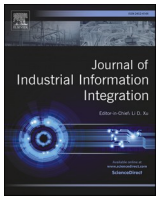
Journal of Industrial Information Integration 39 (2024) 100598

Contents lists available at ScienceDirect

Journal of Industrial Information Integration

journal homepage: www.sciencedirect.com/journal/journal-of-industrial-information-integration



A distributed permutation flow-shop considering sustainability criteria and real-time scheduling

Amir M. Fathollahi-Fard, Lyne Woodward *, Ouassima Akhrif

Department of Electrical Engineering, École de Technologie Supérieure, University of Québec, 1100, Notre-Dame St. W., Montréal, Canada

ARTICLE INFO

Keywords:
Sustainable production
Distributed permutation flow-shop scheduling problem
Lagrangian relaxation
Benders decomposition
Heuristics
Metaheuristics

ABSTRACT

Recent advancements in production scheduling have arisen in response to the need for adaptation in dynamic environments. This paper addresses the challenge of real-time scheduling within the context of sustainable production. We redefine the sustainable distributed permutation flow-shop scheduling problem using an online mixed-integer programming model. The proposed model prioritizes minimizing makespan while simultaneously constraining energy consumption, reducing the number of lost working days and increasing job opportunities within permissible limits. Our approach considers machines operating in different modes, ranging from manual to automatic, and employs two real-time scheduling strategies: predictive-reactive and proactive-reactive scheduling. We evaluate two rescheduling policies: continuous and event-driven. To demonstrate the model's applicability, we present a case study in auto workpiece production. We manage model complexity through various reformulations and heuristics, such as Lagrangian relaxation and Benders decomposition for initial optimization as well as four problem-specific heuristics for real-time considerations. For solving large-scale instances, we employ simulated annealing and tabu search metaheuristic algorithms. Our findings underscore the benefits of the predictive-reactive scheduling strategy and the efficiency of the event-driven rescheduling policy. By addressing dynamic scheduling challenges and integrating sustainability criteria, this study contributes valuable insights into real-time scheduling and sustainable production.

1. Introduction

In today's competitive landscape, manufacturing companies are increasingly striving to establish sustainable production systems that consider economic, environmental, and social criteria [1]. These integrated manufacturing systems aim to seamlessly incorporate sustainability dimensions while effectively addressing uncertainties and disruptions in production schedules. Consequently, manufacturing systems are placing a strong emphasis on incorporating task assignments on machines within their production scheduling processes, particularly to address uncertainties [2]. This approach allows them to enhance sustainability practices and ensure optimal resource allocation in their operations.

State-of-the-art technologies offer various machine operation modes, ranging from manual to highly automated processes. Consequently, production managers must carefully select the appropriate mode for each machine, considering economic, environmental, and social criteria [3]. These operation modes entail varying levels of human interaction. For instance, a machine can be manually operated, requiring significant human intervention, or function in automated modes that minimize the need for human involvement. Introducing a new operation mode or hiring new operators will impact the number of lost working days dedicated to training workers on machine operation, with variations depending on the selected mode [1].

The inherent flexibility in operating modes allows production centers to customize their processes to meet specific requirements, thereby optimizing efficiency. Unlike traditional production systems, where machines primarily function in manual mode, an Industry 4.0-based production system leverages technologies that incorporate advanced automatic modes derived from concepts such as the Internet of Things and cyber-physical systems [4]. Recent advancements in industrial informatics and Industry 4.0 have proven immensely valuable in managing uncertainty in production scheduling [5]. Within an Industry 4.0-based production system, uncertainties can be effectively addressed through real-time scheduling. This involves the integration of simulations, optimization, and probabilistic theories with rescheduling strategies and policies [6]. Given the necessity for real-time optimization and the benefits of a sustainable production system, this study aims to

* Corresponding author.
E-mail address: lyne.woodward@etsmtl.ca (L. Woodward).

https://doi.org/10.1016/j.jii.2024.100598

Available online 12 March 2024

develop a comprehensive optimization model for a sustainable distributed permutation flow-shop, where each factory possesses multiple machines to process tasks in case of disruptive events. The primary objective is to minimize the makespan, representing the maximum completion time for all factories.

According to the ISO 14000 standard [7], environmental sustainability in the manufacturing sector is characterized by a decrease in carbon emissions and energy consumption. Regarding energy consumption, a machine's energy usage can vary based on its operational status and the selected mode of operation. In this study, machine energy consumption is evaluated across three statuses. The first status involves the machine actively processing a task. The second status pertains to idle time when the machine is powered but awaiting a task. The third status is ultra-low idle, where the numerical control device shuts down the servo system, resulting in the lowest power consumption level. When in idle status, a machine consumes a relatively high amount of energy compared to its ultra-idle status, with reported non-task-related energy consumption exceeding 40 % of the total [8]. Reducing energy consumption during idle periods or minimizing idle time significantly improves the energy efficiency of the production process.

The ISO 26000 standard provides a framework for assessing the social performance of manufacturing companies, focusing on enhancing the quality of human life [9]. To this end, social sustainability is a key criterion for evaluating the impact of manufacturing practices on society. In this context, the proposed comprehensive optimization model incorporates various factors, such as the number of created job opportunities and lost working days, as crucial performance indicators to assess social sustainability. Incorporating these metrics into the optimization model enables the measurement and enhancement of the social impact of manufacturing operations, thereby contributing to a more sustainable and socially responsible approach within manufacturing systems.

Social sustainability has a significant impact on people's lives, particularly in countries where the industrial sector comprises a significant portion of the gross domestic product (GDP). For example, China has approximately 80 million employees in its manufacturing sector. As depicted in Fig. 1, in 2020, the agricultural, industrial, and service sectors accounted for 23.6 %, 28.7 %, and 47.7 % of the workforce, respectively. The industrial sector, which generated nearly 32.6 % of China's GDP in 2021, was by far the largest contributor, followed by the wholesale and retail sectors (9.7 %) and the financial sector (8.0 %). Employment opportunities in the manufacturing industry are influenced by various factors. One of these factors is the mode of operation selected. For instance, in manual mode, more workers may be required compared to automated modes of production [1]. These factors emphasize the importance of employment opportunities as a social factor within the framework of Industry 4.0, especially in countries such as China, where the industrial sector presents extensive employment prospects.

To assess social sustainability in the manufacturing industry, the number of workdays lost is a crucial factor from both economic and social perspectives. As noted by Fathollahi-Fard et al. [1], operators may face inability or restrictions in working due to various reasons. For instance, in 2020, the COVID-19 pandemic significantly impacted many workers in China's manufacturing industry due to the high risk of contracting the virus.[2] Furthermore, changes in the work environment represent another cause for lost working days among operators. For example, the introduction of a new automatic operating mode involving an advanced programming system may necessitate worker training. Operators need training to handle machines in this new mode, while engineers and electricians must familiarize themselves with new programming languages to ensure machine maintenance. Recognizing the number of lost workdays is crucial for assessing an essential aspect of social sustainability within the manufacturing industry.

Scheduling machines and tasks often poses challenges due to uncertainties like task arrivals, processing times, and machine breakdowns. Fortunately, recent advancements in Industry 4.0 technologies and industrial informatics have facilitated real-time monitoring and control of these uncertainties [5]. This capability enables production systems to swiftly react and rearrange tasks without interrupting operations. Real-time scheduling approaches [10,11] facilitate task reassignment in case of disruptions, such as sudden task arrivals within the planning horizon. Simulation techniques can map the makespan and production schedule based on disruptive events during this horizon [12,13]. Moreover, the arrival of new tasks due to the development of new products or changes in a machine's operating mode creates uncertainty regarding task processing times. Estimating processing times can be achieved using fuzzy logic or stochastic theory. Machine breakdowns constitute another source of uncertainty, with their occurrence estimated through probabilistic theories. To manage these uncertainties, this study employs real-time scheduling. It employs predictive-reactive and proactive-reactive scheduling strategies, alongside continuous and event-driven rescheduling policies, which are then implemented and assessed. By implementing these approaches, manufacturing companies gain improved prediction, control, and monitoring of disruptions, all while considering economic, environmental, and social criteria in their sustainable production systems.

As widely recognized, permutation flow-shop scheduling problems are known for being NP-hard nature [14]. As a result, substantial efforts in literature have been devoted to developing various metaheuristic algorithms to tackle these complex challenges. In this paper, we not only tailor established metaheuristic algorithms, including tabu search and simulated annealing, and four problem-specific heuristics, but also place significant emphasis on reformulation techniques.

Our reason for incorporating reformulation techniques alongside heuristics and metaheuristics is multifaceted. Firstly, these techniques empower us to leverage existing optimization solvers, such as CPLEX, which can often yield more efficient and effective results compared to developing entirely new metaheuristic algorithms from scratch [15]. Employing reformulation techniques allows us to establish lower bounds for resolving large-scale instances, serving as benchmarks to assess the quality of solutions generated by our metaheuristic algorithms. This efficiency conserves valuable time and resources, particularly when addressing complex scheduling problems [16]. Moreover, these techniques provide invaluable insights into the problem underlying structure, helping identify crucial variables. This enhanced understanding informs more judicious decision-making when devising solution approaches [17]. Furthermore, reformulation techniques offer rigorous mathematical assurances regarding solution quality [16]. This aspect is particularly important in applications where solution quality is of paramount importance, especially in cases where heuristic and metaheuristic algorithms cannot guarantee optimality. Finally, while numerous solution methodologies exist for the distributed permutation flow-shop scheduling problem, our integration of reformulation techniques, specifically Benders decomposition and Lagrangian relaxation alongside constructive heuristics, presents a robust and effective approach compared to our metaheuristic algorithms such as simulated annealing and tabu search.

In conclusion, this study presents a comprehensive optimization model to minimize the makespan of a production system while integrating sustainability concepts through constraints on energy consumption, job creation, and lost working days. In addition, the proposed model addresses uncertainty within the framework of real-time scheduling. Given the complexities inherent in modeling this production scheduling system, efficient solution methods become imperative. The main highlights of this paper are summarized hereafter:

[2] http://www.china.org.cn/business/covid-19-economic-impact/node_8018307.html

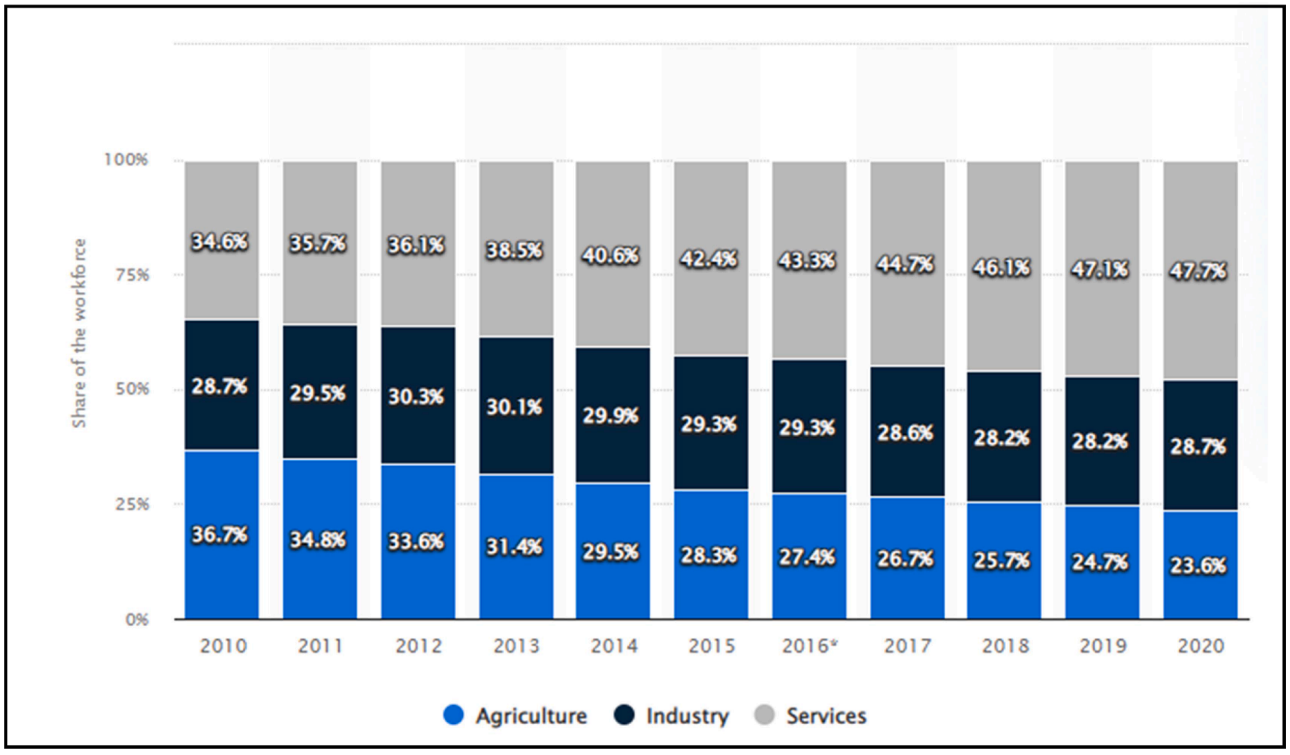


Fig. 1. Total employment in China from 2010 to 2020 for agriculture, industry (manufacturing sector) and services[1].

[1] https://www.statista.com/

- Development of a comprehensive optimization model for the distributed permutation flow-shop, incorporating sustainability criteria and real-time scheduling considerations.
- Introduction of efficient reformulations using Lagrangian relaxation and Benders decomposition to manage the inherent complexity of the optimization problem.
- Creation of problem-specific heuristics and utilization of two state-of-the-art metaheuristics to effectively solve this challenging problem.

The subsequent sections of this article are structured as follows: In Section 2, a comprehensive review of relevant literature in production scheduling is conducted, with a specific focus on uncertainty, sustainability, and distributed permutation flow-shops. Moving to Section 3, the proposed problem is defined, key assumptions are delineated, and the optimization model is introduced. Section 4 delves into the presentation of solution methods, encompassing various approaches such as problem-specific heuristics, customized metaheuristic algorithms, Lagrangian relaxation, and Benders decomposition methods. Transitioning to Section 5, insights derived from computational tests, validation exercises, comparisons, and sensitivity analyses are presented, emphasizing diverse rescheduling policies and strategies. Lastly, in Section 6, the paper concludes by summarizing the main findings and recommendations, while highlighting potential avenues for future research.

2. Literature review

The field of production scheduling has witnessed extensive research over the past century, resulting in numerous significant contributions [18–20]. To elucidate the key contributions relevant to the sustainable distributed flow-shop scheduling problem within the Industry 4.0 framework, the literature review is organized into four distinct subsections. Initially, we examine primary models employed in the context of Industry 4.0, emphasizing the management of uncertainty in production scheduling. Following this, we explore models and algorithms specifically designed for environmentally friendly and sustainable manufacturing systems. Subsequently, our focus shifts to studies concerning distributed permutation flow-shop scheduling. Lastly, we identify research gaps that have motivated the undertaking of this study.

2.1. Production scheduling models under uncertainty

In the context of Industry 4.0, recent advancements in production scheduling are paving the way for smart production systems [21,22]. Drawing from empirical research, Rossit et al. [23] have elucidated the impact of Industry 4.0 concepts on production scheduling. In another insightful survey, Zhang et al. [24] gathered real-time data and assessed a range of job-shop scheduling models developed within the Industry 4.0 framework. Additionally, Dolgui et al. [25] explored the application of optimal control in production scheduling, supply chain, and Industry 4.0-based systems, emphasizing the importance of real-time scheduling in addressing uncertain production scheduling models.

Among the notable contributions in the realm of real-time scheduling, Shen and Yao [26] proposed a multi-objective optimization approach for the flexible job-shop scheduling problem, integrating criteria like energy efficiency and task assignment stability. They employed an evolutionary algorithm to address this challenge. Similarly, Gao et al. [27] utilized a two-stage artificial bee colony algorithm to this problem. The first stage of this algorithm generates an initial task schedule, while the second performs task rescheduling upon the arrival of new tasks. Rahmani and Ramezanian [28] focused on flexible job-shop scheduling, formulating a multi-objective optimization problem with tardiness and scheduling operation stability as objectives, and

used the variable neighborhood search algorithm to find solutions.

Fu et al. [29] developed a flow-shop scheduling model aiming to minimize total makespan and tardiness. Within the context of Industry 4.0, they incorporated the time required to train workers on new technologies and managed uncertainty using stochastic parameters for machine processing times and worker learning curves. They utilized a fireworks algorithm, comparing results with those obtained using a non-dominated sorting genetic algorithm, a multi-objective evolutionary algorithm based on decomposition, and a multi-start simulated annealing algorithm. Additionally, Han et al. [30] proposed a blocking lot-streaming flow-shop scheduling model with stochastic processing time for an Industry 4.0-based system, employing a multi-objective migrating birds' optimization algorithm to solve the model.

Framinan et al. [31] introduced a permutation flow-shop scheduling problem incorporating variable processing times for machines. Their objective was to minimize the makespan, exploring two rescheduling policies, i.e., continuous and periodic rescheduling. In another study focusing on uncertain production scheduling, Gholizadeh et al. [32] addressed a robust optimization problem for the flexible job-shop, taking into account preventive maintenance. They employed a scenario-based genetic algorithm featuring new crossover and mutation operators.

Recent studies have commonly addressed diverse disruptive events, encompassing random task arrivals and machine failures [2,6,33,39]. Shahrabi et al. [39] proposed a job-shop scheduling problem considering random task arrivals and machine failures, employing an event-driven rescheduling policy. They utilized the variable neighborhood search algorithm enhanced by reinforcement learning to solve this problem. Liu et al. [33] devised a heuristic solution based on the tabu search algorithm, incorporating an event-driven rescheduling policy for a mixed-shop scheduling problem considering new task arrivals and machine breakdowns. Al-Behadili et al. [2] defined a multi-objective permutation flow-shop scheduling problem involving multiple disruption events. Their resolution algorithm utilizing a predictive-reactive approach combining randomization and the iterated greedy algorithm. Ghaleb et al. [6] addressed a flexible job-shop scheduling problem with random task arrivals and machine breakdowns, employing continuous and event-driven rescheduling policies. They minimized tardiness using a hybrid genetic algorithm incorporating three problem-specific heuristics as decision rules.

Engin and Yılmaz [34] presented a fuzzy logic-based approach to address the multi-objective hybrid flow-shop scheduling problem involving multi-processor tasks. This approach incorporated fuzzy processing times and due dates. They proposed an efficient genetic algorithm and compared its performance with the simulated annealing algorithm, demonstrating its practicality and effectiveness. In a related study, Engin and İşler [35] introduced a fuzzy hybrid flow-shop scheduling inspired by real apparel production. They developed a parallel greedy algorithm to address this problem, leveraging a fuzzy model to handle uncertainties in setup time, processing time, and due dates using triangular fuzzy numbers.

In a separate study, Zhang et al. [36] proposed a user-friendly real-time data-driven approach for configuring a multi-objective evolutionary algorithm with minimal user intervention, applied to a flow-shop scheduling problem. Lastly, Luo et al. [37] tackled the multi-objective energy-efficient flexible job shop scheduling problem, considering machine breakdowns. They incorporated a rescheduling strategy to mitigate disruptions caused by breakdowns and introduced a knowledge-driven two-stage memetic algorithm to effectively address this challenging problem.

2.2. Green and sustainable manufacturing systems

Research on green and energy-efficient production scheduling has gained significance within the domain of sustainable manufacturing. In a review conducted by Gahm et al. [20], relevant works in sustainable scheduling where classified across three dimensions, i.e., energy supply, energy demand and energetic coverage. Energy supply encompasses the production and availability of energy sources, including electricity, fuel, and other forms of energy. It involves activities such as generation, extraction, refining, processing, and distribution of energy resources to meet demand. Energy demand refers to the amount of energy required by individuals, industries, or societies to support various activities and services. Energetic coverage signifies the extent of energy sources required to satisfy the energy demand of a specific area, region, or country. It indicates the availability and sufficiency of energy resources to meet the population's energy needs. The authors findings highlighted a scarcity of models for energy-efficient job-shop or flow-shop scheduling problems, prompting researchers to develop practical optimization models.

Mansouri et al. [38] introduced the first green flow-shop scheduling problem, considering the interaction between makespan and energy consumption. They defined a new heuristic for solving this problem and compared their results with those obtained using the exact solver from CPLEX software. Mokhtari and Hasani [40] developed a flexible multi-objective job-shop scheduling problem aimed at minimizing total completion time and total energy cost while maximizing overall manufacturing system performance. They utilized an enhanced version of the strength pareto evolutionary algorithm to solve their model. Wu and Sun [41] defined an energy-efficient flexible job-shop scheduling problem including energy-saving criteria, employing a non-dominated sorting genetic algorithm and a heuristic based on the Pareto concept for its solution. Wang et al. [42] formulated an energy-efficient identical parallel machine scheduling problem. Their focus involved machines with identical processing capabilities performing tasks simultaneously, potentially improving energy efficiency and reducing makespan.

Wu and Che [43] addressed the energy-efficient parallel machine scheduling problem, incorporating dynamic speed-scaling techniques. They proposed a memetic differential evolution algorithm with a meta-Lamarckian learning strategy as local search heuristic and compared the results obtained by this approach to those of a non-dominated sorting genetic algorithm and those of a strength Pareto evolutionary algorithm. Dai et al. [44] proposed an energy-efficient flexible job-shop scheduling problem with makespan and energy consumption objectives as well as transportation constraints, employing an enhanced genetic algorithm to generate Pareto solutions. Zhang et al. [45] proposed a hybrid flow-shop scheduling problem with energy efficiency and developed a three-stage decomposition-based multi-objective evolutionary algorithm.

