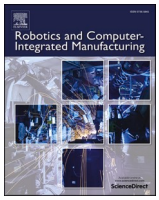
Robotics and Computer–Integrated Manufacturing 98 (2026) 103160

Contents lists available at ScienceDirect

Robotics and Computer-Integrated Manufacturing

journal homepage: www.elsevier.com/locate/rcim



A reinforcement learning-based metaheuristic approach to address the dynamic scheduling problem in cloud manufacturing with task cancellation

Atefeh Rajabi-Kafshgar [a], Mostafa Hajiaghaei-Keshteli [b,*], Mohammad Reza Mohammad Aliha [a]

[a] School of Industrial Engineering, Iran University of Science and Technology, Tehran, Iran
[b] School of Science and Engineering, Tecnologico de Monterrey, Monterrey, Mexico

ARTICLE INFO

Keywords:
Dynamic scheduling problem
Q-learning algorithm
Metaheuristic algorithm
Cloud manufacturing
Task cancellation
Manufacturing services

ABSTRACT

Recent developments in cloud manufacturing (CMg) have highlighted the need for efficient task scheduling and resource allocation in distributed and dynamic environments. To the best of our knowledge, existing studies have not considered dynamic events such as task cancellation, which can lead to resource inefficiencies and disrupt the initial schedule. To address this gap, this paper introduces a novel dynamic task scheduling and service allocation (DTSSA) problem in CMg that considers task cancellation. The proposed model considers logistics time and different arrival times, which directly impact the tasks' completion times. Furthermore, a reinforcement learning-based genetic algorithm is developed to tackle the NP-hardness of the model and solve medium- and large-scale problems in a reasonable time. The algorithm dynamically selects search operators using the Q-learning algorithm and applies a ε-greedy approach to improve search capabilities. In this regard, first, the metaheuristic algorithms' parameters are tuned by the Taguchi method. The proposed algorithms were evaluated using 30 benchmark instances from the literature, as well as example cases inspired by existing studies. Next, the mathematical model is evaluated by implementing small-scale examples using GAMS software. Then, the algorithms are compared with not only some well-known metaheuristic algorithms but also recently developed metaheuristic algorithms using statistical tests and several test problems of different sizes. Additionally, results show that the rescheduling problem provides up to 8.7% better solutions on average than the initial schedule. Lastly, the model's sensitivity analysis reveals that the longer the processing time and logistic time, the longer the maximum completion time for scheduling and rescheduling.

1. Introduction

Cloud manufacturing (CMg) is a service-oriented and networked manufacturing model that integrates distributed manufacturing services (MSs) with cloud computing, the Internet of Things (IoT), artificial intelligence (AI), and advanced technologies to build adaptive and scalable systems [1]. Using an online platform, companies can share their manufacturing resources, such as machines, production capacities, and expertise, as needed to meet market demand without investing heavily in physical infrastructure [2]. In light of CMg's wide range of applications across a wide range of industries, its potential to revolutionize the manufacturing process and enhance operational efficiency is becoming more and more evident.

A wide range of industries have been rapidly implementing CMg over the past decade, including sheet metal, medical devices, mold making, automobile components, paints, steel, food processing equipment, 3D printing, robotics, and large equipment manufacturing [3]. CMg enables real-time monitoring, mass customization, quick service selection, and mass collaboration. It has been widely adopted in industries such as smart factories, customized production, and flexible supply chains [4].

The increasing potential of CMg in resource sharing, flexibility, and global industrial collaboration has led to renewed interest in this approach in recent years [5]. According to the report by Cognitive Market Research, the global CMg market is expected to increase rapidly, from $80.54 billion in 2024 to $234.59 billion by 2031, at a compound annual growth rate (CAGR) of 16.5%[1] (See Fig. 1). The regional analysis shows that North America held over 40% of the global CMg market share in 2024. The automotive industry is primarily driving this growth

* Corresponding author.
E-mail address: mostafahaji@tec.mx (M. Hajiaghaei-Keshteli).

[1] https://www.cognitivemarketresearch.com/cloud-manufacturing-market-report Retrieved on January 29, 2025.

https://doi.org/10.1016/j.rcim.2025.103160
Received 25 March 2025; Received in revised form 25 July 2025; Accepted 30 September 2025
Available online 9 October 2025
0736-5845/© 2025 Elsevier Ltd. All rights are reserved, including those for text and data mining, AI training, and similar technologies.

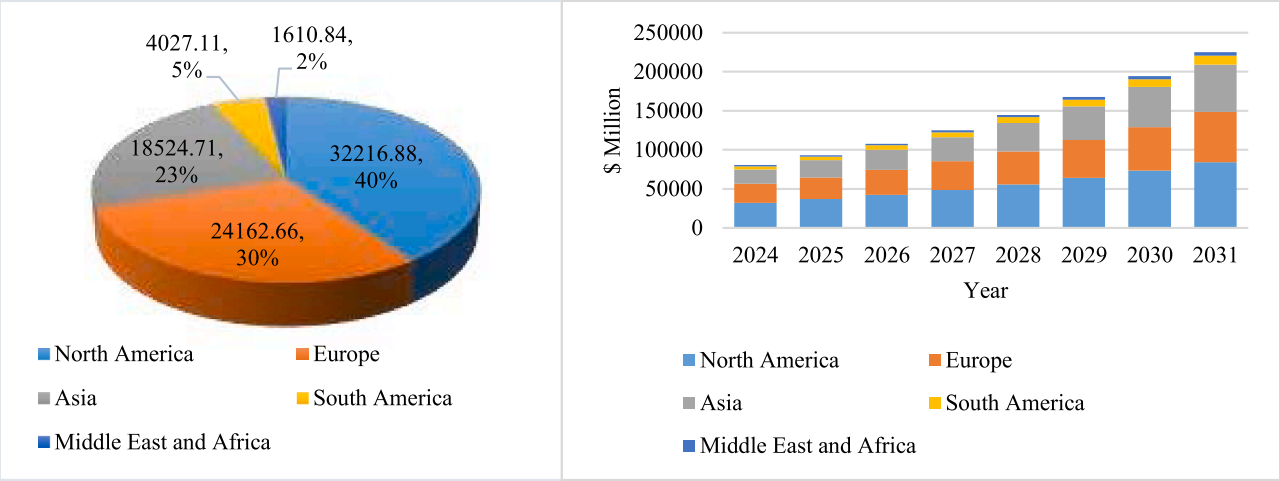


Fig. 1. The global CMg market size by region (Source: www.cognitivemarketresearch.com, ID: CMR440152).

through digital transformation and cloud technology adaptation. Meanwhile, many developing countries are also leveraging the benefits of CMg. For example, Mexico increased its market share to $2931.74 million in 2024, accounting for 3% of the global market, and it is projected to grow at a CAGR of 15.2% by 2031[2]. It has become increasingly crucial to efficiently manage and plan MSs to handle multiple tasks as CMg continues to expand [6].

In CMg, task scheduling and service allocation (TSSA) refer to the process of assigning multiple interdependent tasks to geographically distributed MSs, while determining their execution sequence to meet system-level performance objectives [7]. However, real-world manufacturing environments are dynamic, requiring scheduling approaches that can adapt to unpredicted events such as new task arrivals, service failures, and cancellations [8]. These challenges have led to the development of Dynamic TSSA (DTSSA), which aims to enable real-time adaptation to such disruptions. Among these, task cancellation has received limited attention, despite its potential to cause resource waste and operational inefficiencies if not properly managed [9]. This underscores the importance of adaptive rescheduling mechanisms capable of responding effectively to such disturbances.

The purpose of this paper is to examine how task cancellation affects CMg systems using an event-driven scheduling strategy. As a primary contribution of this work, we aim to minimize the maximum completion time for the DTSSA problem in CMg with task cancellation in mind. In order to mitigate the effects of task cancellation, the system updates its status in real time using IoT-based real-time monitoring, and it develops an optimization problem that dynamically reallocates services and reschedules tasks. Furthermore, the model takes into account not only different task arrival but also the logistics time between service providers (SPs) and the logistics time for delivering tasks to customers. Since logistics delays and fluctuations in task arrivals directly affect task completion times, they are treated as key factors in the task allocation and scheduling process. By incorporating real-time data, the dynamic TSSA model prioritizes these influential factors and adapts scheduling decisions accordingly. To effectively handle the complexity and dynamic behavior of this scheduling problem, an adaptive optimization approach that combines learning and search capabilities is required.

Metaheuristic algorithms are widely used to solve NP-hard problems due to their ability to efficiently explore large solution spaces and provide high-quality solutions within reasonable computational time. In recent years, there has been growing interest in combining reinforcement learning (RL) with metaheuristics, as RL can enhance adaptive decision-making during the search process [10]. Motivated by this trend, this paper proposes an RL-enhanced metaheuristic framework for addressing the challenges of dynamic scheduling in cloud manufacturing. The detailed mechanisms and components of the proposed algorithm are described in Sections 4.

To validate the mathematical model, small-sized test problems are solved using GAMS software with the CPLEX solver. Moreover, a set of test instances with three different sizes is generated to comprehensively evaluate the performance of the proposed algorithm. The Taguchi method is employed to systematically tune the parameters of the metaheuristic algorithms. The proposed algorithm is compared not only with four classical metaheuristics but also with three recently developed algorithms. Besides, statistical techniques such as analysis of variance (ANOVA) and Fisher individual test are utilized to analyze the results. Accordingly, the main contributions of this study are summarized as follows:

- A new Mixed Integer Linear Programming (MILP) model is developed for DTSSA in CMg manufacturing systems, explicitly addressing the challenges associated with task cancellations (DTSSA-TC).
- The proposed model incorporates realistic factors, including task arrival times, inter-service provider logistics, and final delivery time, enabling more accurate and practical scheduling and rescheduling solutions.
- A new combination of the Q-learning algorithm with the genetic algorithm (QLGA) is proposed to identify the best solutions.
- The proposed algorithms were evaluated using both benchmark instances from the literature and example cases inspired by existing studies.
- The proposed metaheuristic algorithm is evaluated not only against conventional metaheuristics but also against QL-enhanced counterparts to demonstrate its effectiveness.

The rest of this paper is structured as follows: the next section reviews the related literature. In Section 3, the scheduling and rescheduling mathematical models are developed. Section 4 describes the solution methodologies. The experimental evaluation, sensitivity analysis, and managerial insights are provided in Section 5. Finally, the main findings and suggestions for further studies are presented in the last section.

2. Literature review

There has been increasing attention in recent years to developing

[2] https://www.cognitivemarketresearch.com/regional-analysis/north-america-cloud-manufacturing-market-report Retrieved on January 29, 2025.

effective operational strategies for improving CMg implementation [2,1]. Among its core components, scheduling plays a critical role and has been reviewed in several studies, highlighting key trends and challenges [11,7,12]. Notably, recent research has emphasized the importance of dynamic and event-driven scheduling approaches to better cope with uncertainties and real-time changes in CMg environments [13]. Accordingly, the following sections focus specifically on TSSA problems, including their dynamic extensions (DTSSA), and explore reinforcement learning-based metaheuristic approaches in this domain.

2.1. A review of TSSA in CMg

Numerous studies have addressed various aspects of TSSA in CMg systems, primarily due to their distributed and complex nature [4]. These investigations have explored a range of modeling frameworks, solution methodologies, and performance metrics to effectively manage the challenges associated with task scheduling under different conditions. One of the earliest studies in this domain was conducted by Liu et al [14], who evaluated a proposed scheduling rule using industrial case studies from the automotive parts manufacturing sector and compared its performance against a random scheduling rule.

As highlighted in the literature, CMg systems require particular attention to logistical considerations due to the geographical dispersion of services and tasks. Akbaripour et al [15] considered a Hub-and-spoke transportation network and designed some MILP problems for different task structures to optimize total cost, time, and the quality of services. The models were validated using the branch and cut method. In the same vein, Fazeli et al [16] and Laili et al [17] incorporated inter-provider logistics into their scheduling formulation to better reflect task transfers between distributed services.

Another prominent focus in the literature is customer satisfaction, reflecting the service-oriented nature of CMg environments. Some researchers have attempted to align scheduling decisions with user expectations by integrating factors such as task tardiness, response time constraints, and customer-specific requirements into their models. For instance, Vahedi-Nouri et al [18] formulated a task selection and scheduling model to maximize customer utility based on tardiness cost, while Valizadeh et al [19] proposed a scheduling framework for customized dental implant manufacturing with an emphasis on reducing service response time. In parallel, energy consumption has emerged as a sustainability-driven objective in recent formulations. A number of studies have jointly considered logistics and energy-related indicators, aiming to reduce both manufacturing and transportation energy usage as part of a comprehensive optimization strategy [20,21].

Despite the growing literature on TSSA, some critical aspects such as task arrival time and final logistics have received limited attention. A few studies have considered task arrival variability and its influence on scheduling decisions [22,23] while only limited efforts have addressed final logistics and its integration into the scheduling process [15,24]. This gap is notable, given the strong influence of these factors on service responsiveness, completion time, and overall system performance. Moreover, most existing research adopts static scheduling assumptions, which fail to reflect the dynamic and event-driven nature of CMg systems. Accordingly, the following subsection provides a focused review of DTSSA.

2.2. A review of DTSSA in CMg

Services and tasks within CMg are often dynamic and uncertain. To address these issues, dynamic scheduling strategies have been developed to adapt to real-time system conditions. Dynamic scheduling approaches in literature are generally classified into three categories: completely reactive, proactive, and predictive-reactive. Completely reactive scheduling relies on local decision-making and real-time dispatching rules, but it cannot ensure global performance due to its immediate responses [25]. In contrast, proactive scheduling anticipates potential disruptions using available information, though its success depends on prediction accuracy [26]. In predictive-reactive approach, widely adopted in modern manufacturing systems, an initial schedule is generated and then updated in response to disturbances [27].

These strategies typically rely on IoT-enabled infrastructure, allowing tasks to be reallocated to available services and rescheduled based on the current system status [28]. This enables the scheduling system to respond effectively to unexpected events, such as changes in task requirements, new task arrivals, task cancellations, machine breakdowns, and material shortages [8].

Within the DTSSA literature, several studies have focused on task-related events. For instance, Zhou et al [29] proposed an event-triggered scheduling strategy to minimize execution time in response to dynamic task arrivals. Yang et al [30] have driven the further development by considering subtask changes. They used a predictive-reactive scheduling model as well as a metaheuristic algorithm to find the best scheduling and rescheduling solutions. Similarly, Mahmoodjanloo et al [31] applied a predictive-reactive approach and proposed two mixed-integer linear programming (MILP) models to manage scheduling and rescheduling in response to newly arriving tasks.

Other researchers have concentrated on service-related events. Zhu et al [32] developed a dynamic scheduling model for a distributed network, where rescheduling was used to prevent substandard products from advancing through the production stages. Xiong et al [33] investigated exceptions such as task requirement changes, service failures, and multiple task assignments to a single service. In another study, Wang et al [34] considered service capacity variations and applied a completely reactive reinforcement learning-based scheduling algorithm. Similarly, Sun et al [35] recently addressed the challenge of random task arrivals in additive manufacturing environments using a reactive scheduling approach.

A particularly critical yet underexplored challenge in DTSSA is task cancellation. Inadequate handling of task cancellations can disrupt the scheduling process, resulting in production inefficiencies, resource waste, and increased energy consumption [9]. Rapid rescheduling of canceled tasks enables better resource utilization and prevents inefficiencies by redirecting allocated services to other tasks [36].

Table 1 provides a summary of abbreviations used for solution methods in the reviewed papers. In addition, Tables 2 and 3 provide a summary of the model characteristics and solution approaches adopted in the literature for addressing TSSA and DTSSA problems in CMg systems. In summary, the DTSSA problem represents an extension of the classical TSSA framework to accommodate the dynamic and uncertain conditions of real-world CMg environments. While both aim to optimize resource allocation and minimize completion time, the presence of dynamic events such as task cancellations significantly increases the complexity of the scheduling problem. Addressing these challenges will require the development of high performance and intelligent solutions.

