	**100**	6.847	0.004	**56.52**	2.965	2.483	**38.61**	14.8	6.474	**17.92**	2.734	6.575
59	(50, 20)	**94.51**	6.041	2.286	**36.58**	1.837	1.401	**26.71**	2.439	4.992	**10.05**	3.568	4.389
60	(50, 20)	**67.14**	3.928	8.027	**23.1**	1.057	0.074	16.86	16.85	12.13	**0.939**	0.121	0.009

To assess the limitations of the MILP model, the CPLEX optimization software was utilized to generate a subset of the optimal PF for a set of relatively small instances. Given the problem's complexity, CPLEX struggled to find optimal solutions within a reasonable runtime for instances larger than 5 jobs, 3 machines, and 2 AGVs. Consequently, three instances denoted as P1, P2, and P3 of this size were optimally solved using the ε-constraint method. This method involves solving the minimum $Cmax$ and minimum TEC subproblems independently and subsequently generating 8 additional solutions by relaxing a constraint on $Cmax$ while optimizing TEC. For this experiment, CPLEX was allotted 8 hours of runtime to find an optimal solution within a 1% tolerance threshold. In parallel, the EMOACA algorithm was applied to these instances using a population of 10 ants, 3 pheromone matrices, and 100 iterations for each run, with a total of 20 independent runs. The solutions generated by the 10 ants were compared with the optimal PF based solely on convergence. A modified version of the convergence measure Υ proposed by Deb et al. [161] was employed for this purpose. In this measure, the obtained optimal and EMOACA-generated fronts were normalized. Subsequently, the minimum Euclidean distance in the objective space needed for a dominated solution from EMOACA to become nondominated was computed and then multiplied by 100% to express it as a percentage gap. If a solution was already nondominated, its distance was set to 0, and the mean distance for all PF points was calculated. The results of the modified convergence measure for the three test instances are presented in Table 4.15. The findings indicate that EMOACA exhibits outstanding performance across all test problems. The average convergence gap is less than 1%, with a maximum of 5% observed in extremely rare cases. In fact, 65.3% of the generated solutions are nondominated by optimal solutions, and 94% of them are within 1% of the optimal PF.

Table 4.15 Results of the modified convergence measure Υ of the solutions generated from the EMOACA with respect to optimal samples.

Test problem	Median (%)	Mean (%)	Minimum (%)	Maximum (%)	Stdv. (%)
P1	0.821	0.52	0.047	1.063	0.395
P2	0.103	0.612	0	5.01	1.2
P3	0.387	0.624	0.013	1.372	0.567

4.5 Conclusion

This chapter delved into the Energy-Efficient Flow Shop Scheduling Problem, considering constraints related to blocking and collision-free transportation. An integrated scheduling model was developed to optimize both total energy consumption and maximum completion time simultaneously. To address this complex problem, an enhanced multi-objective ant colony algorithm was introduced. The algorithm, along with its various enhancement techniques, was rigorously evaluated through a series of computational experiments and compared to multiple multi-objective algorithms from the literature. The results clearly demonstrate the superiority of the proposed algorithm across a wide range of comparison metrics. Furthermore, the enhancement techniques introduced in this study significantly bolstered its performance. Consequently, the algorithm was employed to conduct an in-depth analysis of the impact of problem parameters, such as processing times and the number of AGVs, on energy consumption and makespan.

Conclusion and future work

The thesis at hand presented an inquiry into the broad topic of flow shop scheduling under shared renewable and nonrenewable resources. Two distinct cases of this problem were investigated, mainly: the Flow Shop and Maintenance Scheduling Problem under Shared Nonrenewable Resource constraint (FMSP-SNR) and the Energy-Efficient Flow Shop Scheduling Problem with Blocking and Collision-Free Transportation Constraints (EFSP-BCFT).

In the initial part of this scholarly work, Chapter 1 offered a comprehensive overview of the entire thesis, emphasizing the study's objectives and significance, while also highlighting its relevance from both academic and practical perspectives. Chapter 2 was dedicated to an extensive review of the existing literature concerning FMSP-SNR and EFSP-BCFT. A concise summary and synthesis of previous research was provided.

Chapter 3 concentrated on studying the FMSP-SNR with the objectives to minimize both the makespan and total production costs, encompassing various factors such as maintenance, resource consumption, and resource inventory costs in the context of a permutation flow shop environment. A Non-Linear Mixed Integer programming formulation was developed to model the problem and generate optimal solutions to some small-size instances. To approximate the Pareto optimal front for larger instance of this bi-objective problem, the study proposed a novel bi-objective randomized local search (BORLS) and employed two prominent metaheuristic methods: the nondominated sorting genetic algorithm (NSGA-II) and a bi-objective adaptation of the particle swarm optimization (BOPSO) as contender methodologies to test the effectiveness of the proposed BORLS procedure. The results of the computational experiments undertaken revealed valuable insights. In the first set of experiments, a comparison between the two metaheuristics using purely random initial populations showcases the superiority of adapted BOPSO over NSGA-II. In the second set, where the initial populations were enriched with heuristically generated solutions from BORLS, the algorithms demonstrate comparable performance for large problems. Notably, the BORLS method stands out by generating superior solutions when the emphasis is on total production cost. These findings contribute significantly to the understanding and solution of the FMSP-

SNR problem, offering valuable tools and insights for practitioners and researchers in the fields of production scheduling, maintenance planning and resource allocation and management.

Transitioning to the latter case of the flow shop scheduling under shared renewable and nonrenewable resources, Chapter 4 was dedicated to the EFSP-BCFT while considering Sequence Dependent Setup Times (SDST). This problem extends its scope to the scheduling of Automated Guided Vehicles (AGVs) with the objective of minimizing both makespan and total energy consumption. To grapple with this multi-objective problem, the thesis introduces the Enhanced Multi-Objective Ant Colony Algorithm (EMOACA), a novel algorithm designed to tackle the complexities of the EFSP-BCFT. A collection of innovative techniques were developed and integrated into the EMOACA framework. These include – but not exhaustively – a novel low-high resolution strategy, the utilization of heuristic information for AGV dispatching, and an energy reduction improvement procedure founded on identifying critical paths in the schedule. A comprehensive array of computational experiments was conducted to validate the effectiveness of the proposed method and its enhancements, benchmarking them against conventional and state-of-the-art metaheuristics. These experiments encompass diverse problem scenarios, ranging from case studies to different problem sizes, demonstrating the ability of EMOACA and its enhancements to consistently produce high-quality solutions which allowed for a deep analysis of the impact of the different problem parameters on the scheduling outcome.

In summary, this thesis makes significant contributions to the realms of scheduling and optimization within the manufacturing and logistics domains. By dissecting and addressing two complex scheduling problems – the FMSP-SNR and EFSP-BCFT with SDST – it advances the current state-of-the-art and equips both researchers and practitioners with innovative tools and insights to tackle them effectively. The combination of rigorous problem formulations, advanced metaheuristic algorithms, and extensive computational experiments underscores the thesis's commitment to pushing the boundaries of knowledge in these critical areas of operations research and industrial engineering.