Tirkolaee et al. [46] introduced a variant of flow-shop scheduling that considers the possibility of outsourcing just-in-time delivery to simultaneously minimize total cost and energy consumption. They contributed a fuzzy model to handle uncertainty and developed a self-adaptive artificial fish swarm algorithm to solve their model. They compared their results with those obtained by the epsilon constraint method. Shukla et al. [47] proposed a bi-objective model incorporating type-2 fuzzy sets to address an uncertain energy-efficient parallel machine scheduling problem. They proposed an enhanced multi-objective evolutionary algorithm in comparison with the non-dominated sorting genetic algorithm. Sin et al. [48] proposed a green scheduling model considering electricity cost and preventive maintenance. They developed a hybrid multi-objective genetic algorithm to find a balance between total electricity cost and machine unavailability. Anghinolfi et al. [49] focused on minimizing makespan and total energy consumption in an identical parallel machine scheduling problem, employing a hybrid method combining a constructive heuristic proposed by Wang et al. [42] with local search heuristics using a greedy search. They compared their algorithm with non-dominated sorting genetic and decomposition-based multi-objective evolutionary algorithms.

Hong et al. [50] proposed an energy-efficient flexible flow-shop for a multi-cell manufacturing system with objectives including makespan, energy consumption, and total handling distance. They presented an

enhanced version of a decomposition-based multi-objective evolutionary algorithm and compared it with the original algorithm and other powerful methods from the literature. Marichelvam and Geetha [51] defined an energy-efficient flow-shop scheduling problem under uncertainty, considering stochastic processing times. They developed a hybrid evolutionary algorithm with variable neighborhood search to solve the model.

In addition, Jiang et al. [52] introduced an energy-efficient flexible job-shop problem, solving it with an improved artificial bee colony algorithm and comparing it with the state-of-the-art methods. They demonstrated the applicability of their research to complex components of the aerospace industry in China. Li et al., [53] addressed the energy-efficient flexible job shop scheduling problem with the aim of minimizing both makespan and energy consumption. They introduced a novel approach named popular-based adaptive memetic algorithm.

Collectively, these studies collectively contribute to advancing the understanding and development of energy-efficient and sustainable production scheduling methods, with implications for various industries and sectors. However, few studies have focused on contributing to energy-efficient and sustainable production scheduling methods for the distributed permutation flow-shop configuration.

2.3. Distributed permutation flow-shop problems

Naderi and Ruiz [14], considering the makespan as the optimization performance criterion, defined and modeled the first distributed permutation flow-shop scheduling problem. Unlike more traditional flow-shop scheduling problems that focus on scheduling tasks to be performed in a single factory, this problem encompasses multiple factories in the production scheduling, thereby increasing its complexity. In this regard, they proposed two decision rules to heuristically assign tasks to factories, subsequently enhancing these solutions through variable neighborhood procedures. Furthermore, the same problem was addressed using various approaches, including a genetic algorithm based on local search strategies [54], a modified iterated greedy search algorithm [55], a scatter search heuristic [56] and another metaheuristic algorithm inspired by chemical reactions [57].

Fernandez-Viagas et al., [58] also studied the distributed permutation flow-shop scheduling problem, focusing on minimizing the total flow-time, which represents the sum of completion times across all factories. Pan et al. [59] addressed the same problem using local search heuristics, while Ruiz et al., [60] proposed a simplified version of an iterated greedy heuristic. Meng et al. [61] expanded the problem by incorporating the reception of orders from diverse customers. They developed an evolutionary swarm-based optimization algorithm to solve this modified problem. Allali et al. [62] introduced a multi-objective optimization problem in the context of distributed no-wait permutation flow shop scheduling, incorporating sequence-dependent setup times. Their objective was to minimize both makespan and maximum tardiness criteria simultaneously.They employed nature-inspired metaheuristics, including a genetic algorithm, artificial bee colony algorithm, and migratory bird optimization to solve this complex problem. Han et al. [63] presented an enhanced iterated greedy algorithm aiming to minimize makespan while considering sequence-dependent setup times within a distributed permutation flow-shop system. In a related context, Wang et al. [64] introduced a cooperative iterated greedy algorithm with the objective of minimizing the total tardiness time.

The research on distributed permutation flow-shop scheduling has also delved into handling uncertainty. As an example, Baysal et al. [65] addressed the issue of uncertain processing times for jobs on machines by employing triangular fuzzy numbers to model a distributed permutation flow-shop scheduling problem. They applied the artificial bee colony algorithm to solve this particular challenge. In their subsequent research, outlined by Baysal et al. [66], they extended their analysis and conducted further comparisons. Expanding on this theme, Başar et al. [67] introduced a distributed no-wait flow shop scheduling problem with fuzzy due dates. They tackled this challenge using a parallel kangaroo algorithm.

Recent developments in this field have incorporated the concept of environmental sustainability. Wang and Wang [68] introduced an energy-efficient distributed permutation flow-shop scheduling model targeting the minimization of makespan and energy consumption. Fu et al. [69] approached this problem using a brainstorm optimization algorithm, while Wang et al. [70] applied a multi-objective whale optimization algorithm. Luo et al., [71] delved into a comprehensive framework, employing a multi-objective knowledge-driven evolutionary algorithm based on decomposition. Their focus was on the energy-efficient scheduling of distributed permutation flow shop in heterogeneous factories, aiming to minimize both makespan and total energy consumption. Qin et al. [72] introduced an energy-efficient distributed hybrid flow-shop scheduling problem that incorporates blocking constraints. They presented an enhanced iterative greedy algorithm geared towards optimizing the energy consumption within the job sequence.

Incorporating a processing time penalty as a negative social factor, Lu et al. [73] expanded their energy-efficient distributed permutation flow-shop problem. However, addressing social sustainability, as per ISO 26000, encompasses various indicators such as job opportunities, lost working days, workplace injuries, and local social development. They employed a multi-objective memetic optimization method and compared the results with those of other well-known algorithms. Fathollahi-Fard et al. [1] developed a multi-objective sustainable distributed permutation flow-shop scheduling problem, aiming to minimize the makespan, energy consumption, and lost working days while maximizing job opportunities. They proposed a learning-based social engineering optimizer for their deterministic model. Lastly, Yue et al. [74] addressed energy-efficient scheduling in the printed circuit board manufacturing industry with a bi-objective mathematical model. They proposed a hybrid Pareto spider monkey optimization algorithm and compared its effectiveness with other multi-objective evolutionary algorithms.

2.4. Research gaps and contributions

To clearly identify the gaps in existing research that underlie the proposed approach, Table 1 illustrates the primary contributions found in the literature to date in comparison to those provided by this research. We identified the most relevant works as those considering multiple sustainability criteria or at least addressing uncertainty. This table categorizes these relevant works based on the configuration of the production system, followed by sustainability criteria such as economic, environmental and social factors. Under the economic criterion, we defined makespan, flow-time and tardiness. The category of uncertainty encompasses various disruptive events, including random task arrivals, machine failures, variable processing times, and the utilization of rescheduling policies aimed at managing uncertainties. These rescheduling policies may be continuous, periodic, or event-driven in nature. Additionally, the ability to select multiple operating modes on machines, a concept relevant to Industry 4.0-based systems, is considered. Lastly, the table includes algorithms and heuristics used to solve the problem.

After analyzing the most relevant works in this field and organizing them in Table 1, several conclusions can be drawn:

- Only Lu et al. [73] and Fathollahi-Fard et al. [1] attempted to consider all economic, environmental and social factors simultaneously. However, their models were deterministic and did not consider energy consumption, job opportunities and lost workdays as constraints.
- No paper has considered a production system modeled as a distributed permutation flow-shop scheduling problem dealing with multiple uncertainties.

Table 1
Most relevant studies related to the proposed approach with regards to sustainability and uncertainty.

Reference	Production configuration	Sustainability criteria			Uncertainties							Solution algorithm
		Economic	Environmental	Social	Random task arrival	Machine's breakdown	Variable process time	Continuous rescheduling policy	Periodic rescheduling policy	Event-driven rescheduling policy	Operating mode	
Shen and Yao [26]	FJS	✓	✓	–	✓	–	–	–	–	–	–	MEA
Gao et al., [27]	FJS	✓	–	–	✓	–	–	–	–	–	–	TSABC
Rahmani and Ramezanian [28]	FJS	✓	–	–	✓	–	–	–	–	–	–	VNS
Shahrabi et al., [39]	JS	✓	–	–	✓	✓	–	–	–	✓	–	VNS with RL
Liu et al., [33]	MS	✓	–	–	✓	✓	–	–	–	✓	–	TS
Wang and Wang [68]	DPFS	✓	✓	–	–	–	–	–	–	–	–	KCA
Fu et al., [29]	FS	✓	–	–	–	–	✓	–	–	–	–	MFS
Han et al., [30]	BFS	✓	–	–	–	–	✓	–	–	–	–	MMBO
Fu et al., [69]	DPFS	✓	✓	–	–	–	–	–	–	–	–	MOBSO
Framinan et al., [31]	PFS	✓	–	–	–	–	✓	✓	✓		–	–
Wang et al., [70]	DPFS	✓	✓	–	–	–	–	–	–	–	–	MOWOA
Han et al., [72]	BFS	✓	✓	–	–	–	–	–	–	–	–	DEMO
Lu et al., [73]	DPFS	✓	✓	✓	–	–	–	–	–	–	–	MOMOA
Al-Behadili et al., [2]	PFS	✓	–	–	✓	✓	–	–	–	✓	–	MBRIG
Ghaleb et al., [6]	FJS	✓	–	–	✓	✓	–	✓	–	✓	–	HGA and Heuristics
Fathollahi-Fard et al., [1]	DPFS	✓	✓	✓	–	–	–	–	–	–	✓	LSEO
Gholizadeh et sl., [32]	FJS	✓	–	–	–	–	✓	–	–	–	–	SGA
Engin and Yılmaz [34]	MS	✓	–	–	–	–	✓	–	–	–	–	GA and SA
Engin and İşler [35]	MS	✓	–	–	–	–	✓	–	–	–	–	Parallel greedy heuristic
Luo et al. [37]	FJS	✓	✓	–	–	✓	–	✓	–	–	–	Knowledge-driven MEA
Tirkolaee et al. [46]	MS	✓	✓	–	–	–	✓	–	–	–	–	Self-adaptive AFS
Shukla et al. [47]	JS	✓	✓	–	–	–	✓	–	–	–	–	Enhanced MEA
Marichelvam and Geetha [51]	FS	✓	✓	–	–	–	✓	–	–	–	–	Hybrid MEA and VNS
Jiang et al., [52]	FJS	✓	✓	–	–	–	–	–	–	–	–	TSABC
Yue et al., [74]	FJS	✓	✓	–	–	–	–	–	–	–	–	HPSMO
Luo et al., [71]	DPFS	✓	✓	–	–	–	–	–	–	–	–	Multi-objective knowledge-driven evolutionary algorithm based on decomposition
Qin et al., [72]	DPFS	✓	✓	–	–	–	–	–	–	–	–	Enhanced iterative greedy algorithm
This study	DPFS	✓	✓	✓	✓	✓	✓	✓	–	✓	✓	Lagrangian relaxation, Benders decomposition, Heuristics, SA and TS

*Abbreviations are DPFS: Distributed permutation flow-shop scheduling; KCA: Knowledge-based cooperative algorithm; MOBSO: Multi-objective brain storm optimization; MOMOA: Multi-objective memetic optimization algorithm; DEMO: Discrete evolutionary multi-objective optimization; BFS: Blocking flow-shop scheduling; MOWOA: Multi-objective whale optimization algorithm; LSEO: Learning-based social engineering optimizer; FJS: Flexible job-shop scheduling; MEA: Multi-objective evolutionary algorithms; TSABC: Two-stage artificial bee colony; VNS: Variable neighborhood search; FS: Flow-shop scheduling; MFS: Multi-objective firework algorithm; MMBO: Multi-objective migrating birds' optimization; PFS: Permutation flow-shop scheduling; JS: Job-shop scheduling; RL: Reinforcement learning; MS: Mixed-shop scheduling; TS: Tabu search; HGA: Hybrid genetic algorithm; MBRIG: Multi-objective biased randomized iterated greedy; SGA: Scenario-based genetic algorithm; HPSMO: Hybrid Pareto spider monkey optimization algorithm; AFS: Artificial fish swarm algorithm;.

- None of the papers simultaneously studied all disruptive events, e.g., variable processing time, random task arrivals and machine failures.
- Although one feature of Industry 4.0 is the use of advanced technologies in production systems, except for Fathollahi-Fard et al. [1], no study has considered the possibility of operating mode selection for machines.
- Only a few studies have considered the rescheduling policies [2,6,31,33,39]. However, these studies did not integrate environmental and social criteria, two of the three criteria defining sustainability.

To bridge these research gaps, this study develops a comprehensive optimization model for a sustainable distributed permutation flow shop, considering various dynamic factors crucial for real-time scheduling. Our aim is to contribute to the field by introducing an uncertain model and focusing on online optimization, distinguishing our research from our previous work [1].

In the literature review, Fathollahi-Fard et al. [1] presented a deterministic multi-objective optimization model for sustainable distributed permutation flow-shop scheduling using an adaptive memory-based SEO algorithm, offering valuable insights. Nevertheless, in this paper, we introduce several novel aspects that distinguish our research:

- Unlike our previous deterministic approach, this paper adopts an uncertain model that accounts for various uncertainties in the production process. The integration of real-time scheduling strategies allows adaptability to uncertainties, enhancing responsiveness and flexibility in production systems.
- Alongside the uncertain model, we explore and compare continuous and event-driven rescheduling policies. This investigation aligns with established practices in production scheduling research, offering insights into the effectiveness of these policies in dynamic manufacturing environments.
- To address the complexity of our optimization problem, we develop and assess efficient reformulations using advanced techniques like Lagrangian relaxation and Benders decomposition. These approaches help streamline the computational processes, particularly for managing large-scale instances.
- We introduce problem-specific heuristics tailored to evaluate rescheduling policies, providing practical insights into the performance of proposed scheduling strategies.
- For solving large-scale instances, we employ well-known metaheuristics such as SA and TS. However, in contrast, Fathollahi-Fard et al. [1] used a metaphor-based algorithm utilizing social engineering technique.

In summary, while our prior work by Fathollahi-Fard et al. [1] laid the foundation for sustainable distributed permutation flow-shop scheduling, this paper significantly advances the field by focusing on uncertainty, real-time scheduling, and exploring a wide range of optimization techniques. These contributions collectively contribute to deeper understanding of sustainable scheduling in dynamic manufacturing environments.

3. Proposed problem

The primary objective of the proposed problem is to determine the optimal sequence of N tasks distributed across F factories to be processed on M machines, each capable of operating in P different production modes. The sequence of these tasks performs O operations. In the following subsections, we formalize the mathematical descriptions of sustainability, real-time scheduling, and uncertainties. Then, the decision variables and the objective are defined. Finally, the main notations and formulation of the proposed optimization model are explained.

3.1. Sustainability

The proposed problem includes features related to economic, environmental and social dimensions defining the concept of sustainability. This includes considerations such as energy consumption affecting both economic and environmental criteria, yield loss influencing the economic aspect, and metrics related to job opportunities created and working days lost, linked to the social dimension.

Given that machines predominantly consume non-renewable energy and contribute to carbon emissions, managing energy consumption within the production system is significantly important in this study. To address this, the energy consumed by machines is categorized into three levels corresponding to their operating states: ultra-low idle, idle, or processing states, denoted EC_{mpf}, IEC_{mf} and UEC_{mpf}, respectively. These energy consumption levels must be adhere to predefined upper limits ($UBEC$). Additionally, machines can operate in manual or automatic modes, each associated with an error rate (RW_{mpf}). Typically, an automatic operating mode generates fewer wastes than a manual mode.

The social dimension of sustainability is accounted for by considering the number of job opportunities created and working days lost, aimed at enhancing workers' quality of life and environment [1]. Depending on the production mode, each machine requires varying numbers of workers to process tasks (JO_{mpf}). Generally, manual operation demands more workers compared to automatic mode. From the point of view of social sustainability, a higher number of workers is favorable. Thus, a lower bound for the expected job opportunities created (LBJ) is defined. Moreover, the level of knowledge required by operators varies based on the chosen mode of operation. For example, machines operated in advanced automatic modes driven by programmable logic controllers (PLC) [76] or automatic position control (APC) systems [77] may require a training periods for operators. These training periods measured in days, contribute to the metric of lost working days (LD_{mpf}). Reducing this factor is advantageous from both economic and social perspectives. Hence, an upper bound (UBL) is set to limit the allowable number of lost working days, aligning with the goal of optimizing economic and social criteria.

3.2. Real-time scheduling

Rescheduling refers to the process of updating a production schedule in response to disruptive events, involving three fundamental terms: strategies, policies and methods. A rescheduling strategy entails modifying production plans in reaction to disruptive events, considering the sequence of tasks in the initial schedule. On the other hand, a rescheduling policy delineates the specific criteria dictating when and how rescheduling is performed [31]. Finally, various rescheduling methods are used to update schedules. Table 2 outlines the classification of strategies, policies and methods used in real-time scheduling, as established by Ghaleb et al. [6].

The literature review has identified three distinct rescheduling strategies, namely, completely-reactive, proactive-reactive, and predictive-reactive scheduling strategies. In the completely-reactive scheduling strategy, no pre-schedule is generated beforehand. Thus, real-time local decisions dictate schedule construction [31]. Conversely, both proactive-reactive and predictive-reactive strategies aim to establish an initial deterministic schedule assuming the absence of disruptive events.

Within the proactive-reactive scheduling strategy, the initial task schedule is followed until a moment, t, when disruptive events occur. At this time, all possible scenarios for assigning the remaining tasks are considered to determine the optimal sequence for their assignment. On the other hand, in the predictive-reactive scheduling strategy, the optimal schedule is determined upon the announcement of an upcoming disruptive event, even before its occurrence (before t). This strategy involves rescheduling not only tasks after the disruptive event but also

Table 2
Real-time scheduling concepts.

Strategies	Policies When-to-reschedule	How-to-reschedule Fixed sequencing	Rescheduling	Methods
• Completely-reactive scheduling • Predictive-reactive scheduling • Proactive-reactive scheduling	• Continuous rescheduling • Periodic rescheduling • Event-driven rescheduling	Right shift rescheduling	• Partial rescheduling • Complete rescheduling	• Dispatch rules • Optimization algorithms • Simulation-based scheduling • Machine learning-based scheduling

includes all remaining tasks known from the moment the impending disruptive event is identified.

All these strategies incorporate machine failures effects, utilizing fuzzy, stochastic, or probabilistic theory to estimate their durations. Within the context of production scheduling, predicting machine breakdowns is commonly referred to as preventive maintenance [32]. Prior research [6,31] confirms that both predictive-reactive and proactive-reactive scheduling strategies exhibit superior time efficiency compared to a completely reactive scheduling approach.