2.3. RL-based metaheuristics algorithm in scheduling problems

In distributed scheduling problems, mathematical programming methods often cannot provide solutions within a reasonable amount of time because of their NP-hard nature [19]. As outlined in Tables 1 and 2, much of the work on scheduling and allocation problems in CMg have focused on utilizing metaheuristic algorithms. While most CMg scheduling studies have relied on conventional metaheuristics, recent research demonstrates growing interest in integrating reinforcement learning to enhance search performance [10] as further illustrated in Fig. 2.

RL has been applied to scheduling problems primarily through two paradigms. The first group is learning dispatching or priority policy in dynamic environments which often aims to replace traditional rule-based scheduling with data-driven decisions and typically requires modeling high-dimensional state-action spaces [41,50]. The second

Table 1
The abbreviations of algorithms in Tables 3 and 4.

Algorithms	Abbreviations	Algorithms	Abbreviations
Differential evolution	DE	Scatter search	SS
Particle swarm optimization	PSO	Simulated annealing	SA
Biogeography-based optimization	BBO	Genetic algorithm	GA
Chaos optimization algorithm	COA	Artificial bee colony	ABC
Teaching-learning based optimization	TLBO	Social spider optimization	SSO
Variable neighborhood search	VNS	Ant colony optimization	ACO
Non-dominated neighbor immune algorithm	NNIA	Gray wolf optimizer	GWO
Strength pareto evolutionary algorithm 2	SPEA2	Harris hawks optimizer	HHO
Multi-objective firefly algorithm	MOFA	Equilibrium optimizer	EO
Multi-objective gray wolf optimizer	MOGWO	Iterative greedy algorithm	IGA
Multi-objective evolutionary algorithm	MOEA	Memetic algorithm	MA
Non-dominated sorting genetic algorithm II	NSGA-II	Evolutionary algorithm	EA
Multi-objective particle swarm optimization	MOPSO	Invasive weed optimization	IWO
Non-dominated sorting genetic algorithm III	NSGA-III		

paradigm, which forms the foundation of this study, employs RL internally to adapt components of the search process, such as operator selection, within metaheuristic frameworks [51].

For instance, Yu et al [52] proposed a Q-learning-based metaheuristics to solve a distributed flow-shop scheduling problem, in which the RL agent adaptively selected local search operators from artificial bee colony, particle swarm optimization, genetic algorithm, and the Jaya algorithm. This operator-level integration of RL has also been adopted in several other scheduling contexts [53,54,51,55,56]. In addition to operator selection, some studies used RL to dynamically adjust the parameters of search operators, such as crossover and mutation rates, or reinitialization frequencies [57–59]. Table 4 summarizes recent studies that integrate Q-learning into metaheuristic algorithms for scheduling.

While RL-based metaheuristic algorithms have been successfully applied to various scheduling problems, they have not yet been explored in the context of CMg scheduling problems. In our approach, Q-learning is used to adapt operator selection within the optimization process. Unlike RL-based priority rule approaches (Table 3), our method defines the state and action space based on the internal status of the optimization process, such as operator performance, rather than problem-specific variables like the number of jobs or machines. As a result, the state-action space remains small and does not scale with the problem size, enabling more efficient and stable learning.

2.4. Research gaps

Global attention has recently increased to collaboration in a

Table 2
Overview of model characteristics in existing literature on TSSA and DTSSA in CMg.

Papers	DTSSA	CMg	Performance measure	Task arrival	Middle logistic	Final logistic	Dynamic event
[14]	-	✓	Time, cost, reliability	✓	✓	-	-
[15]	-	✓	Time, cost, quality		✓	✓	-
[37]	-	✓	Completion time	-	-	-	-
[38]	-	✓	Task load, provider efficiency, task reliability, IoT matching	-	-	-	-
[16]	-	✓	Completion time, cost	-	✓	-	-
[17]	-	✓	Completion time, cost	-	✓	-	-
[18]	-	✓	Customer utility, factory utility	-	-	-	-
[19]	-	✓	Completion time	-	-	-	-
[39]	-	✓	Completion time	-	✓	-	-
[40]	-	✓	Completion time	-	-	-	-
[41]	-	✓	Time, cost, reliability	-	✓	-	-
[36]	-	✓	Makespan, cost, quality	✓	✓	-	-
[23]	-	✓	Completion time, reliability, cost	✓	✓	-	-
[22]	-	✓	Makespan, cost, reliability	✓	✓	-	-
[42]	-	✓	Time, cost	-	✓	-	-
[43]	-	✓	Makespan	-	✓	-	-
[21]	-	✓	Time, cost, reliability, energy	-	✓	-	-
[24]	-	✓	Cost	-	✓	✓	-
[20]	-	✓	Customer satisfaction, energy	-	✓	-	-
[44]	-	✓	Completion time, cost, reliability	-	✓	-	-
[29]	✓	✓	Execution time	✓	✓	-	Dynamic task arrival
[45]	✓	✓	Time, cost, reliability, stability, robustness	-	-	-	Service interruption
[46]	✓	✓	Time, cost, quality, capability	-	✓	-	Requirements deviation
[30]	✓	✓	Time and cost	-	✓	-	Process time & cost change
[47]	✓	✓	Time, cost, schedule reliability	-	✓	-	New task
[48]	✓	✓	Makespan	-	✓	-	Service change
[31]	✓	-	Total lateness	-	✓	-	New task
[32]	✓	-	Makespan, energy	-	✓	-	Inspection result
[33]	✓	✓	Total time, cost	-	✓	-	Task & service change
[13]	✓	✓	Makespan, cost, quality, stability	✓	✓	-	New task
[49]	✓	✓	Makespan, cost	-	✓	-	Service capacity change
[9]	✓	-	Makespan, energy consumption	-	✓	-	Task cancellation
[50]	✓	-	Average tardiness	✓	✓	-	New task
[34])	✓	-	Makespan, energy consumption	-	✓	-	Machine breakdown
[35]	✓	✓	Tardiness	✓	-	-	Dynamic task arrival
This work	✓	✓	Makespan, stability	✓	✓	✓	Task cancellation

Table 3
Overview of solution approaches adopted in TSSA and DTSSA studies in CMg.

Papers	Exact	Metaheuristic/Heuristic	RL-based	Industrial example
[14]	-	Workload-based priority rule	-	Automotive parts
[15]	CPLEX	-	-	Motorcycle parts
[37]	-	ACO	-	-
[38]	-	COA	-	-
[16]	-	SSO, GA, SA	-	-
[17]	-	MOEA, NSGA-II, NSGA-III	-	Electronic product
[18]	CPLEX	NSGA-II, MOGWO, MOPSO, MOFA	-	-
[19]	CPLEX	PSO	-	Dentistry implant
[39]	-	NSGA-II, SPEA2	-	Large cement equipment
[40]	Gurobi	PSO, GA, IGA	-	-
[41]	-	-	RL-based priority rule	Robotic services
[36]	CPLEX	GA	-	-
[23]	-	EA	-	-
[22]	-	-	RL-based priority rule	Automobile engine parts
[42]	-	MOPSO, MOGWO, NNIA, MOEA, NSGA-II	-	-
[43]	-	VNS, GA, ACO, SA	-	-
[21]	-	NSGA-II	-	Iron and steel company
[24]	-	GA, ACO	-	-
[20]	Gurobi	MOEA, NSGA-II, MOPSO, TLBO	Ql-based heuristic	-
[44]	-	-	RL-based priority rule	-
[29]	-	Event-triggered heuristic	-	CNC production
[45]	-	GA, PSO	-	-
[46]		Hybrid of GA and ACO		
[30]	-	GA	-	-
[47]	-	BBO, GA, GWO, PSO	-	-
[48]	-	-	RL-based priority rule	Aircraft part
[31]	CPLEX	Hybrid of EO and VNS, DE	-	-
[32]	-	MA, NSGA-II, NSGA-III, NNIA	-	-
[33]	-	MOWOA, HHO, MOPSO, NSGA-II	-	Auto part industry
[13]	-	GA, PSO, GA	-	-
[49]	-	-	RL-based priority rule	Automotive parts
[9]	-	MA, NSGA-II, NSGA-III, NNIA	-	-
[50]	-	-	RL-based priority rule	-
[34]	-	NSGA-III, EA	-	-
[35]	-	-	QL-based priority rule	3D printing services
This work	CPLEX	GWO, GA, SA, SEO, HHO	QL-based GA	-

networked, cloud-based manufacturing system as Industry 4.0 and advanced technologies become more prevalent. By designing an efficient dynamic scheduling model, the system can not only address key performance metrics such as completion time but also flexibly respond to unforeseen events in dynamic environments.

Although a significant body of research has focused on scheduling under static conditions, such models inherently fail to capture the impact of dynamic events, including task cancellation. Furthermore, as discussed in Subsections 2.1 and 2.2, while some studies have considered task arrival times and final logistics, these aspects remain insufficiently modeled in both static and dynamic contexts, despite evidence of their influence on scheduling performance. Thus, there is a clear need for DTSSA models that explicitly address these challenges and incorporate realistic assumptions, such as task arrival times, delivery constraints, and final logistics stages.

From a methodological standpoint, most scheduling solutions rely on metaheuristic algorithms, which are effective for large-scale problems but typically lack learning capabilities (Table 3). This limitation can reduce their ability to adapt in dynamic environments. While some recent works (reviewed in Subsection 2.4) have integrated RL into metaheuristics, such efforts have not been applied to CMg problems. In this way, new algorithms for DTSSA problems can be developed based on the perspective of the solution methodology. To address these gaps, this study proposes the following research contributions to address this research gap:

- This study introduces novel MILP models for the DTSSA with task cancellation.
- This problem not only incorporates logistics between SPs and task delivery time to customers but also considers dynamic task arrivals, making it more realistic.
- A search-dependent Q-learning mechanism is applied for operator selection in GA.
- A domain adaptation strategy is explored to support generalization across problem instances.

3. Problem statement and formulation

In the following sections, a detailed description of the proposed model is provided, along with a mathematical formulation of scheduling and rescheduling problems.

3.1. Problem definition

CMg is primarily known for its ability to enable the effective sharing of manufacturing resources through the integration of a wide range of resources, including machines, tools, and software from a variety of vendors. A CMg system's performance depends on allocating tasks to the right services and scheduling subtasks to minimize maximum completion time. Consumers can easily access these services on demand. The following sections are considered as part of the CMg platform for the TSSA problem as illustrated in Fig. 3.

It is considered that the CMg platform includes m distributed service providers ($K = \{SP_1, SP_2, \ldots, SP_k, \ldots, SP_m\}$), which have already been deployed in different geographical areas. Each service provider offers one or more manufacturing services. For example, the k-th service provider (SP_k) offers q_k services ($R_k = \{MS_{k1}, MS_{k2}, \ldots, MS_{kr}, \ldots, MS_{kq_k}\}$). MSs can execute multiple specific operations defined by the CMg system. All MSs and their capabilities will be encapsulated to determine the particular operations they can perform and the associated time.

On the other hand, consumers submit their requirements (manufacturing tasks) as standard electronic data, enabling the cloud system to interpret the inputs automatically. The set of requirements contains n independent tasks denoted as $I = \{T_1, \ldots, T_i, \ldots, T_n\}$. Each task is decomposed into multiple interdependent subtasks structured sequentially. The set of subtasks task i (T_i) contains p_i subtasks and denoted as $J_i = Sub_{i1}, Sub_{i2}, \ldots, Sub_{ij}, \ldots, Sub_{ip_i}\}$, where Sub_{ij} denotes j-th subtask of the task i. The CMg system matches the product requirements and service capabilities to determine the candidate services proper for each subtask and estimated processing times. Each subtask is supposed to be performed at least by one manufacturing service.

The main decisions of this section consist of two main sub-problems: allocating subtasks to the candidate services and scheduling subtasks on the allocated machines with a minimum maximum completion time. It is worth noting that task completion time is supposed to consider the processing time, the logistics between service providers, and the final logistics for delivering the completed task to the customers. Besides, the tasks aren't available at the beginning of the schedule and arrive at

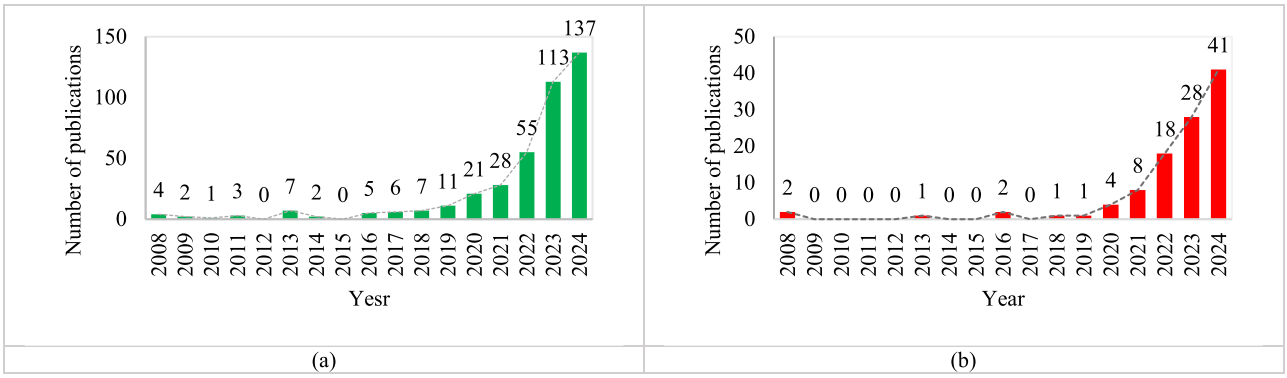


Fig. 2. Trend of publications in: (a) RL-based metaheuristics and (b) RL-based metaheuristics in scheduling problems (Source: Scopus).

Table 4
Overview of Q-learning applications operator management in metaheuristic for scheduling problems.

Paper	SE	Metaheuristic(s)	Action	Description
[57]	FS, JS	GA	Operator parameter selection	Crossover and mutation rate
[52]	DFS	ABC, PSO, GA	Operator selection	Local search of jobs and factories
[60]	FS	IGA	Operator parameter selection	Perturbation operator
[61]	DL	VNS	Operator selection	Swap, double swap, inverse, insertion, bind insertion, block insertion
[62]	DFS	PSO	Operator selection	Local search of factories
[54]	HFS	ABC	Operator selection	Insert, swap, pair swap, inverse swap, lot split mutation
[63]	JS	ABC	Operator selection	Local search of jobs and machines
[64]	HFS	MRLM	Operator selection	Swap, insertion, destruction–construction mutation, local search
[59]	FJS	NSGA-III	Operator parameter selection	Crossover and mutation rate
[53]	DFS	EA	Operator selection	Genetic operator, brainstorm operator, grey wolf operator
[58]	FJS	EA	Operator parameter selection	Reinitialization rate
[65]	JS	MA	Operator parameter selection	Crossover rate
[66]	FJS	MA	Operator selection	Local search of jobs and machines
[51]	HFFS	IGA	Operator selection	Insert, swap, inverse
[56]	DFS	SS, IWO, ABC	Operator selection	Local search of jobs and factories
[67]	PB	SA	Operator selection	Local search of jobs, operations, batches, and workloads
[55]	PM	IGA-PSO	Operator selection	Insert, swap, inverse
This work	CMg	GA	Operator selection	Pairwise swap, swap, inversion, one-point, two-point, and uniform crossover

*Scheduling environment (SE): Flow shop (FS), Distributed FS (DFS), Hybrid FS (HFS), Parallel machine (PM), Hybrid flexible flow shop (HFFS), disassembly line (DL), Job shop (JS), Parallel batch (PB), Flexible JS (FJS).

different times.