In future research regarding the topics related to the FMSP-SNR, our focus will be on investigating the impact of incorporating multiple resources with different configurations, including global, local, and partially-shared resources, into the model, and understanding its implications on complexity. Additionally, we are interested in developing extensions of the algorithms presented in this study to accommodate diverse performance criteria and constraints in the system. Moreover, we plan to explore how these techniques can be adapted to handle other production environments such as parallel machines or job shop setups. By delving into these areas, we aim to further enhance the applicability and versatility of the proposed methods.

While this thesis has made significant strides in addressing key aspects of the EFSP-BCFT, such as incorporating variable AGV speeds, collision prevention, and battery management – areas often overlooked in existing literature – it is important to highlight the need for further research in several areas. Regarding the transportation aspect, future studies could explore the implementation of a bi-directional guide path with shortcuts and track maintenance. Additionally, extending the model to account for variable transportation capacity, including both the number of AGVs and load capacity, while considering factors like AGV breakdown and deployment costs, would be a valuable extension of the current framework. In terms of machine scheduling, a potential avenue for future investigation could involve examining the flexible flow shop with passing, providing insights into more complex scheduling environments.

References

1. Wang, S., & Liu, M. (2015). Multi-objective optimization of parallel machine scheduling integrated with multi-resources preventive maintenance planning. Journal of Manufacturing Systems, 37, 182-192. https://doi.org/10.1016/j.jmsy.2015.07.002

2. EIA, U.S.Energy Information Administration. International Energy Outlook 2016.

3. Dai, M., Tang, D., Giret, A., Salido, M. A., & Li, W. D. (2013). Energy-efficient scheduling for a flexible flow shop using an improved genetic-simulated annealing algorithm. Robotics and Computer-Integrated Manufacturing, 29(5), 418–429. https://doi.org/10.1016/j.rcim.2013.04.001

4. Liu, Y., Dong, H., Lohse, N., Petrovic, S., & Gindy, N. (2014). An investigation into minimising total energy consumption and total weighted tardiness in job shops. Journal of Cleaner Production, 65, 87–96. https://doi.org/10.1016/j.jclepro.2013.07.060

5. EIA. In: International Energy Outlook 2018.

6. European Council (2014) Euco 169/14 Brussels, Belgium, European Council. https://www.consilium.europa.eu/uedocs/cms_data/docs/pressdata/de/ec/145424.pdf

7. Javied, T., Bakakeu, J., Gessinger, D., & Franke, J. (2018). Strategic energy management in industry 4.0 environment. 2018 Annual IEEE International Systems Conference (SysCon). https://doi.org/10.1109/syscon.2018.8369610

8. Li, W., Zein, A., Kara, S., & Herrmann, C. (2011). An Investigation into Fixed Energy Consumption of Machine Tools. Glocalized Solutions for Sustainability in Manufacturing, 268–273. https://doi.org/10.1007/978-3-642-19692-8_47

9. Mori, M., Fujishima, M., Inamasu, Y., & Oda, Y. (2011). A study on energy efficiency improvement for machine tools. CIRP Annals, 60(1), 145–148. https://doi.org/10.1016/j.cirp.2011.03.099

10. Neugebauer, R., Wabner, M., Rentzsch, H., & Ihlenfeldt, S. (2011). Structure principles of energy efficient machine tools. CIRP Journal of Manufacturing Science and Technology, 4(2), 136–147. https://doi.org/10.1016/j.cirpj.2011.06.017

11. Rahimifard, S., Seow, Y., & Childs, T. (2010). Minimising Embodied Product Energy to support energy efficient manufacturing. CIRP Annals, 59(1), 25–28. https://doi.org/10.1016/j.cirp.2010.03.048

12. Liu, C.-H., & Huang, D.-H. (2013). Reduction of power consumption and carbon footprints by applying multi-objective optimisation via genetic algorithms. International Journal of Production Research, 52(2), 337–352. https://doi.org/10.1080/00207543.2013.825740

13. Ding, J.-Y., Song, S., & Wu, C. (2016). Carbon-efficient scheduling of flow shops by multi-objective optimization. European Journal of Operational Research, 248(3), 758–771. https://doi.org/10.1016/j.ejor.2015.05.019

14. European Environment Agency. (2020). Greenhouse gas: Emissions share by sector in EU28 (2017) Accessed online 10 Apr. 2020 https://www.eea.europa.eu/data

15. Gao, K., Huang, Y., Sadollah, A., & Wang, L. (2019). A review of energy-efficient scheduling in intelligent production systems. Complex & Intelligent Systems. https://doi.org/10.1007/s40747-019-00122-6

16. Fazlollahtabar, H., & Saidi-Mehrabad, M. (2013). Methodologies to Optimize Automated Guided Vehicle Scheduling and Routing Problems: A Review Study. Journal of Intelligent & Robotic Systems, 77(3-4), 525–545. https://doi.org/10.1007/s10846-013-0003-8.

17. Peterson, S. B., & Michalek, J. J. (2013). Cost-effectiveness of plug-in hybrid electric vehicle battery capacity and charging infrastructure investment for reducing US gasoline consumption. Energy Policy, 52, 429–438. https://doi.org/10.1016/j.enpol.2012.09.059

18. Lyon, T. P., Michelin, M., Jongejan, A., & Leahy, T. (2012). Is “smart charging” policy for electric vehicles worthwhile? Energy Policy, 41, 259–268. https://doi.org/10.1016/j.enpol.2011.10.045

19. Kabir, Q. S., & Suzuki, Y. (2018). Comparative analysis of different routing heuristics for the battery management of automated guided vehicles. International Journal of Production Research, 1–18. https://doi.org/10.1080/00207543.2018.1475761

20. De Ryck, M., Versteyhe, M., & Shariatmadar, K. (2020). Resource management in decentralized industrial automated guided vehicle systems. Journal of Manufacturing Systems, 54, 204–214. https://doi.org/10.1016/j.jmsy.2019.11.003

21. Vis, I. F. A. (2006). Survey of research in the design and control of automated guided vehicle systems. European Journal of Operational Research, 170(3), 677–709. https://doi.org/10.1016/j.ejor.2004.09.020

22. Meißner, M., & Massalski, L. (2020). Modeling the electrical power and energy consumption of automated guided vehicles to improve the energy efficiency of production systems. The International Journal of Advanced Manufacturing Technology, 110(1-2), 481–498. https://doi.org/10.1007/s00170-020-05796-8

22. Pinedo, M. L. (2008). Scheduling: Theory, algorithms, and systems. Springer Science & Business Media. https://doi.org/10.1007/978-3-319-26580-3

23. Demeulemeester, E. L., & Herroelen, W. S. (2006). Project scheduling: a research handbook (Vol. 49). Springer Science & Business Media.