In our real-time scheduling approach for the distributed permutation flow-shop system, we conduct an extensive analysis and comparison of predictive-reactive and proactive-reactive scheduling strategies. To effectively illustrate the nuances between these rescheduling strategies within our distributed permutation flow-shop scheduling, we employ a benchmarked numerical example from Fathollahi-Fard et al. [1]. This example involves two factories (F_1 and F_2), each equipped with two machines (M_1 and M_2) capable of operating in two distinct modes (manual and automatic modes). Four tasks (1 to 4, highlighted in blue, orange, green, and yellow respectively in Fig. 2) are to be scheduled. A machine breakdown whose duration (including the subsequent recovery period) is identified in purple in Fig. 2, is considered. Considering the distributed permutation flow-shop problem, tasks are allocated to specific factories, and once assigned, each task must undergo processing on all machines within the designated factory. In this context, O_{nmpf} denotes the operation of task n on machine m using operating mode p in factory f. Before generating the sequence, an operating mode

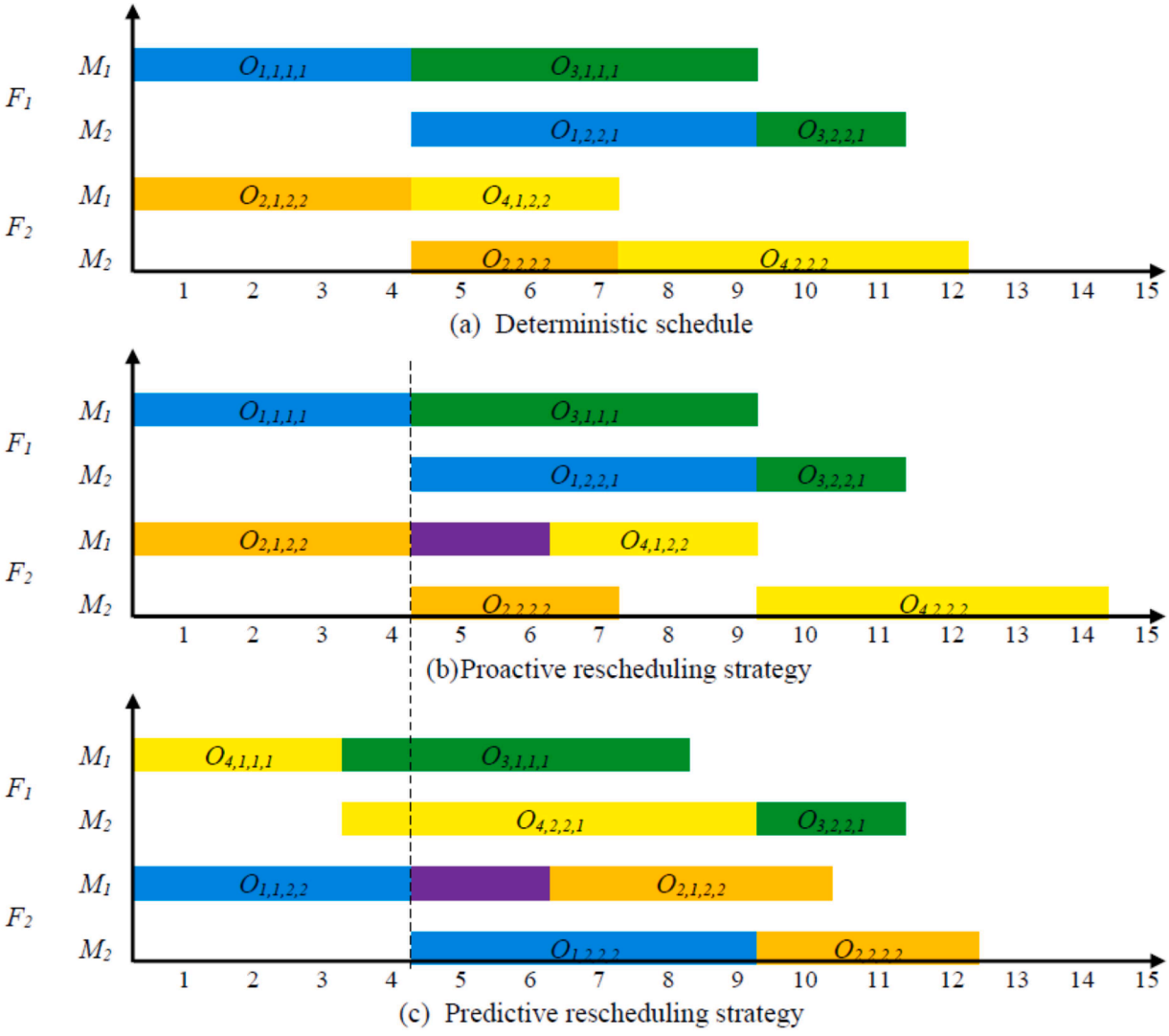


Fig. 2. An example of our distributed permutation flow-shop for (a) optimal deterministic schedule (no disruptive event considered), (b) proactive and (c) predictive rescheduling strategies.

is assigned to each machine. Based on the benchmarked example from Fathollahi-Fard et al. [1], we allocate the manual mode to the first machine (M_1) in the first factory (F_1), while in the same facility, the second machine (M_2) is designated with the automatic mode. In the second factory (F_2), both machines are set to operate in automatic mode.

In the absence of disruptive events, the deterministic schedule initially allocates tasks {1, 3} to the first factory and tasks {2, 4} to the second, resulting in a makespan of 12 units of time (Fig. 2(a)). However, at $t = 4$, a machine failure occurs with machine M_1 in F_2. For $t < 4$, tasks {1, 2} were respectively assigned to machines M_1 in F_1 and F_2. The proactive-reactive strategy therefore aims to optimize the assignment of tasks {3, 4} to the factories considering a recovery time of two units for the broken machine (M1 in F2). The optimal makespan achieved by the proactive-reactive strategy (Fig. 2(b)) amounts to 14 units of time. According to the predictive rescheduling strategy (Fig. 2(c)), adjustments to the deterministic schedule are done before the occurrence of the disruptive event. In this example, we assume prior knowledge of the disruptive event from time $t = 0$. As a result, the whole task sequence is reconfigured, allocating tasks {4, 3} to the first factory and tasks {1, 2} to the second factory, resulting in a makespan of 12 units of time.

As outlined in Table 2, three well-known types of rescheduling policies determine the timing of rescheduling activities, i.e., continuous, periodic and event-driven. A periodic rescheduling policy revises or creates schedules at set intervals, whereas an event-driven policy reacts to specific occurrences, such as rush order arrivals or order cancellations [2]. Existing literature highlights that an event-driven policy typically leads to a shorter makespan compared to a periodic rescheduling policy [31].

Furthermore, continuous rescheduling, a subtype of event-driven rescheduling, involves recalibrating the production schedule each time an uncertain event, such as task arrivals or machine failures, occurs [33]. It encompasses regular updates to the schedule based on real-time information. On the other hand, an event-driven rescheduling policy reacts to specific triggers or events necessitating a schedule update, like changes in task dependencies, resource availability, unexpected events, or disruptions. Our study compares the continuous rescheduling policy with the event-driven rescheduling policy.

Regarding how-to-reschedule policies, various approaches exist for rescheduling tasks in response to disruptions. The right shift rescheduling policy involves delaying each remaining operation to ensure schedule feasibility [26,28]. Partial rescheduling focuses solely on adjusting the affected operations while maintaining the remainder of the schedule unchanged [27]. Finally, full rescheduling entails recomputing the entire schedule using optimization algorithms. Generally, full rescheduling is expected to yield a better solution compared to partial rescheduling, as it optimizes the entire schedule, considering all tasks and their dependencies. Hence, this study uses a comprehensive optimization approach to perform full rescheduling.

Real-time scheduling can be achieved using several methods, including dispatch rules, optimization algorithms, simulation-based techniques, and machine learning-based algorithms. For instance, dispatch rules manage manufacturing systems by assigning tasks to available machines as they become accessible, without generating a production schedule [28]. Dispatch rules typically yield local optimal solutions, whereas optimization algorithms iteratively seek global optimal solutions [1]. However, unlike simulation-based and machine learning-based techniques, heuristics or metaheuristic algorithms, as well as dispatch rules, require defining an optimization problem.

Generally, this paper employs a real-time scheduling approach that integrates two strategic scheduling methods, i.e., predictive-reactive and proactive-reactive scheduling strategies. Additionally, it incorporates two rescheduling policies, i.e., continuous and event-driven rescheduling policies. To effectively address the inherent complexities of real-time scheduling in the proposed distributed permutation flow-shop scheduling problem, a synergistic combination of metaheuristic and heuristic algorithms has been implemented.

3.3. Uncertainties

The model defined in this paper simulates an uncertain production system. To manage the uncertainties' impact on the production system's performance, it's crucial to estimate processing time and predict disruptions, including new task arrivals and machine breakdowns. First, the variable processing time (PT_{nmpf}) of task n on machine m operated in mode p in factory f, can be estimated through pessimistic, realistic and optimistic scenarios, without considering the impact of machine breakdowns. Employing the fuzzy method proposed by Jiménez, et al., [78] and denoting PT^{pes}_{nmpf}, PT^{rea}_{nmpf} and PT^{opt}_{nmpf} as pessimistic, realistic and optimistic estimations respectively, the expected processing time (EPT_{nmpf}) is defined as follows:

$$EPT_{nmpf} = \frac{PT^{pes}_{nmpf}, +2PT^{rea}_{nmpf} + PT^{opt}_{nmpf}}{4} \tag{1}$$

To enhance our estimation, machine failure and repair rates are introduced into the processing time estimate. In this proposed distributed permutation flow-shop system, machines are classified into two states concerning their ability to process a task: machine m operating in mode p is either capable to process task n or requires repair. Referring to the studies of He and Sun, [79] and Mehta and Uzsoy, [80], random machine breakdowns follow an exponential distribution-based probabilistic function [81]. It is assumed in this study that each machine m using operating mode p has fixed failure and repair rates denoted as γ_{mp} and δ_{mp} respectively. Consequently, the mean time to failure and the mean time to repair are $\frac{1}{\gamma_{mp}}$ and $\frac{1}{\delta_{mp}}$ respectively. Hence, the computation of the processing time (PC_{nmpf}) of operation (O_{nmpf}) involves adding the expected processing time (EPT_{nmpf}) and the production delay caused by machine breakdowns and the time required for repairs using the probabilistic theory of Ghaleb et al., [6]:

$$PC_{nmpf} = EPT_{nmpf} + \left\{ \left(TF_{nmpf} + \frac{1}{\delta_{mp}} \right) \times \left(\frac{e^{-\gamma_{mp} EPT_{nmpf}}}{1 - e^{-\gamma_{mp} EPT_{nmpf}}} \right) \right\} \tag{2}$$

where TF_{nmpf} as the failure time occurring within EPT_{nmpf} is estimated as follows:

$$TF_{nmpf} = \frac{\frac{1}{\gamma_{mp}} (1 - e^{-\gamma_{mp} EPT_{nmpf}}) - EPT_{nmpf} e^{-\gamma_{mp} EPT_{nmpf}}}{1 - e^{-\gamma_{mp} EPT_{nmpf}}} \tag{3}$$

In addition to estimating process and failure times, the machine state, denoted as MS_{mpft}, is set to 1 if the machine is engaged in an operation. Otherwise, when the machine is not actively involved in an operation ($MS_{mpft} = 0$), it allows for maintenance or repair activities. The variable RP_{mpft} defines the duration during which a machine undergoes repair, meaning the machine is not engaged in an operation ($MS_{mpft} = 0$). On the other hand, AV_{mpft} represents the time required for the machine to process a task, indicating the machine's engagement in an operation ($MS_{mpft} = 1$). Last but not least, task processing on a machine is feasible only if the machine is capable of processing it, which is represented by the variable H_{nimpft}.

Thus, the proposed online model is designed to handle two distinct uncertainties encountered in production scheduling: machine breakdowns and the arrival of new tasks. Machine breakdowns are addressed by using probabilistic theory to estimate the downtime of machines. On the other hand, the handling of new task arrivals within a specific time horizon, starting at time t, falls under the purview of real-time scheduling. In this study, a deterministic schedule signifies a scenario where no real-time event occurs ($t = 0$). However, the model also covers stochastic scheduling, which accounts for real-time events within the operational planning horizon ($t > 0$). Stochastic scheduling involves scenarios where one or more real-time events occur, such as random arrivals of new tasks. This allows the model to be flexible in adapting to various types of production environments, ensuring optimal scheduling under uncertain conditions.

3.4. Decision variables and objective function

Two main decision variables are defined in the proposed problem:

- The selected mode of operation for each machine (Y_{mpf});
- The assignment of tasks to machines (X_{nimpft}), defining their sequence;

These two binary variables define the optimization problem's search space . Moreover, four auxiliary decision variables are linked to these main decision variables:

- The expected time at which a task commences processing on a machine (S_{impft}), aligned with the task sequence (defined by variables X_{nimpft})
- The expected number of tasks allocated to each factory (A_{ft}), utilizing the task assignment variable (X_{nimpft});
- The completion time of each task (C_{impft}), reliant on both the task assignment variable (X_{nimpft}) and the expected start time (S_{impft});
- The expected time for completing tasks in a factory (CT_{ft}), calculated using task completion times (C_{impft}).

The objective function is to minimize the expected total makespan across all factories ($CMAX_t$), which relies on the expected time for completing tasks within these factories (CT_{ft}). This objective underscores the primary goal of production scheduling, emphasizing the reduction of the overall time required for task completion [14]. However, the model accounts for social and environmental sustainability considerations, such as energy consumption, lost working days, and job opportunities. These factors are integrated into the scheduling model as constraints. While the model prioritizes minimizing makespan, it ensures, that schedules generated also adhere to these sustainability factors, treating them as essential constraints. Incorporating these factors as additional objective functions could overly complicate the optimization problem, making it challenging to solve. Additionally, the model might struggle to generate an optimal solution that satisfies all objectives simultaneously. By treating these sustainability factors as constraints, the optimization problem remains more manageable, increasing the likelihood of generating feasible and practical schedules.

3.5. Notations and problem formulation

Before establishing our optimization problem, all indices, parameters and decision variables are briefly defined hereafter:

Indices:	
f	Index of factories, $f \in \mathscr{F} = \{1, 2, \ldots, F\}$;
m	Index of machines, $m \in \mathscr{M} = \{1, 2, \ldots, M\}$;
n	Index of tasks, $n \in \mathscr{N} = \{1,2,\ldots,N\}$;
p	Index of operating modes, $p \in \mathscr{P} = \{1,2,\ldots,P\}$;
i	Index of task positions in a schedule, $i \in \mathscr{N} = \{1,2,\ldots,N\}$;
t	The time at which a real-time event takes place;
Parameters:	
B	Maximum budget allowed for the installation of machines and operating modes in the production system, including the salary of workers (in $);
CO_{mpf}	Cost of operating machine m in mode p in factory f (in $);
JO_{mpf}	Number of job opportunities created by the use of machine m according to mode of operation p in factory f;
CJ_{mpf}	Salary of workers operating machine m in mode p in factory f (in $) during the planning horizon;
LD_{mpf}	Number of days needed to train the workers to operate machine m in a new operating mode p in factory f;
MW	Maximum waste ratio allowed in all factories;
RW_{mpf}	Waste ratio of machine m when operated in mode p in factory f;
O_{nmpf}	Operation defined as the process of task n by machine m operating in mode p in factory f;
PC_{nmpf}	Expected processing time of operation O_{nmpf} (in *hours*);
UEC_{mpf}	Amount of useful energy consumed by machine m operating in mode p in factory f (in *KWh);*

(continued on next column)

(continued)

EC_{mpf}	Amount of energy consumed by machine m in mode p in factory f during the total time period it is in the ultra-low idle status (in *KWh);*
IEC_{mf}	Amount of energy consumed by machine m in factory f during the total time period it is in the idle status (in *KWh)*
$UBEC$	Upper bound of energy consumption (in *KWh);*
LBJ	Lower bound of the number of job opportunities generated;
UBL	Upper bound of the number of lost working days;
MS_{mpft}	Status of machine m whose operating mode p is selected in factory f at time t; it equals to 1 if the machine is processing a task; otherwise, 0;
AV_{mpft}	Time (in *hours*) where a machine m whose operating mode p is selected in factory f at time t is necessary to process a task; it is a positive value if the machine is busy on an operation ($MS_{mpt} = 1$); otherwise, 0;
RP_{mpft}	Time (in *hours*) needed by machine m whose operating mode p is selected in factory f at time t to recover; it is a positive value if the machine fails at time t ($MS_{mpt} = 0$); otherwise, 0;
H_{nimpft}	It gets 1 if machine m whose operating mode p is selected in factory f at time t is capable to process operation O_{nmpf}; otherwise, 0.
Decision variables:	
Y_{mpf}	If the operating mode p is assigned to machine m in factory f, 1; otherwise, 0;
ST_{impft}	Expected starting time (in *hours*) of task processing at position i on machine m whose operating mode p is selected in factory f at time t;
X_{nimpft}	If the task n is set at position i on machine m whose operating mode p is selected in factory f at time t, 1; otherwise, 0;
A_{ft}	Expected number of tasks assigned to factory f at time t. This is an auxiliary variable depending on X_{nimpft};
C_{impft}	Expected completion time (in *hours*) of a task at position i on machine m whose operating mode p is selected in factory f at time t. This is an auxiliary variable depending on X_{nimpft} and ST_{impft};
CT_{ft}	Expected time (in *hours*) for completing tasks in factory f at time t. This is an auxiliary variable depending on C_{impft};
$CMAX_t$	Expected makespan (in *hours*) for completing tasks in all factories at time t. This is an auxiliary variable depending on CT_{ft} for each factory $f \in \mathscr{F}$.

Using these notations, an online mixed integer linear programming model addressing the sustainability dimensions and uncertainties with real-time scheduling capabilities is now developed.

$$Z = \min\left(CMAX_t = \max\left(CT_{ft}\right)\right) \tag{4}$$

s.t.

$$\sum_{m=1}^{M}\sum_{p=1}^{P}\sum_{f=1}^{F}\left(Y_{mpf} \times JO_{mpf} \times CJ_{mpf}\right) + \sum_{m=1}^{M}\sum_{p=1}^{P}\sum_{f=1}^{F}\left(Y_{mpf} \times CO_{mpf}\right) \le B \tag{5}$$

$$\sum_{m=1}^{M}\sum_{p=1}^{P}\sum_{f=1}^{F}\left(Y_{mpf} \times RW_{mpf}\right) \le MW \tag{6}$$

$$\sum_{i=1}^{N}\sum_{f=1}^{F} X_{nimpft} = 1, \quad \forall n \in \mathscr{N}, m \in \mathscr{M}, p \in \mathscr{P},\ time\ t \tag{7}$$

$$\sum_{n=1}^{N}\sum_{f=1}^{F} X_{nimpft} = 1, \quad \forall i \in \mathscr{N}, m \in \mathscr{M}, p \in \mathscr{P},\ time\ t \tag{8}$$

$$\sum_{n=1}^{N}\sum_{i=1}^{N}\sum_{m=1}^{M}\sum_{p=1}^{P}\left(X_{nimpft}\right) = A_{ft}, \quad \forall f \in \mathscr{F},\ time\ t \tag{9}$$

$$\sum_{n=1}^{N}\sum_{i=1}^{N} X_{nimpft} < N \times Y_{mpf}, \quad \forall m \in \mathscr{M}, p \in \mathscr{P}, f \in \mathscr{F},\ time\ t \tag{10}$$

$$\sum_{p=1}^{P} Y_{mpf} = 1, \quad \forall m \in \mathscr{M}, f \in \mathscr{F} \tag{11}$$

$$X_{nimpft} \le H_{nimpft} \quad \forall i, n \in \mathscr{N},\ m \in \mathscr{M}, p \in \mathscr{P}, f \in \mathscr{F},\ time\ t \tag{12}$$

$$ST_{i,m,pft} \geq \sum_{n=1}^{N} \left(X_{nimpft} \times \left\{MS_{mpft} AV_{mpft} + \left(1 - MS_{mpft}\right) RP_{mpft}\right\}\right),\ \forall i \in \mathscr{N}, m \in \mathscr{M}, p \in \mathscr{P}, f \in \mathscr{F},\ time\ t \tag{13}$$

$$C_{impft} \geq ST_{i,m-1,pft} + \sum_{n=1}^{N} \left(X_{nimpft} \times PC_{nmpf}\right),\ \forall i \in \mathscr{N}, m > 1, p \in \mathscr{P}, f \in \mathscr{F},\ time\ t \tag{14}$$

$$C_{impft} \geq ST_{i-1,mpft} + \sum_{n=1}^{N} \left(X_{nimpft} \times PC_{nmpf}\right),\ \forall i > 1, m \in \mathscr{M}, p \in \mathscr{P}, f \in \mathscr{F},\ time\ t \tag{15}$$

$$CT_{ft} \geq \sum_{i=1}^{I} \sum_{m=1}^{M} \sum_{p=1}^{P} C_{impft}, \quad \forall f \in \mathscr{F},\ time\ t \tag{16}$$

$$\sum_{m\in M} \sum_{p\in P} \sum_{f\in F} \left(Y_{mpf} \times EC_{mpf}\right) + \sum_{m\in M} \sum_{p\in P} \sum_{f\in F} \sum_{n\in N} UEC_{mpf} \times Y_{mpf} \times PC_{nmpf} + \sum_{m\in M} \sum_{f\in F} IEC_{mf} \times \left(\sum_{p\in P} Y_{mpf}\right) \leq UBEC \tag{17}$$

$$\sum_{m=1}^{M} \sum_{p=1}^{P} \sum_{f=1}^{F} \left(Y_{mpf} \times JO_{mpf}\right) \geq LBJ \tag{18}$$

$$\sum_{m=1}^{M} \sum_{p=1}^{P} \sum_{f=1}^{F} \left(Y_{mpf} \times LD_{mpf}\right) \leq UBL \tag{19}$$

$$A_{ft},\ ST_{impft},\ C_{impft}, CT_{ft}, CMAX_t \geq 0 \tag{20}$$

$$Y_{mpf},\ X_{nimpft} \in \{1, 0\} \tag{21}$$

The objective function minimizes the makespan, i.e., the maximum completion time in all factories as given in Eq. (4) by adjusting the decision variables which are bounded by Eqs. (20) and (21). The objective function is subject to the set of constraints (5) to (19), the meaning of which is provided hereafter:

Meaning of constraints:	
Constraint (5)	Costs of implementing an operating mode on a machine in addition to the salary of workers in all factories must not exceed the maximum budget;
Constraint (6)	The total ratio of broken products must be lower than a predefined ratio;
Constraints (7) to (8)	The schedule of tasks must be unique;
Constraints (9)	Calculation of the number of tasks assigned to each factory at time *t*;
Constraints (10)	Decision variables for the selection of operating modes on machines are linked to the decision variables for the assignment of tasks defining the sequence;
Constraints (11)	For each machine, an operating mode must be selected;
Constraints (12)	The assignment of tasks must be performed according to the capability of each machine to process the task;
Constraints (13)	The expected starting time of a task must be set once the time of operation or recovery (after a failure) has been completed on the machine;
Constraints (14) and (15)	The completion time of a task must consider the tasks schedules on machines;
Constraints (16)	The expected time for completing tasks in a factory is computed;
Constraints (17) to (19)	The amount of energy consumed by machines, the number of job opportunities created and the number of working days lost are bounded;

The incorporation of sustainability and uncertainties in the proposed comprehensive optimization model renders it more complex compared to the traditional distributed permutation flow-shop scheduling model. This increased complexity arises from the inclusion of additional constraints, such as constraints (5), (6), (10) and (11) governing the selection of operating modes. Additionally, constraints (12) and (13) address disruptions, such as machine breakdowns and new task arrivals, while constraints (17) to (19) handle sustainability dimensions. This greater complexity motivates us to develop efficient reformulations and heuristics, essential for effectively solving the proposed optimization problem.