There are two types of scheduling problems: static and dynamic. The next subsection presents a mathematical model for a static scheduling problem. The static model is then extended for a dynamic scheduling problem in order to deal with the cancellation of tasks. Dynamic scheduling is achieved by using an event-driven rescheduling strategy, which enables all unprocessed operations to be rescheduled after task cancellation, while maintaining the best schedule stability possible. By adding a penalty term to the dynamic model's objective function, deviations of service provider allocation will be controlled.

3.2. Problem suppositions

The assumptions of the model are listed as follows.

- It is assumed that all tasks are decomposed into sequential structures [68].
- At least one service is available for each subtask in the matching process.
- It is supposed that tasks are independent.
- Tasks are considered to arrive dynamically and at different times.
- All services are available at the beginning of the scheduling process.
- The logistics time between service providers and delivery time to customers are considered.
- The manufacturing processes cannot be interrupted unless the task has been canceled.

3.3. Static Scheduling Model

In this subsection, we develop the formulation of the static scheduling problem. The static problem optimizes the allocation of services and the scheduling of tasks without taking cancellation into account. The model notations are provided in Tables 5-7.

3.3.1. Objective function

In manufacturing systems task completion time is one of the crucial performance metrics for improving system efficiency, ensuring timely delivery, and reducing operational costs [69,36,70]. Eq. (1) illustrates the objective function as minimizing maximum completion time.

$$\min Z = C_{max} \tag{1}$$

3.3.2. Constraints

Following are the constraints that ensure that the optimization model operates within a feasible solution space, adhering to the logical and practical constraints of task allocation, task arrival times, sequencing, and logistics.

$$Y_{ijkr} \le S_{ijkr}, \ \forall i \in I, j \in J_i, k \in K, r \in q_k \tag{2}$$

$$\sum_{k=1}^{m} Z_{ijk} = 1, \ \forall i \in I, j \in J_i \tag{3}$$

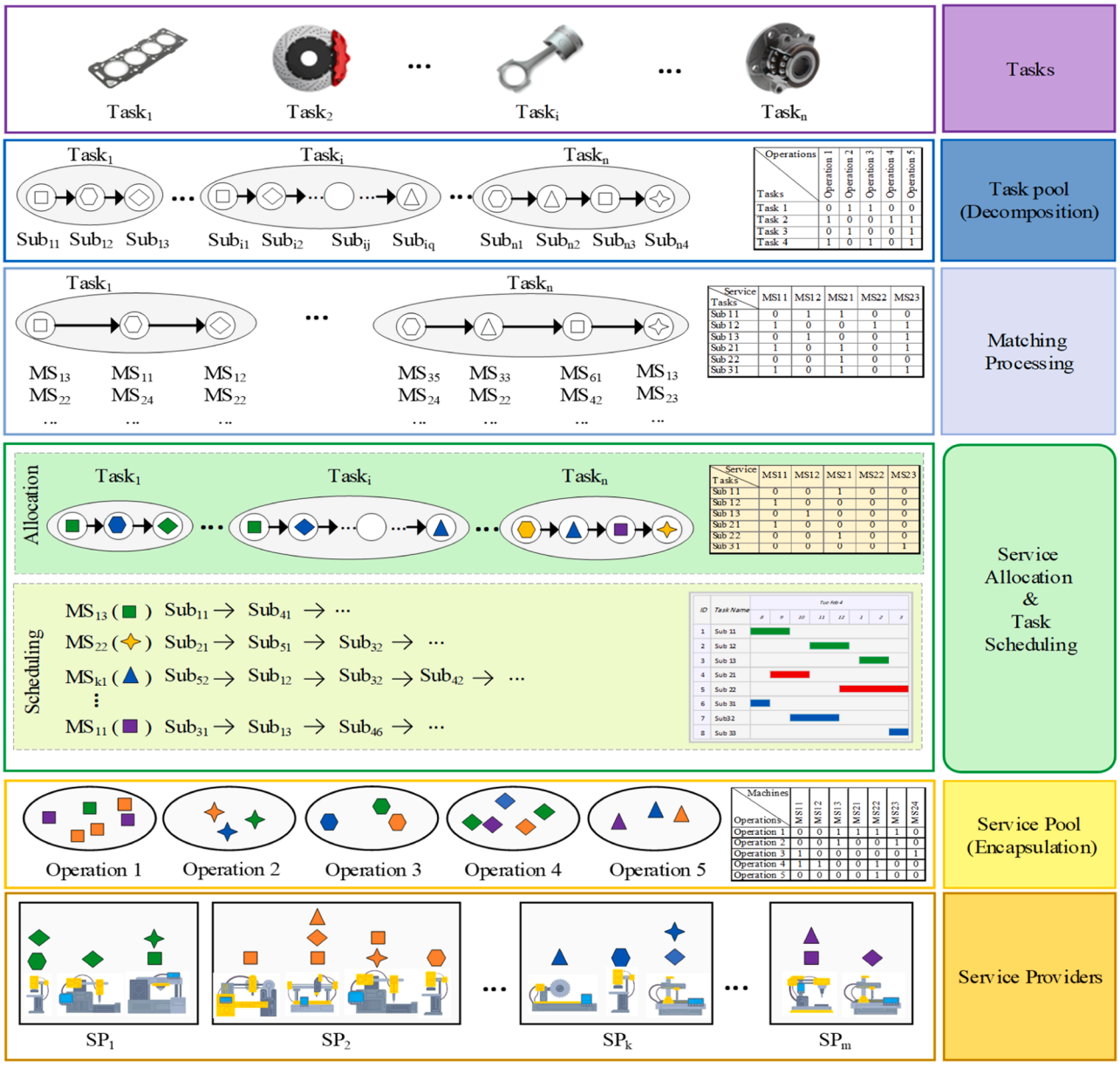


Fig. 3. The framework of the TSSA problem in the CMg platform.

Table 5
Sets and indices of the scheduling model.

Sets and indices	Description
$i,i' = 1,2,\ldots,n$	Indices of tasks/customers
$k,k' = 1,2,\ldots,m$	Indices of service providers/factories,
$j,j' = 1,2,\ldots,p_i$	Indices of subtasks of T_i
$r,r' = 1,2,\ldots,q_k$	Indices of manufacturing services/machines of SP_k
$I = \{T_1,\ldots, T_i,\ldots, T_n\}$	Set of tasks/customers
$K = \{SP_1, SP_2,\ldots, SP_k,\ldots, SP_m\}$	Set of service providers/factories
$J_i = \{Sub_{i1}, Sub_{i2},\ldots, Sub_{ij},\ldots, Sub_{ip_i}\}$	Set of subtasks of T_i
$R_k = \{MS_{k1}, MS_{k2},\ldots, MS_{kr},\ldots, MS_{kq_k}\}$	Set of services/machines of SP_k

Table 6
Parameters of the scheduling model.

Parameters	Description
n	Total number of tasks
m	Total number of service providers
p_i	Total number of subtasks of T_i
q_k	Total number of MSs of SP_k
At_i	Arrival time of T_i
P_{ijkr}	Processing time of Sub_{ij} on MS_{kr}
$TSS_{kk'}$	logistics time between SP_k and $SP_{k'}$; 0 if $k = k'$
TSC_{ki}	logistics time between SP_k and location of T_i
A_{ijkr}	Availability of MS_{kr} for Sub_{ij}; 1, if MS_{kr} can execute Sub_{ij}; otherwise, 0.
M	A big positive number

Table 7
Variables of the scheduling model.

Variables	Description
C_{max}	Maximum completion time
Ct_i	Completion time of T_i
S_{ij}	Start time of Sub_{ij}
F_{ij}	Finish time of Sub_{ij}
$\gamma_{ijkk'}$	Equal one if Sub_{ij} transported from SP_k to $SP_{k'}$; otherwise 0.
Y_{ijkr}	Equal one if Sub_{ij} allocated to MS_{kr}; otherwise 0.
Z_{ijk}	Equal one if Sub_{ij} allocated to SP_k; otherwise 0.
$X^{kr}_{iji'j'}$	Equal one if Sub_{ij} and $Sub_{i'j'}$ both processed on MS^r_k and Sub_{ij} proceeds $Sub_{i'j'}$; otherwise 0.

$$\sum_{r=1}^{q_k} Y_{ijkr} = Z_{ijk},\ \forall i \in I, j \in J_i, k \in K \tag{4}$$

$$X^{kr}_{i'j'ij} + X^{kr}_{iji'j'} \le Y_{ijkr},\ \forall i, i' \in I, j, j' \in J_i, k \in K, r \in R_k, (i \ne i') \tag{5}$$

$$X^{kr}_{i'j'ij} + X^{kr}_{iji'j'} + 1 \ge Y_{ijkr} + Y_{i'j'kr},\ \forall i, i' \in I, j, j' \in J_i, k \in K, r \in R_k, (i \ne i') \tag{6}$$

$$S_{i'j'} \ge S_{ij} + P_{ijkr} - M\left(1 - X^{kr}_{iji'j'}\right),\ \forall i, i' \in I, j, j' \in J_i, k \in K, r \in R_k, (i \ne i') \tag{7}$$

$$Y_{i(j-1)kr} + Y_{ijk'r'} - 1 \le \gamma_{ijkk'},\ \forall i \in I, j \in J_i (j > 1), k, k' \in K, r \in R_k \tag{8}$$

$$Y_{i(j-1)kr} \ge \gamma_{ijkk'},\ \forall i \in I, j \in J_i\ (j > 1), k, k' \in K, r \in R_k \tag{9}$$

$$Y_{ijk'r'} \ge \gamma_{ijkk'},\ \forall i \in I, j \in J_i\ (j > 1), k, k' \in K, r \in R_k \tag{10}$$

$$S_{ij} \ge At_i,\ \forall i \in I, j \in J_i\ (j = 1) \tag{11}$$

$$S_{ij} \ge F_{i(j-1)} + \sum_{k=1}^{m}\sum_{k'=1}^{m} \gamma_{ijkk'}\ TSS_{kk'},\ \forall i \in I, j \in J_i\ (j > 1) \tag{12}$$

$$F_{ij} \ge S_{ij} + \sum_{k=1}^{m}\sum_{r=1}^{q_k} Y_{ijkr} P_{ijkr},\ \forall i \in I, j \in J_i \tag{13}$$

$$Ct_i \ge F_{ip_ikr} + \sum_{k=1}^{m}\sum_{r=1}^{q_k} Y_{ip_ikr} TSC_{ki},\ \forall i \in I, k \in K, r \in R_k \tag{14}$$

$$C_{max} \ge Ct_i,\ \forall i \in I \tag{15}$$

$$X^{kr}_{jij'i'}, Y_{ijkr},\ \gamma_{ijkk'} \in \{0, 1\},\ \forall i \in I, j \in J_i, k \in K, r \in R_k \tag{16}$$

$$Z_{ijk} \in \{0, 1\},\ \forall i \in I, j \in J_i, k \in K \tag{17}$$

$$F_{ij}, S_{ij} \ge 0,\ \forall i \in I, j \in J_i \tag{18}$$

$$Ct_i \ge 0,\ \forall i \in I \tag{19}$$

Constraints (2) to (4) define the allocation constraints, ensuring that each subtask is assigned to exactly one service provider and one manufacturing service. These constrains prevent overlapping assignments and guarantee a well-structured task distribution. Constraints (5) to (7) establish the precedence relationships among subtasks on the same machine. Thes Eq.s indicate that when multiple subtasks are allocated to the same service only one sequence can be accommodated. Constraints (8) to (10) consider the logistics of subtask Sub_{ij} from service provider SP_k to $SP_{k'}$. These constraints account for the transportation time required between providers, which directly impacts the completion time of the task. By incorporating logistics into the model, realistic execution times are ensured. Constraint (11) represents that the start time of the first subtask (Sub_{i1}) cannot be earlier than the task arrival time. In other words, no subtask can begin before the task itself is available for processing. Constraint (12) state that for subtasks beyond the first one ($j > 1$), the start time must be at least as large as the finish time of the preceding subtask ($j - 1$), plus the required logistics time for transferring the subtask from service provider SP_k to $SP_{k'}$. This constraint ensures that the execution sequence is respected and that any necessary transportation time between providers is properly considered. Constraint (13) determines the finish time of subtask Sub_{ij}, which is equal to its start time plus the processing time required for execution. Constraint (14) indicates the completion time of each task. A task is considered completed only when its final subtask has finished execution and the final product has been delivered to the customer. This constraint plays a crucial role in minimizing the makespan of the entire process. Constraint (15) calculates the maximum completion time. Finally, Constraints (16) to (19) characterize the type of decision variables, specifying whether they are binary, integer, or continuous.

3.4. Rescheduling Model

The proposed model was developed in this subsection for use in a dynamic environment. The dynamic event of task cancellation in a production system would result in a suboptimal schedule. Therefore, rescheduling is an effective method of adjusting the schedule dynamically and utilizing the available time slots efficiently. By incorporating real-time adjustments, the system can improve overall productivity and minimize idle time caused by unexpected task cancellations. As a means of addressing this challenge, this paper proposes a reactive and event-driven rescheduling strategy for CMg tasks. To extend the rescheduling model, new parameters and sets are introduced, as presented in Tables 8 and 9.

It is considered that tasks are canceled at the time $t^{(c)}$ and the reactive time ($t^{(r)}$) of the scheduling system is after responding time (Δt) of service providers as they may have different conditions to manage their systems. Thus, the rescheduling process begins at $t^{(r)} = t^{(c)} + \Delta t$, ensuring that service providers have sufficient time to assess the new situation and reallocate their resources accordingly. During this response period, providers may need to evaluate their current workload, adjust machine assignments, coordinate workforce availability, and manage material flow disruptions. Additionally, system updates and communication with other stakeholders are necessary to prevent conflicts and inefficiencies. By incorporating this responding time, the rescheduling model accounts for the practical constraints of dynamic production environments, leading to more feasible and efficient task reallocations. The current status of the shop floor is determined according to the data provided by IoT system analysis [30].

The rescheduling problem can be formulated as a MILP model. The objective function, given in Eq. (20), consists of two components. The first component aims to minimize the maximum completion time in the updated set I', ensuring efficient task execution in the rescheduled environment. The second component minimizes the penalty associated with deviation from the original service provider allocation determined in the static problem. This term promotes stability in service provider allocation by considering the previous optimized service provider allocation variable (Z^*_{ijk}) as a parameter in dynamic model.

$$\min Z = C'_{max} + \sum_{i \in I'}\sum_{j \in J'_i}\sum_{k \in K} \delta \left| Z_{ijk} - Z^*_{ijk} \right| \tag{20}$$

Table 8
New sets of rescheduling problem.

Sets	Description
I''	Set of canceled tasks $I'' \subseteq I$
I'	Set of current tasks $I' \subseteq I$
J'_i	Set of unprocessed subtasks of T_i
$J^{(u)}_i$	Set of first unprocessed subtasks of tasks in I'

Table 9
New parameters of rescheduling problem.