24. Baker, K. R., & Trietsch, D. (2013). Principles of sequencing and scheduling. John Wiley & Sons.

25. Bolz, R. W. (2012). Manufacturing automation management: a productivity handbook. Springer Science & Business Media.

26. Kuo, Y. H., Leung, J. M., & Yano, C. A. (2014). Scheduling of multi-skilled staff across multiple locations. Production and Operations Management, 23(4), 626-644. https://doi.org/10.1111/poms.12184

27. Nolz, P. C. (2021). Optimizing construction schedules and material deliveries in city logistics: A case study from the building industry. Flexible Services and Manufacturing Journal, 33(3), 846-878. https://doi.org/10.1007/s10696-020-09391-7

28. Gahm, C., Denz, F., Dirr, M., & Tuma, A. (2016). Energy-efficient scheduling in manufacturing companies: A review and research framework. European Journal of Operational Research, 248(3), 744-757. https://doi.org/10.1016/j.ejor.2015.07.017

29. Fang, K., Uhan, N., Zhao, F., & Sutherland, J. W. (2011). A new approach to scheduling in manufacturing for power consumption and carbon footprint reduction. Journal of Manufacturing Systems, 30(4), 234-240. https://doi.org/10.1016/j.jmsy.2011.08.004

30. Kwon, B., & Lee, G. M. (2015). Spatial scheduling for large assembly blocks in shipbuilding. Computers & Industrial Engineering, 89, 203-212. https://doi.org/10.1016/j.cie.2015.04.036

31. Itoh, T. D., Horinouchi, T., Uchida, H., Takahashi, K., & Ozaki, H. (2021). Optimal scheduling for laboratory automation of life science experiments with time constraints. SLAS TECHNOLOGY: Translating Life Sciences Innovation, 26(6), 650-659. https://doi.org/10.1177/24726303211021790

32. Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. In Annals of discrete mathematics (Vol. 5, pp. 287-326). Elsevier. https://doi.org/10.1016/S0167-5060(08)70356-X

33. Zarandi, M. F., Mosadegh, H., & Fattahi, M. (2013). Two-machine robotic cell scheduling problem with sequence-dependent setup times. Computers & Operations Research, 40(5), 1420-1434. https://doi.org/10.1016/j.cor.2012.09.006

34. Lunardi, W. T., Birgin, E. G., Laborie, P., Ronconi, D. P., & Voos, H. (2020). Mixed integer linear programming and constraint programming models for the online printing

shop scheduling problem. Computers & Operations Research, 123, 105020. https://doi.org/10.1016/j.cor.2020.105020

35. Lee, K., Lei, L., & Pinedo, M. (2012). Production scheduling with history-dependent setup times. Naval Research Logistics (NRL), 59(1), 58-68. https://doi.org/10.1002/nav.21472

36. Bitran, G. R., & Gilbert, S. M. (1989). Sequencing production on parallel machines with two magnitudes of sequence dependent setup costs.

37. Naderi, B., Zandieh, M., & Fatemi Ghomi, S. M. T. (2009). Scheduling sequence-dependent setup time job shops with preventive maintenance. The International Journal of Advanced Manufacturing Technology, 43, 170-181. https://doi.org/10.1007/s00170-008-1693-0

38. Chen, L., Wang, J., & Zhu, Y. (2020). Reentrant hybrid flow shop scheduling problem with non-anticipatory sequence-dependent setup time. International Journal of Industrial and Systems Engineering, 36(4), 560-576. https://doi.org/10.1504/IJISE.2020.112073

39. Zandieh, M., & Gholami, M. (2009). An immune algorithm for scheduling a hybrid flow shop with sequence-dependent setup times and machines with random breakdowns. International Journal of Production Research, 47(24), 6999–7027. https://doi:10.1080/00207540802400636

40. Allahverdi, A., Gupta, J. N., & Aldowaisan, T. (1999). A review of scheduling research involving setup considerations. Omega, 27(2), 219-239. https://doi.org/10.1111/j.1937-5956.2000.tb00137.x

41. Allahverdi, A., Ng, C. T., Cheng, T. E., & Kovalyov, M. Y. (2008). A survey of scheduling problems with setup times or costs. European journal of operational research, 187(3), 985-1032. https://doi.org/10.1016/j.ejor.2006.06.060

42. Shrouf, F., Ordieres-Meré, J., García-Sánchez, A., & Ortega-Mier, M. (2014). Optimizing the production scheduling of a single machine to minimize total energy consumption costs. Journal of Cleaner Production, 67, 197-207. https://doi.org/10.1016/j.jclepro.2013.12.024

43. Mansouri, S. A., Aktas, E., & Besikci, U. (2016). Green scheduling of a two-machine flowshop: Trade-off between makespan and energy consumption. European journal of operational research, 248(3), 772-788. https://doi.org/10.1016/j.ejor.2015.08.064

44. Moon, D. H., Kim, D. K., & Jung, J. Y. (2004). An operator load-balancing problem in a semi-automatic parallel machine shop. Computers & Industrial Engineering, 46(2), 355-362. https://doi.org/10.1016/j.cie.2003.12.015

45. Selvarajah, E., & Zhang, R. (2014). Supply chain scheduling at the manufacturer to minimize inventory holding and delivery costs. International Journal of Production Economics, 147, 117-124. https://doi.org/10.1016/j.ijpe.2013.08.015

46. Gulati, R., & Smith, R. (2009). Maintenance and reliability best practices. Industrial Press Inc.

47. Campbell, J. D., Reyes-Picknell, J. V., & Kim, H. S. (2015). Uptime: Strategies for excellence in maintenance management. CRC Press.

48. Meyers, F. E., & Stephens, M. P. (2005). Manufacturing facilities design and material handling. Pearson Educación.

49. Coyle, J. J., Novack, R. A., Gibson, B., & Bardi, E. J. (2015). Transportation: a global supply chain perspective. Cengage Learning.

50. Ullrich, G. (2015). Automated guided vehicle systems. Springer-Verlag Berlin Heidelberg. doi, 10, 978-3. https://doi.org/10.1007/978-3-662-44814-4

51. De Ryck, M., Versteyhe, M., & Debrouwere, F. (2020). Automated guided vehicle systems, state-of-the-art control algorithms and techniques. Journal of Manufacturing Systems, 54, 152-173. https://doi.org/10.1016/j.jmsy.2019.12.002

52. Le-Anh, T., & De Koster, M. B. M. (2006). A review of design and control of automated guided vehicle systems. European Journal of Operational Research, 171(1), 1-23. https://doi.org/10.1016/j.ejor.2005.01.036

53. Arora, S., & Barak, B. (2009). Computational complexity: a modern approach. Cambridge University Press.

54. Hochba, D. S. (Ed.). (1997). Approximation algorithms for NP-hard problems. ACM Sigact News, 28(2), 40-52. https://doi.org/10.1145/261342.571216

55. Lawler, E. L., Lenstra, J. K., Kan, A. H. R., & Shmoys, D. B. (1993). Sequencing and scheduling: Algorithms and complexity. Handbooks in operations research and management science, 4, 445-522. https://doi.org/10.1016/S0927-0507(05)80189-6

56. Pan, C. H. (1997). A study of integer programming formulations for scheduling problems. International Journal of Systems Science, 28(1), 33-41. https://doi.org/10.1080/00207729708929360

57. Bellman, R. (1966). Dynamic programming. Science, 153(3731), 34-37. https://doi.org/10.1126/science.153.3731.34

58. Boyd, S., & Mattingley, J. (2007). Branch and bound methods. Notes for EE364b, Stanford University, 2006, 07.

59. Johnson, S. M. (1954). Optimal two-and three-stage production schedules with setup times included. Naval research logistics quarterly, 1(1), 61-68. https://doi.org/10.1002/nav.3800010110

60. Nawaz, M., Enscore Jr, E. E., & Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. Omega, 11(1), 91-95. https://doi.org/10.1016/0305-0483(83)90088-9

61. Carrera, S. (2010). Planification et ordonnancement des plateformes logistiques (Doctoral dissertation, Institut National Polytechnique de Lorraine-INPL).