4. Proposed solution methods

The most effective approach to handling a complex NP-hard model is through reformulation, aiming to decrease its overall complexity [82]. Here, two different types of reformulations are developed for this purpose, i.e., Lagrangian relaxation and Benders decomposition. Although the proposed reformulation models are efficient for solving small-scale problems, they are still time-consuming when large-scale problems are tackled [15]. More specifically, reformulations can be used to find a deterministic schedule (without real-time event) while they are not able to deal with a stochastic schedule (essential for managing real-time events occurring at $t > 0$). To solve large-scale problems and find a stochastic schedule, we introduce simple and rapid heuristics. These heuristics are subsequently refined using two well-known metaheuristic algorithms. In what follows, we first explain our heuristics (Section 4.1) and metaheuristics (Section 4.2). Then, we present our Lagrangian relaxation (Section 4.3) and Benders decomposition (Section 4.4) reformulations.

4.1. Heuristics

This study proposes four different problem-specific heuristics for both scheduling tasks and rescheduling in the face of uncertainty. While our reformulations can only determine a deterministic schedule ($t = 0$), our heuristics are able of adapting to uncertainty by generating a new schedule whenever an event occurs (at $t > 0$). The first step in our approach involves selecting the operating mode of each machine (Y_{mpf}). Then, we determine the task sequence (through the variable X_{nimpft}) to be processed on the machines both before and after the time ($t > 0$) when the disruptive event occurs. These binary variables build the search space of our optimization problem. Other non-binary decision variables are computed based on their dependencies on these two primary binary variables, derived from constraints (9), (10), (13), (14), (15) and (16).

The proposed approach is exemplified through a scenario involving two production centers denoted by F_1 and F_2. Each center comprises two machines operating in different modes: M_1 and M_2 are located in F_1, while M_3 and M_4 are located in F_2. The proposed approach assigns operating modes and task sequences to each machine in response to a real-time event occurring at time *t*. Fig. 3 depicts a potential solution for this example, presented in two segments. Fig. 3(a) showcases the assignment of operating modes to the machines. In this example, M_1 and M_4 are set to operate in automatic mode, while M_2 and M_3 operate in manual mode. Fig. 3(b) shows the sequence of 10 tasks distributed among machines and factories. The allocation of tasks is correlated with the time when the real-time event occurs. For example, tasks 10, 4 and 6 are assigned to the first factory (F_1) following the occurrence of the disruptive event at time *t*. It is worth noting that the initial segment of the solution definition (Fig. 3(a)) remains consistent across all heuristics proposed in the study, while the latter segment (Fig. 3(b)) is obtained by different decision rules that vary based on the selected algorithm.

The first step in any of the proposed heuristics is to select the operating mode on every machine in each factory. For this purpose, the following steps must be performed:

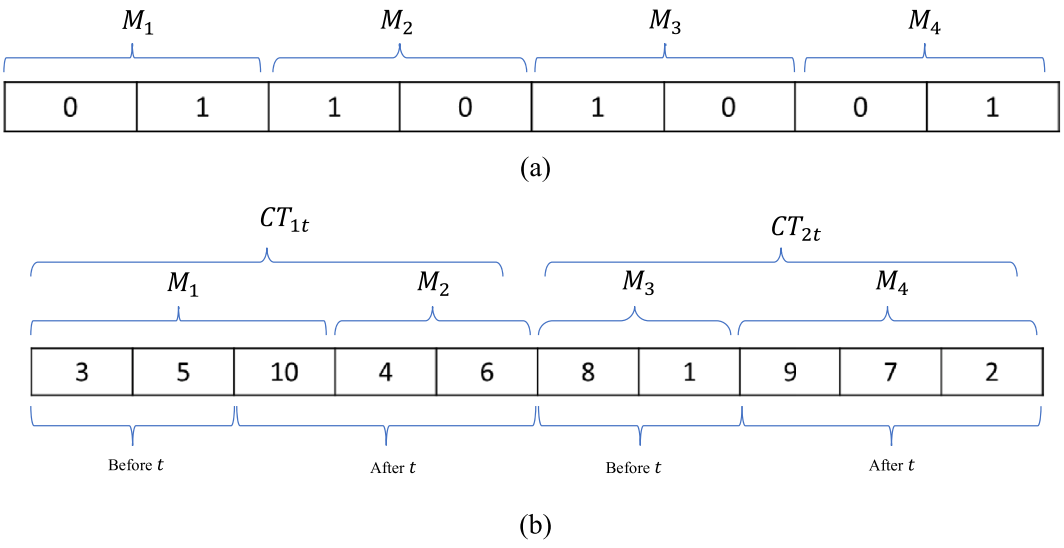


Fig. 3. Solution definition, i.e., (a) assignment of operating modes, (b) sequence of tasks assigned to each machine.

Step 0: For each operating mode, each machine, and each factory, compute the average processing time of jobs ($\sum_{n\in N} PC_{nmpf}/N$).
Step 1: For each machine in every factory, select the operating mode (Y_{mpf}) that leads to the lowest average processing time.
Step 2: Verify the satisfaction of constraints (5), (6), (17), (18), and (19). If any of these constraints are found to be violated, the subsequent solution feasibility process should be implemented. Otherwise, proceed directly to Step 8.
Step 3: If constraint (5) is unsatisfied, identify the machine with the highest implementation cost. Adjust the selected operating mode for this machine and see if the implementation cost decreases. Iterate this process for the machine with the highest implementation cost among those not yet tested until compliance with this constraint is achieved.
Step 4: If constraint (6) is not satisfied, identify the machine with the highest waste rate. Change the chosen operating mode for this machine and evaluate if the waste rate diminishes. If improved, retain this selection. Otherwise, revert to the previously selected operation mode for this machine. Repeat this procedure for the machine with the highest waste rate among those not yet tested until the constraint is satisfied.
Step 5: If constraint (17) is violated, identify the machine with the greatest total energy consumption ($UEC_{mpf} + EC_{mpf} + IEC_{mf}$). Change the operating mode for this machine and see if the total energy consumption decreases. Repeat until the constraint is satisfied.
Step 6: If constraint (18) is not feasible, identify the machine requiring the fewest employed workers (JO_{mpf}). Change the operating mode for this machine and see if the number of employed workers increases. If enhanced, maintain this choice. If not, revert to the previous operating mode for this machine. Repeat this step until compliance with this constraint is achieved.
Step 7: If constraint (19) is violated, identify the machine resulting in the highest number of lost working days. Modify the operating mode for this machine and assess if the number of lost working days decreases. If reduced, maintain this selection. If not, revert to the previous operating mode for this machine. Repeat until the constraint is satisfied.
Step 8: In the absence of any violations in constraints (5), (6), (17), (18) and (19), proceed to use the determined decision variables for the operating mode selection (Y_{mpf}) in the scheduling and rescheduling phases of our heuristics. Otherwise, return to Step 2 to rectify any constraint violations.

The scheduling phase of our heuristics is based on NR1 and NR2 decision rules initially introduced by Naderi and Ruiz [14] for task assignment within a factory. The original definitions of these decision rules, NR1 and NR2, are as follows:

- NR1: Assign task *n* to the factory that had the smallest makespan before including this task.
- NR2: Assign task *n* to the factory that would have the smallest makespan once this task is included.

In both rules above, the task is processed by the first available machine within the chosen factory. However, these decision rules are not directly applicable to our proposed sustainable distributed permutation flow-shop scheduling problem due to the inclusion of different operating modes and the stochastic nature of machine breakdowns and task rescheduling.

In our heuristics, there exist two main phases: before and after time *t*, which marks the occurrence of a disruption. The decision rules NR1 and NR2 are applicable before time *t*. To delineate the task schedule after time *t*, we have created two decision rules, AF1 and AF2, centered around the failure recovery time (RP_{mpft}) impacting the expected start time outlined in constraint set (13) at time *t*. These are outlined as follows:

- AF1: In each factory, assign task *n* to the machine with the shortest failure recovery time and calculate the makespan without including task *n*.
- AF2: In each factory, assign task *n* to the machine with the shortest failure recovery time and calculate the makespan including task *n*.

In summary, by executing Steps 0 to 8 and using decision rules NR1 and NR2 before time *t*, alongside AF1 and AF2 after time *t*, four distinct heuristics- designated as H1, H2, H3 and H4 - are formulated as follows:

H1: Determine (Y_{mpf}) from Steps 0 to 8. Before time *t*, apply NR1 to establish the task assignment sequence (X_{nimpft}) and compute the makespan ($CMAX_t$). After time *t*, apply AF1 to determine the task assignment sequence and compute the makespan.
H2: Determine (Y_{mpf}) from Steps 0 to 8. Before time *t*, apply NR2 to establish the task assignment sequence (X_{nimpft}) and compute the makespan ($CMAX_t$). After time *t*, apply AF1 to determine the task assignment sequence and compute the makespan.

H3: Determine (Y_{mpf}) from Steps 0 to 8. Before time t, apply NR1 to establish the task assignment sequence (X_{nimpft}) and compute the makespan ($CMAX_t$). After time t, apply AF2 to determine the task assignment sequence and compute the makespan.
H4: Determine (Y_{mpf}) from Steps 0 to 8. Before time t, apply NR2 to establish the task assignment sequence (X_{nimpft}) and compute the makespan ($CMAX_t$). After time t, apply AF2 to determine the task assignment sequence and compute the makespan.

4.2. Metaheuristics

Metaheuristic algorithms employ iterative random search procedures to explore new neighboring solutions in the pursuit of finding the global optimal solution. In this study, we have implemented two well-established algorithms, i.e., simulated annealing (SA) and tabu search (TS). These algorithms are renowned for their robustness and have been extensively documented in the literature [75]. It is important to note that the solution representation and exploration of the search space in these metaheuristics align those used in our heuristics. In the following, we provide a description of the main loop of these metaheuristics along with their implementation details in our study.

4.2.1. Simulated annealing

SA, a metaheuristic algorithm inspired by the annealing process in metals, operates based on a neighborhood exploration approach [83]. This algorithm starts with an initial solution, selected in this study as the best solution resulting from our heuristics. During each iteration $it \in \{1, 2, \ldots, MaxIt\}$, a new neighboring solution is generated. If this new solution is superior to the best solution found so far, it is accepted. However, if the new solution is not better, a probabilistic decision rule, influenced by the principle of temperature reduction observed during the annealing process, is applied.

Let Sol_{it} and $f_{(Sol_{it})}$ represent the solution and its corresponding makespan at iteration it, respectively. It is assumed that Sol_{it} is the best solution obtained so far (Sol^*) at iteration it ($Sol^* = Sol_{it}$). New_Sol_{it} denotes a new solution found through the neighborhood operator. If the value of $f_{(New\_Sol_{it})}$ is greater than $f_{(Sol_{it})}$, the SA decision rule is employed, involving the determination of the acceptance probability for this solution, as outlined below:

$$prob = e^{-\Delta/Tem} \text{ where } \Delta = \left| f_{(New\_Sol_{it})} - f_{(Sol^*)} \right| \tag{22}$$

where Tem represents the current temperature and is iteratively updated using the equation:

$$Tem = redu \times Tem \tag{23}$$

The acceptance probability calculation incorporates the temperature damping factor ($redu$). If the calculated acceptance probability is lower than a randomly generated probability, the solution (Sol_{it+1}) is accepted as input for the neighborhood procedure. This procedure encompasses various operators such as swap, reversion, and insertion [84].

Fig. 4 illustrates the application of these operators to the solution depicted in Fig. 3, which serves as the initial solution for the neighborhood procedure. In this instance, the swap operator interchanges the positions of tasks 10 and 1. Conversely, the reversion operator reverses a segment of the task sequence from {10, 4, 6, 8, 1} to {1, 8, 6, 4, 10}. The insertion operator, depicted in Fig. 4, inserts a task at the start of the selected sequence, shifting all subsequent tasks. Notably, the insertion operator demonstrates the repositioning of task 1 at the start of the sequence, before task 10.

At each iteration, one of the neighborhood procedures depicted in Fig. 4 is randomly selected, resulting in the creation of a new neighboring solution. In essence, the SA metaheuristic algorithm follows these key steps:

Step 0: Configure the parameters of the SA algorithm and choose the appropriate dataset.
Step 1: Execute the heuristics and set the best solution obtained from them as the initial solution (Sol_0), updating the record of the best solution (Sol^*) attained.
Step 2: Randomly select one of the neighborhood procedures and generate a new solution.
Step 3: If this new solution surpasses the best solution obtained so far, update the best solution accordingly. Otherwise, employ the decision rule to determine whether to accept or reject the new solution.
Step 4: Adjust the temperature for the decision rule.
Step 5: If the maximum number of iterations is reached, output the best solution obtained so far. If not, proceed to the next iteration by returning to **Step 2**.

4.2.2. Tabu search

TS is a widely-used metaheuristic algorithm inspired by the cognitive processes of the human brain, designed to effectively avoid revisiting previously explored solutions [85]. Similar to other metaheuristic methods, the process begins with an initial solution, specifically derived in this study from the best solution obtained through our heuristics. Throughput each iteration, the algorithm randomly selects one of the neighborhood procedures as explained earlier in Fig. 4, and evaluates the solution by considering the tabu list. The TS maintains a restricted record of solutions within the tabu list. If the new solution already exists in the tabu list, it is marked as tabu, prompting the algorithm to generate an alternative neighboring solution [84]. Moreover, if this solution outperforms the best solution obtained so far, the algorithm updates the record of the best solution accordingly. Upon reaching the maximum number of iterations, the algorithm terminates, and the best solution attained so far is presented as the output. Generally, the main steps of the TS are as follows:

Step 0: Configure the TS parameters and select the dataset.
Step 1: Execute the heuristics and designate the best solution as the initial solution (Sol_0), updating the record of the best solution (Sol^*) obtained so far.
Step 2: Randomly select one of the neighborhood procedures and generate a new solution.
Step 3: Cross-check the new solution with the tabu list. If the solution already exists in the tabu list, label it as tabu and generate an alternative neighboring solution.

Swap

Sol_{it}	3	5	10	4	6	8	1	9	7	2
Sol_{it+1}	3	5	1	4	6	8	10	9	7	2

Reversion

Sol_{it}	3	5	10	4	6	8	1	9	7	2
Sol_{it+1}	3	5	1	8	6	4	10	9	7	2

Insertion

Sol_{it}	3	5	10	4	6	8	1	9	7	2
Sol_{it+1}	3	5	1	10	4	6	8	9	7	2

Fig. 4. Neighborhood procedures of our metaheuristic algorithms.

Table 3
Lagrangian relaxation reformulations .

Reformulation model	Relaxed constraints
LG1	Constraint sets (14) and (15)
LG2	Constraint set (14)
LG3	Constraint set (15)
LG4	Constraints (5) and (6)
LG5	Constraint set (17)
LG6	Constraints (18) and (19)
LG7	Constraints (17), (18) and (19)
LG8	Constraints (5), (6) and (17) to (19)

Step 4: If the new solution outperforms the best solution obtained thus far, update the record of the best solution accordingly.
Step 5: If the maximum number of iterations is reached, output the best solution obtained so far. Otherwise, proceed to the next iteration by returning to **Step 2**.

4.3. Lagrangian relaxation reformulation

The Lagrangian relaxation reformulation aims to relax a set of hard inequality constraints from the original model by adding them into the objective function as soft constraints using Lagrange multipliers [86]. In the case of a minimization problem, the relaxed model provides a lower bound solution which is infeasible for the original problem. Two main criteria are defined to evaluate the quality of the solution obtained from a Lagrangian relaxation reformulation: the CPU time required to identify this lower bound solution and its optimality gap (OG) from the exact solution of the optimization problem [87]. Based on these criteria, the best reformulation in finding a lower bound is selected. Then, an iterative algorithm updates the Lagrangian relaxation multipliers to improve this lower bound [17]. In this regard, this algorithm aims to minimize the deviation between the updated lower bound and a fixed upper bound. The fixed upper bound is established using the best solution derived from NR1 and NR2, which have the capability of identifying feasible solutions that meet all constraints without disruptive events. The iterative process ends upon discovering a feasible lower bound or upon reaching the maximum number of iterations.

The most important step in finding an efficient Lagrangian relaxation reformulation is the selection of the set of constraints to be relaxed. If constraints (14) and (15) are considered to be relaxed with respect to the original model, the Lagrangian relaxation reformulation is as follows:

$$LB = \min\Bigg(CMAX_t + \sum_{i=1}^{N}\sum_{m=2}^{M}\sum_{p=1}^{P} \times \sum_{f=1}^{F}\pi_{impft}\left(ST_{i,m-1,pft} + \sum_{n=1}^{N}\left(X_{nimpft}\times PC_{nmpf}\right) - C_{impft}\right) + \sum_{i=2}^{N}\sum_{m=1}^{M} \times \sum_{p=1}^{P}\sum_{f=1}^{F}\varphi_{impft}\left(ST_{i-1,m,pft} + \sum_{n=1}^{N}\left(X_{nimpft}\times PC_{nmpf}\right) - C_{impft}\right)\Bigg) \quad (24)$$

$$s.t. Constraints(5) to (13) and (16) to (21)$$

where π_{impft} and φ_{impft} are the Lagrange multipliers. We call this reformulation LG1. Other reformulations are defined in Table 3. These reformulations are established by relaxing hard constraints that have a significant impact on the computation time compared to the original model. Four sustainable distributed permutation flow-shop scheduling test problems from Fathollahi-Fard et al. [1] are used to assess and compare the performance of these Lagrangian relaxation reformulations. All reformulations as well as the original problem are solved using the CPLEX[3] solver. It should be noted that the application of the CPLEX solver only provides deterministic schedule. Hence, this comparison is undertaken with the assumption that no real-time events are occurring.

To ensure a fair comparison, the initial values of the Lagrange multipliers were uniformly fixed at one across all reformulations. The CPU time was measured on a laptop equipped with an Intel(R) Core (TM) i7-10850H CPU @ 2.70 GHz, running at a speed of 2.71 GHz.

The results are shown in Table 4, where CPU times are provided in seconds. The OG is expressed as the relative deviation of the lower bound found using the reformulation from the exact solution of the original problem. Thus, a lower value of OG means a better solution. Fig. 5 depicts the performance difference of the 8 reformulations concerning CPU time and OG criteria. Notably, all reformulations demonstrate better solvability compared to the original problem in terms of CPU time. Among these reformulations, LG1 emerges as the fastest, whileLG5 exhibits the slowest. Moreover, in terms of optimality gap, LG1 shows the best performance, while LG8 ranks as the least efficient model. In conclusion, LG1 stands out as the most efficient Lagrangian reformulation for solving our optimization problem.

We propose four problem-specific heuristics aimed at providing an upper bound (UB) for our Lagrangian reformulation. Hence, the Lagrangian reformulation LG1 has four versions, each based on one of these heuristics. The Lagrange multipliers given in Eq. (24) are sequentially updated as follows:

$$\pi_{impft}^{it+1} = \max\left(\pi_{impft}^{it} + f^{it}\times\frac{\left(UB - LB^{it}\right)}{\left(UB - LB^{it+1}\right)^2}\times\left[\left(ST_{i,m-1,pft} + \sum_{n=1}^{N}\left(X_{nimpft}\times PC_{nmpf}\right) - C_{impft}\right)\right], 0\right) \quad (25)$$

$$\varphi_{impft}^{it+1} = \max\left(\varphi_{impft}^{it} + f^{it}\times\frac{\left(UB - LB^{it}\right)}{\left(UB - LB^{it+1}\right)^2}\times\left[\left(ST_{i-1,m,pft} + \sum_{n=1}^{N}\left(X_{nimpft}\times PC_{nmpf}\right) - C_{impft}\right)\right], 0\right) \quad (26)$$

UB is a fixed upper bound, identified as the best solution among those obtained using heuristics H1 to H4. Moreover, LB^{it} is the lower bound solution found at iteration *it,* while f^{it} is a randomly generated scalar number between zero and two [88]. Interested readers who would like to have more details about the approach used for updating the Lagrange multipliers, are referred to Tautenhain, et al., [89].

4.4. Benders decomposition reformulation

This study also proposes a Benders decomposition (BD) reformulation [90] to address the original model using a mixed integer programming approach.BD reformulation is applicable to an optimization model when it implies both continuous and integer variables. However, its application in the field of production scheduling [88] is limited due to the intricate nature of many auxiliary decision variables and constraints that are challenging to separate. The feasibility of BD implementation is enhanced when an optimization model has fewer decision variables and constraints. In a relevant study, Hamzadayı [16] proposed an efficient BD reformulation for the traditional distributed permutation flow-shop

[3] https://www.ibm.com/analytics/cplex-optimizer

Table 4
Comparison of Lagrangian relaxation reformulations in their ability to find a deterministic schedule.