Parameters	Description
n'	Number of canceled task(s)
$t^{(c)}$	Task cancellation time
Δt	Responding time
$t^{(r)}$	Reactive time ($t^{(r)} = t^{(c)} + \Delta t$)
δ	Penalty of change in the service provider allocation
MT_{kr}	Earliest available time of machine k

To update the constraints in the initial problem, the sets I and J_i should be replaced by sets I' and J'_i, respectively. Furthermore, the start time of the first unprocessed subtask depends on both the finish time of the previous subtask and the earliest available time of the assigned machine (MT_{kr}). Moreover, if a task is canceled, work on it stops and the machine becomes available for another subtask. Constraint (21) calculates the value of the earliest available time of machine MS_{kr}. In addition, Constraint (22) considers the available time of machines at the start time of the first unprocessed subtasks.

$$MT_{kr} = \begin{cases} t^{(r)} & MS_{kr} \text{ is ideal in } t^{(r)} \text{ or executing subtasks in } I'' \\ \max\{t^{(r)}, Y_{ijkr} F_{ij^{(u)}-1}\} & MS_{kr} \text{ is executing } Sub_{ij^{(u)}-1} \text{ in } t^{(r)} \end{cases} \tag{21}$$

$$S_{ij} \geq MT_{kr} - M(1 - Y_{ijkr}),\ \forall i \in I', j \in J_i^{(u)}, k \in K, r \in R_k \tag{22}$$

To clarify the calculation of S_{ij} and MT_{kr}, consider a simple example in Fig. 4. In this example, there are two service providers, three machines, and four tasks which task 4 is canceled in $t^c = 115$. According to Constraint (21), the earliest available time of MS_{11}, MS_{21} and MS_{22} are 166, 166, and 215. Besides, Constraints (11) and (12), and Constraint (22) should be considered to determine the start time of the first unprocessed subtasks. The key constraint for calculating the start times of subtasks Sub_{22} and Sub_{32} is Constraint (22), while subtask Sub_{13} is defined by Constraint (12).

To accommodate the updated scheduling environment, the model revises Eq.s (2) to (19) by replacing previous tasks set I with I' and original subtasks' set J_i with J'_i. Furthermore, Eq.s (23) to (27) provide a linearized representation of Eq.s (20) to (22) and should be integrated into the existing set of constraints. Here, p_{ijk} and q_{ijk} are auxiliary variables used to linearize the stability section in the objective function.

$$\min Z = C'_{max} + \sum_{i \in I'} \sum_{j \in J'_i} \sum_{k \in K} \delta \left(p_{ijk} + q_{ijk}\right) \tag{23}$$

Constraints $(2) - (19),\ \forall i \in I', j \in J'_i, k, k' \in K, r \in R_k$

$$MT_{kr} \geq t^{(r)},\ \forall k \in K, r \in R_k \tag{24}$$

$$MT_{kr} \geq F_{ij-1^{(u)}} Y_{ijkr},\ \forall i \in I', j \in J_i^{(u)}, k \in K, r \in R_k \tag{25}$$

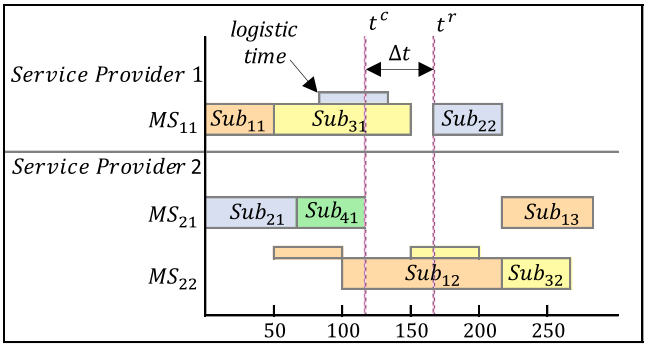


Fig. 4. Example of the first unprocessed subtasks in the rescheduling stage.

$$S_{ij} \geq MT_{kr} - M(1 - Y_{ijkr}),\ \forall i \in I', j \in J_i^{(u)}, k \in K, r \in R_k \tag{26}$$

$$Z_{ijk} - Z^*_{ijk} + p_{ijk} - q_{ijk} = 0,\ \forall i \in I', j \in J'_i, k \in K, r \in R_k \tag{27}$$

4. Solution methodology

The TSSA in CMg extends the classical job shop scheduling problem, which is classified as an NP-hard problem [71,72]. As the size of the problem increases, solving it using exact methods within polynomial time becomes impractical [20]. While these methods guarantee optimality, they are often limited by computational intensity, making them infeasible for large-scale problems [73]. Several works have shown that this challenge in TSSA problems can be overcome using metaheuristic algorithms to provide (near-) optimal solutions that balance solution quality and computational time [60,7,36,23].

Metaheuristic algorithms efficiently solve NP-hard combinatorial optimization problems by providing near-optimal solutions within a reasonable time [74]. Their flexibility allows them to handle diverse problems without extensive modifications, unlike exact methods [75]. By balancing intensification and diversification, they prevent premature convergence and offer a scalable approach for complex optimization where exact solutions are impractical [76]. The limitations of these algorithms, despite their advantages, include the absence of learning mechanisms and the limited use of past search information as a guide. The proposed model was solved with a new Q-learning-based genetic algorithm (QLGA) as well as some classical and recent metaheuristic algorithms.

As mentioned in Section 3.1, the problem environment is classified into static and dynamic scheduling problems. Therefore, the methodology consists of two stages: the first stage generates the initial scheduling scheme, while the second stage focuses on rescheduling. Accordingly, all applied metaheuristic algorithms are executed following the flowchart shown in Fig. 5.

4.1. Encoding and decoding plan

Encoding and decoding are essential components affecting the performance of metaheuristic algorithms [77]. During the encoding process, a problem is transformed into a mathematical representation that includes the objective function, constraints, and decision variables. Encoding properly ensures an accurate representation of the problem as well as an effective search space [78]. On the other hand, decoding translates the mathematical representation back into a form that can be understood and evaluated within the original context of the problem [79]. The effectiveness of metaheuristics heavily depends on these processes, making their careful design and implementation crucial for achieving high-quality solutions efficiently [80].

The Random Key (RK) representation is commonly used in permutation-based combinatorial problem [81]. RK representations represent chromosomes as vectors of random variables between 0 and 1. RK representations in this study are used to allocate subtasks to machines and to schedule subtasks on machines. Additionally, the number of genes in each chromosome corresponds to the number of subtasks. For instance, if there are three tasks with 3, 3, and 2 subtasks respectively, the chromosome would be represented by an 8-element vector (See Fig. 6).

As a result of the proposed TSSA problem, determining the sequence of tasks on machines is divided into two sub-problems. Firstly each gene maps to a feasible service allocation solution by Eq. (28) proposed by [23].

$$\beta_{ij} = round\left(\rho * \left(\sum_{k=1}^{m} \sum_{r=1}^{q_k} A_{ijkr} - 1\right) + 1\right) \tag{28}$$

Where ρ is the value of the gene, $\sum_{k=1}^{m} \sum_{r=1}^{q_k} A_{ijkr}$ is the total available

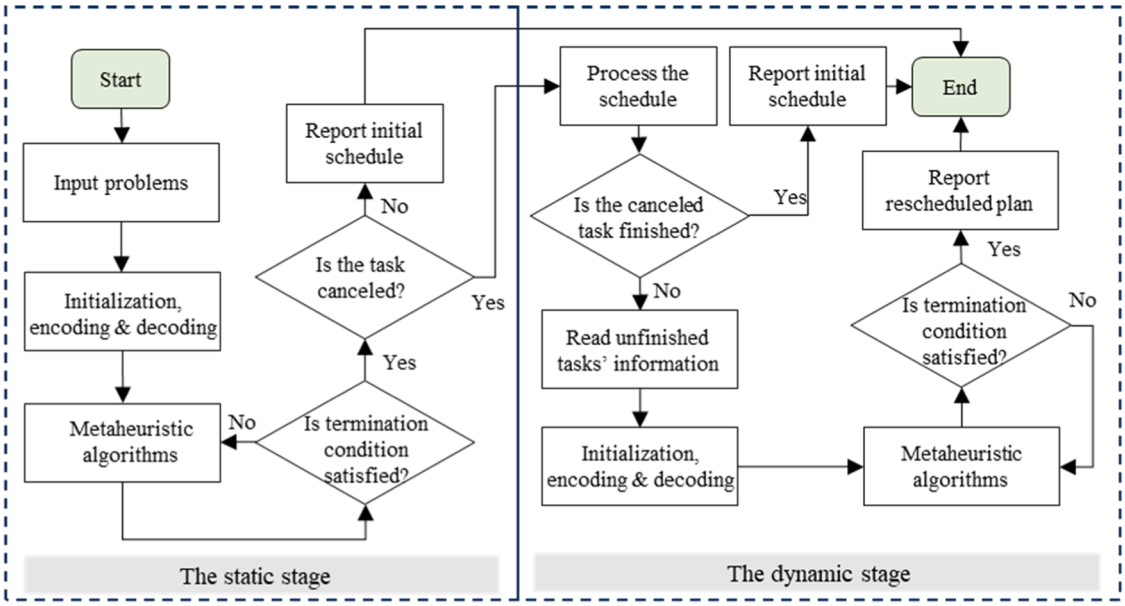


Fig. 5. The flowchart of static and dynamic stages of solution methodologies.

	Task 1			Task 2			Task 3	
	Sub_{11}	Sub_{12}	Sub_{13}	Sub_{21}	Sub_{22}	Sub_{23}	Sub_{31}	Sub_{32}
	0.098	0.278	0.574	0.127	0.913	0.632	0.615	0.906
Candidate services	1,4	3,5,6	2,4	2,3,6	1,3,4,5	3	1,4,6	1,3
AllocTij	1	2	2	1	4	1	2	2
selected service	m1	m5	m4	m2	m5	m3	m4	m3

Fig. 6. The representation of the random key method and decoding process.

services for Sub_{ij}, and β_{ij} represents the number of allocated services. As shown in Fig. 6, if four services are available for Sub_{22}, the fourth one will be allocated. The next step is to sequence the subtasks on allocated services. Since the structure of tasks is sequential, subtasks with higher priority are selected first. If multiple subtasks are allocated to a machine, the tasks with lower gene value (ρ) is selected and the remaining tasks are considered in the next sequencing phase. Then, tasks with subsequent priorities are scheduled until no tasks are left for schedule. Fig. 7 depicts the pseudocode of the decoding process.

4.2. Metaheuristics algorithms

As NP-hard scheduling problems become computationally impractical, metaheuristic algorithms are commonly used due to their flexibility and efficiency [82]. To benchmark the performance of the proposed method, both classical and modern algorithms were implemented, including Tabu Search (TS) [83], Simulated Annealing (SA) [84], Genetic Algorithm (GA) [85], Particle Swarm Optimization (PSO) [86], Gray Wolf Optimizer (GWO) [87], Harris Hawks Optimization (HHO) [88], and Social Engineering Optimizer (SEO) [89].

TS uses a memory-based tabu list to avoid revisiting recently evaluated solutions, while SA applies a probabilistic acceptance strategy that allows worse solutions in early stages to escape local optima [90,91]. In contrast, GA and PSO belong to population-based algorithms that evolve a set of solutions through genetic operators or swarm dynamics, enabling broad exploration [92,93]. Recent nature-inspired methods such as GWO, HHO, and SEO enhance the search process by using hierarchical coordination, dynamic exploration-exploitation strategies, or adaptive interactions between solutions [94,95]. These algorithms were implemented in their standard forms and used as baselines for comparison with the proposed QLGA framework.

4.3. Q-Learning algorithm

In contrast to metaheuristic algorithms, the Q-Learning algorithm combines model-free and policy-free Reinforcement Learning (RL) strategies [96]. The algorithm includes three main elements: Reward Tables, Q-tables, and the Bellman Eq. (Fig. 8). In each stage, a Q-learning agent's main task is to update its state by choosing an action with the highest Q-value. After each action, the agent receives a reward or penalty based on the generated result [97]. The Q-table is updated for the next step ($Q_{(t+1)}$) using the Bellman Eq. represented in Eq. (29).

$$Q_{(t+1)}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t\left(r_t + \gamma \underset{a}{max}(Q_t(s_{t+1}, a)) - Q_t(s_t, a_t)\right) \quad (29)$$

Where a_t is current action, s_t and s_{t+1} are the current and the next states respectively. The learning in the Q-Learning has two components: the discount factor (γ) and the learning rate (α_t), both within [0,1]. The discount factor determines the importance of future reward and the learning rate determines how fast new information is replaced by old information. The positive or negative reward is defined based on the improvement of the objective function value. If the current action improves the best solution, the agent gets positive reward (+1). Conversely, r_t is a negative penalty (-1) if the action worsens the best

```
Decoding of solution representation into a TSSA problem
Calling proposed solution and model parameters
Allocate candidate services to subtasks by equation (28)
TotalST<-indices of all subtasks
while ~isempty (TotalST)
    SST <- []
    m <- first unprocessed subtask of each task
    C <- indices of allocated machines to subtasks in m
    K1 <- machines in C with one subtask
    K2 <- machines in C with more than one subtask
    SST <- [SST subtasks of K1]
    for j = [indices of machines in K2]
        S <- subtask of j with minimum gene
        SST <- [SST S]
    end for
    for index= [indices of subtasks in SST]
        if index is the first subtask
            start time of index <- max (task arrival time, machine earliest available time)
            finish time of index <- Start time of index + processing time of index
            machine earliest available time<- finish time of index
        else
            Start time of index <- max (finish time of previous subtask +logistic between
            service providers, machine earliest available time)
            finish time of index <- Start time of index + processing time of index
            machine earliest available time<- finish time of index
        end if
    TotalST(index)=[];
    end for
end while
completion time <- finish time of last subtasks + delivery time to customers
```

Fig. 7. Decoding of solution representation into a TSSA problem.

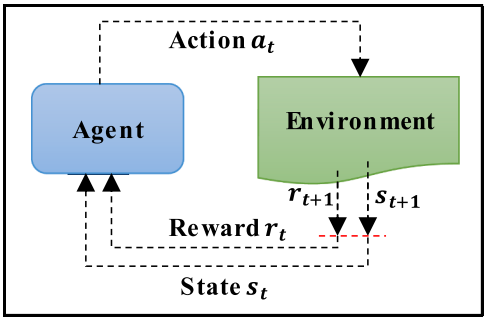


Fig. 8. The Q-learning algorithm architecture.

solution. Fig. 9 represents the pseudocode of the Q-Learning algorithm.

```
Initialize
Set the state S = [s_1, s_2, ..., s_n] and action A = [a_1, a_2, ..., a_n]
for each state and action
    Set Q(s_t, a_t) = 0
end for
Set initial state randomly
while stopping criterion is not met do
    Select the best a_t for the s_t from Q-table
    Execute a_t and get the immediate r_t using Equation (31)
    Get the maximum Q-value for the next state s_{t+1}
    Update q-table using Equation (29)
    Update the current state
end while
Return the updated Q(s_t, a_t)
```

Fig. 9. The pseudocode of Q_learning.

4.4. The proposed algorithm

In the developed metaheuristic algorithm (QLGA), the base algorithm is a classical GA with three mutation operators (pairwise, swap, and inversion) and three crossover operators (one-point, two-point, and uniform). Fig. 10 illustrates examples of applying crossover and mutation operators. In applied GA in the TSSA problem in CMg, a single mutation operator and a single crossover operator are consistently applied in all iterations [24,46,23]. In this study, the operator selection of QLGA is based on the Q-Learning algorithm. The Q-table is considered a 9×9 matrix in which rows represent the states and columns represent the action. Actions and states are initially selected randomly (i.e., for the first 50 iterations) to train the Q-learner. Then, the actions are determined according to the Q-table. The following subsections provide a detailed explanation of each component of the proposed algorithm.

4.4.1. Initialization phase

The initial population consists of randomly generated chromosomes, each encoding a feasible schedule of tasks to CMg resources. Each individual is evaluated using objective function. At this stage, the Q-table, a 9×9 matrix, is also initialized. This matrix is used to guide the operator selection process, where each row represents a state (combination of mutation and crossover operators used in the current generation), and each column represents an action (operator combination selected for the next generation). To ensure learning at early stages, the algorithm uses random action selection for the first few generations (e.g., 50 iterations), allowing the agent to explore diverse strategies and populate the Q-table with reward-based feedback.