62. Talbi, E. G. (2009). Metaheuristics: from design to implementation. John Wiley & Sons.

63. Deb, K., Sindhya, K., & Hakanen, J. (2016). Multi-objective optimization. In Decision sciences (pp. 161-200). CRC Press.

64. Collette, Y., & Siarry, P. (2004). Multiobjective optimization: principles and case studies. Springer Science & Business Media.

65. Voß, S., Martello, S., Osman, I. H., & Roucairol, C. (Eds.). (2012). Meta-heuristics: Advances and trends in local search paradigms for optimization.

66. Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM computing surveys (CSUR), 35(3), 268-308. https://doi.org/10.1145/937503.937505

67. Riquelme, N., Von Lucken, C., & Baran, B. (2015). Performance metrics in multi-objective optimization. 2015 Latin American Computing Conference (CLEI). https://doi:10.1109/clei.2015.7360024

68. Zitzler, E., & Thiele, L. (1998, September). Multiobjective optimization using evolutionary algorithms—a comparative case study. In International conference on parallel problem solving from nature (pp. 292-301). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/BFb0056872

69. Czyzżak, P., & Jaszkiewicz, A. (1998). Pareto simulated annealing—a metaheuristic technique for multiple-objective combinatorial optimization. Journal of multi-criteria decision analysis, 7(1), 34-47. https://doi.org/10.1002/(SICI)1099-1360(199801)7:1<34::AID-MCDA161>3.0.CO;2-6

70. Knowles, J., & Corne, D. (2002, May). On metrics for comparing nondominated sets. In Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600) (Vol. 1, pp. 711-716). IEEE. https://doi.org/10.1109/CEC.2002.1007013

71. Wu, J., & Azarm, S. (2001). Metrics for quality assessment of a multiobjective design optimization solution set. J. Mech. Des., 123(1), 18-25. https://doi.org/10.1115/1.1329875

72. Holland, J. H. (1992). Genetic algorithms. Scientific american, 267(1), 66-73.

73. Kennedy, J., & Eberhart, R. (1995, November). Particle swarm optimization. In Proceedings of ICNN'95-international conference on neural networks (Vol. 4, pp. 1942-1948). IEEE. https://doi.org/10.1109/ICNN.1995.488968

74. Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. IEEE Computational Intelligence Magazine, 1(4), 28–39. https://doi:10.1109/mci.2006.329691

76. Palmer, D. S. (1965). Sequencing jobs through a multi-stage process in the minimum total time—a quick method of obtaining a near optimum. Journal of the Operational Research Society, 16(1), 101-107. https://doi.org/10.1057/jors.1965.8

77. Campbell, H. G., Dudek, R. A., & Smith, M. L. (1970). A heuristic algorithm for the n job, m machine sequencing problem. Management science, 16(10), B-630. https://doi.org/10.1287/mnsc.16.10.B630

78. Ebeling, C. E. (2019). An introduction to reliability and maintainability engineering. Waveland Press.

79. Lee, C. Y. (1997). Minimizing the makespan in the two-machine flowshop scheduling problem with an availability constraint. Operations research letters, 20(3), 129-139. https://doi.org/10.1016/S0167-6377(96)00041-7

80. Schmidt, G. (2000). Scheduling with limited machine availability. European Journal of Operational Research, 121(1), 1-15. https://doi.org/10.1016/S0377-2217(98)00367-1

81. Aggoune, R. (2004). Minimizing the makespan for the flow shop scheduling problem with availability constraints. European Journal of Operational Research, 153(3), 534-543. https://doi.org/10.1016/S0377-2217(03)00261-3

82. Kubzin, M. A., Potts, C. N., & Strusevich, V. A. (2009). Approximation results for flow shop scheduling problems with machine availability constraints. Computers & Operations Research, 36(2), 379-390. https://doi.org/10.1016/j.cor.2007.10.013

83. Hadda, H., Dridi, N., & Hajri-Gabouj, S. (2010). An improved heuristic for two-machine flow shop scheduling with an availability constraint and nonresumable jobs. 4OR, 8, 87-99. https://doi.org/10.1007/s10288-009-0102-3

84. Choi, B. C., Lee, K., Leung, J. Y. T., & Pinedo, M. L. (2010). Flow shops with machine maintenance: Ordered and proportionate cases. European Journal of Operational Research, 207(1), 97-104. https://doi.org/10.1016/j.ejor.2010.04.018

85. Cui, W. W., Lu, Z., Zhou, B., Li, C., & Han, X. (2016). A hybrid genetic algorithm for non-permutation flow shop scheduling problems with unavailability constraints. International Journal of Computer Integrated Manufacturing, 29(9), 944-961. https://doi.org/10.1080/0951192X.2015.1130247

86. Berrichi, A., Amodeo, L., Yalaoui, F., Châtelet, E., & Mezghiche, M. (2009). Bi-objective optimization algorithms for joint production and maintenance scheduling: application to the parallel machine problem. Journal of intelligent manufacturing, 20, 389-400. https://doi.org/10.1007/s10845-008-0113-5

87. Kaabi, J., Varnier, C., & Zerhoni, N. (2002, October). Heuristics for scheduling maintenance and production on a single machine. In IEEE International Conference on Systems, Man and Cybernetics (Vol. 5, pp. 5-pp). IEEE. https://doi.org/10.1109/ICSMC.2002.1176339

88. Kaabi, J., Varnier, C. & Zerhouni, N., (2003). Genetic algorithm for scheduling production and maintenance in a Flow Shop. In Proceedings of the IMS04 Conference. Arles, France, 2003.