Methods	Tests	T1	T2	T3	T4
	Size of tests ($F \times M \times P \times N$)	$2 \times 2 \times 2 \times 4$	$2 \times 2 \times 2 \times 8$	$2 \times 4 \times 2 \times 20$	$3 \times 4 \times 3 \times 30$
Exact solution	CPU time (s)	10.98	14.75	34.72	65.74
LG1	CPU time (s)	4.65	5.34	12.75	18.39
	OG	0.14	0.23	0.23	0.18
LG2	CPU time (s)	7.95	8.74	23.33	26.11
	OG	0.14	0.29	0.34	0.32
LG3	CPU time (s)	7.03	10.79	21.93	39.84
	OG	0.14	0.32	0.29	0.37
LG4	CPU time(s)	9.33	12.68	26.04	55.22
	OG	0.43	0.56	0.49	0.52
LG5	CPU time (s)	10.16	13.57	30.64	57.19
	OG	0.38	0.42	0.56	0.44
LG6	CPU time (s)	8.89	11.94	28.12	53.24
	OG	0.52	0.75	0.68	0.71
LG7	CPU time (s)	7.56	10.76	25.38	47.96
	OG	0.69	0.65	0.51	0.79
LG8	CPU time (s)	6.54	7.91	19.32	25.74
	OG	0.82	0.86	0.94	0.78

scheduling problem. However, given the complexity of the proposed model in this study, the previously suggested BD reformulation is not applicable. Thus, we introduce a novel BD reformulation well-adapted to our complex optimization model.

The original optimization model needs to be split into two separate models: the master problem (MP) and the primary subproblem (PS). Most notably, the PS subproblem will be solved using a linear programming approach. In this regard, binary variables must be fixed. Consequently, all variables of the PS subproblem are non-negative and continuous. Formulated based on a feasible solution for X_{nimpft}, the PS subproblem is expressed as follows:

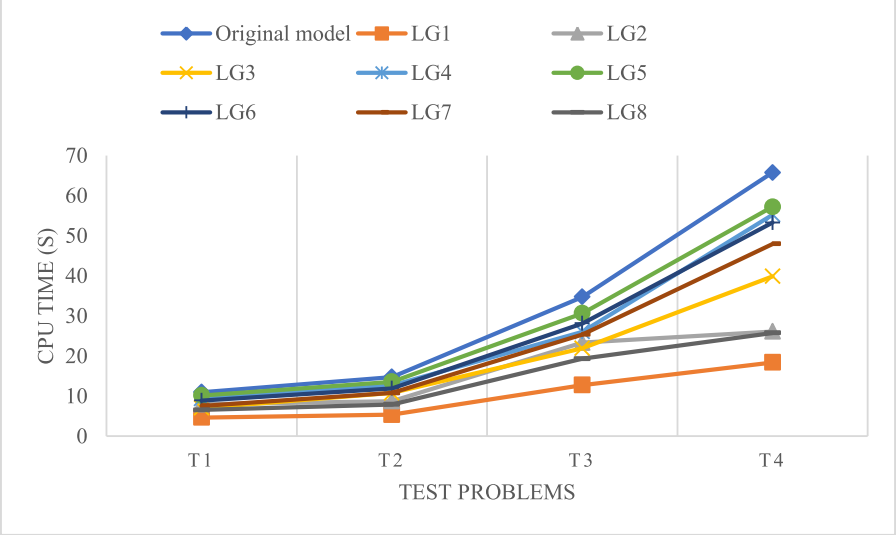
$$PS = \min\left(CMAX_t = \max\left(CT_{ft}\right)\right) \tag{27}$$

s.t.

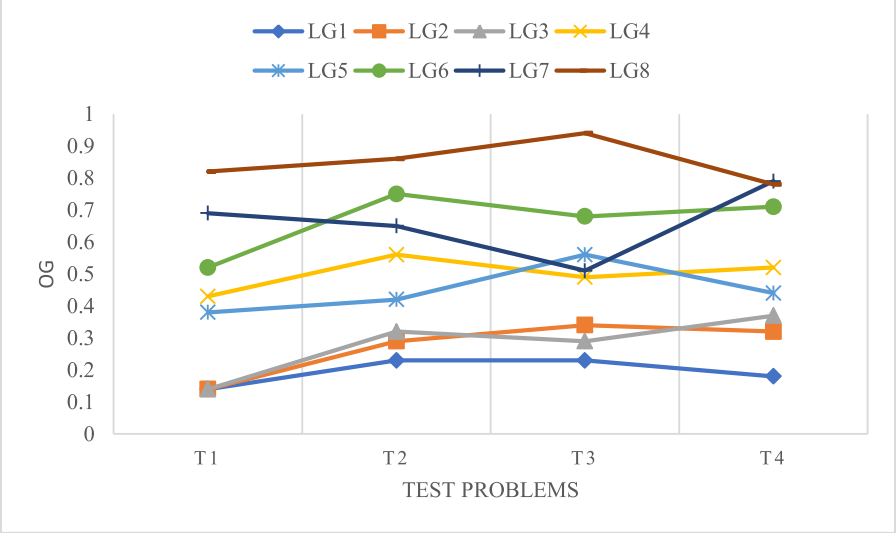
$$Constraints\ (9),\ (13),\ (14),\ (15),\ (16),\ (20)$$

$$A_{ft},\ ST_{impft},\ C_{impft}, CT_{ft}, CMAX_t\ \geq 0 \tag{28}$$

If X_{nimpft} is feasible, the PS subproblem is also feasible and so is its dual. For each constraint set of (9), (13), (14), (15), (16) and (20), a



(a)



(b)

Fig. 5. Comparison of Lagrangian relaxation reformulations based on the criteria of (a) CPU time and (b) OG.

continuous decision variable is defined for the dual of PS (DPS). Based on this reformulation, an upper bound for the DPS formulation is defined as follows:

$$DPS \geq \left(\sum_{i=1}^{N}\sum_{m=1}^{M}\sum_{p=1}^{P}\sum_{f=1}^{F}\beta_{impft}\left(\sum_{n=1}^{N}\left(X_{nimpft}\times\left\{MS_{mpft}AV_{mpft}+\left(1-MS_{mpft}\right)RP_{mpft}\right\}\right)\right)+\sum_{i=1}^{N}\sum_{m=2}^{M}\sum_{p=1}^{P}\sum_{f=1}^{F}\varepsilon_{impft}\left(\sum_{n=1}^{N}\left(X_{nimpft}\times PC_{nmpf}\right)\right)+\sum_{i=2}^{N}\sum_{m=1}^{M}\sum_{p=1}^{P}\times\sum_{f=1}^{F}\sigma_{impft}\left(\sum_{n=1}^{N}\left(X_{nimpft}\times PC_{nmpf}\right)\right)\right) \tag{29}$$

An upper bound for the DPS is a lower bound for the PS subproblem (Eq. (29) is a lower bound for the PS subproblem). Thus, Eq. (29) is inserted into the MP subproblem to provide an optimality cut for the BD reformulation:

$$MP = \min(CMAX_t) \tag{30}$$

s.t.

Constraints (5), (6), (7), (8), (10), (11), (12), (17), (18), (19), (21)

$$CMAX_t \geq \left(\sum_{i=1}^{N}\sum_{m=1}^{M}\sum_{p=1}^{P}\beta_{impft}\left(\sum_{n=1}^{N}\left(X_{nimpft}\times\left\{MS_{mpft}AV_{mpft}+\left(1-MS_{mpft}\right)RP_{mpft}\right\}\right)\right)+\sum_{i=1}^{N}\sum_{m=2}^{M}\sum_{p=1}^{P}\varepsilon_{impft}\left(\sum_{n=1}^{N}\left(X_{nimpft}\times PC_{nmpf}\right)\right)+\sum_{i=2}^{N}\sum_{m=1}^{M}\sum_{p=1}^{P}\sigma_{impft}\left(\sum_{n=1}^{N}\left(X_{nimpft}\times PC_{nmpf}\right)\right)\right),\ \forall f\in\mathcal{F} \tag{31}$$

$$CMAX_t \geq 0 \tag{32}$$

where β_{impft}, ε_{impft} and σ_{impft} are fixed values obtained from the solution of the dual of the PS subproblem. After solving the MP subproblem, the variable X_{nimpft} is sent to the PS subproblem to update the optimality cut. In this respect, iteratively, the MP subproblem calls the PS subproblem and updates the optimality cut and the binary variables. It goes without saying that the MP subproblem provides an efficient BD reformulation for the original problem. In fact, each subproblem of this reformulation has fewer constraints than the LG1 reformulation presented in Section 4.3, which makes the BD reformulation more efficient for solving large-scale test datasets. To the best of our knowledge, our comprehensive optimization model has not been studied before, and the proposed BD reformulation is introduced for the first time.

5. Computational results

In addition to the small-scale size test problems presented in Section 4.2, we also consider in this section, medium and large-scale tests as benchmarked by Fathollahi-Fard et al. [1]. Table 5 presents the size of each test problem along with the initial values of the Lagrange multipliers and the maximum allowable number of iterations employed for both the Lagrangian relaxation and the Benders decomposition reformulations. The table further displays the adjusted values of parameters for our metaheuristic algorithms. Additionally, it should be noted that for the SA algorithm, the initial temperature was set at 10,000, with a damping ratio of 0.99. Due to space constraints in this journal, the calibration analyses of our SA and TS algorithms are not provided in this section. The range of values for each parameter benchmarked from Fathollahi-Fard et al., [1] and Ghaleb et al. [6] is shown in Table 6. It should be noted that the expected processing time is evaluated using Eqs. (1) to (3).

In particular, our analyses are delineated into four distinct sections. Firstly, an evaluation of our heuristics, metaheuristics, and

Table 5
Size of test problems.

Complexity level	Number of test studies	Size of the test Number of factories (F)	Number of machines (M)	Number of operating modes (P)	Number of tasks (N)	Initial value of Lagrange multipliers	Number of iterations for running the reformulations	Time limit of our metaheuristic algorithms (seconds)	Size of the tabu list
Small	T1	2	2	2	4	1	10	50	10
	T2	2	2	2	8	1	10	50	10
	T3	2	4	2	20	1	10	50	10
	T4	3	4	3	30	1	10	50	10
Medium	T5	3	6	2	30	10	30	500	50
	T6	3	6	3	40	10	30	500	50
	T7	4	8	4	30	10	30	500	50
	T8	4	8	5	40	10	30	500	50
Large	T9	6	12	4	80	10	50	1000	100
	T10	6	12	5	100	10	50	1000	100
	T11	8	16	6	80	10	50	1000	100
	T12	10	16	6	100	10	50	1000	100

Table 6
Values of the parameters.

Parameter	Range
PT^{opt}_{nmpf}	$randi([2,4],N, M, P, F)$
PT^{rea}_{nmpf}	$randi([4,6],N, M, P, F)$
PT^{pes}_{nmpf}	$randi([6,8],N, M, P, F)$
CO_{mpf}	$randi([8,20],M, P, F)*10^4$
JO_{mpf}	$randi([2,9],M, P, F)$
CJ_{mpf}	$randi([8,20],M, P, F)$
LD_{mpf}	$randi([8,30],M, P, F)$
LBJ	$round\ (sum(JO_{mpf}/3))$
UBL	$round\ (sum(LD_{mpf} * \left(\frac{2}{3}\right)))$
RW_{mpf}	$rand(M, P, F)*0.1$
IEC_{mf}	$(randi([8,12],M, P, F)+rand())*10^5$
UEC_{mpf}	$(randi([2,7],M, P, F)+rand())*10^5$
EC_{mpf}	$(randi([20,40],M, P, F)+rand())*10^5$
$UBEC$	$round\ (sum((IEC_{mpf} + UEC_{mpf} + EC_{mpf}) * \left(\frac{2}{3}\right)))$
B	$randi([round(sum(JO_{mtf.} * CJ_{mtf} + CO_{mtf})/2), round(sum(JO_{mtf.} * CJ_{mtf} + CO_{mtf}))])$
MS_{mpft}	$round(rand(M, P)*0.8)$
H_{nimpft}	$round(rand(N, I, M, P, F)*0.9)$
RP_{mpft},AV_{mpft}	**if** $MS_{mpt} == 0$ $RP_{mpft} = normrnd\ (sum(H_{nimpft} * PC_{nmpf}),\ 2 * sum(H_{nimpft} * PC_{nmpf}))$**else** $AV_{mpft} = exp(3 * sum(H_{nimpft} * PC_{nmpf}))$ **end**
γ_{mp}	$\frac{1}{7 * sum(H_{nimpft} * EPT_{nmpf})}$
δ_{mp}	$\frac{1}{\frac{3}{2} * sum(H_{nimpft} * EPT_{nmpf})}$
MW	**if** $sum(RW_{mpf})>1$ $randi([round(sum(RW_{mpf})/2), round(sum(RW_{mpf}))])$ **else** $rand()+(sum(RW_{mpf})/2)$**end**

*$randi$, $rand$, $normrnd$, $round$, sum, exp are taken from MATLAB function definitions.

reformulations is conducted in comparison to the precise solution of the original problem obtained using the CPLEX solver. Subsequently, with the identification of the optimal reformulation model, an in-depth sensitivity analysis is carried out to assess the effectiveness of our proposed comprehensive optimization model in a detailed case study. Furthermore, a comprehensive examination of the capabilities of our heuristics and metaheuristics in facilitating real-time scheduling is presented. This analysis is further complemented by exploring various scheduling strategies and rescheduling policies in response to disruptive events. Finally, the study culminates with extensive discussions focusing on managerial insights and key findings derived from our comprehensive results. It should be noted that all codes for our reformulations and heuristics are computed on a laptop with an Intel(R) Core (TM) i7-10850H CPU @ 2.70 GHz 2.71 GHz.

5.1. Comparison of our solution methods

Here, we conduct a comparative analysis of our reformulations to assess their individual performance. Utilizing the CPLEX software, we solved the original model and the two reformulations for the small-scale tests, within a reasonable time. As mentioned earlier, LG1 and BD reformulations can be applied to find a deterministic schedule (there is no real-time events in this case). However, in this comparison, heuristics and metaheuristics are able to generate a stochastic schedule in real-time whenever an event occurs at time t within a user-defined time range. Here, $0 < t \leq 5$, i.e. $t = randi([1,5])*rand$. Each metaheuristic algorithm was executed 10 times and the best, worst and average solutions alongside the standard deviation of the solutions were reported in Table 7 for small-scale instances.

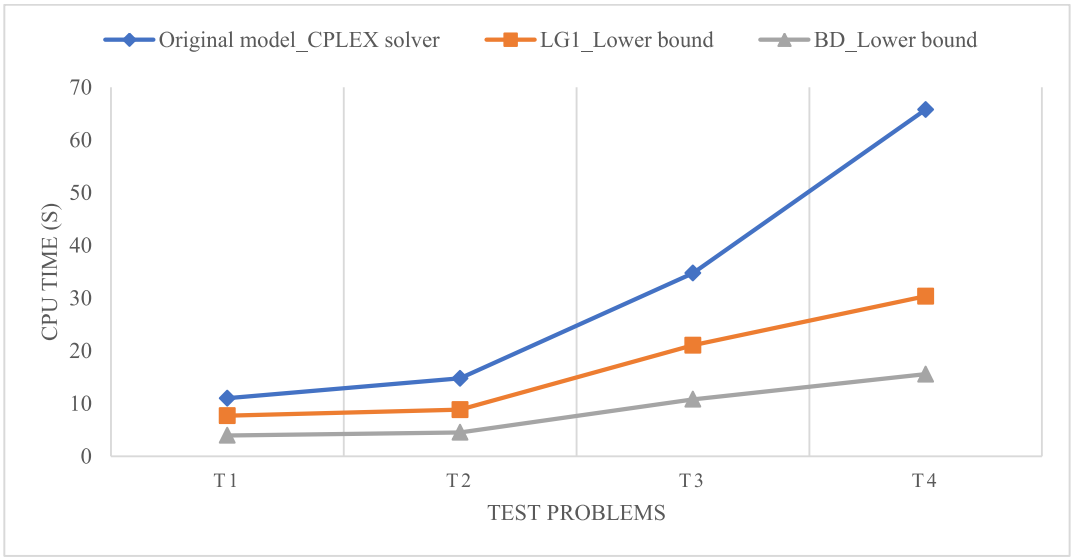
It is obvious that the occurrence of these random events leads to a longer makespan compared to the one obtained by the deterministic schedules generated using our reformulations. However, the objective of comparing these random schedules with our reformulations is to assess the quality of the deterministic schedules. Moreover, the solutions derived from the LG1 and BD reformulations serve as a lower bound, while the solutions obtained from the heuristics act as upper bounds for the proposed problem. As reported in Table 7, the reformulations are compared with each other and with the exact solution of the original problem for small-scale tests. Results of heuristics and metaheuristics are also reported under the occurrence of a disruptive event. It should be noted that, for each test problem, the same disruptive event is simulated to ensure an unbiased comparison as the problem size increases.

Fig. 6 illustrates a comparison of solutions based on CPU time and the identified best makespan. The results indicate that both reformulations are more efficient in solving compared to the original problem. Additionally, the BD reformulation exhibits a shorter solution time than the LG1 reformulation, as depicted in Fig. 6(a) and detailed in Table 7. Notably, the CPU time required to solve the LG1 reformulation is longer than the value indicated in Table 4. In this case, we apply an iterative algorithm using Eqs. (25) and (26), while the results in Table 4 were derived using a fixed upper bound. However, it is important to emphasize that Eqs. (25) and (26) have no impact on the determined lower bound; in other words, the lower bounds of LG1 remain consistent with those presented in Table 4.

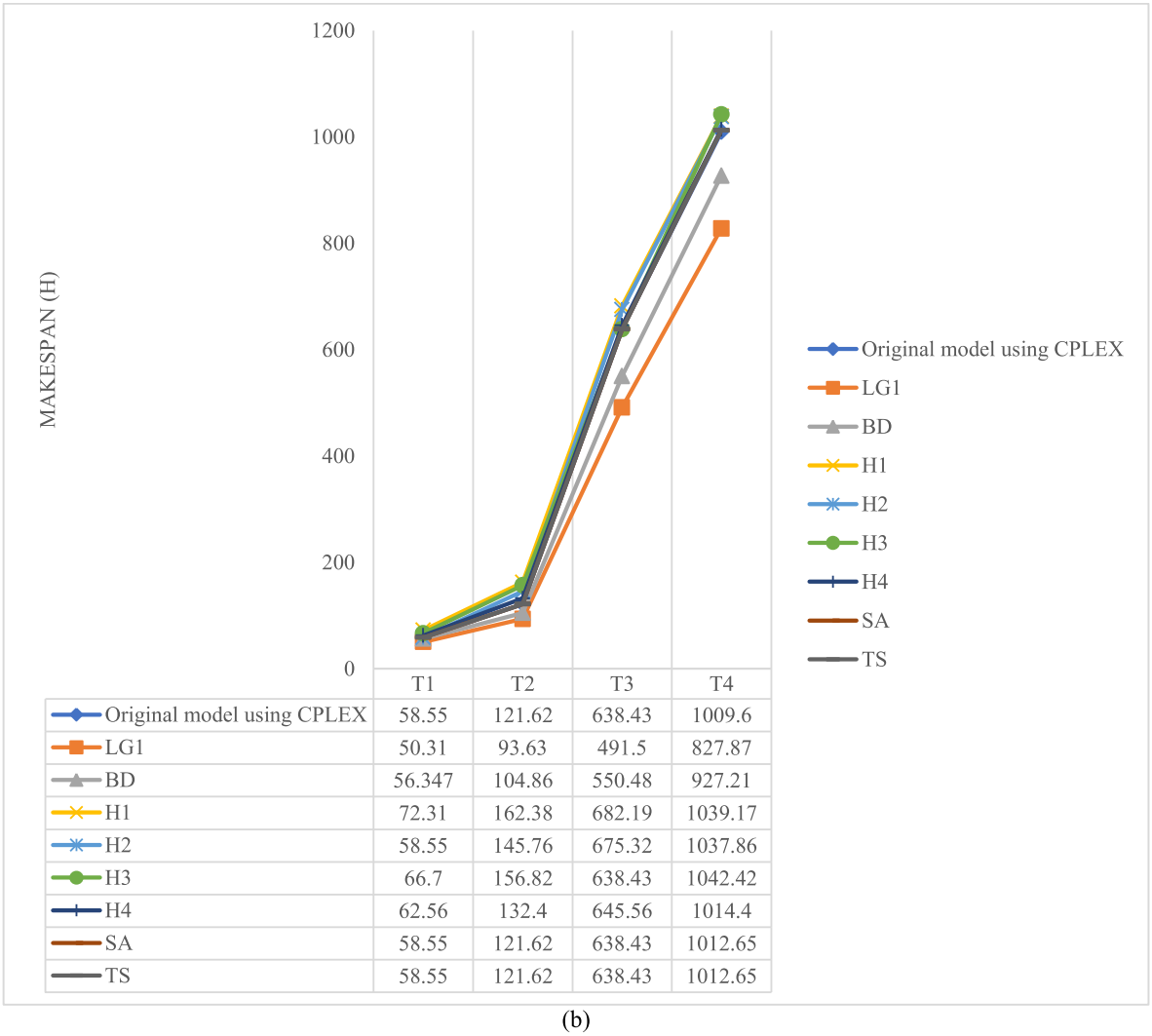
Additionally, the BD reformulation produces a stronger lower bound for our model since its solutions are closer to the exact solution, i.e., the optimality gap is lower as shown in Fig. 6(b). Although our proposed heuristics can provide an upper bound for the original model, there is little differentiation among them in this regard. The metaheuristic algorithms, namely SA and TS, were allotted the same running time to ensure an unbiased comparison. Additionally, it is evident that our metaheuristic algorithms outperform our heuristics, as they effectively improve upon the best solutions identified by the heuristics. Notably, our metaheuristic algorithms successfully identified the optimal solution

Table 7
Comparison of our solution methods in small-scale tests.