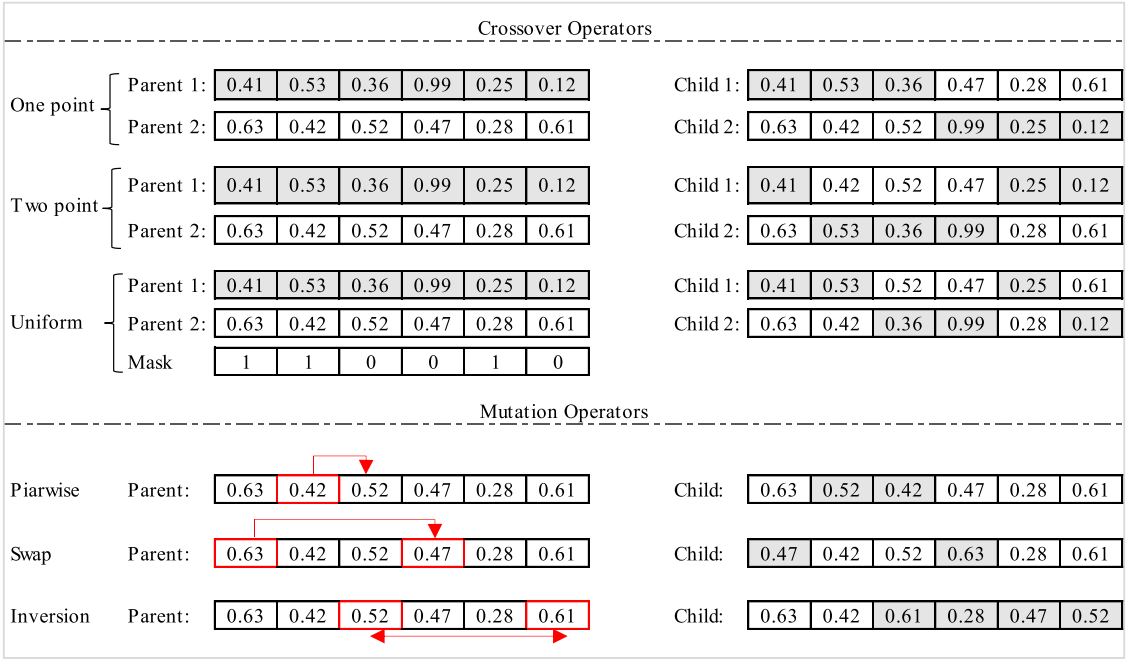


Fig. 10. Examples of applying crossover and mutation operators.

4.4.2. *Evolutionary phase*

The main evolutionary process proceeds in generations and includes selection, crossover, reproduction, and mutation. Unlike traditional genetic algorithms with fixed operators, the proposed algorithm employs a Q-learning agent that dynamically selects a pair of genetic operators (crossover and mutation) in each generation, based on historical performance. At each iteration, the current state (s_t) is defined based on the operator pair used in the current generation. Then, an action (a_t), a new operator pair, is selected using an ε-greedy strategy as shown in Eq. (30):

$$A = \begin{cases} nAction * rand() & rand < \varepsilon \\ \underset{a}{\text{argmax}}(Q_t(s_{t+1}, a)) & otherwise \end{cases} \tag{30}$$

After selecting the operators, the algorithm performs selection using the Roulette Wheel Selection method. This technique assigns selection probabilities proportional to the fitness of each individual, allowing higher-quality chromosomes to have a greater chance of being chosen as parents while preserving population diversity. Based on the predefined crossover probability (P_c), a proportion of the selected parents (i.e., $P_c \times N$, where N is the population size) undergo crossover using the chosen crossover operator to generate offspring. The remaining ($P_r = 1 - P_c$) portion of chromosomes are transferred to the next generation through reproduction based on an elitist strategy, where chromosomes with higher fitness values are directly copied. A reward r_t is assigned based on the fitness comparison in Eq. (31).

$$r_t = \begin{cases} +1 & \text{if } f_{new} < f_{best} \\ 0 & \text{if } f_{new} = f_{best} \\ -1 & \text{if } f_{new} > f_{best} \end{cases} \tag{31}$$

This reward is used to update the Q-table using the Bellman Eq.. Besides, to ensure convergence of the learning process over time, the learning rate α_t is progressively decreased using Eq. (32), where *maxit* denotes the total number of iterations and α_{t+1} the learning rate of the next iteration.

$$\alpha_{t+1} = \alpha_t - (\alpha_t / (maxit - 1)) \tag{32}$$

Fig. 11 illustrates the complete framework of the proposed reinforcement learning-based metaheuristic algorithm, which operates in static (left) and dynamic (right) scheduling stages. In the static stage, the initial schedule is generated using Q-learning to guide operator selection (crossover and mutation). If no cancellation occurs, the system proceeds to finalize the initial schedule. Once a cancellation event is detected, the system transitions to the dynamic stage, where the schedule is adapted in real time based on unfinished tasks. In both stages, a Q-learning mechanism supports adaptive operator selection via an ε-greedy strategy, allowing the system to balance exploration and exploitation based on the evolving environment.

4.4.3. *Q-Learning components in QLGA*

To enhance adaptability and learning capability in the dynamic CMg environment, the proposed algorithm integrates Q-learning into its metaheuristic algorithm. This integration transforms the operator selection process into a reinforcement learning task, where the Q-learning components are defined as follows:

Agent: The agent in the proposed framework is responsible for selecting the most appropriate pair of genetic operators (mutation and crossover) at each generation of the algorithm. Its goal is to improve solution quality over time by learning from the outcomes of previous operator choices. The agent uses the Q-table as a memory structure to store and update the expected rewards of different operator combinations under varying search conditions [51,55].

State: In RL-enhanced metaheuristics, state definitions are typically classified as search-dependent, problem-dependent, or instance-dependent [97]. Inspired by [67] and [64], the state in this study is search-dependent, based on the current operator pair used in each generation. This design avoids reliance on problem-specific features and keeps the state space small and constant, regardless of the problem size. Specifically, the Q-learning framework defines 9 distinct states corresponding to all possible combinations of the three mutation (pairwise, swap, inversion) and three crossover (one-point, two-point, uniform) operators. Each state maps to one row in the Q-table, guiding operator selection throughout the optimization process.

Action Space: The action space consists of the set of possible decisions the agent can take, i.e., the selection of a new pair of genetic operators to be applied in the next generation. Similar to the state

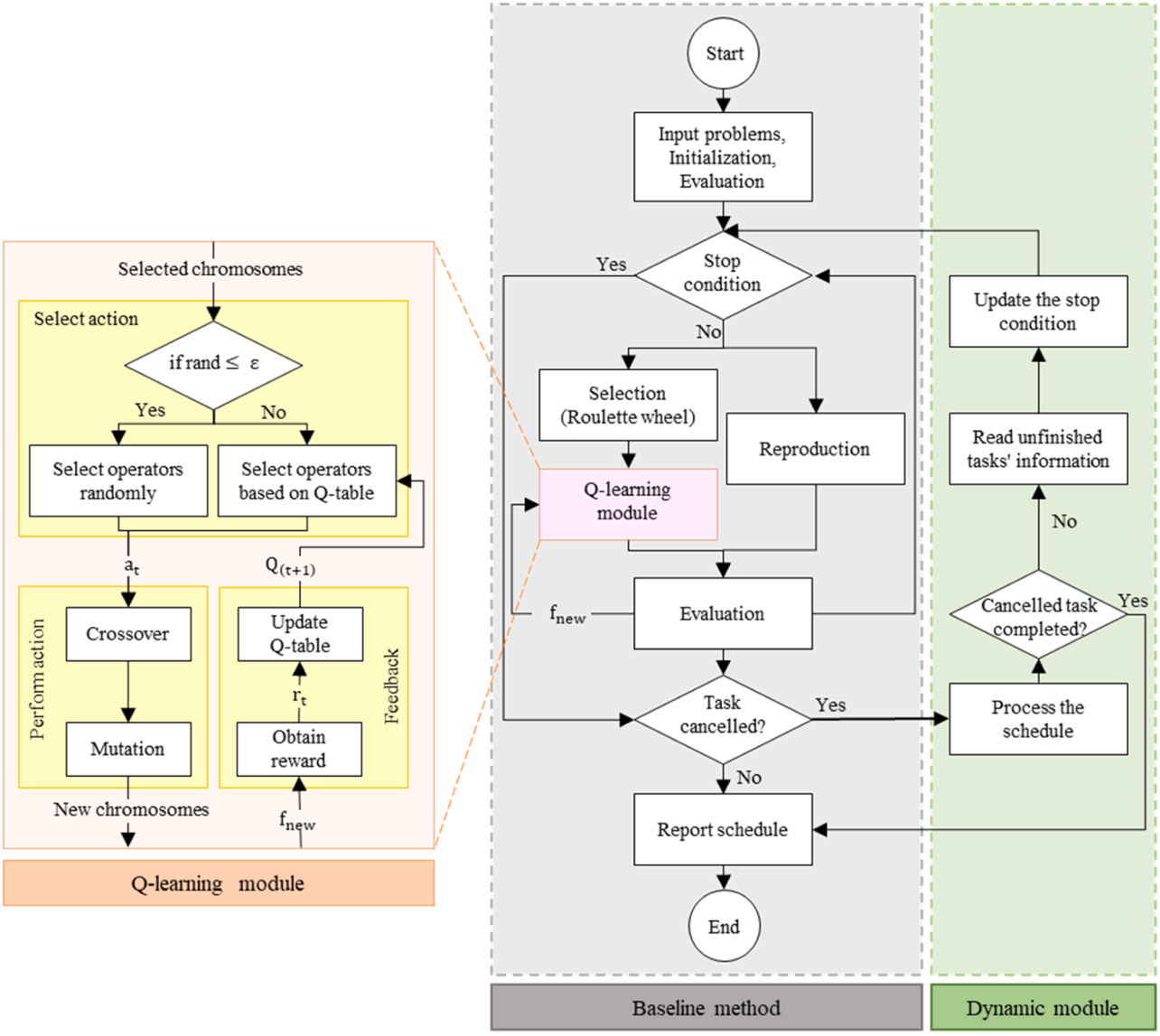


Fig. 11. The flowchart of proposed two-stage Q-learning based GA.

space, there are 9 possible actions corresponding to all valid combinations of one mutation and one crossover operator. This action is selected using an ε-greedy exploration-exploitation strategy [57]. With a small probability ε, a random operator pair is selected (exploration), while with probability $(1-\varepsilon)$, the action with the highest Q-value in the current state is selected (exploitation) [56].

Environment: In our framework, the environment represents the optimization process itself. It evaluates the outcomes of the agent's decisions, specifically, the selection of operator pairs, and provides reward feedback based on search performance improvements. This reward is then used to update the Q-table, enabling the agent to learn which operator pairs are more effective. In this way, the environment connects the agent's decisions with actual performance outcomes in the scheduling problem [60].

It is important to highlight that the state-action space in the proposed Q-learning framework is fixed and remains small, with only 9 states and 9 actions based on the possible combinations of mutation and crossover operators. Unlike other RL-based methods that work with complex scheduling environments, such as job or machine configurations, our method keeps the learning space separate from the problem details. As a result, the size of the state-action space does not increase as the problem becomes larger, which helps the learning process stay stable and efficient even in large-scale scheduling problems.

5. Numerical results

As a result of implementing the proposed model and solution approaches, computational results are presented in this section. In the first step, the parameters of the model are defined, and test problems of various complexity are generated across a range of sizes. Afterward, algorithm parameters are adjusted to ensure optimal performance. When solving small-size problems, CPLEX solver is applied, while when solving medium and large-size problems, metaheuristic algorithms are utilized. The performance of algorithms is then compared in terms of solution quality and computational time. In addition, a sensitivity analysis is conducted to determine how changes in key parameters affect the models, providing valuable managerial insights for practical use. All metaheuristic algorithms are simulated in MATLAB and are performed on a Core i5-4210 M CPU with 8 GB RAM.

5.1. Setting model parameters

This section describes the parameters used to generate instance

Table 10
Model parameters for instance problems.

Parameter	Range	Problem size
Number of tasks, n	$U \sim (1,10)$	Small
	$U \sim (10,20)$	Medium
	$U \sim (20,35)$	Large
Number of providers, m	$U \sim (1,5)$	Small
	$U \sim (5,7)$	Medium
	$U \sim (7,10)$	Large
Subtasks of a task, p_i	$U \sim (1,5)$	Small
	$U \sim (1,6)$	Medium
	$U \sim (1,7)$	Large
Arrival time,	$Exp \sim (100)$, Minute	All size
Processing time, P_{ijkr}	$U \sim (30,180)$, Minute	All size
Logistics time, $TSS_{kk'}$	$U \sim (30,240)$, Minute	All size
Delivery time, TSC_{ki}	$U \sim (30,240)$, Minute	All size
Cancellation time, $t^{(c)}$	$Exp \sim (120)$, Minute	All size
Responding time, Δt	$U \sim (10,30)$, Minute	All size

problems. To test the generality and validity of solution methodologies across different problem sizes, two sets of instances are created in three scales: small, medium, and large. Using predefined parameters, 15 static problems (SP1–SP15) and 15 dynamic problems (DP1–DP15) are generated. The details of these parameters are summarized in Tables 10 and 11, following the parameters values in previous studies [42,31]. The instances provide a diverse dataset for testing the effectiveness of solution methodologies in solving scheduling and service allocation problems in cloud manufacturing.

5.2. Tunning algorithms parameters

A metaheuristic algorithm's performance heavily depends on its parameter settings. In this study, the Taguchi method was used to calibrate the parameters of the implemented algorithms. Experimental design was employed instead of adjusting parameters individually, which may not capture interactions between parameters. Taguchi's method ensures efficient tuning while minimizing the number of experiments required by defining key parameters and their corresponding levels [56].

Table 12 presents the algorithm parameters and their corresponding levels. Based on the number of parameters and levels, the Taguchi method recommended an orthogonal array of L9 for SEO, SA, TS, GA, GWO, and HHO, while L27 was used for PSO and QLGA. Three representative problems of different sizes (problems 3, 8, and 13) were selected and run thirty times each to determine the optimal parameter levels. The results were evaluated using relative percentage deviation (RPD) for the minimizing objective, as calculated in Eq. (33).

$$RPD = \frac{Alg_{sol} - Best_{sol}}{Best_{sol}} \tag{33}$$

Where Alg_{sol} represents the objective function value obtained by the algorithm, and $Best_{sol}$ denotes the best-found value. After executing the model, the objective function values were converted into RPD values, and the mean RPD was considered as the response for the Taguchi method. The parameter setting that results in the minimum RPD was selected as the optimal level. Fig. 12 and Table 12 propose the best level of parameters, which are applied in the subsequent section for further experiments.

5.3. Assess the efficiency of the proposed solution methods

Metaheuristic approaches and exact optimization method, GAMS using the CPLEX solver, are implemented in this section to determine their efficiencies. By analyzing their computational performance and solution quality, this assessment aims to provide a comprehensive comparison of these methods. This comparison uses different performance metrics, such as the objective function values obtained, the time required to reach the best solution (hitting time), and statistical analysis. As a result of employing these metrics, each approach is highlighted in terms of its strengths and limitations, highlighting the trade-off between solution accuracy and computational efficiency. Additionally, the statistical analysis offers deeper insight into how well the proposed metaheuristic handles small and large problems, further confirming the validity of the observed performance differences.

To ensure reasonable computational feasibility, the exact method was formulated in GAMS and solved using the CPLEX solver, with a maximum execution time of 3600 seconds for both SPs and DPs. We updated the schedule status and ran dynamic problems (DP1-DP15) after solving the static problems (SP1-SP15). In order to evaluate the performance of the metaheuristic approach, GAMS results serve as a benchmark since exact methods guarantee optimality. Nevertheless, as the problem size increases, the exact solution method suffers from a significant drop in efficiency, especially for medium- and large-sized problems, due to its computational complexity. After problem 5, this method reaches the time limit and cannot find a feasible solution within a reasonable time frame. By contrast, metaheuristic algorithms have been able to find near-optimal solutions in a shorter amount of time.