89. Benbouzid-Sitayeb, F., Guebli, S. A., Bessadi, Y., Varnier, C., & Zerhouni, N. (2011). Joint scheduling of jobs and preventive maintenance operations in the flowshop sequencing problem: a resolution with sequential and integrated strategies. International Journal of Manufacturing Research, 6(1), 30-48. https://doi.org/10.1504/IJMR.2011.037912

90. Cassady, C. R., & Kutanoglu, E. (2003). Minimizing job tardiness using integrated preventive maintenance planning and production scheduling. IIE transactions, 35(6), 503-513. https://doi.org/10.1080/07408170304416

91. Cassady, C. R., & Kutanoglu, E. (2005). Integrating preventive maintenance planning and production scheduling for a single machine. IEEE Transactions on reliability, 54(2), 304-309. https://doi.org/10.1109/TR.2005.845967

92. Sortrakul, N., Nachtmann, H. L., & Cassady, C. R. (2005). Genetic algorithms for integrated preventive maintenance planning and production scheduling for a single machine. Computers in industry, 56(2), 161-168. https://doi.org/10.1016/j.compind.2004.06.005

93. Wang, S., & Liu, M. (2016). Two-machine flow shop scheduling integrated with preventive maintenance planning. International Journal of Systems Science, 47(3), 672-690. https://doi.org/10.1080/00207721.2014.900137

94. Ruiz, R., García-Díaz, J. C., & Maroto, C. (2007). Considering scheduling and preventive maintenance in the flowshop sequencing problem. Computers & Operations Research, 34(11), 3314-3330. https://doi.org/10.1016/j.cor.2005.12.007

95. Berrichi, A., & Yalaoui, F. (2013). Efficient bi-objective ant colony approach to minimize total tardiness and system unavailability for a parallel machine scheduling problem. The International Journal of Advanced Manufacturing Technology, 68, 2295-2310. https://doi.org/10.1007/s00170-013-4841-0

96. Wang, S., & Liu, M. (2014). Two-stage hybrid flow shop scheduling with preventive maintenance using multi-objective tabu search method. International Journal of Production Research, 52(5), 1495-1508. https://doi.org/10.1080/00207543.2013.847983

97. Khatami, M., & Zegordi, S. H. (2017). Coordinative production and maintenance scheduling problem with flexible maintenance time intervals. Journal of Intelligent Manufacturing, 28, 857-867. https://doi.org/10.1007/s10845-014-1001-9

98. Naderi, B., Zandieh, M., & Fatemi Ghomi, S. M. T. (2009). A study on integrating sequence dependent setup time flexible flow lines and preventive maintenance scheduling. Journal of intelligent manufacturing, 20, 683-694. https://doi.org/10.1007/s10845-008-0157-6

99. Naderi, B., Zandieh, M., & Aminnayeri, M. (2011). Incorporating periodic preventive maintenance into flexible flowshop scheduling problems. Applied Soft Computing, 11(2), 2094-2101. https://doi.org/10.1016/j.asoc.2010.07.008

100. Allaoui, H., Lamouri, S., Artiba, A., & Aghezzaf, E. (2008). Simultaneously scheduling n jobs and the preventive maintenance on the two-machine flow shop to minimize the makespan. International Journal of Production Economics, 112(1), 161-167. https://doi.org/10.1016/j.ijpe.2006.08.017

101. Pan, E., Liao, W., & Xi, L. (2010). Single-machine-based production scheduling model integrated preventive maintenance planning. The International Journal of Advanced Manufacturing Technology, 50, 365-375. https://doi.org/10.1007/s00170-009-2514-9

102. Hadidi, L. A., Al-Turki, U. M., & Rahim, M. A. (2011). An integrated cost model for production scheduling and perfect maintenance. International Journal of Mathematics in Operational Research, 3(4), 395-413. https://doi.org/10.1504/IJMOR.2011.040875

103. Hadidi, L. A., Al-Turki, U. M., & Rahim, A. (2012). Integrated models in production planning and scheduling, maintenance and quality: a review. International Journal of Industrial and Systems Engineering, 10(1), 21-50. https://doi.org/10.1504/IJISE.2012.044042

104. Aramon Bajestani, M., & Beck, J. C. (2015). A two-stage coupled algorithm for an integrated maintenance planning and flowshop scheduling problem with deteriorating machines. Journal of Scheduling, 18, 471-486. https://doi.org/10.1007/s10951-015-0416-2

105. Mokhtari, H., Mozdgir, A., & Kamal Abadi, I. N. (2012). A reliability/availability approach to joint production and maintenance scheduling with multiple preventive maintenance services. International Journal of Production Research, 50(20), 5906-5925. https://doi.org/10.1080/00207543.2011.637092

106. Salmasnia, A., & Mirabadi-Dastjerd, D. (2017). Joint production and preventive maintenance scheduling for a single degraded machine by considering machine failures. Top, 25(3), 544-578. https://doi.org/10.1007/s11750-017-0445-4

107. Daniels, R. L., & Mazzola, J. B. (1993). A tabu-search heuristic for the flexible-resource flow shop scheduling problem. Annals of Operations Research, 41(3), 207-230. https://doi.org/10.1007/BF02023075

108. Daniels, R. L., & Mazzola, J. B. (1994). Flow shop scheduling with resource flexibility. Operations Research, 42(3), 504-522. https://doi.org/10.1287/opre.42.3.504

109. Gupta, J. N., Krüger, K., Lauff, V., Werner, F., & Sotskov, Y. N. (2002). Heuristics for hybrid flow shops with controllable processing times and assignable due dates. Computers & Operations Research, 29(10), 1417-1439. https://doi.org/10.1016/S0305-0548(01)00040-5

110. Mokhtari, H., Abadi, I. N. K., & Cheraghalikhani, A. (2011). A multi-objective flow shop scheduling with resource-dependent processing times: trade-off between makespan and cost of resources. International Journal of Production Research, 49(19), 5851-5875. https://doi.org/10.1080/00207543.2010.523724

111. Behnamian, J., & Ghomi, S. F. (2011). Hybrid flowshop scheduling with machine and resource-dependent processing times. Applied Mathematical Modelling, 35(3), 1107-1123. https://doi.org/10.1016/j.apm.2010.07.057

112. Edis, E. B., & Oguz, C. (2012). Parallel machine scheduling with flexible resources. Computers & Industrial Engineering, 63(2), 433-447. https://doi.org/10.1016/j.cie.2012.03.018

113. Wong, C. S., Chan, F. T. S., & Chung, S. H. (2013). A joint production scheduling approach considering multiple resources and preventive maintenance tasks. International Journal of Production Research, 51(3), 883-896. https://doi.org/10.1080/00207543.2012.677070

114. Nguyen, N. Q., Yalaoui, F., Amodeo, L., Chehade, H., & Toggenburger, P. (2018). Total completion time minimization for machine scheduling problem under time windows constraints with jobs' linear processing rate function. Computers & Operations Research, 90, 110-124. https://doi.org/10.1016/j.cor.2017.09.015

115. Belkaid, F., Yalaoui, F., Bennekrouf, M., & Sari, Z. (2015, December). Métaheuristiques pour l'optimisation des services de maintenance et de production sous des contraintes de ressources consommables. In Xème Conférence Internationale: Conception et Production Intégrées.