Solution methods	Evaluation criteria	Test instances T1	T2	T3	T4
Original model using CPLEX	Optimal makespan (h)	58.55	121.62	638.43	1009.6
	CPU time (s)	10.98	14.75	34.72	65.74
LG1 ($t = 0$)	Lower bound (h)	50.31	93.63	491.5	827.87
	CPU time (s)	7.67	8.81	21.03	30.34
BD ($t = 0$)	Lower bound (h)	56.347	104.86	550.48	927.21
	CPU time (s)	3.93	4.51	10.78	15.55
Heuristics ($0 < t \leq 5$)	Upper bound (h) for H1	72.31	162.38	682.19	1039.17
	Upper bound (h) for H2	58.55	145.76	675.32	1037.86
	Upper bound (h) for H3	66.7	156.82	638.43	1042.42
	Upper bound (h) for H4	62.56	132.4	645.56	1014.4
SA ($0 < t \leq 5$)	Best solution for makespan (h)	58.55	121.62	638.43	1012.65
	Worst solution for makespan (h)	58.55	129.26	638.43	1014.4
	Average solution for makespan (h)	58.55	125.47	638.43	1013.75
	Standard deviation	0	9.55	0	1.946
TS ($0 < t \leq 5$)	Best solution for makespan (h)	58.55	121.62	638.43	1012.65
	Worst solution for makespan (h)	58.55	130.57	638.43	1014.4
	Average solution for makespan (h)	58.55	127.38	638.43	1014.12
	Standard deviation	0	13.6082	0	2.06803



(a)



	T1	T2	T3	T4
Original model using CPLEX	58.55	121.62	638.43	1009.6
LG1	50.31	93.63	491.5	827.87
BD	56.347	104.86	550.48	927.21
H1	72.31	162.38	682.19	1039.17
H2	58.55	145.76	675.32	1037.86
H3	66.7	156.82	638.43	1042.42
H4	62.56	132.4	645.56	1014.4
SA	58.55	121.62	638.43	1012.65
TS	58.55	121.62	638.43	1012.65

(b)

Fig. 6. Comparison of reformulations for small-scale tests, i.e., T1 to T4 based on the CPU time (a) and optimal solutions of the different approaches (b).

Table 8
Comparison of heuristics and metaheuristics with BD reformulation.

Methods		T5	T6	T7	T8	T9	T10	T11	T12
H1	Makespan (h)	196.27	228.54	235.82	319.65	863.11	1093.6	1153.69	1485.95
	CPU time (s)	0.29	0.76	1.93	2.87	10.87	23.76	31.82	41.73
	OG	0.3494	0.1557	0.2861	0.2071	0.1042	0.113	0.1186	0.1357
H2	Makespan (h)	158.35	226.87	198.288	305.75	874.32	1061.65	1136.23	1595.32
	CPU time (s)	0.52	0.98	1.87	3.16	9.64	25.43	36.23	44.62
	OG	0.0886	0.1475	0.0814	0.1546	0.1186	0.0804	0.1016	0.2193
H3	Makespan (h)	167.27	220.55	221.34	285.84	844.28	1142.72	1185.54	1567.54
	CPU time (s)	0.58	1.15	1.52	3.25	12.76	27.97	37.43	48.54
	OG	0.1596	0.1197	0.2034	0.0756	0.0894	0.1633	0.1492	0.1932
H4	Makespan (h)	157.75	213.88	202.81	297.54	856.75	1075.54	1169.5	1412.86
	CPU time (s)	0.67	1.71	2.24	4.89	14.44	28.18	35.89	46.35
	OG	0.084	0.0818	0.1075	0.1264	0.0949	0.0563	0.1394	0.0794
BD	Makespan (h)	145.44	197.75	183.36	264.8	781.6	982.56	1031.36	1308.32
	CPU time (s)	30.99	79.28	103.86	224.85	667.97	6391.69	9060.74	14,169.86
SA	Best solution for makespan (h)	157.3076	212.4125	195.864	284.52	829.1051	1012.867	1082.45	1350.868
	Worst solution for makespan (h)	157.75	212.993	197.6682	284.9721	842.5107	1057.081	1136.23	1397.145
	Average solution for makespan (h)	157.5431	212.6361	196.345	284.4666	837.5043	1032.128	1134.86	1378.429
	Standard deviation	0.3602	0.3064	0.7902	0.313364	5.3142	14.3021	3.8754	15.8024
	OG	0.000328	0	0	0	0.00032	0.000412	0.049537	0.003714
TS	Best solution for makespan (h)	157.3076	212.4125	195.864	284.52	829.1051	1012.867	1132.29	1350.868
	Worst solution for makespan (h)	157.75	212.993	197.6682	284.9721	844.28	1061.65	1136.23	1412.86
	Average solution for makespan (h)	157.6684	212.876	197.2445	284.7461	836.6926	1037.259	1134.26	1381.864
	Standard deviation	0.8564	0.5367	1.0671	0.395588	13.27804	42.68513	3.4475	54.243
	OG	0.000328	0	0	0	0.00032	0.000412	0.09649	0.003714

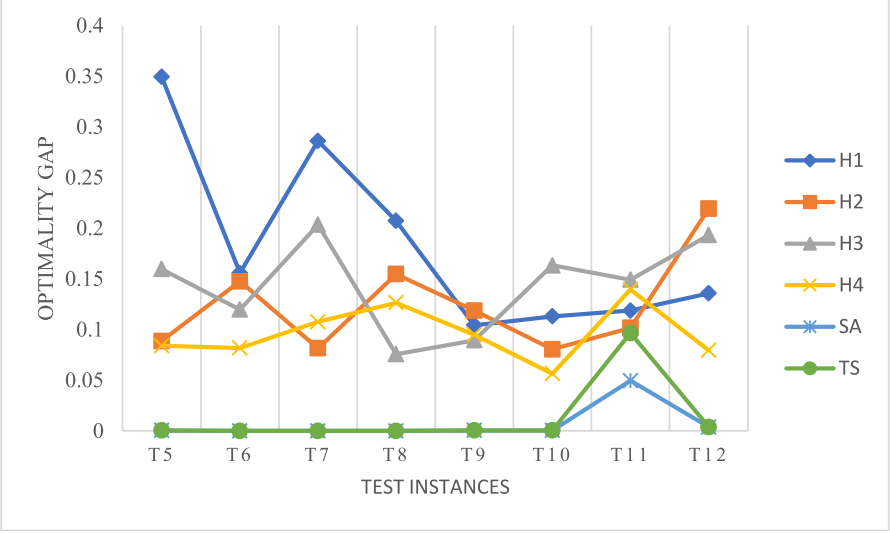


Fig. 7. Optimality gap of the heuristics and metaheuristics compared to the BD reformulation.

in tests instances i.e., T1 and T3, matching the results obtained from the original model using CPLEX software. Among the two, it is observed that SA demonstrates greater robustness, indicated by the lower standard deviations compared to TS.

Overall, our experiments demonstrate that the solution found by the BD reformulation is significantly closer to the exact solution than the one found by the LG1 reformulation. Moreover, BD reformulation solution can be found in a shorter CPU time than the LG1 solution. BD reformulation is the best method in this comparison because its solution is closer to the optimal solution and it is found in a shorter time than when we solve the original problem.

Further analysis for larger instances is presented in Table 8. In large-scale test scenarios, pinpointing exact solutions is often impossible. Hence, the BD reformulation serves as a reference point for evaluating and comparing the results of both our heuristics and metaheuristics. As displayed in Table 8, our heuristics demonstrate quicker processing times in comparison to the proposed BD reformulation. Moreover, there is no noticeable distinction among the CPU times of the four heuristics. Notably, heuristics operate in a single iteration, while metaheuristics follow an iterative approach. To ensure an unbiased comparison between TS and SA, we run these algorithms in the same maximum CPU time as given in Table 5. Furthermore, it is worth noting that while the BD reformulation (i.e., deterministic schedules) is expected to yield optimal solutions, our heuristics including H1, H2, H3, and H4 as well as the metaheuristics from literature including SA and TS are designed to be employed following the occurrence of real-time events. Fig. 7 illustrates the optimality gap calculations for the heuristics and metaheuristics, indicating that both SA and TS exhibit a smaller optimality gap when compared to the heuristics.

The quality of the solutions obtained by our heuristics is measured by the optimality gap compared to the solutions derived from the proposed BD reformulation. To this end, the graph in Fig. 8 presents the means and errors of this quality evaluation, calculated at a 95 % confidence level using the Student's t distribution. The vertical axis of this graph normalized OG values of our heuristics according to the fundamental schedules identified by the BD reformulation. It is important to note that

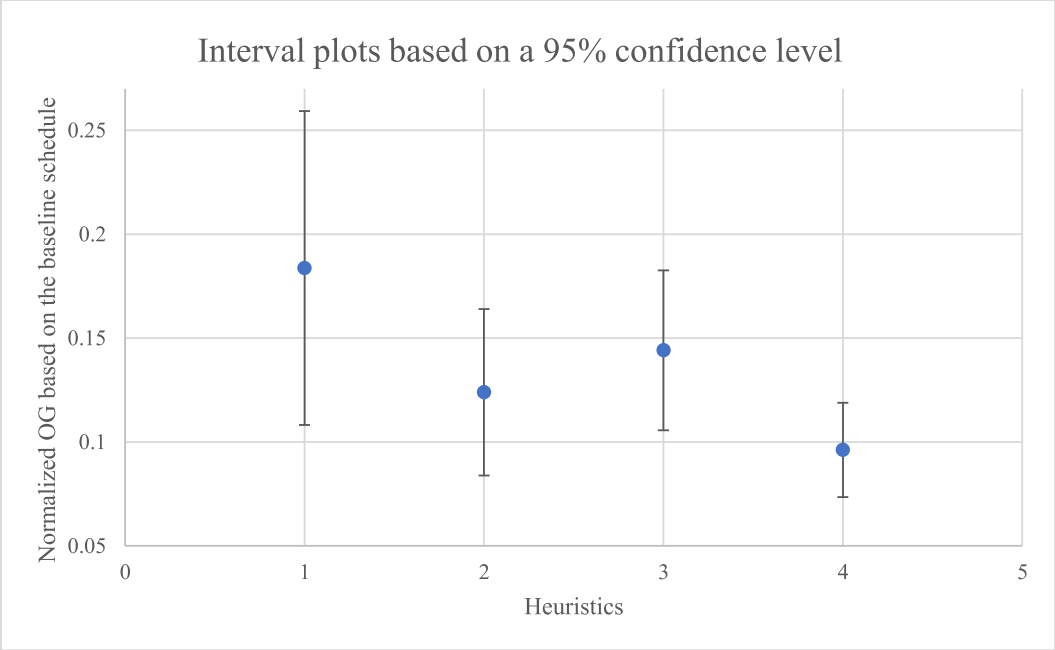


Fig. 8. Means plot with 95 % confidence level for the assessment of heuristics based on the base schedules found by the BD reformulation.

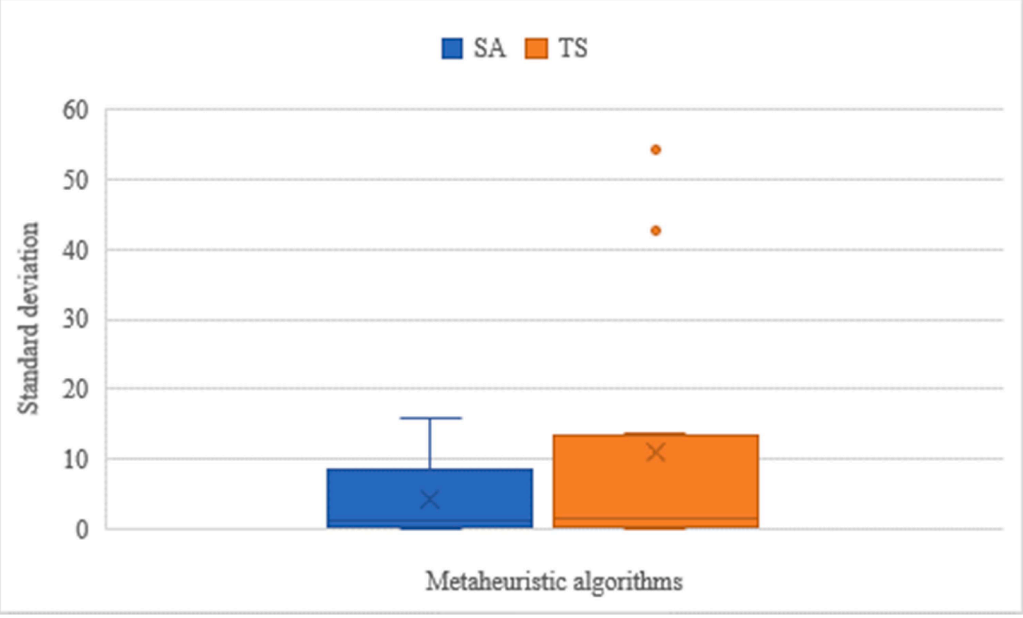


Fig. 9. Box plot with 95 % confidence level for the assessment of metaheuristics based on the standard deviation.

while the OG values in Table 4 are based on the solutions of the original model, those in Table 8 are derived from the solutions of the BD reformulation, representing a lower bound for the original problem. The primary observations from these computations indicate that H1 demonstrates superior speed compared to the other heuristics. However, as depicted in Fig. 8, H4 showcases robustness and efficiency, outperforming the other heuristics. Additionally, in this comparison, H2 performs better than H3, while H1 is revealed to be the least effective among all algorithms.

Furthermore, an analysis of the algorithms' robustness is conducted based on the standard deviation of the solutions from our metaheuristic algorithms. This analysis is presented using a box plot with a 95 % confidence level derived from the Student's t distribution in Fig. 9. Notably, the findings suggest that SA demonstrates greater effectiveness than TS in solving our optimization problems.

5.2. Application of the proposed approach to a case study

Based on the data provided by Wuhan Huazhong Numerical Control company, a case study was defined to verify the effectiveness of the proposed optimization model. The case study is the production of flanges as shown in Fig. 10 used in automobile construction. This production consumes a non-negligible amount of energy resulting from the multiple levels of energy consumed by the machines according to three statuses i.e., ultra-idle, idle or processing.

The production of a flange requires ten tasks of turning, milling,

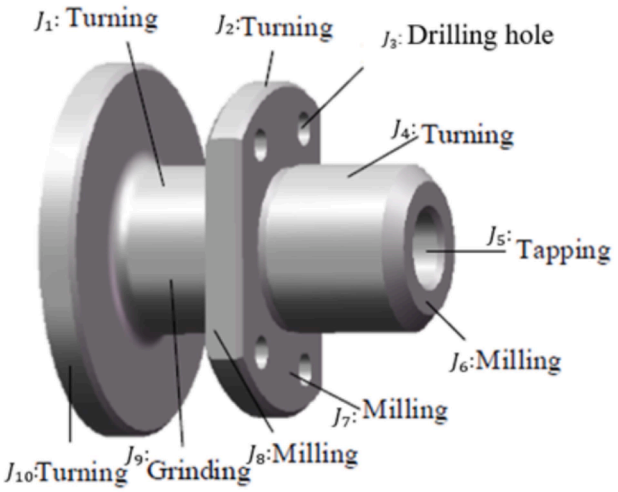


Fig. 10. The product and its processing tasks as provided by Wuhan Huazhong Numerical Control Co.

drilling, tapping and grinding. These tasks are carried out using five different CNC machines located in two separate factories. Each of the five CNC machines can be operated under three different modes of production [89], i.e., a manual production mode (MAN) and two automatic production modes based on two advanced operating systems (PLC or APC). In MAN mode, the CNC machine behaves like a conventional or standard machine. In this regard, the operator of a CNC machine is able to press buttons, turn handwheels, and activate switches to operate the process of tasks while improving its functional performance. However, in PLC or APC modes, all of these activities are performed automatically. In order to obtain more information on these CNC machine modes readers can refer to the Wuhan Huazhong Numerical Control Co.[4] as well as studies by Alphonsus and Abdullah [76] and Shilyaev et al., [77] which describe PLC and APC systems respectively.

The processing times ($PT_{nmpf} = (PT^{pes}_{nmpf}, PT^{rea}_{nmpf}, PT^{opt}_{nmpf})$) required by the machines to process these ten tasks according to their three different modes of production are shown in Table 9. Note that not every task can be processed on every machine. Thus, some rows of this table are empty. Other parameters such as energy consumption levels (IEC_{mpf}, UEC_{mpf}, EC_{mpf}) as well as economic and social factors (CO_{mpf}, CJ_{mpf}, JO_{mpf}, LD_{mpf}, RW_{mpf}) are given in Table 10. The case study is solved in the context where there is no disruptive event ($t = 0$). Thus, only a deterministic schedule is provided where the BD reformulation is used to obtain an optimal makespan of 488.19 min in a computational time of 8.45 s.

In addition to highlighting the impact of key parameters, including the number of tasks (N) and factories (F), budget (B), upper limits of energy consumption ($UBEC$), number of working days lost (UBL), and the lower limit of job opportunities created (LBJ) on the results, a sensitivity analysis is conducted using various values for these parameters. The nominal value of each parameter is uniformly increased across four additional cases; for instance, the nominal value of 10 tasks is increased to 20, 30, 40, and then 50 tasks. For each case, the makespan is evaluated. Fig. 11 illustrates the variation in the makespan (in minutes) resulting from the alteration of these parameters.

As shown in Fig. 11(a), an increase in the number of tasks significantly increases the makespan while an increase in the number of factories reduces the makespan (Fig. 11(b)) as the number of tasks assigned to each factory is reduced. Fig. 11(c) shows that an increase in the upper bound of the budget can reduce the makespan. However, this improvement is limited. Fig. 11(d) reveals that increasing the allowable limit of total energy consumption can reduce the makespan. However, as with the upper bound of the budget, this improvement is limited. Fig. 11 (e) confirms that the influence of the upper bound of lost working days on the makespan is generally the same as that of the maximum allowable energy consumption. Finally, as shown in Fig. 11(f), an increase in the lower bound of the number of job opportunities created does not lead to a reduction of the makespan. However, even if it increases the makespan, its main objective is more of a social nature by offering more job opportunities.

5.3. Comparison of scheduling strategies and rescheduling policies

As previously discussed, the proposed real-time scheduling concept focuses on the incorporation of new tasks arriving at time t, alongside the estimated machine failure based on probabilistic theories. The implementation of predictive-reactive and proactive-reactive scheduling strategies facilitates this real-time scheduling process, with the evaluation and comparison of their performances based on the makespan. These strategies diverge in the application of decision rules AF1 and AF2 within our heuristics. In the case of an anticipated disruptive event at time t, the predictive-reactive scheduling strategy applies AF1 and AF2 at time t-($PCmin$), where $PCmin$ denotes the minimum processing time among all unassigned tasks. On the other hand, the proactive-reactive scheduling strategy employs the decision rules AF1 and AF2 at time t.

According to the results outlined in Table 11, the average makespan values obtained through the predictive-reactive strategy are notably shorter compared to those resulting from the proactive-reactive strategy. To compare the performance of the four heuristics, the makespan values of each test were normalized, and the normalized mean values are presented in Fig. 12 with a 95 % confidence level utilizing the Student's t-distribution. For the predictive-reactive scheduling strategy (Fig. 12 (a)), it was observed that the H4 heuristic emerged as the most effective optimization algorithm. Additionally, H2 outperformed H3, while H1 exhibited the least favorable results within this strategy. Similarly, for the proactive-reactive scheduling strategy (Fig. 12(b)), the findings revealed that H4 and H1 were respectively the best and worst heuristics in this context.

Another important concept is the cost of rescheduling referring to the time lost due to rescheduling. This cost is computed by the deviation of the makespan of the stochastic schedule from the makespan of the deterministic schedule. Table 12 reports the comparison of the continuous rescheduling policy with the event-driven rescheduling policy. In each test problem case, we assume that a number of tasks corresponding 25 % of the total number of tasks arrive as new tasks at time $0 < t \leq CMAX_t$. with $CMAX_t$ being the makespan at time zero (i.e., the makespan resulting from the deterministic schedule) found by the BD reformulation for each test problem. After generating 0.25 of the total number of tasks in each test problem as real-time events and applying heuristics, the rescheduling cost for each rescheduling policy is computed as the difference in hours between the makespan obtained using the rescheduling policy and the makespan of the deterministic schedule. Based on the results taken from Table 12,Fig. 13 shows that continuous rescheduling has a higher cost in terms of time than the event-driven rescheduling policy. Moreover, the behavior of the four heuristics for each of the policies is similar with respect to the solution found.

In conclusion, the predictive-reactive scheduling strategy is more efficient than the proactive-reactive strategy in performing real-time scheduling for our proposed sustainable distributed permutation flowshop system. Regarding the how-to-reschedule process, the event-driven rescheduling policy is more efficient than the continuous rescheduling policy. Finally, in all the analyses, our heuristic H4 in majority of analyses, shows the best performance among the other proposed alternatives.

[4] https://huazhongcnc.en.made-in-china.com/

Table 9
Processing times of tasks (PT^{pes}_{nmpf}, PT^{rea}_{nmpf}, PT^{opt}_{nmpf} in minutes) .