All metaheuristic algorithms are applied to both dynamic and static versions of the problems thirty times to consider the inherent variability of metaheuristic methods. Tables 13 and 14 show the average values (Avg.) and standard deviation (SD) of the solutions proposed by each algorithm. It is evident that the solution quality varies between the metaheuristics. The objective function values for all test problems were plotted separately based on problem size and for static and dynamic scenarios to facilitate comparison (see Fig. 13). Furthermore, hitting

Table 11
The dimensions of instance problems.

Scale	Problems	n	m	$\sum_k q_k$	$\sum_i p_i$	$t^{(c)}$	Γ	Δ
Small	1	3	2	4	9	50.278	3	22.026
	2	4	2	5	13	39.804	1	28.443
	3	5	4	9	19	28.352	3	20.918
	4	7	5	13	25	43.340	7	29.110
	5	9	6	14	31	11.329	2	24.734
Medium	6	11	7	18	37	103.605	11	26.294
	7	13	9	21	48	51.361	13,2	24.898
	8	15	9	25	58	66.372	15,8	24.862
	9	17	11	26	66	98.541	7,3	11.297
	10	19	12	42	83	20.190	12,19	19.446
Large	11	26	13	49	101	79.234	12,1,8	17.979
	12	30	15	51	121	41.127	20,17,5	20.748
	13	32	17	61	132	95.014	14,6,18	25.091
	14	38	18	72	146	97.199	10,16,9	13.762
	15	40	19	79	172	50.278	3,20,19	27.453

Table 12
The level of parameters in metaheuristics algorithms and the selected levels.

Algorithms	Parameters	Levels			Selected
		1	2	3	
SA	A: Initial temperature	100	500	1000	**100**
	B: Reduction rate	0.95	0.98	0.99	**0.95**
	C: Iteration	100	200	300	**300**
SEO	A: Collecting data rate	0.15	0.20	0.25	**0.2**
	B: Connecting attacker rate	0.03	0.05	0.07	**0.05**
	C: Number of connections	15	25	35	**25**
	D: Iteration	100	200	300	**300**
TS	A: Tabu list size	4	5	6	**5**
	B: Neighborhood size	5	10	15	**15**
	C: Iteration	100	200	300	**200**
PSO	A: Population size	15	25	35	**35**
	B: Inertia weight	0.45	0.65	0.85	**0.65**
	C: Personal learning rate	1.2	1.3	1.4	**1.3**
	D: Global learning rate	1.3	1.4	1.5	**1.4**
	E: Iteration	100	200	300	**100**
HHO	A: Population size	15	25	35	**15**
	B: Iteration	100	200	300	**300**
GWO	A: Population size	15	25	35	**25**
	B: Iteration	100	200	300	**300**
GA	A: Population size	15	25	35	**35**
	B: Mutation percentage	0.05	0.1	0.2	**0.2**
	C: Crossover percentage	0.7	0.8	0.9	**0.7**
	D: Iteration	100	200	300	**200**
QLGA	A: Population size	15	25	35	**35**
	B: Mutation percentage	0.05	0.1	0.2	**0.2**
	C: Crossover percentage	0.7	0.8	0.9	**0.7**
	D: Epsilon	0.1	0.2	0.3	**0.1**
	E: Learning rate	0.9	0.85	0.8	**0.9**
	F: Discount factor	0.9	0.85	0.8	**0.85**
	G: Iteration	100	200	300	**300**

time, which represents the time required to reach the best solution, is also recorded for each instance and reported in Table 15. Based on the results, the proposed metaheuristic is capable of providing competitive solutions within a feasible computational time compared to the exact method. Metaheuristics remain effective when exact methods become impractical for large instances, where the exact solver guarantees optimality for small instances.

Generally, QLGA and GA seem to have slightly lower median solution values and their distributions indicate stability with fewer extreme variations. The variance in solution values shows that algorithms like SA, TS, and PSO exhibit higher variability indicating less consistent performance. Conversely, GA, GWO, and QLGA show relatively lower variability, suggesting more stable solution quality. SEO and HHO display moderate variability. A key observation is that the ranking of metaheuristics remains relatively stable across the three problem sizes, though some methods exhibit higher variability depending on the instance size.

Since problem sizes vary, direct comparison of objective function values may not be meaningful. Therefore, as mentioned in section 5.2, the RPD is used to normalize the results. Fig. 14 represents the RPD values of static and dynamic problems. Analysis of RPD values suggests that the algorithms perform similarly in static and dynamic situations. TS and GA also perform well, maintaining relatively low RPD values. QLGA achieves the lowest RPD values in most problems, demonstrating superior solution quality. HHO has the highest variances and generally higher RPD values, indicating lower solution quality. PSO, SEO, and SA display moderate performance with fluctuations across problem sizes. For a more in-depth analysis, the statistical tests were employed to determine the best-performing algorithm overall.

To compare the performance of the algorithms accurately, statistical tests were conducted. Since the performance of the algorithms in solving static and dynamic problems is relatively similar in terms of RPD values, all results were considered together for comparing the performance of the algorithms. Firstly, the normality assumption was checked using the Anderson-Darling test. A normal distribution was found for all algorithms, except one, and Bartlett's test indicated that the assumption of homogeneity of variance was violated. To determine whether there were statistically significant differences among the algorithms, Welch's ANOVA, which is robust to unequal variances, was applied.

In all cases, the Welch ANOVA test indicated statistically significant differences in algorithm performance across the small, medium, and large problem sizes, as well as for the overall dataset. According to this finding, not all algorithms perform equally, with at least one algorithm performing significantly better or worse than the others. Aside from providing a visual representation of the performance variation among different algorithms across small, medium, and large problem sizes, interval plots also provide an indication of metaheuristic performance overall.

As seen in Fig 15, QLGA consistently achieves the lowest RPD values across all problem sizes, with minimal variation, indicating its stability and effectiveness in finding high-quality solutions. The greater the problem size, the better QLGA's relative performance, while HHO, SA and PSO struggle significantly in large-scale problems. The performance of HO, PSO, and SA is not significantly better than that of other methods. Not only do these algorithms struggle with scalability, they also tend to perform inconsistently across different problem sizes.

Compared to recently developed algorithms, classical algorithms such as GA and TS perform better when the problem is more complex. The two algorithms have low RPD values and small confidence intervals, suggesting consistent performance across different problem sizes. There is less consistency across different problem sizes when using recently proposed algorithms such as GWO and SEO, as they exhibit varying RPD values and wider confidence intervals. A statistically significant difference exists among the algorithms across all problem sizes, confirming that the general trend remains consistent across all problem sizes. To further investigate these differences, post-hoc tests are required in order to determine which algorithms differed significantly from each other.

The Games-Howell post-hoc test was utilized to conduct a pairwise comparison of metaheuristic algorithms. This test account for unequal variances, make it ideal for evaluating the performance of applied metaheuristics. Fig. 16 provides a statistical comparison of the mean differences between different metaheuristic algorithms. The confidence intervals that do not include zero indicate a significant difference between the corresponding algorithms. Most comparisons involving QLGA and other algorithms show significant positive differences which means QLGA performs statistically different.

According to Table 16, GWO, GA, and QLGA, on the other hand, rank in the upper groups and provide much better solutions according to the grouping information provided by the Games-Howell method. This study demonstrates that QLGA is a promising approach for solving DTSSA in CMg systems in comparison with other single solution methods, such as TS and SEO.

Hitting time is another crucial performance metric that evaluates the efficiency of an algorithm in reaching the best solution. A lower hitting time indicates that an algorithm converges faster, which is particularly important in dynamic and large-scale problems where computational efficiency is a key concern. Fig. 17 and Fig. 18 illustrate the hitting time performance of various algorithms, distinguishing between single-solution-based and population-based methods, and analyzing their efficiency in static (SP) and dynamic (DP) problems. This distinction allows for a more detailed comparison of how algorithmic structures influence convergence speed under different conditions.

As a result of their iterative process of generating and exploring solutions, population-based algorithms tend to have higher hit times. Among them, QLGA and GWO have the highest hitting times, while PSO performs relatively better. The convergence speed of single-solution algorithms is superior, with SA and TS consistently maintaining the lowest hitting times. The computational performance of metaheuristics differs between static and dynamic problems. Hitting times are significantly higher in dynamic problems (DPs) compared to static problems (SPs).

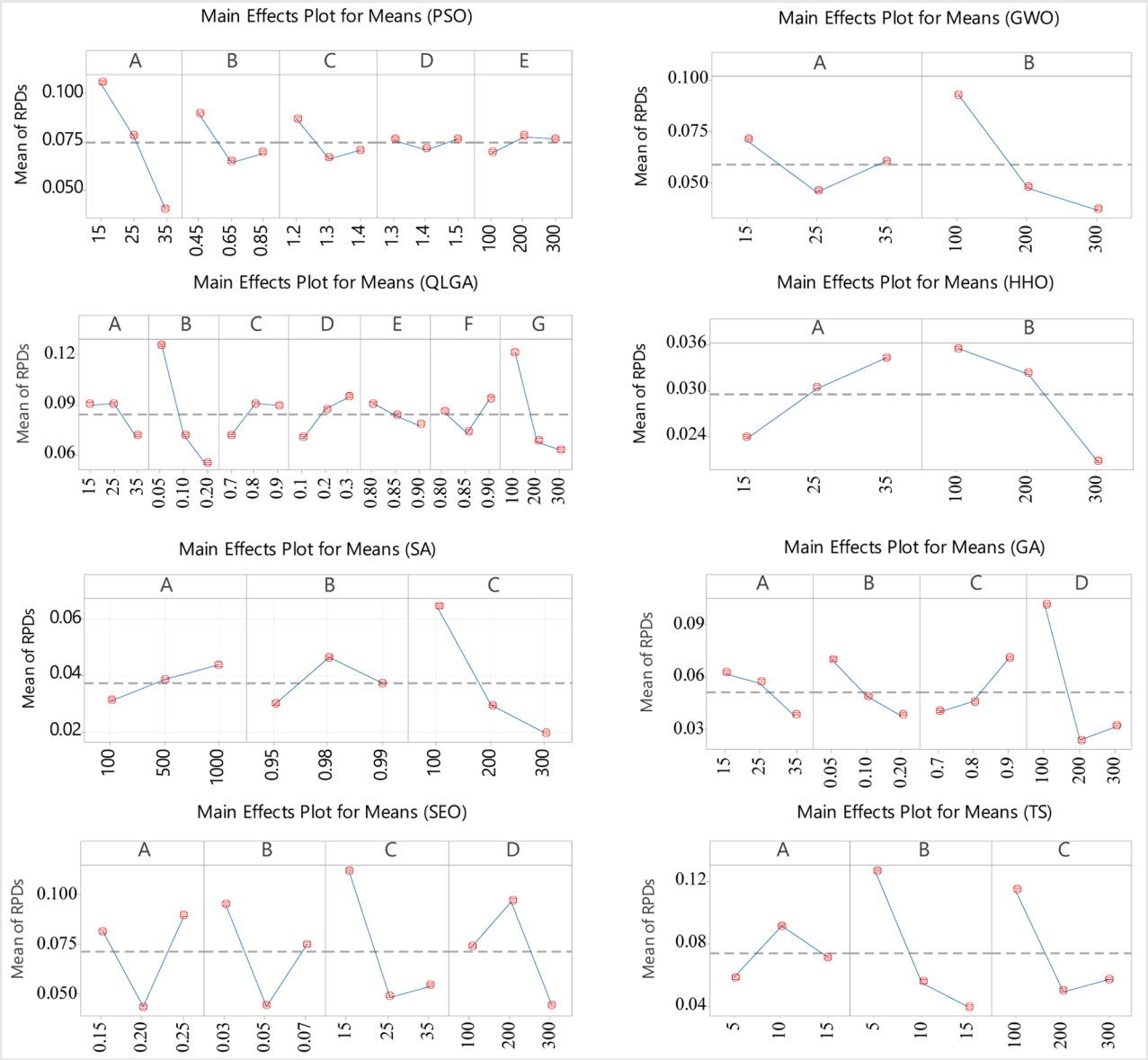


Fig. 12. The main effect plot of mean of RPD for all algorithms.

SA and TS maintain stable performance across both static and dynamic settings, reinforcing their adaptability. QLGA shows a substantial increase in hitting time for dynamic problems, indicating its sensitivity to environmental changes.

To statistically compare the algorithms' hitting times, the Friedman non-parametric test was applied (Table 17). A lower ranking indicates a faster convergence rate, resulting in a faster hitting time. This test ranks the algorithms by their average performance across multiple instances. The results indicate that SA has the lowest hitting times across all categories, making it the most efficient in terms of fast convergence. TS follows closely behind SA, showing stable performance in both static and dynamic conditions. Its results outperform other population-based algorithms, but it is still behind classical single-solution algorithms in terms of convergent time. The highest hitting times are found in SEO and GWO, both of which have the longest convergence times.

SA is the most effective solution approach if fast convergence is the main concern. TS is next, followed by QLGA when solution quality (in terms of mean performance) is the main concern. A balanced approach shows that TS ranks second in hitting time and third in mean performance. This consistency across both speed and quality puts it at the top of the list for overall reliability.

5.4. Benchmark

To verify the adaptability and effectiveness of the proposed QLGA, a total of 30 DTSSA benchmark instances named DTSs 01–30 were constructed based on the datasets proposed by [98] where extended from the flexible job shop problem benchmark [99]. DTSs were divided into three scales: small-scale instances (DTSs 01–10), medium-scale instances (DTSs 11–20) and large-scale instances (DTSs 21–30). These problems are extended from problems (01-10), (26-35), and (41-50) in benchmark, respectively. For the instances of different scales, the processing times were set as the first worker's processing time; the delivery time (TSC_{ki}) and logistics time ($TSS_{kk'}$) for all size problems were set $U \sim (30, 240)$; the arrival time of all tasks were set as $exp \sim (100)$; Cancellation time were set as $exp \sim (120)$. The constructed instances DSTs 01–30 can be downloaded from https://github.com/Rajabi9291/Scheduling-benchmark/blob/main/Data.zip.

Table 13
Detail results of the solution methodologies on static and dynamic test problems (Part I).

Problems		Gams	SA		TS		SEO		PSO	
			Ave.	Std.	Ave.	Std.	Ave.	Std.	Ave.	Std.
Small	SP1	932.69	945.94	2.02	946.15	1.90	951.35	8.49	964.26	38.37
	DP1	893.58	900.96	**0.00**	900.96	**0.00**	900.96	**0.00**	956.06	104.32
	SP2	709.10	858.87	48.28	798.60	63.67	829.62	53.49	837.03	46.72
	DP2	695.73	811.29	67.73	**719.22**	**9.42**	752.96	42.61	767.85	41.62
	SP3	668.37	970.08	75.31	901.95	61.61	889.25	84.22	983.52	78.27
	DP3	645.31	936.37	77.66	869.69	47.13	865.12	60.55	962.45	101.03
	SP4	777.19	1181.38	98.24	1115.98	79.23	1121.39	69.15	1130.53	81.84
	DP4	758.05	1101.42	123.55	1029.30	93.12	1072.35	23.60	1119.26	104.30
	SP5	655.14	1168.68	**14.25**	1099.51	99.38	1169.80	67.24	1183.41	24.00
	DP5	649.51	1155.54	**22.21**	991.97	56.13	1106.96	80.64	1157.13	90.24
Medium	SP6	1747.14	1713.25	74.51	1696.13	50.82	1582.71	117.43	1705.98	136.00
	DP6	-	1695.49	86.99	1582.12	162.37	1574.31	117.51	1694.79	112.97
	SP7	-	1829.70	57.52	1655.11	127.13	1747.31	130.87	1678.46	59.19
	DP7	-	1784.37	83.04	**1372.24**	66.98	1577.35	57.12	1675.55	101.30
	SP8	-	2112.69	124.20	2094.54	184.19	1971.76	167.10	2020.14	78.30
	DP8	-	2074.49	160.71	1809.26	89.66	1878.23	233.72	1958.70	108.45
	SP9	-	1910.04	182.92	1795.43	115.60	1881.14	87.12	1828.74	37.89
	DP9	-	1875.72	117.60	1585.80	89.64	1689.64	121.13	1823.69	103.54
	SP10	-	2362.68	101.45	2243.06	152.11	2366.58	42.97	2331.07	207.10
	DP10	-	2308.16	189.03	1996.81	132.93	2143.57	**31.67**	2269.27	295.23
Large	SP11	-	2048.48	131.13	1968.60	138.09	2091.50	97.40	2094.08	62.02
	DP11	-	2015.05	**32.83**	1726.91	37.06	1874.04	154.10	2080.76	179.45
	SP12	-	2477.28	95.28	2442.81	255.08	2527.53	92.63	2520.42	96.90
	DP12	-	2437.90	130.10	2106.77	116.10	**2219.50**	114.88	2491.19	137.84
	SP13	-	2522.04	**18.69**	2332.48	168.46	2407.50	110.31	2400.28	136.72
	DP13	-	2519.06	72.81	2140.77	111.05	2128.19	102.92	2388.10	168.74
	SP14	-	2856.87	231.19	2742.76	200.78	3021.12	152.12	2940.15	132.82
	DP14	-	2827.51	120.96	2433.20	73.63	2646.89	183.63	2895.77	190.49
	SP15	-	2725.99	254.36	2457.94	178.78	2661.48	105.68	2454.30	**55.83**
	DP15	-	2655.89	75.58	2270.18	70.48	2349.24	42.89	2384.89	87.43

Table 14
Detail results of the solution methodologies on static and dynamic test problems (Part II).