116. Panagiotis Kouvelis, Michael W. Kim, (1992) Unidirectional Loop Network Layout Problem in Automated Manufacturing Systems. Operations Research 40(3), 533–550. https://doi.org/10.1287/opre.40.3.533

117. Tanchoco, J. M. A., and Sinriech, D. (1992). OSL—optimal single-loop guide paths for AGVS. International Journal of Production Research, 30(3), 665–681. https://doi.org/10.1080/00207549208942917

118. Qiu, L., Hsu, W.-J., Huang, S.-Y., & Wang, H. (2002). Scheduling and routing algorithms for AGVs: A survey. International Journal of Production Research, 40(3), 745–760. https://doi.org/10.1080/00207540110091712

119. Ulusoy, G., Sivrikaya-Şerifoğlu, F., & Bilge, Ü. (1997). A genetic algorithm approach to the simultaneous scheduling of machines and automated guided vehicles. Computers & Operations Research, 24(4), 335–351. https://doi.org/10.1016/S0305-0548(96)00061-5

120. Ihsan Sabuncuoglu & Suleyman Karabuk (1998) A beam search-based algorithm and evaluation of scheduling approaches for flexible manufacturing systems. IIE Transactions, 30(2), 179–191. https://doi.org/10.1080/07408179808966449

121. Behnamian, J., Fatemi Ghomi, S. M. T., Jolai, F., & Amirtaheri, O. (2012). Minimizing makespan on a three-machine flowshop batch scheduling problem with transportation using genetic algorithm. Applied Soft Computing, 12(2), 768–777. https://doi.org/10.1016/j.asoc.2011.10.015

122. Elmi, A., & Topaloglu, S. (2013). A scheduling problem in blocking hybrid flow shop robotic cells with multiple robots. Computers & Operations Research, 40(10), 2543–2555. https://doi.org/10.1016/j.cor.2013.01.024

123. Zabihzadeh, S. S., & Rezaeian, J. (2016). Two meta-heuristic algorithms for flexible flow shop scheduling problem with robotic transportation and release time. Applied Soft Computing, 40, 319–330. https://doi.org/10.1016/j.asoc.2015.11.008

124. Ahmadi-Javid, A., & Hooshangi-Tabrizi, P. (2015). A mathematical formulation and anarchic society optimisation algorithms for integrated scheduling of processing and transportation operations in a flow-shop environment. International Journal of Production Research, 53(19), 5988–6006. https://doi.org/10.1080/00207543.2015.1035812

125. Amirteimoori, A., Kia, R. (2022). Concurrent scheduling of jobs and AGVs in a flexible job shop system: a parallel hybrid PSO‑GA meta‑heuristic. Flexible Services and Manufacturing Journal. https://doi.org/10.1007/s10696-022-09453-y

126. Nishi, T., Hiranaka, Y., & Grossmann, I.E. (2011). A bilevel decomposition algorithm for simultaneous production scheduling and conflict-free routing for automated guided vehicles. Computers & Operations Research 38(5), 876–888. https://doi.org/10.1016/j.cor.2010.08.012

127. Li, X., Chan, F. T. S., & Chung, S. H. (2015). Optimal multi-degree cyclic scheduling of multiple robots without overlapping in robotic flowshops with parallel machines. Journal of Manufacturing Systems, 36, 62–75. https://doi.org/10.1016/j.jmsy.2015.03.003

128. Saidi-Mehrabad, M., Dehnavi-Arani, S., Evazabadian, F., & Mahmoodian, V. (2015). An Ant Colony Algorithm (ACA) for solving the new integrated model of job shop scheduling and conflict-free routing of AGVs. Computers & Industrial Engineering, 86, 2–13. https://doi.org/10.1016/j.cie.2015.01.003

129. Lyu, X.-F., Song, Y.-C., He, C.-Z., Qi, L., & Guo, W.-F. (2019). Approach to Integrated Scheduling Problems Considering Optimal Number of Automated Guided Vehicles and Conflict-free Routing in Flexible Manufacturing Systems. IEEE Access, 1–1. https://doi.org/10.1109/access.2019.2919109

130. Shouwen, J., Di, L., Zhengrong, C., & Dong, G. (2020): Integrated scheduling in automated container terminals considering AGV conflict-free routing, Transportation Letters, https://doi.org/10.1080/19427867.2020.1733199

131. Murakami, K. (2020). Time-Space Network Model and MILP Formulation of the Conflict-Free Routing Problem of a Capacitated AGV system. Computers & Industrial Engineering, 106270. https://doi.org/10.1016/j.cie.2020.106270

132. Zou, W.-Q., Pan, Q.-K., Meng, T., Gao, L., & Wang, Y.-L. (2020). An effective discrete artificial bee colony algorithm for multi-AGVs dispatching problem in a matrix manufacturing workshop. Expert Systems with Applications, 113675. https://doi.org/10.1016/j.eswa.2020.113675

133. Xin, J., Negenborn, R. R., & Lodewijks, G. (2014). Energy-aware control for automated container terminals using integrated flow shop scheduling and optimal control. Transportation Research Part C: Emerging Technologies, 44, 214–230. https://doi.org/10.1016/j.trc.2014.03.014

134. Lu, C., Gao, L., Li, X., Pan, Q., & Wang, Q. (2017). Energy-efficient permutation flow shop scheduling problem using a hybrid multi-objective backtracking search algorithm. Journal of Cleaner Production, 144, 228–238. https://doi.org/10.1016/j.jclepro.2017.01.011

135. Jiang, E., & Wang, L. (2018). An improved multi-objective evolutionary algorithm based on decomposition for energy-efficient permutation flow shop scheduling problem with sequence-dependent setup time. International Journal of Production Research, 1–16. https://doi.org/10.1080/00207543.2018.1504251

136. Xin, X., Jiang, Q., Li, S., Gong, S., & Chen, K. (2021). Energy-efficient scheduling for a permutation flow shop with variable transportation time using an improved discrete whale swarm optimization. Journal of Cleaner Production, 293, 126121. https://doi.org/10.1016/j.jclepro.2021.126121

137. Dai, M., Tang, D., Giret, A., & Salido, M. A. (2019). Multi-objective optimization for energy-efficient flexible job shop scheduling problem with transportation constraints. Robotics and Computer-Integrated Manufacturing, 59, 143–157. https://doi.org/10.1016/j.rcim.2019.04.006

138. Li, M., & Lei, D. (2021). An imperialist competitive algorithm with feedback for energy-efficient flexible job shop scheduling with transportation and sequence-dependent setup times. Engineering Applications of Artificial Intelligence, 103, 104307. https://doi.org/10.1016/j.engappai.2021.104307

139. Zhang, B., Pan, Q.-K., Gao, L., Meng, L.-L., Li, X.-Y., & Peng, K.-K. (2019). A Three-Stage Multiobjective Approach Based on Decomposition for an Energy-Efficient Hybrid Flow Shop Scheduling Problem. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 1–16. https://doi.org/10.1109/TSMC.2019.2916088

140. Tan, W., Yuan, X., Huang, G., & Liu, Z. (2021). Low-carbon joint scheduling in flexible open-shop environment with constrained automatic guided vehicle by multi-objective particle swarm optimization. Applied Soft Computing, 111, 107695. https://doi.org/doi:10.1016/j.asoc.2021.107695

141. He, L., Chiong, R., Li, W., Budhi, G. S., Zhang, Y. (2022). A multiobjective evolutionary algorithm for achieving energy efficiency in production environments

integrated with multiple automated guided vehicles. Knowledge-Based Systems 243, 108315. https://doi.org/10.1016/j.knosys.2022.108315