Machine	Operating mode	Tasks J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8	J_9	J_{10}
CNC 1 turning	PLC	(5.6, 5.5, 5.4)	(4.9, 4.8, 4.7)	–	(5.2, 5.1, 5)	–	–	–	–	–	(5.6, 5.5, 5.4)
	APC	(5.6, 5.45, 5.35)	(5, 4.9, 4.8)	–	(5.3, 5.2, 5.1)	–	–	–	–	–	(5.65, 5.5, 5.35)
	MAN	(6, 5.9, 5.7)	(6, 5.8, 5.7)	–	(6.8, 6.5, 6.4)	–	–	–	–	–	(7, 6.7, 6.5)
CNC 2 milling	PLC	–	–	–	–	–	(3.6, 3.5, 3.4)	(3.4, 3.3, 3.2)	(3.5, 3.4, 3.3)	–	–
	APC	–	–	–	–	–	(3.7, 3.6, 3.5)	(3.5, 3.4, 3.3)	(3.6, 3.5, 3.4)	–	–
	MAN	–	–	–	–	–	(6, 5.5, 5)	(6, 5.7, 5.5)	(5.5, 5.3, 5)	–	–
CNC 3 drilling	PLC	–	–	(4.7, 4.6, 4.4)	–	–	–	–	–	–	–
	APC	–	–	(4.8, 4.7, 4.6)	–	–	–	–	–	–	–
	MAN	–	–	(7, 6, 5.5)	–	–	–	–	–	–	–
CNC 4 tapping	PLC	–	–	–	–	(3.9, 3.8, 3.7)	–	–	–	–	–
	APC	–	–	–	–	(4, 3.9, 3.8)	–	–	–	–	–
	MAN	–	–	–	–	(6, 5, 4)	–	–	–	–	–
CNC 5 grinding	PLC	–	–	–	–	–	–	–	–	(4.2, 4.1, 4)	–
	APC	–	–	–	–	–	–	–	–	(4.3, 4.2, 4.1)	–
	MAN	–	–	–	–	–	–	–	–	(6.4, 5.8, 5)	–

Table 10
Parameters of our numerical example .

Machine	Operating mode	IEC_{mpf} (kWh)	UEC_{mpf} (kWh)	EC_{mpf} (kWh)	CO_{mpf} ($)	CJ_{mpf} ($)	JO_{mpf} (Person)	LD_{mpf} (Days)	RW_{mpf}
CNC 1-turning	PLC	0.5	4.1	2.9	32.4×10^3	2	3	7	0.04
	APC	0.45	4.15	3	34.2×10^3	2	3	7	0.03
	MAN	0.52	4.3	3.2	20.4×10^3	1	8	2	0.14
CNC 2- milling	PLC	0.3	3.8	3.1	41.5×10^3	3	2	7	0.02
	APC	0.35	3.75	3.15	42.1×10^3	3	2	7	0.02
	MAN	0.57	4.5	5.2	28.5×10^3	1	6	2	0.12
CNC 3- drilling	PLC	0.2	2.6	1.8	31.2×10^3	3	4	7	0.02
	APC	0.3	2.75	1.9	32.4×10^3	3	4	7	0.01
	MAN	0.4	3.5	2.4	16.3×10^3	2	8	2	0.17
CNC 4- tapping	PLC	0.5	3.1	1.9	23.3×10^3	4	5	7	0.01
	APC	0.45	3.2	2	22.5×10^3	4	5	7	0.03
	MAN	0.75	4.5	3.2	11.7×10^3	2	6	2	0.15
CNC 5- grinding	PLC	0.3	2.6	1.3	32.1×10^3	4	4	10	0.02
	APC	0.35	2.65	1.4	31.7×10^3	4	4	10	0.03
	MAN	0.8	3.76	2.3	18.5×10^3	1	8	3	0.15

5.4. Discussions and managerial insights

In the realm of production scheduling, recent progress has emphasized the need to redefine task sequences in response to real-time events and uncertainties, such as new task arrivals and machine breakdowns. This shift aims to enhance adaptability and responsiveness in dynamic production environments. Concurrently, integrating sustainability criteria across economic, environmental, and social dimensions has emerged as a pivotal challenge. This study contributes to this evolving discourse by redefining a distributed permutation flow-shop in alignment with sustainability criteria.

Our optimization model prioritizes makespan minimization while accounting for energy consumption, lost working days, and job opportunities within defined constraints. Real-time scheduling is achieved through predictive-reactive and proactive-reactive strategies, complemented by continuous and event-driven rescheduling policies. These approaches serve as invaluable tools for production managers, enabling them to balance intricate scheduling objectives while responding effectively to dynamic events.

The evaluation of reformulations, heuristics, and metaheuristics provides essential insights into their effectiveness in achieving scheduling solutions. Notably, the BD reformulation, among various reformulations, exhibits exceptional speed and generates solutions approaching optimality. This underscores the potential of incorporating reformulations to significantly reduce computational time and establish robust lower bounds for our distributed permutation flow-shop system. The stochastic nature of our proposed heuristics offers varying degrees of efficiency, with the H4 heuristic standing out for its robustness and effectiveness. Among the metaheuristic algorithms, SA demonstrates noteworthy robustness compared to TS, evident from the analysis of optimality gaps and standard deviations.

Summarizing the key results, Fig. 2 employs Gantt charts to provide a comparative visualization of the deterministic schedule and our rescheduling strategies. Figs. 3 and 4 illustrate potential solutions and the neighborhood procedure in our metaheuristic algorithms, respectively. Performance comparisons highlight the superiority of the BD

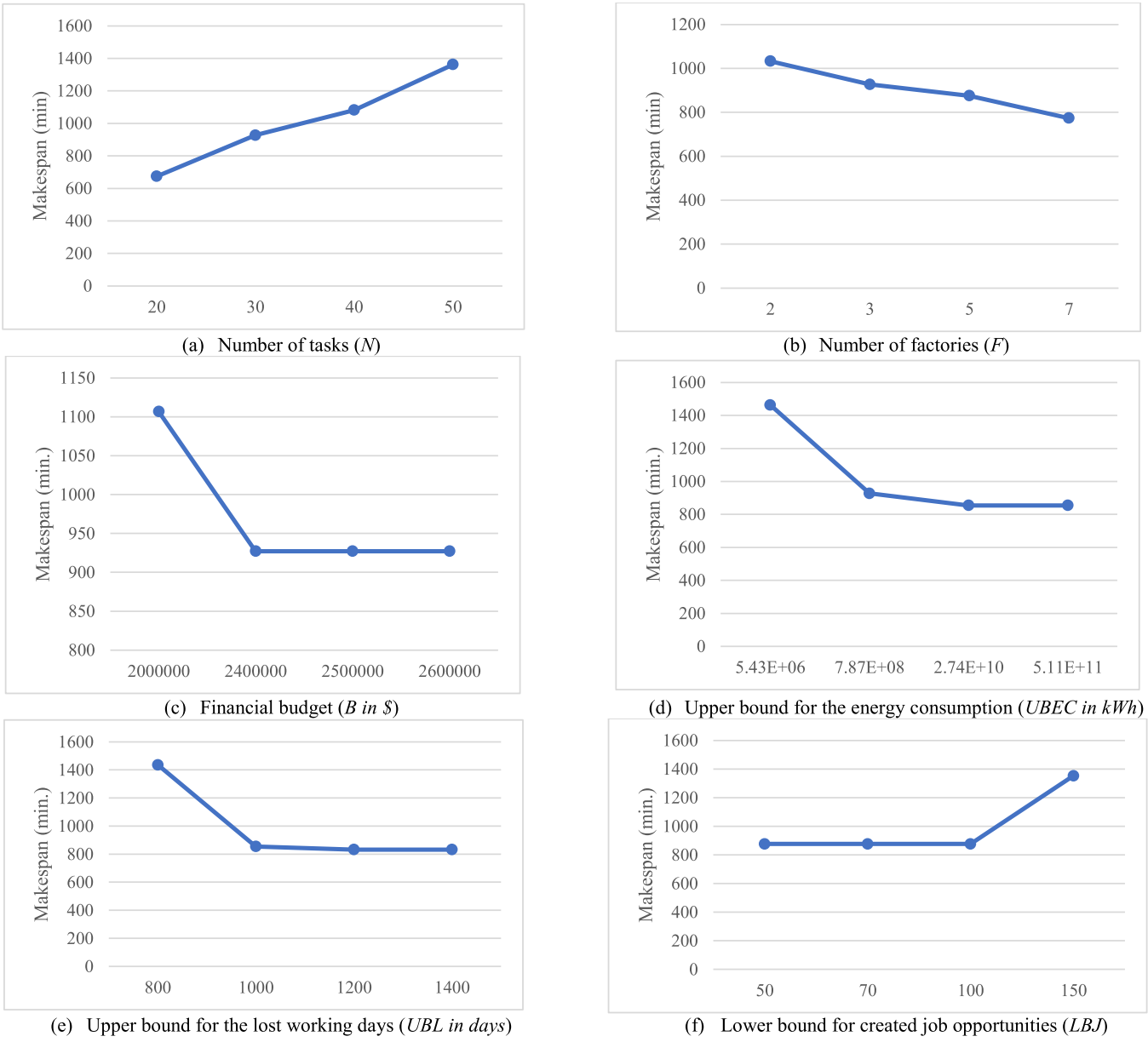
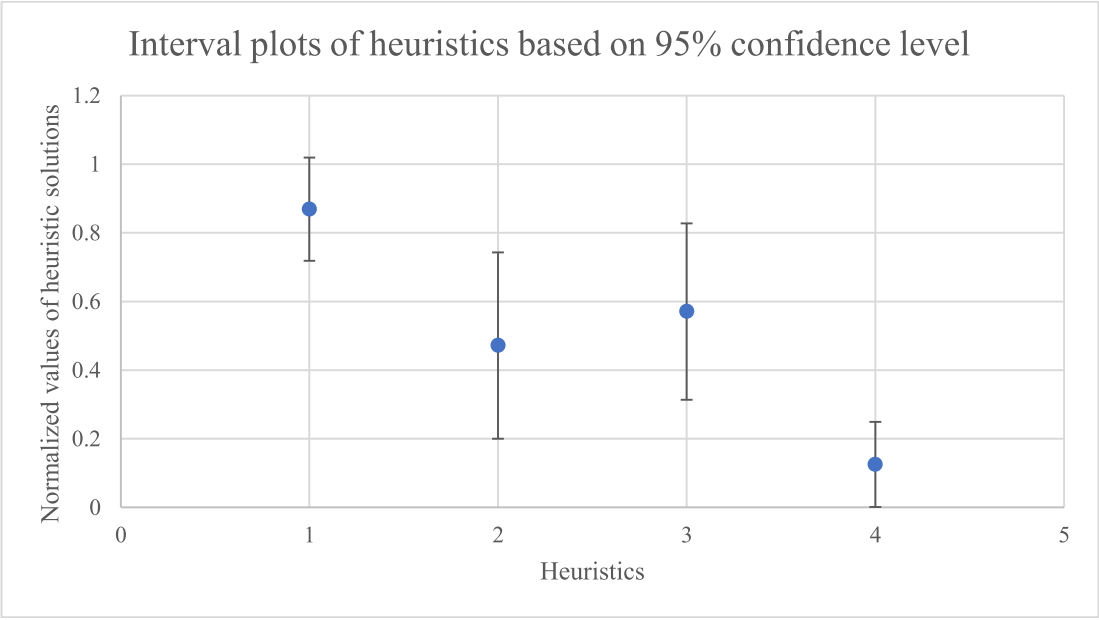


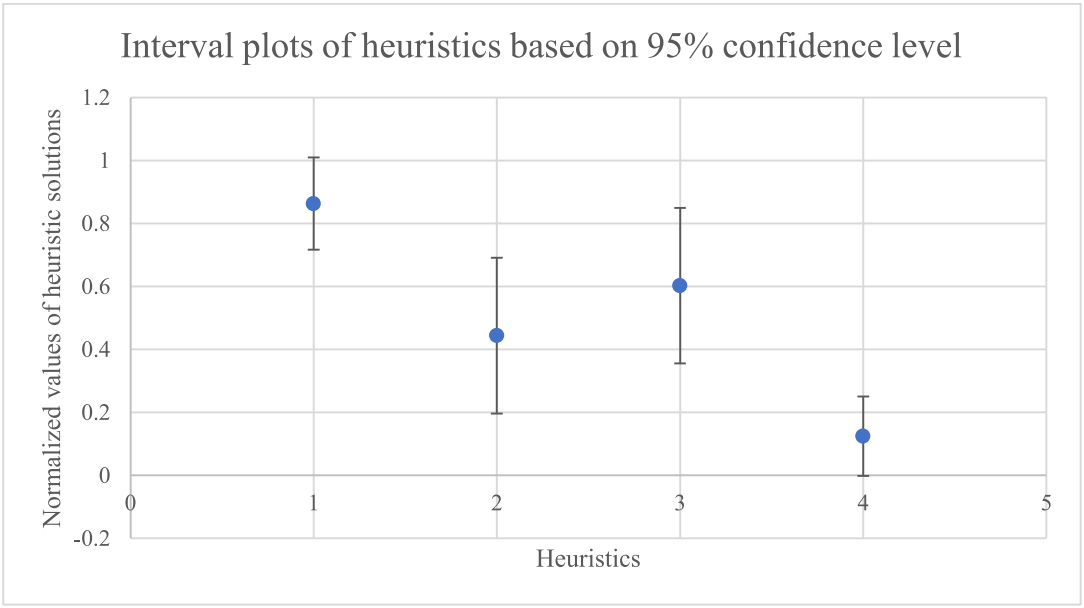
Fig. 11. Sensitivity analyses on key parameters.

Table 11
Comparison of the makespan (h) of the scheduling strategies.

Test problem	Predictive-reactive scheduling				Proactive-reactive scheduling			
	H1	H2	H3	H4	H1	H2	H3	H4
T1	80.87	66.07	77.34	65.13	95.27	76.25	92.16	72.22
T2	173.29	156.54	160.15	135.64	187.54	164.19	172.62	145.39
T3	690.36	684.77	647.47	652.06	704.21	693.33	662.34	657.23
T4	1052.13	1050.48	1051.14	1024.66	1067.35	1060.32	1066.86	1033.88
T5	220.88	164.15	176.28	166.92	228.66	170.25	191.65	180.33
T6	242.75	233.09	235.71	223.81	252.58	245.85	247.51	236.84
T7	248.6	216.45	240.59	214.82	258.82	227.38	248.34	225.05
T8	333.94	324.67	303.3	310.58	346.07	331.51	316.53	322.19
T9	879.18	879.49	857.23	871.12	887.95	890.22	866.44	881.97
T10	1120.89	1068.46	1163.17	1080.45	1129.64	1082.63	1170.03	1096.38
T11	1175.88	1155.64	1200.31	1171.67	1187.48	1169.33	1209.22	1182.15
T12	1502.47	1603.35	1577.28	1425.15	1517.96	1610.84	1592.14	1438.57
Average	643.436	633.596	640.83	**611.834**	655.294	643.583	652.986	**622.683**



(a)



(b)

Fig. 12. Normalized average values of the makespan obtained according to the scheduling strategies, i.e., (a) predictive-reactive scheduling and (b) proactive-reactive scheduling.

reformulation (Fig. 6) and the top-performing heuristic H4 (Fig. 8). A detailed case study (Fig. 10) further demonstrates the practical applicability of our approach, supported by efficiency data in Table 9. Sensitivity analysis (Fig. 11) and real-time scheduling assessments (Table 11) offer nuanced insights into critical parameters and rescheduling policies.

In conclusion, this study provides valuable managerial insights, showcasing the potential of integrating sustainability criteria into real-time scheduling. The comparison of methods emphasizes the efficiency and robustness of reformulations and heuristics. Sensitivity analysis sheds light on the pivotal role of critical parameters, guiding decision-making for production managers and scheduling programmers. Scheduling strategy and rescheduling policy selection emerge as crucial factors, with recommendations favoring certain algorithms, such as SA over TS. Overall, this research equips production managers with practical tools and knowledge to enhance the efficiency and sustainability of their production systems.

6. Conclusions, findings and future research avenues

In summary, this paper introduces a comprehensive optimization model that addresses sustainability and uncertainty concerns within the context of a distributed permutation flow-shop scheduling problem. The primary objective is to minimize the makespan while incorporating constraints related to energy consumption, job opportunities, and lost working days. The model also accounts for the influence of different machine operating modes on social criteria, including the number of workers and training needs, along with three energy consumption levels

Table 12
Comparison of rescheduling policies based on the cost of rescheduling (h).

	Event-driven rescheduling				Continuous rescheduling			
	H1	H2	H3	H4	H1	H2	H3	H4
T1	2.43	1.98	2.32	1.95	7.99	6.62	7.822	5.941
T2	5.2	4.703	4.812	4.075	15.88	14.15	15.16	13.06
T3	20.74	20.57	19.45	19.59	62.48	61.87	59.28	59.03
T4	31.61	31.56	31.58	30.78	95.63	95.26	95.71	93.06
T5	6.637	4.93	5.296	5.01	20.52	15.25	16.29	15.95
T6	7.29	7.003	7.082	6.72	22.08	21.48	21.54	21.03
T7	7.46	6.504	7.22	6.45	22.54	20.21	22.45	19.66
T8	10.03	9.755	9.11	9.33	30.76	30.05	28.25	28.7
T9	26.41	26.42	25.75	26.17	79.67	79.344	77.919	79.2
T10	33.68	32.105	34.95	32.46	101.56	97.074	105.56	97.57
T11	35.33	34.72	36.06	35.2	106.054	104.95	108.796	105.69
T12	45.14	48.17	47.39	42.82	135.894	145.5	142.237	129.35
Average	19.33	19.03	19.25	**18.384**	58.424	57.65	58.416	**55.697**

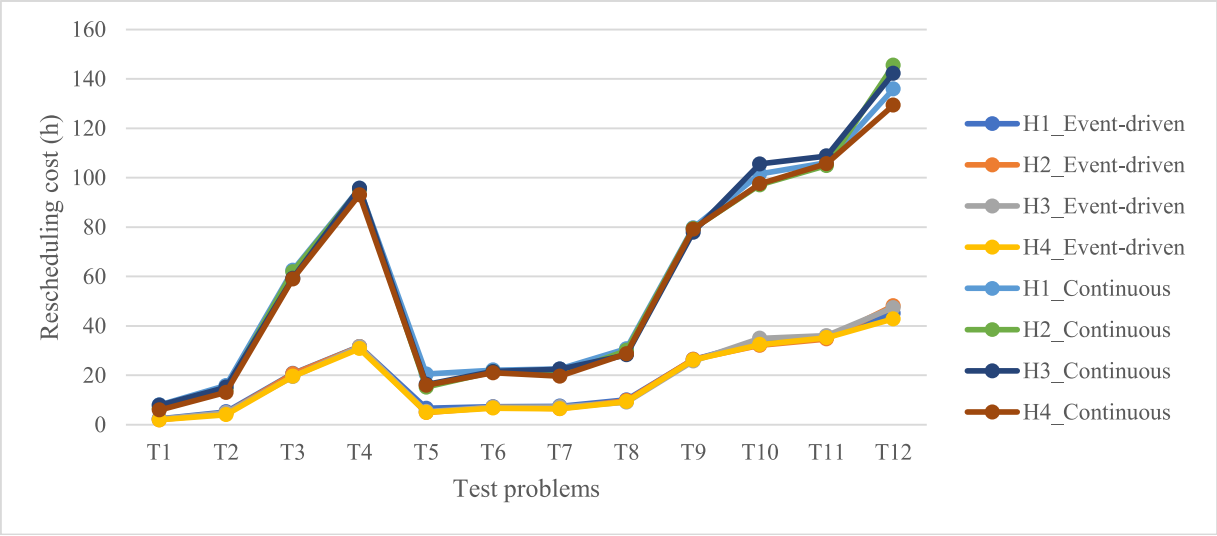


Fig. 13. Rescheduling cost (in *hours*) of the heuristics according to continuous and event-driven rescheduling policies.

based on machine states. To address the uncertainty, real-time scheduling is employed, utilizing predictive-reactive, proactive-reactive, continuous, and event-driven rescheduling approaches.

For solving the proposed optimization model, efficient reformulations of the optimization model through Lagrangian relaxation and Benders decomposition are proposed, demonstrating superior solutions compared to an exact solver. To expedite the solution-finding process, four tailored heuristics (H1, H2, H3, and H4) and two powerful metaheuristic algorithms (SA and TS) are deployed, collectively enhancing the overall optimization process.

Upon comparing algorithm efficiencies, sensitivity analyses highlight the profound impact of key parameters on scheduling outcomes, emphasizing the need for resource-efficient practices. Factors such as the number of tasks, factories, and budget constraints significantly influence makespan, while estimating energy consumption limits, sustainable thresholds, job opportunity limits, and lost workdays underscore the broader societal implications of scheduling decisions.

Analysis of predictive-reactive and proactive-reactive scheduling strategies reveals the former's tendency to result in shorter makespans, indicating the effectiveness of anticipating real-time events. Moreover, the event-driven rescheduling policy outperforms continuous rescheduling in terms of time savings, emphasizing the importance of responsive rescheduling to minimize disruptions. Heuristic H4 consistently emerges as the most efficient algorithm, showcasing robustness in handling dynamic scheduling scenarios and offering valuable insights for production managers optimizing processes while considering sustainability factors.

While this study significantly contributes to efficient solutions for sustainable distributed permutation flow-shop scheduling under uncertainties, certain limitations indicate potential avenues for future research. Integrating real-time events with probabilistic scenarios could enhance scenario-based robust optimization models. Considering additional criteria like task assignment stability and tardiness could transform the model into a multi-objective optimization framework. Exploring metaheuristic algorithms beyond SA and TS, such as adaptive large neighborhood search, presents a promising direction for future research. Lastly, combining the proposed BD reformulation with machine learning tools may yield even more efficient solutions for the presented model. These avenues represent promising directions to advance the field and address the identified limitations.