Problems		HHO		GWO		GA		QLGA	
		Ave.	Std.	Ave.	Std.	Ave.	Std.	Ave.	Std.
Small	SP1	**944.57**	2.19	972.67	30.92	**944.57**	2.19	945.79	**1.81**
	DP1	900.96	**0.00**	935.67	61.78	906.62	9.80	**900.96**	**0.00**
	SP2	798.10	**18.76**	828.67	44.23	788.09	31.43	**773.03**	46.18
	DP2	787.11	60.89	743.39	36.23	760.49	43.84	741.93	32.91
	SP3	939.72	58.48	911.00	**40.36**	889.31	47.12	**859.79**	66.64
	DP3	893.68	105.09	844.00	**22.34**	841.33	70.93	**805.48**	61.40
	SP4	1245.48	73.52	1078.67	49.24	1064.42	**15.77**	**1027.44**	44.54
	DP4	1160.01	53.64	1045.67	113.52	1044.78	**21.11**	**951.62**	53.62
	SP5	1157.86	39.74	1088.27	27.85	1071.42	91.73	**1039.95**	67.22
	DP5	1089.29	45.01	1040.83	132.29	1001.69	81.94	**931.34**	40.76
Medium	SP6	1858.05	51.07	1650.33	103.24	1568.22	**34.73**	**1507.98**	41.23
	DP6	1755.89	58.95	1628.67	114.65	1441.36	**52.03**	**1441.08**	85.26
	SP7	1843.35	144.38	1754.33	30.60	1629.47	**7.41**	**1588.11**	107.24
	DP7	1640.55	93.78	1606.67	121.33	1470.57	**42.07**	1454.42	105.37
	SP8	2232.90	80.08	2043.00	190.86	**1902.59**	66.03	1937.5	**51.16**
	DP8	2025.32	70.11	1968.00	110.50	1812.86	**34.87**	**1768.58**	88.67
	SP9	2086.36	**16.94**	1780.67	95.92	1723.22	72.46	**1653.61**	108.51
	DP9	1907.57	96.39	1719.33	95.35	1622.15	**78.62**	**1514.21**	127.90
	SP10	2380.73	61.31	2284.33	43.36	2173.61	**42.50**	**2139.19**	116.54
	DP10	2249.86	114.63	2186.33	176.74	1965.18	113.70	**1949.88**	64.91
Large	SP11	2238.00	79.00	2044.33	39.63	1953.17	**21.60**	**1863.35**	52.81
	DP11	1932.23	43.52	1932.00	63.53	1891.21	94.14	**1701.81**	55.35
	SP12	2761.91	223.95	2523.33	100.90	2365.81	**78.28**	**2289.99**	130.94
	DP12	2311.18	83.91	2211.00	**54.74**	2162.83	147.83	**2097.93**	118.13
	SP13	2530.02	69.62	2547.67	276.02	2310.00	54.86	**2296.23**	103.72
	DP13	2431.88	**14.98**	2442.67	83.93	2225.70	155.10	2128.81	59.24
	SP14	3067.22	125.69	2814.00	125.90	2757.63	**40.78**	**2688.4**	112.51
	DP14	2661.42	**18.88**	2597.00	89.60	2449.79	26.79	**2349.51**	69.88
	SP15	2771.90	82.05	2624.00	93.74	2371.91	66.22	**2307.47**	68.55
	DP15	2470.09	153.06	2378.33	**36.35**	2282.84	42.93	**2068.98**	57.33

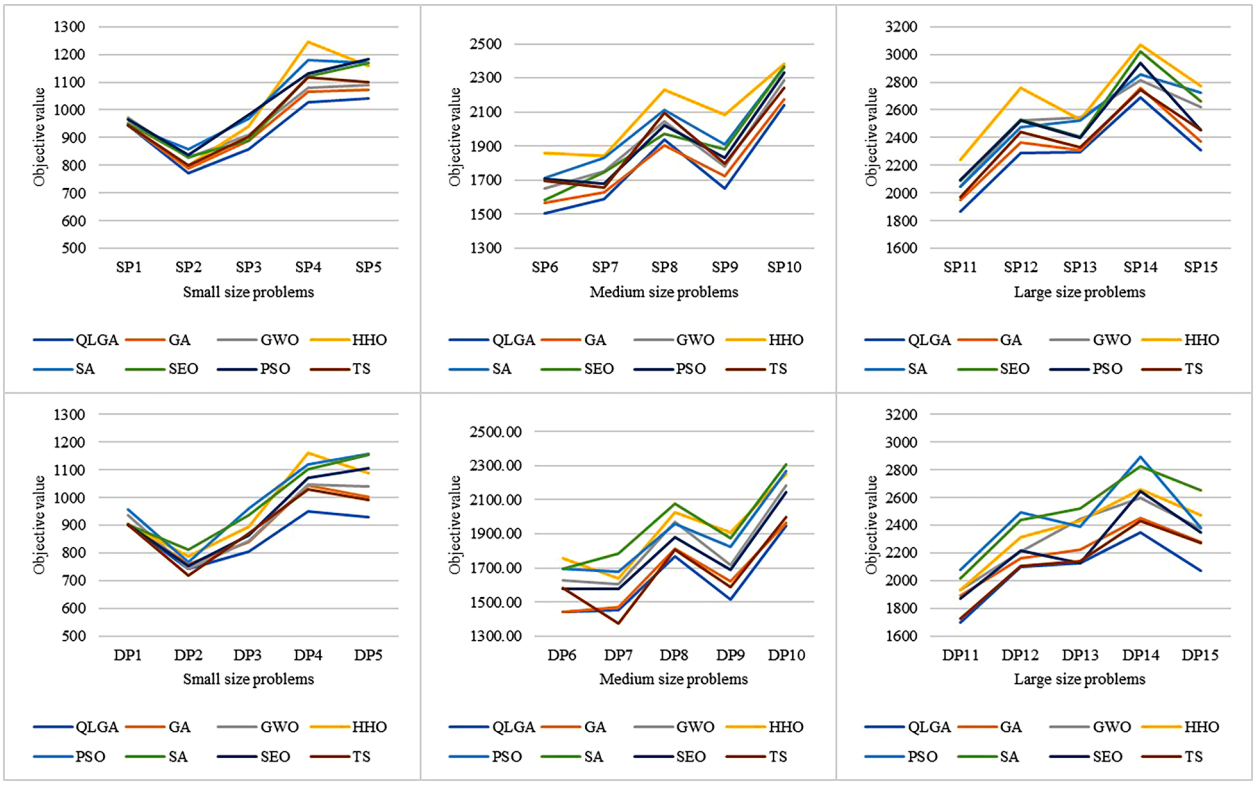


Fig. 13. Objective value of three size static and dynamic problems.

Table 15
The hitting time of solution methodologies.

Problems		Exact	Metaheuristics							
		Gams (Time/Gap)	SA	TS	SEO	PSO	HHO	GWO	GA	QLGA
Small	SP1	1.87	0.533	0.886	3.585	0.149	4.556	0.135	2.684	0.609
	DP1	0.47	1.991	2.766	5.354	0.177	4.805	5.198	0.515	0.954
	SP2	3.80	0.229	1.181	3.333	0.396	4.549	0.418	3.751	3.442
	DP2	1.67	1.284	3.875	5.790	0.342	0.691	6.836	0.601	3.023
	SP3	44.70	0.570	1.231	5.466	1.069	5.727	1.336	5.287	3.512
	DP3	12.22	2.277	5.836	11.958	0.918	12.196	9.301	7.213	5.188
	SP4	222.40	0.551	1.352	5.033	1.324	4.737	5.774	5.285	4.831
	DP4	21.91	2.218	5.830	10.330	1.600	12.966	10.350	8.747	5.496
	SP5	2452.38	0.648	1.461	7.559	1.491	6.907	5.718	5.850	6.408
	DP5	69.047	3.567	4.886	19.608	1.015	26.262	12.474	9.711	22.462
Medium	SP6	46.19 %	0.754	1.820	12.608	3.173	7.318	9.813	8.376	10.603
	DP6	-	4.814	6.787	29.857	7.127	25.776	23.057	22.092	36.273
	SP7	-	1.053	2.870	13.745	3.195	6.767	10.827	10.771	13.033
	DP7	-	3.514	10.093	34.361	13.592	27.053	30.966	25.665	32.717
	SP8	-	0.902	2.867	20.731	5.938	8.730	18.947	10.896	12.908
	DP8	-	2.766	6.712	43.793	4.983	19.532	29.989	23.674	26.403
	SP9	-	0.772	2.335	17.442	8.440	6.621	16.589	10.844	13.937
	DP9	-	3.202	9.886	55.373	7.309	20.160	28.232	26.166	41.244
	SP10	-	1.123	2.609	23.653	10.403	10.042	22.732	14.413	14.655
	DP10	-	5.211	7.218	86.706	5.825	31.056	46.686	28.891	51.999
Large	SP11	-	0.993	1.382	29.081	11.422	2.343	23.101	13.248	1.064
	DP11	-	5.433	11.230	49.419	22.272	52.996	59.555	31.397	3.386
	SP12	-	1.156	2.471	28.961	13.146	8.968	21.414	16.026	5.316
	DP12	-	5.682	13.791	77.835	32.120	26.295	73.408	46.633	5.025
	SP13	-	1.249	3.387	24.505	15.527	13.852	21.613	17.182	23.413
	DP13	-	2.712	15.596	138.271	18.814	58.472	71.508	51.705	33.824
	SP14	-	1.362	3.743	26.060	19.210	6.556	23.006	20.075	29.352
	DP14	-	6.009	15.316	198.232	28.218	36.763	82.754	64.145	26.600
	SP15	-	1.370	4.037	30.827	15.087	13.588	25.316	20.212	38.994
	DP15	-	4.909	9.643	194.203	53.412	71.447	93.170	65.345	48.830

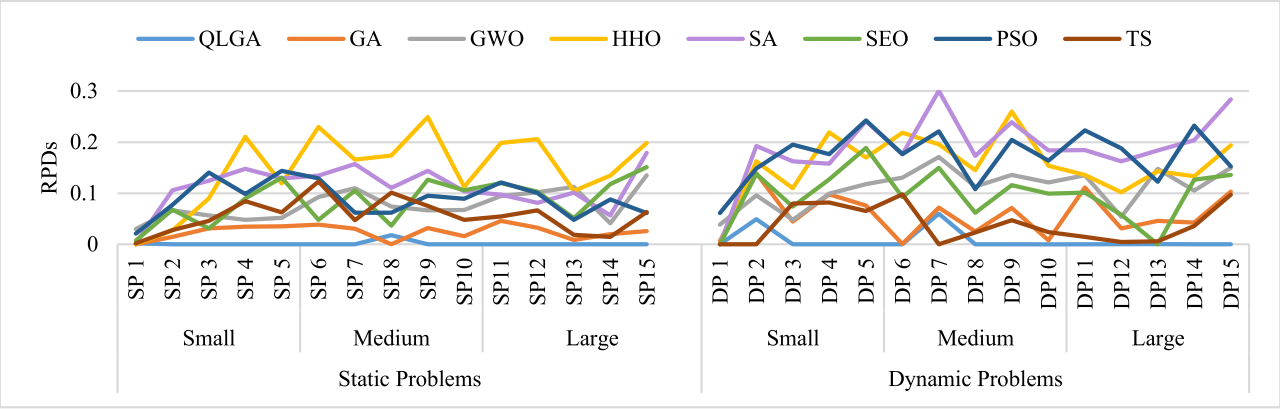


Fig. 14. RPD values of static and dynamic problems.

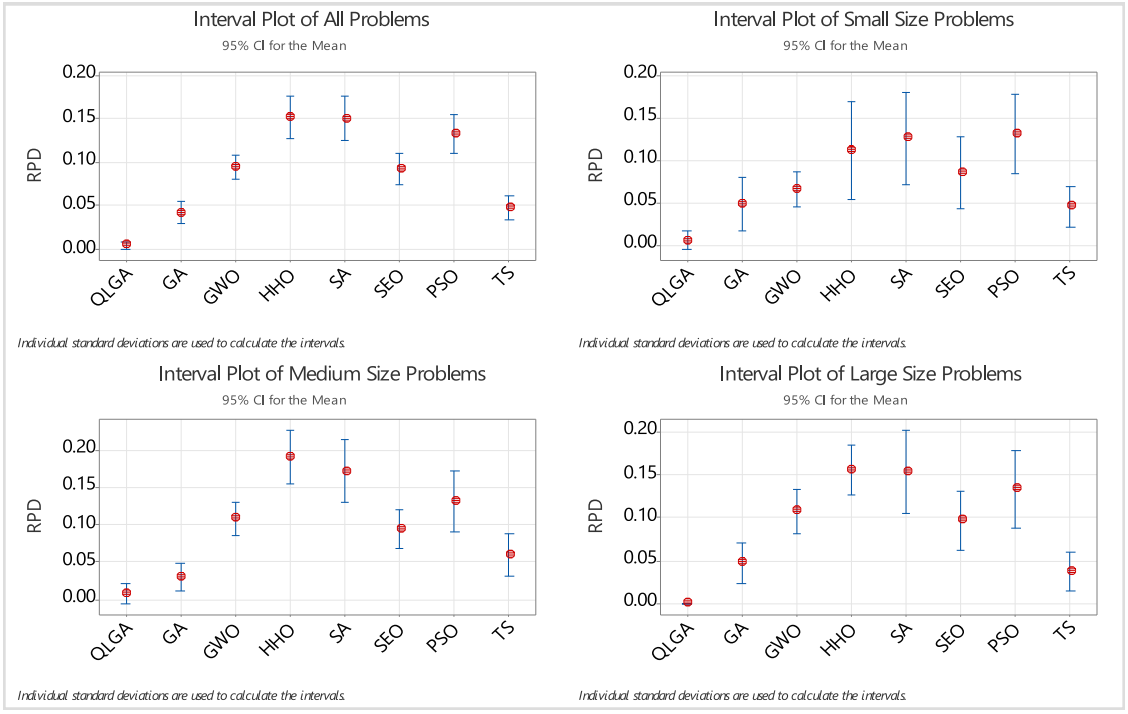


Fig. 15. The interval plot of RPD values across different sizes.