142. McHaney, R. (1995). Modelling battery constraints in discrete event automated guided vehicle simulations. International Journal of Production Research, 33(11), 3023–3040. https://doi.org/10.1080/00207549508904859

143. Kabir, Q. S., & Suzuki, Y. (2018). Increasing manufacturing flexibility through battery management of automated guided vehicles. Computers & Industrial Engineering, 117, 225–236. https://doi.org/10.1016/j.cie.2018.01.026

144. Kawakami, T., Takata, S. (2012). Battery Life Cycle Management for Automatic Guided Vehicle Systems. In: Matsumoto, M., Umeda, Y., Masui, K., Fukushige, S. (eds) Design for Innovative Value Towards a Sustainable Society. Springer, Dordrecht. https://doi.org/10.1007/978-94-007-3010-6_77

145. Yongguo Mei, Yung-Hsiang Lu, Hu, Y. C., & Lee, C. S. G. (n.d.). A case study of mobile robot's energy consumption and conservation techniques. ICAR '05. Proceedings., 12th International Conference on Advanced Robotics, 2005. https://doi:10.1109/.2005.1507454

146. Mousavi, M., Yap, H. J., Musa, S. N., Tahriri, F., & Md Dawal, S. Z. (2017). Multi-objective AGV scheduling in an FMS using a hybrid of genetic algorithm and particle swarm optimization. PLOS ONE, 12(3), e0169817. https://doi.org/10.1371/journal.pone.0169817

147. Barak, S., Moghdani, R., & Maghsoudlou, H. (2021). Energy-efficient multi-objective flexible manufacturing scheduling. Journal of Cleaner Production, 283, 124610. https://doi.org/10.1016/j.jclepro.2020.124610

148. Abderrahim, M., Bekrar, A., Trentesaux, D., Aissani, N., & Bouamrane, K. (2020). Manufacturing 4.0 Operations Scheduling with AGV Battery Management Constraints. Energies, 13(18), 4948. https://doi.org/10.3390/en13184948

149. Carlier, J., & Kan, A. R. (1982). Scheduling subject to nonrenewable-resource constraints. Operations Research Letters, 1(2), 52-55. https://doi.org/10.1016/0167-6377(82)90045-1

150. Toker, A., Kondakci, S., & Erkip, N. (1991). Scheduling under a non-renewable resource constraint. Journal of the Operational Research Society, 42, 811-814. https://doi.org/10.1057/jors.1991.152

151. Grigoriev, A., Holthuijsen, M., & Van De Klundert, J. (2005). Basic scheduling problems with raw material constraints. Naval Research Logistics (NRL), 52(6), 527-535. https://doi.org/10.1002/nav.20095

152. Györgyi, P., & Kis, T. (2014). Approximation schemes for single machine scheduling with non-renewable resource constraints. Journal of Scheduling, 17, 135-144. https://doi.org/10.1007/s10951-013-0346-9

153. Györgyi, P., & Kis, T. (2015). Approximability of scheduling problems with resource consuming jobs. Annals of Operations Research, 235(1), 319-336. https://doi.org/10.1007/s10479-015-1993-3

154. Herr, O., & Goel, A. (2016). Minimising total tardiness for a single machine scheduling problem with family setups and resource constraints. European Journal of Operational Research, 248(1), 123-135. https://doi.org/10.1007/s10479-015-1993-3

155. Györgyi, P., & Kis, T. (2018). Minimizing the maximum lateness on a single machine with raw material constraints by branch-and-cut. Computers & Industrial Engineering, 115, 220-225. https://doi.org/10.1016/j.cie.2017.11.016

156. Slowiński, R. (1984). Preemptive scheduling of independent jobs on parallel machines subject to financial constraints. European Journal of Operational Research, 15(3), 366-373. https://doi.org/10.1016/0377-2217(84)90105-X

157. Yeh, W.-C., Chuang, M.-C. & Lee, W.-C., 2014. Uniform parallel machine scheduling with resource consumption constraint. Applied Mathematical Modelling, doi: http://dx.doi.org/10.1016/j.apm.2014.10.012.

158. Györgyi, P., & Kis, T. (2017). Approximation schemes for parallel machine scheduling with non-renewable resources. European Journal of Operational Research, 258(1), 113-123. https://doi.org/10.1016/j.ejor.2016.09.007

159. Pinheiro, J. C., Arroyo, J. E. C., & Fialho, L. B. (2020, July). Scheduling unrelated parallel machines with family setups and resource constraints to minimize total tardiness. In Proceedings of the 2020 genetic and evolutionary computation conference companion (pp. 1409-1417). https://doi.org/10.1145/3377929.3398150

160. von Hoyningen-Huene, W., & Kiesmüller, G. P. (2015). Evaluation of the expected makespan of a set of non-resumable jobs on parallel machines with stochastic failures. European Journal of Operational Research, 240(2), 439-446. https://doi.org/10.1016/j.ejor.2014.07.044

161. Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, 6(2), 182–197. https://doi:10.1109/4235.996017

162. Frutos, M., Olivera, A. C., & Tohmé, F. (2010). A memetic algorithm based on a NSGAII scheme for the flexible job-shop scheduling problem. Annals of Operations Research, 181, 745-765. https://doi.org/10.1007/s10479-010-0751-9

163. Han, Y. Y., Gong, D. W., Sun, X. Y., & Pan, Q. K. (2014). An improved NSGA-II algorithm for multi-objective lot-streaming flow shop scheduling problem.

International Journal of Production Research, 52(8), 2211-2231. https://doi.org/10.1080/00207543.2013.848492

164. Ahmadi, E., Zandieh, M., Farrokh, M., & Emami, S. M. (2016). A multi objective optimization approach for flexible job shop scheduling problem under random machine breakdown by evolutionary algorithms. Computers & operations research, 73, 56-66. https://doi.org/10.1016/j.cor.2016.03.009

165. Basseur, M., Seynhaeve, F., & Talbi, E. G. (2003). Adaptive mechanisms for multi-objective evolutionary algorithms. In IMACS multiconference, Computational Engineering in Systems Applications (CESA'03).