CRediT authorship contribution statement

Amir M. Fathollahi-Fard: Conceptualization, Data curation, Investigation, Methodology, Resources, Software, Validation, Writing – original draft, Writing – review & editing. **Lyne Woodward:** Conceptualization, Funding acquisition, Supervision, Visualization, Writing – original draft, Writing – review & editing. **Ouassima Akhrif:** Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no conflict of interest.

Data availability

The data that has been used is confidential.

Acknowledgement

The authors presented a condensed version of this paper at the 5th International Conference on Industry 4.0 and Smart Manufacturing,[5] where they exclusively introduced our mixed integer programming model by a set of small instances and the case study. The paper titled "An Optimization Model for Smart and Sustainable Distributed Permutation Flow Shop Scheduling" published in Procedia Computer Science journal is the condensed version of this paper. It is important to note that the authors have no affiliations or financial interests, either non-financial ones, with organizations such as Wuhan Huazhong Numerical Control Company and the People's Republic of China, which are mentioned in this paper Finally, the authors extend their gratitude for the financial support received for this research from the Discovery Grant Program of the National Sciences and Engineering Research Council of Canada (NSERC), grant number RGPIN-2019-05853.

References

[1] A.M. Fathollahi-Fard, L. Woodward, O. Akhrif, Sustainable distributed permutation flow-shop scheduling model based on a triple bottom line concept, J. Ind. Inf. Integr. (2021) 100233.
[2] M. Al-Behadili, D. Ouelhadj, D. Jones, Multi-objective biased randomised iterated greedy for robust permutation flow shop scheduling problem under disturbances, J. Oper. Res. Soc. 71 (11) (2020) 1847–1859.
[3] T. Varelmann, N. Erwes, P. Schäfer, A. Mitsos, Simultaneously optimizing bidding strategy in pay-as-bid-markets and production scheduling, Comput. Chem. Eng. 157 (2022) 107610.
[4] A.G. Frank, L.S. Dalenogare, N.F. Ayala, Industry 4.0 technologies: implementation patterns in manufacturing companies, Int. J. Prod. Econ. 210 (2019) 15–26.
[5] L.S. Dalenogare, G.B. Benitez, N.F. Ayala, A.G. Frank, The expected contribution of Industry 4.0 technologies for industrial performance, Int. J. Prod. Econ. 204 (2018) 383–394.
[6] M. Ghaleb, H. Zolfagharinia, S. Taghipour, Real-time production scheduling in the Industry-4.0 context: addressing uncertainties in job arrivals and machines breakdowns, Comput. Oper. Res. (2020) 105031.
[7] C.J. Corbett, D.A. Kirsch, International diffusion of ISO 14000 certification, Prod. Oper. Manage 10 (3) (2001) 327–342.
[8] C. Li, L. Li, Y. Tang, Y. Zhu, L. Li, A comprehensive approach to parameters optimization of energy-aware CNC milling, J. Intell. Manuf. 30 (1) (2019) 123–138.
[9] J. Llach, F. Marimon, M del Mar Alonso-Almeida, Social Accountability 8000 standard certification: analysis of worldwide diffusion, J. Clean. Prod. 93 (2015) 288–298.
[10] F. Manríquez, H. González, N. Morales, Short-term open-pit production scheduling optimizing multiple objectives accounting for shovel allocation in stockpiles, Optim. Eng. (2022) 1–27.
[11] Z. Zhuang, Y. Li, Y. Sun, W. Qin, Z.H. Sun, Network-based dynamic dispatching rule generation mechanism for real-time production scheduling problems with dynamic job arrivals, Robot. Comput. Integr. Manuf. 73 (2022) 102261.
[12] L.R. Abreu, J.O. Cunha, B.A. Prata, J.M. Framinan, A genetic algorithm for scheduling open shops with sequence-dependent setup times, Comput. Oper. Res. 113 (2020) 104793.
[13] C.M. Harmonosky, S.F. Robohn, Real-time scheduling in computer integrated manufacturing: a review of recent research, Int. J. Comput. Integr. Manuf. 4 (6) (1991) 331–340.
[14] B. Naderi, R. Ruiz, The distributed permutation flowshop scheduling problem, Comput. Oper. Res. 37 (4) (2010) 754–768.
[15] H. Soleimani, P. Chhetri, A.M. Fathollahi-Fard, S.M.J. Mirzapour Al-e-Hashem, S Shahparvari, Sustainable closed-loop supply chain with energy efficiency: lagrangian relaxation, reformulations and heuristics, Ann. Oper. Res. (2022) 1–26.
[16] A. Hamzadayı, An effective benders decomposition algorithm for solving the distributed permutation flowshop scheduling problem, Comput. Oper. Res. 123 (2020) 105006.
[17] S.M. Ali, A.M. Fathollahi-Fard, R. Ahnaf, K.Y. Wong, A multi-objective closed-loop supply chain under uncertainty: an efficient Lagrangian relaxation reformulation using a neighborhood-based algorithm, J. Clean. Prod. (2023) 138702.
[18] S.C. Graves, A review of production scheduling, Oper. Res. 29 (4) (1981) 646–675.
[19] L. Tang, J. Liu, A. Rong, Z. Yang, A review of planning and scheduling systems and methods for integrated steel production, Eur J Oper Res 133 (1) (2001) 1–20.
[20] C. Gahm, F. Denz, M. Dirr, A. Tuma, Energy-efficient scheduling in manufacturing companies: a review and research framework, Eur J Oper Res 248 (3) (2016) 744–757.
[21] D. Rossit, F. Tohmé, Scheduling research contributions to Smart manufacturing, Manuf. Lett. 15 (2018) 111–114.
[22] M. Parente, G. Figueira, P. Amorim, A. Marques, Production scheduling in the context of Industry 4.0: review and trends, Int. J. Prod. Res. 58 (17) (2020) 5401–5431.
[23] D.A. Rossit, F. Tohmé, M. Frutos, Industry 4.0: smart scheduling, Int. J. Prod. Res. 57 (12) (2019) 3802–3813.
[24] J. Zhang, G. Ding, Y. Zou, S. Qin, J. Fu, Review of job shop scheduling research and its new perspectives under Industry 4.0, J. Intell. Manuf. 30 (4) (2019) 1809–1830.
[25] A. Dolgui, D. Ivanov, S.P. Sethi, B. Sokolov, Scheduling in production, supply chain and Industry 4.0 systems by optimal control: fundamentals, state-of-the-art and applications, Int. J. Prod. Res. 57 (2) (2019) 411–432.
[26] X.N. Shen, X. Yao, Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems, Inf Sci (Ny) 298 (2015) 198–224.
[27] K.Z. Gao, P.N. Suganthan, T.J. Chua, C.S. Chong, T.X. Cai, Q.K. Pan, A two-stage artificial bee colony algorithm scheduling flexible job-shop scheduling problem with new job insertion, Expert. Syst. Appl. 42 (21) (2015) 7652–7663.
[28] D. Rahmani, R. Ramezanian, A stable reactive approach in dynamic flexible flow shop scheduling with unexpected disruptions: a case study, Comput. Ind. Eng. 98 (2016) 360–372.
[29] Y. Fu, J. Ding, H. Wang, J. Wang, Two-objective stochastic flow-shop scheduling with deteriorating and learning effect in Industry 4.0-based manufacturing system, Appl. Soft. Comput. 68 (2018) 847–855.
[30] Y. Han, J.Q. Li, D. Gong, H. Sang, Multi-objective migrating birds optimization algorithm for stochastic lot-streaming flow shop scheduling with blocking, IEEe Access. 7 (2018) 5946–5962.
[31] J.M. Framinan, V. Fernandez-Viagas, P. Perez-Gonzalez, Using real-time information to reschedule jobs in a flowshop with variable processing times, Comput. Ind. Eng. 129 (2019) 113–125.
[32] H. Gholizadeh, H. Fazlollahtabar, A.M. Fathollahi-Fard, M.A. Dulebenets, Preventive maintenance for the flexible flowshop scheduling under uncertainty: a waste-to-energy system, Environ. Sci. Pollut. Res. (2021) 1–20.
[33] F. Liu, S. Wang, Y. Hong, X. Yue, On the robust and stable flowshop scheduling under stochastic and dynamic disruptions, IEEE Trans. Eng. Manage. 64 (4) (2017) 539–553.
[34] O. Engin, M.K. Yılmaz, A fuzzy logic based methodology for multi-objective hybrid flow shop scheduling with multi-processor tasks problems and solving with an efficient genetic algorithm, J. Intell. Fuzzy Syst. 42 (1) (2022) 451–463.
[35] O. Engin, M. İşler, An efficient parallel greedy algorithm for fuzzy hybrid flow shop scheduling with setup time and lot size: a case study in apparel process, J. Fuzzy Extens. Appl. 3 (3) (2022) 249–262.
[36] B. Zhang, L.L. Meng, C. Lu, J.Q. Li, Real-time data-driven automatic design of multi-objective evolutionary algorithm: a case study on production scheduling, Appl. Soft. Comput. 138 (2023) 110187.
[37] C. Luo, W. Gong, C. Lu, Knowledge-driven two-stage memetic algorithm for energy-efficient flexible job shop scheduling with machine breakdowns, Expert. Syst. Appl. 235 (2024) 121149.
[38] S.A. Mansouri, E. Aktas, U. Besikci, Green scheduling of a two-machine flowshop: trade-off between makespan and energy consumption, Eur. J. Oper. Res. 248 (3) (2016) 772–788.
[39] J. Shahrabi, M.A. Adibi, M. Mahootchi, A reinforcement learning approach to parameter estimation in dynamic job shop scheduling, Comput. Ind. Eng. 110 (2017) 75–82.
[40] H. Mokhtari, A. Hasani, An energy-efficient multi-objective optimization for flexible job-shop scheduling problem, Comput. Chem. Eng. 104 (2017) 339–352.
[41] X. Wu, Y. Sun, A green scheduling algorithm for flexible job shop with energy-saving measures, J. Clean. Prod. 172 (2018) 3249–3264.
[42] S. Wang, X. Wang, J. Yu, S. Ma, M. Liu, Bi-objective identical parallel machine scheduling to minimize total energy consumption and makespan, J. Clean. Prod. 193 (2018) 424–440.
[43] X. Wu, A. Che, A memetic differential evolution algorithm for energy-efficient parallel machine scheduling, Omega (Westport) 82 (2019) 155–165.
[44] M. Dai, D. Tang, A. Giret, M.A. Salido, Multi-objective optimization for energy-efficient flexible job shop scheduling problem with transportation constraints, in: Robot. Comput. Integr. Manuf., 59, 2019, pp. 143–157.
[45] B. Zhang, Q.K. Pan, L. Gao, X.Y. Li, L.L. Meng, K.K. Peng, A multiobjective evolutionary algorithm based on decomposition for hybrid flowshop green scheduling problem, Comput. Ind. Eng. 136 (2019) 325–344.
[46] E.B. Tirkolaee, A. Goli, G.W. Weber, Fuzzy mathematical programming and self-adaptive artificial fish swarm algorithm for just-in-time energy-aware flow shop scheduling problem with outsourcing option, IEEE Trans. Fuzzy Syst. 28 (11) (2020) 2772–2783.
[47] A.K. Shukla, R. Nath, P.K. Muhuri, Q.D. Lohani, Energy efficient multi-objective scheduling of tasks with interval type-2 fuzzy timing constraints in an Industry 4.0 ecosystem, Eng. Appl. Artif. Intell. 87 (2020) 103257.
[48] I.H. Sin, B. Do Chung, Bi-objective optimization approach for energy aware scheduling considering electricity cost and preventive maintenance using genetic algorithm, J. Clean. Prod. 244 (2020) 118869.
[49] D. Anghinolfi, M. Paolucci, R. Ronco, A bi-objective heuristic approach for green identical parallel machine scheduling, Eur. J. Oper. Res. 289 (2) (2020) 416–434.

[5] https://www.msc-les.org/ism2023/

[50] Z. Hong, Z. Zeng, L. Gao, Energy-efficiency scheduling of multi-cell manufacturing system considering total handling distance and eligibility constraints, Comput. Ind. Eng. 151 (2021) 106998.

[51] M.K. Marichelvam, M. Geetha, A memetic algorithm to solve uncertain energy-efficient flow shop scheduling problems, Int. J. Adv. Manuf. Technology 115 (1–2) (2021) 515–530.

[52] X. Jiang, Z. Tian, W. Liu, Y. Suo, K. Chen, X. Xu, Z. Li, Energy-efficient scheduling of flexible job shops with complex processes: a case study for the aerospace industry complex components in China, J. Ind. Inf. Integr. 27 (2022) 100293.

[53] R. Li, W. Gong, L. Wang, C. Lu, X. Zhuang, Surprisingly popular-based adaptive memetic algorithm for energy-efficient distributed flexible job shop scheduling, IEEe Trans. Cybern. (2023), https://doi.org/10.1109/TCYB.2023.3280175.

[54] J. Gao, R. Chen, A hybrid genetic algorithm for the distributed permutation flowshop scheduling problem, Int. J. Comput. Intell. Syst. 4 (4) (2011) 497–508.

[55] S.W. Lin, K.C. Ying, C.Y. Huang, Minimising makespan in distributed permutation flowshops using a modified iterated greedy algorithm, Int. J. Prod. Res. 51 (16) (2013) 5029–5038.

[56] B. Naderi, R. Ruiz, A scatter search algorithm for the distributed permutation flowshop scheduling problem, Eur. J. Oper. Res. 239 (2) (2014) 323–334.

[57] H. Bargaoui, O.B. Driss, K. Ghédira, A novel chemical reaction optimization for the distributed permutation flowshop scheduling problem with makespan criterion, Comput. Ind. Eng. 111 (2017) 239–250.

[58] V. Fernandez-Viagas, P. Perez-Gonzalez, J.M. Framinan, The distributed permutation flow shop to minimise the total flowtime, Comput. Ind. Eng. 118 (2018) 464–477.

[59] Q.K. Pan, L. Gao, L. Wang, J. Liang, X.Y. Li, Effective heuristics and metaheuristics to minimize total flowtime for the distributed permutation flowshop problem, Expert. Syst. Appl. 124 (2019) 309–324.

[60] R. Ruiz, Q.K. Pan, B. Naderi, Iterated Greedy methods for the distributed permutation flowshop scheduling problem, Omega (Westport) 83 (2019) 213–222.

[61] T. Meng, Q.K. Pan, L. Wang, A distributed permutation flowshop scheduling problem with the customer order constraint, Knowl. Based. Syst. 184 (2019) 104894.

[62] K. Allali, S. Aqil, J. Belabid, Distributed no-wait flow shop problem with sequence dependent setup time: optimization of makespan and maximum tardiness, Simul. Model. Pract. Theory. 116 (2022) 102455.

[63] X. Han, Y. Han, Q. Chen, J. Li, H. Sang, Y. Liu, Y. Nojima, Distributed flow shop scheduling with sequence-dependent setup times using an improved iterated greedy algorithm, Complex Syst. Model. Simul. 1 (3) (2021) 198–217.

[64] Y. Wang, Y. Han, Y. Wang, M.F. Tasgetiren, J. Li, K. Gao, Intelligent optimization under the makespan constraint: rapid evaluation mechanisms based on the critical machine for the distributed flowshop group scheduling problem, Eur. J. Oper. Res. 311 (3) (2023) 816–832.

[65] M.E. Baysal, A. Sarucan, K. Büyüközkan, O. Engin, Distributed fuzzy permutation flow shop scheduling problem: a bee colony algorithm, in: International conference on intelligent and fuzzy systems, Springer International Publishing, Cham, 2020, pp. 1440–1446.

[66] M.E. Baysal, A. Sarucan, K. Büyüközkan, O. Engin, Artificial bee colony algorithm for solving multi-objective distributed fuzzy permutation flow shop problem, J. Intell. Fuzzy Syst. 42 (1) (2022) 439–449.

[67] R. Başar, K. Büyüközkan, O. Engin, Distributed no-wait flow shop with fuzzy environment, in: International Conference on Intelligent and Fuzzy Systems, Springer International Publishing, Cham, 2022, pp. 279–286.

[68] J.J. Wang, L. Wang, A knowledge-based cooperative algorithm for energy-efficient scheduling of distributed flow-shop, IEEE Trans. Syst. Man Cybern. Syst. (99) (2018) 1–15.

[69] Y. Fu, G. Tian, A.M. Fathollahi-Fard, A. Ahmadi, C. Zhang, Stochastic multi-objective modelling and optimization of an energy-conscious distributed permutation flow shop scheduling problem with the total tardiness constraint, J. Clean. Prod. 226 (2019) 515–525.

[70] G. Wang, L. Gao, X. Li, P. Li, M.F. Tasgetiren, Energy-efficient distributed permutation flow shop scheduling problem using a multi-objective whale swarm algorithm. Swarm. Evol. Comput., 2020 100716.

[71] C. Luo, W. Gong, R. Li, C. Lu, Problem-specific knowledge MOEA/D for energy-efficient scheduling of distributed permutation flow shop in heterogeneous factories, Eng. Appl. Artif. Intell. 123 (2023) 106454.

[72] H. Qin, Y. Han, Q. Chen, L. Wang, Y. Wang, J. Li, Y. Liu, Energy-efficient iterative greedy algorithm for the distributed hybrid flow shop scheduling with blocking constraints, IEEe Trans. Emerg. Top. Comput. Intell. 7 (5) (2023) 1442–1457.

[73] C. Lu, L. Gao, W. Gong, C. Hu, X. Yan, X. Li, Sustainable scheduling of distributed permutation flow-shop with non-identical factory using a knowledge-based multi-objective memetic optimization algorithm. Swarm. Evol. Comput., 2020 100803.

[74] L. Yue, H. Wang, J. Mumtaz, M. Rauf, Z. Li, Energy-efficient scheduling of a two-stage flexible printed circuit board flow shop using a hybrid Pareto spider monkey optimisation algorithm, J. Ind. Inf. Integr. 31 (2023) 100412.

[75] Y. Han, J. Li, H. Sang, Y. Liu, K. Gao, Q. Pan, Discrete evolutionary multi-objective optimization for energy-efficient blocking flow shop scheduling with setup time, Appl. Soft. Comput. (2020) 106343.

[76] E.R. Alphonsus, M.O. Abdullah, A review on the applications of programmable logic controllers (PLCs), Renew. Sustain. Energy Rev. 60 (2016) 1185–1205.

[77] P.V. Shilyaev, I.Y. Andryushin, V.V. Golovin, A.A. Radionov, A.S. Karandaev, V. R. Khramshin, Algorithms of a digital automatic system for tension and loop control in a wide-strip hot-rolling mill, Russ. Electr. Eng. 84 (10) (2013) 533–541.

[78] M. Jiménez, M. Arenas, A. Bilbao, M.V. Rodrı, Linear programming with fuzzy parameters: an interactive method resolution, Eur. J. Oper. Res. 177 (3) (2007) 1599–1609.

[79] W. He, D.H. Sun, Scheduling flexible job shop problem subject to machine breakdown with route changing and right-shift strategies, Int. J. Adv. Manuf. Technol. 66 (1–4) (2013) 501–514.

[80] S.V. Mehta, R.M. Uzsoy, Predictable scheduling of a job shop subject to breakdowns, IEEE Trans. Robot. Autom. 14 (3) (1998) 365–378.

[81] M.S. Ross, Introduction to Probability Models, Elsevier books, 2019.

[82] R. Rahmaniani, T.G. Crainic, M. Gendreau, W. Rei, The Benders decomposition algorithm: a literature review, Eur. J. Oper. Res. 259 (3) (2017) 801–817.

[83] Van Laarhoven, P.J., Aarts, E.H., van Laarhoven, P.J., & Aarts, E.H. (1987). *Simulated Annealing* (pp. 7–15). Springer Netherlands.

[84] P. Seydanlou, M. Sheikhalishahi, R. Tavakkoli-Moghaddam, A.M. Fathollahi-Fard, A customized multi-neighborhood search algorithm using the tabu list for a sustainable closed-loop supply chain network under uncertainty, Appl. Soft. Comput. (2023) 110495.

[85] F. Glover, Tabu search: a tutorial, Interfaces. (Providence) 20 (4) (1990) 74–94.

[86] M.L. Fisher, The Lagrangian relaxation method for solving integer programming problems, Manage. Sci. 27 (1) (1981) 1–18.

[87] J. Gmys, M. Mezmaz, N. Melab, D. Tuyttens, A computationally efficient branch-and-bound algorithm for the permutation flow-shop scheduling problem, Eur. J. Oper. Res. 284 (3) (2020) 814–833.

[88] Tordecilla, R.D., Montoya-Torres, J.R., Quintero-Araujo, C.L., Panadero, J., & Juan, A.A. (2022). The location routing problem with facility sizing decisions. Int. Trans. Oper. Res. https://doi.org/10.1111/itor.13125.

[89] C.P. Tautenhain, A.P. Barbosa-Povoa, B. Mota, M.C. Nascimento, An efficient Lagrangian-based heuristic to solve a multi-objective sustainable supply chain problem, Eur. J. Oper. Res. 294 (1) (2021) 70–90.

[90] J.F. Benders, Partitioning procedures for solving mixed-variables programming problems, Numer. Math. (Heidelb) 4 (1) (1962) 238–252.