166. Pedersen, G. K. M., & Goldberg, D. E. (2004). Dynamic Uniform Scaling for Multiobjective Genetic Algorithms. Genetic and Evolutionary Computation–GECCO 2004. GECCO 2004. Lecture Notes in Computer Science, 3103. https://doi.org/10.1007/978-3-540-24855-2_2

167. Tasgetiren, M. F., Liang, Y. C., Sevkli, M., & Gencyilmaz, G. (2007). A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. European journal of operational research, 177(3), 1930-1947. https://doi.org/10.1016/j.ejor.2005.12.024

168. Hu, X., & Eberhart, R. (2002, May). Multiobjective optimization using dynamic neighborhood particle swarm optimization. In Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600) (Vol. 2, pp. 1677-1681). Ieee. https://doi.org/10.1109/CEC.2002.1004494

169. Sierra, M. R., & Coello Coello, C. A. (2005, March). Improving PSO-based multi-objective optimization using crowding, mutation and∈-dominance. In International conference on evolutionary multi-criterion optimization (pp. 505-519). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-31880-4_35

170. Di Pierro, F., Khu, S. T., & Savic, D. A. (2007). An investigation on preference order ranking scheme for multiobjective evolutionary optimization. IEEE transactions on evolutionary computation, 11(1), 17-45. https://doi.org/10.1109/TEVC.2006.876362

171. Garza-Fabre, M., Pulido, G. T., & Coello, C. A. C. (2009, November). Ranking methods for many-objective optimization. In Mexican international conference on artificial intelligence (pp. 633-645). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-05258-3_56

172. Wang, Y., & Yang, Y. (2009). Particle swarm optimization with preference order ranking for multi-objective optimization. Information Sciences, 179(12), 1944-1959. https://doi.org/10.1016/j.ins.2009.01.005

173. Zhang, Q., & Li, H. (2007). MOEA/D: A multiobjective evolutionary algorithm based on decomposition. IEEE Transactions on evolutionary computation, 11(6), 712-731. https://doi.org/10.1109/TEVC.2007.892759

174. Gajpal, Y., Rajendran, C., & Ziegler, H. (2005). An ant colony algorithm for scheduling in flowshops with sequence-dependent setup times of jobs. The International Journal of Advanced Manufacturing Technology, 30(5-6), 416–424. https://doi:10.1007/s00170-005-0093-y

175. Yagmahan, B., & Yenisey, M. M. (2008). Ant colony optimization for multi-objective flow shop scheduling problem. Computers & Industrial Engineering, 54(3), 411–420. https://doi:10.1016/j.cie.2007.08.003

176. Zhang, H., Zhang, Q., Ma, L., Zhang, Z., & Liu, Y. (2019). A hybrid ant colony optimization algorithm for a multi-objective vehicle routing problem with flexible time windows. Information Sciences. https://doi:10.1016/j.ins.2019.03.070

177. Mu, C., Zhang, J., Liu, Y., Qu, R., & Huang, T. (2019). Multi-objective ant colony optimization algorithm based on decomposition for community detection in complex networks. Soft Computing. https://doi:10.1007/s00500-019-03820-y

178. Qingfu Zhang, & Hui Li. (2007). MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. IEEE Transactions on Evolutionary Computation, 11(6), 712–731. https://doi:10.1109/tevc.2007.892759

179. Saaty, T. L. (1994). How to Make a Decision: The Analytic Hierarchy Process. Interfaces, 24(6), 19–43. https://doi:10.1287/inte.24.6.19

180. Ruiz, R., Maroto, C., & Alcaraz, J. (2005). Solving the flowshop scheduling problem with sequence dependent setup times using advanced metaheuristics. European Journal of Operational Research, 165(1), 34–54. https://doi:10.1016/j.ejor.2004.01.022

Abstract: This study introduces the Flow shop and Maintenance Scheduling Problem with a Shared Nonrenewable Resource (FMSP-SNR) and the Energy-Efficient Flow shop Scheduling Problem with Blocking and Collision-Free Transportation constraints (EFSP-BCFT). For the FMSP-SNR, a bi-objective model is proposed to minimize makespan and total production costs. The study compares a developed bi-objective randomized local search (BORLS) with NSGA-II and a proposed bi-objective adaptation of PSO. Results show that BORLS is effective in solving large problems and as a solution enhancement strategy. In the EFSP-BCFT case, the study addresses AGV scheduling considering collision-free transportation, speed control, and battery management. An Enhanced Multi-Objective Ant Colony Algorithm (EMOACA) is introduced to minimize makespan and total energy consumption. The algorithm incorporates innovative techniques, including a low-high resolution search strategy and problem-specific heuristics for AGV dispatching. Extensive experiments validate the effectiveness of EMOACA, emphasizing the importance of AGV speed control, battery management, and idle power consumption for improved performance.

Keywords: Flow shop scheduling, Resource constraints, Maintenance, Transportation, Energy.

Résumé: Cette étude présente le problème d’ordononcement des machines en série et de maintenance avec une ressource non renouvelable partagée (FMSP-SNR) et le problème de d’ordononcement économe en énergie avec des contraintes de blockage et transport sans collision (EFSP-BCFT). Pour le FMSP-SNR, un modèle bi-objectif est proposé pour minimiser le makespan et les coûts totaux de production. L'étude compare une recherche locale randomisée bi-objectif développée (BORLS) avec NSGA-II et une adaptation bi-objectif proposée du PSO. Les résultats montrent que BORLS est efficace pour résoudre des problèmes larges et comme stratégie d'amélioration des solutions. Dans le cas EFSP-BCFT, l'étude aborde la planification des AGV en tenant compte du transport sans collision, du contrôle de la vitesse et de la gestion de la batterie. Un algorithme amélioré de colonies de fourmis multi-objectifs (EMOACA) est introduit pour minimiser le makespan et la consommation totale d'énergie. L'algorithme intègre des techniques innovantes, notamment une stratégie de recherche basse-haute résolution et des heuristiques spécifiques aux problème pour la répartition des AGV. Des expériences approfondies valident l'efficacité d'EMOACA, soulignant l'importance du contrôle de la vitesse de l'AGV, de la gestion de la batterie et de la consommation d'énergie en état inactif pour de meilleures performances.

Mots clés: Ordonancement des machines en série, Contraintes de ressources, Maintenance, Transport, Energie.

ملخص: تقدم هذه الدراسة اشكالية جدولة الإنتاج المتسلسل والصيانة مع موارد مشتركة غير المتجددة ومشكلة الجدولة الموفرة للطاقة مع قيود الإعتراض و النقل الخالي من الاصطدام. بالنسبة للاشكالية الاولى ، تم اقتراح نموذج ثنائي الهدف لتقليل مدة التصنيع وتكاليف الإنتاج الإجمالية. تقارن الدراسة بحثًا موضعيًا عشوائيًا ثنائي الهدف تم تطويره مع الخوارزمية الجينية للفرز غير السائد وتكيف ثنائي الهدف مقترح خوارزمية سرب الجسيمات. أظهرت النتائج أن البحث الموضعي العشوائي فعال في حل الاشكاليات الكبيرة وكإستراتيجية لتعزيز الحلول. بالنسبة للاشكالية الثانية ، تتناول الدراسة جدولة المركبات الموجهة آليا مع الأخذ في الاعتبار النقل الخالي من الاصطدام والتحكم في السرعة وإدارة البطارية. تم تقديم خوارزمية مستعمرة النمل المحسنة متعددة الأهداف لتقليل مدة التصنيع و الاستهلاك الإجمالي للطاقة. تشتمل الخوارزمية على تقنيات مبتكرة، بما في ذلك استراتيجية بحث منخفضة و مرتفعة الدقة واستدلالات خاصة بالاشكالية لإرسال المركبات. تؤكد التجارب الموسعة فعالية الخوارزمية، مع التركيز على أهمية التحكم في سرعة المركبات، وإدارة البطارية، واستهلاك الطاقة في وضع السكون لتحسين الأداء.

الكلمات المفتاحية: جدولة الإنتاج المتسلسل، قيود الموارد، الصيانة، النقل، الطاقة.