Tables 18 and 19 represent the objective function value proposed by metaheuristic algorithms for static and dynamic instance respectively. The computational results reveal distinct performance patterns between the metaheuristic algorithms across static and dynamic problems. For the static problems, QLGA consistently demonstrates superior efficiency, achieving the best solutions in 25 out of 30 instances, while TS follows with 5 successful instances. In dynamic problems, the performance gap between the algorithms widens. QLGA emerges as the most effective, achieving the best solutions medium and large size problems. Although TS performs well in small scale problems, its effectiveness decreases as the complexity of the dynamic environment increases.

To further clarify the differences between the performance of the applied algorithms, the RPD values are demonstrated in Figs. 19 and Fig. 20. The results indicate that the performance of the algorithms remained consistent across both static and dynamic problems. In both cases, QLGA achieved the best results, followed by TS and GA. The remaining algorithms, including SA, SEO, PSO, HHO, and GWO, ranked lower in comparison. This consistency highlights the robustness of QLGA and TS, particularly due to their effective use of memory and past data, which enables them to deliver superior solutions in both types of problem scenarios.

To further validate the effectiveness of the proposed Q-learning-based metaheuristic algorithm, a comparative analysis was conducted against two recent and highly relevant RL-based methods, QLIGS [51] and QLPSOIGS [55]. Without loss of generality, these two algorithms were selected as representative RL-based approaches due to their methodological similarity and relevance to our problem setting. The results shown in Tables 18 and 19 indicate that the proposed method outperformed both approaches across all tested instances in scheduling and rescheduling stages. This confirms the effectiveness of the proposed algorithm for solving dynamic scheduling problems in cloud manufacturing.

5.5. *Domain Adaptation through Q-Table Initialization*

Domain adaptation is a form of transfer learning in RL where knowledge from a source domain is used to accelerate learning in a related target domain [100]. One typical implementation is value

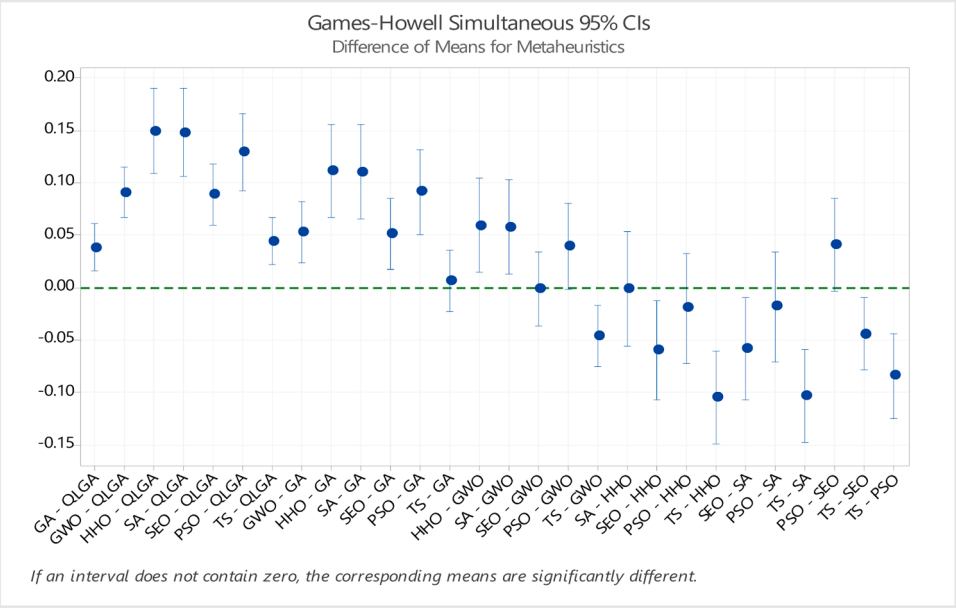


Fig. 16. The pairwise comparison of Games-Howell method.

Table 16
Grouping information using the Games-Howell method.

Factor	Mean	Grouping				
HHO	0.1528	A				
SA	0.1513	A				
PSO	0.1324	A	B			
GWO	0.09427		B			
SEO	0.09269		B			
TS	0.04767			C		
GA	0.04195			C		
QLGA	0.00431					D

transfer, where Q-values learned in the source domain are used to initialize the policy or value function in the target setting [101,102].

To explore the potential of reusing learned knowledge across different problem instances, a basic domain adaptation mechanism based on Q-table initialization were implemented. Specifically, the average Q-values obtained from previously solved experimental instances (source domain) were calculated and used to initialize the Q-table for benchmark problems of similar scale (target domain). Despite the simplicity of this approach, it resulted in improved performance in 10 instances. These preliminary results demonstrate the feasibility of value-based domain adaptation within Q-learning based metaheuristics and motivate further investigation. Table 20 reports the comparative performance of the algorithm with and without Q-table initialization across the benchmark instances.

5.6. Impact of rescheduling

In this subsection, the impact of rescheduling on the performance of dynamic problems was analyzed. Since unexpected disruptions and real-time variations can significantly affect scheduling efficiency, rescheduling plays a crucial role in minimizing delays and optimizing resource utilization. By adjusting task assignments and reordering operations, rescheduling aims to improve overall system performance and reduce the maximum completion time.

Fig. 21 presents a comparison between the initial schedule and the rescheduled outcomes for 15 dynamic problem instances. Rescheduling consistently results in a shorter completion time in dynamic problems, as shown in the left chart, which shows that rescheduling consistently leads to shorter completion times. As seen in the right chart, the improvement percentage varies from problem to problem, with a typical range of 5% to 15%. While some problems benefit modestly, others benefit significantly.

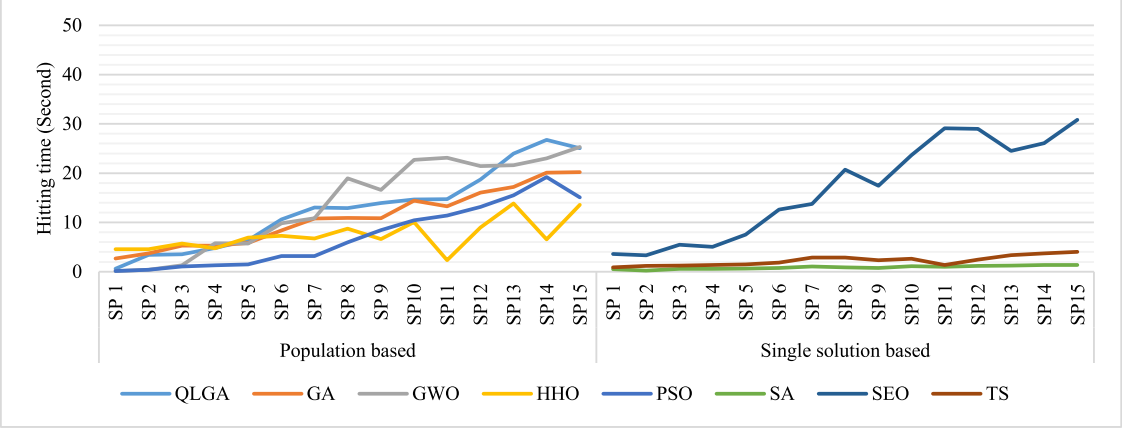


Fig. 17. The hitting time of metaheuristics for static problems.

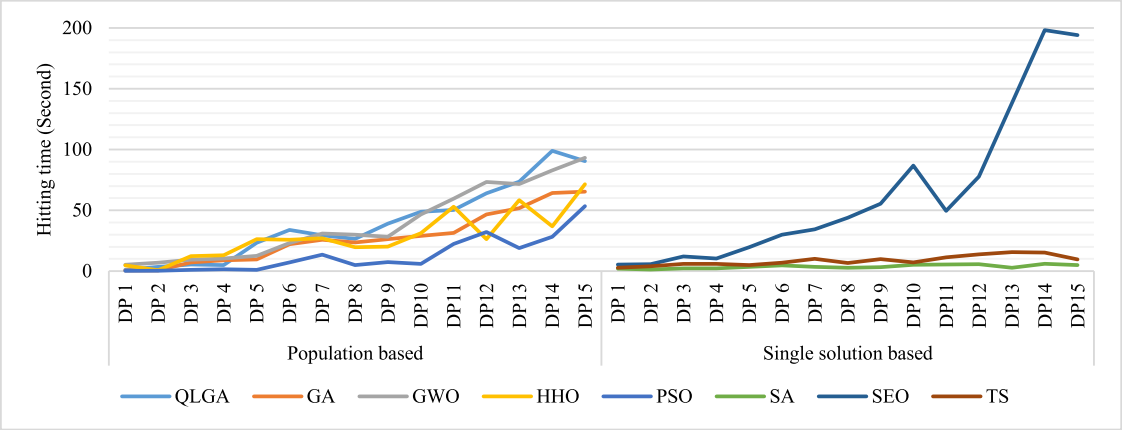


Fig. 18. The hitting time of metaheuristics for dynamic problems.

Table 17
The result of Friedman test across different problem sizes.

Algorithms	Small size problems		Medium size problems		Large size problems		All problems	
	Median	Ranks	Median	Ranks	Median	Ranks	Median	Ranks
GA	4.847	5	17.839	5	25.948	5	13.438	5
GWO	5.060	6	22.577	7	39.739	7	18.650	7
HHO	6.483	8	15.224	4	20.656	4	11.757	4
PSO	0.818	1	7.464	3	19.037	3	7.334	3
QLGA	4.230	4	21.278	6	36.840	6	16.899	6
SA	0.942	2	2.907	1	3.187	1	1.033	1
SEO	5.609	7	25.092	8	42.064	8	21.551	8
TS	2.241	3	5.343	2	7.338	2	2.821	2
Overall	3.779		14.716		24.351		11.685	

Table 18
The objective function values of applied algorithms for benchmark instance in static stage.

Problems		Metaheuristics							QL-based metaheuristics		
		SA	TS	SEO	PSO	HHO	GWO	GA	QLIGS	QLPSOIGS	QLGA
DTS01	(10 × 5 × 2 × 6)	1000.67	847.81	983.25	1016.70	1058.12	1012.62	905.96	983.89	976.91	**793.97**
DTS02	(10 × 5 × 2 × 6)	1029.38	932.22	956.56	983.54	1087.69	940.10	1007.94	1097.32	1099.65	**917.00**
DTS03	(10 × 5 × 2 × 6)	1050.46	824.24	902.58	892.28	982.66	895.92	834.83	960.07	961.02	**800.72**
DTS04	(10 × 5 × 2 × 6)	1175.88	**862.38**	1065.66	1198.27	1087.89	1016.09	1096.76	1047.71	1184.99	982.25
DTS05	(10 × 5 × 2 × 6)	752.98	**576.36**	714.96	744.84	742.19	689.95	628.51	683.16	742.75	621.94
DTS06	(15 × 5 × 2 × 6)	1174.84	953.95	1016.44	1132.50	1158.07	1094.96	994.92	1117.43	1070.59	**919.97**
DTS07	(15 × 5 × 2 × 6)	1258.65	**931.39**	1198.84	1348.68	1260.40	1157.09	1140.55	1221.23	1183.13	971.56
DTS08	(15 × 5 × 2 × 6)	1536.52	1304.55	1489.67	1364.70	1586.97	1372.73	1360.36	1468.79	1534.09	**1207.63**
DTS09	(15 × 5 × 2 × 6)	1313.99	1047.22	1297.60	1246.56	1267.74	1312.61	1183.36	1295.58	1283.38	**1030.04**
DTS10	(15 × 5 × 2 × 6)	1083.44	940.49	1015.75	1034.05	1140.18	1108.24	1010.15	1107.51	1071.42	**937.68**
DTS11	(20 × 10 × 4 × 8)	2119.28	1839.35	2162.72	1987.54	2336.87	2170.63	1928.90	2125.06	2019.90	**1836.28**
DTS12	(20 × 10 × 4 × 8)	2623.41	2307.77	2718.65	2546.73	2756.91	2516.63	2404.04	2569.38	2400.87	**2283.50**
DTS13	(20 × 10 × 4 × 8)	2832.37	2558.40	2692.97	2917.46	2830.68	2769.89	2476.75	2706.16	2544.74	**2452.35**
DTS14	(20 × 10 × 4 × 8)	2523.98	2187.81	2552.20	2527.95	2760.21	2504.91	2406.01	2385.37	2512.11	**2175.59**
DTS15	(20 × 10 × 4 × 8)	2896.60	2618.66	2773.46	2722.39	2999.07	2917.33	2588.33	2833.82	2642.83	**2574.90**
DTS16	(30 × 10 × 4 × 8)	3337.98	2879.59	3253.00	3280.56	3370.60	3166.34	2987.86	3217.25	3095.41	**2727.84**
DTS17	(30 × 10 × 4 × 8)	2869.72	**2427.32**	2883.66	2657.89	3025.21	2872.72	2494.08	2778.45	2707.73	2435.35
DTS18	(30 × 10 × 4 × 8)	3396.71	2941.32	3260.70	3209.38	3480.83	3402.07	2971.80	3226.59	3197.21	**2843.24**
DTS19	(30 × 10 × 4 × 8)	2766.22	2281.25	2606.08	2601.00	2719.16	2760.37	2513.64	2551.20	2629.40	**2267.37**
DTS20	(30 × 10 × 4 × 8)	2659.69	2373.06	2601.51	2564.07	2754.90	2536.69	2257.62	2650.79	2485.88	**2252.70**
DTS21	(40 × 10 × 6 × 10)	2899.01	2667.08	2869.68	2916.92	3003.86	2918.13	2822.76	2826.83	2727.30	**2520.89**
DTS22	(40 × 10 × 6 × 10)	3051.70	**2523.01**	2886.71	2880.39	3059.50	2989.10	2570.09	2786.58	2675.67	2622.40
DTS23	(40 × 10 × 6 × 10)	3208.31	2871.01	3038.76	3009.60	3240.18	3152.81	2902.65	3083.54	3115.87	**2734.47**
DTS24	(40 × 10 × 6 × 10)	2925.78	2677.89	2860.61	2906.04	3071.81	2841.35	2883.56	3137.49	2870.54	**2528.19**
DTS25	(40 × 10 × 6 × 10)	3275.87	2839.03	3268.60	3041.10	3401.71	3260.28	3068.91	3236.48	3114.19	**2838.42**
DTS26	(60 × 10 × 6 × 10)	4378.06	3971.38	4077.23	3974.27	4340.60	4219.30	3967.23	4284.82	4293.75	**3726.19**
DTS27	(60 × 10 × 6 × 10)	3716.68	3306.49	3465.50	3282.33	3702.97	3522.23	3364.82	3542.65	3414.08	**3233.22**
DTS28	(60 × 10 × 6 × 10)	3685.49	3509.05	3752.29	3576.11	3733.48	3737.38	3491.85	3707.17	3633.72	**3258.19**
DTS29	(60 × 10 × 6 × 10)	3992.78	3563.00	4047.31	3768.92	4231.92	3913.92	3714.22	3967.61	3744.87	**3500.85**
DTS30	(60 × 10 × 6 × 10)	3621.28	3227.60	3641.91	3546.44	3752.54	3588.10	3516.86	3798.96	3333.69	**3208.88**

Table 19
The objective function values of applied algorithms for benchmark instance in dynamic stage.

Problems		Metaheuristics							QL-based metaheuristics		
		SA	TS	SEO	PSO	HHO	GWO	GA	QLIGS	QLPSOIGS	QLGA
DTS01	(10 × 5 × 2 × 6)	976.07	818.25	930.62	973.98	964.58	989.83	824.77	950.28	919.97	**759.17**
DTS02	(10 × 5 × 2 × 6)	942.62	887.75	949.54	966.41	969.89	855.49	949.83	937.52	1080.35	**849.73**
DTS03	(10 × 5 × 2 × 6)	923.37	**745.07**	838.93	877.00	862.06	868.34	789.06	832.06	875.76	783.23
DTS04	(10 × 5 × 2 × 6)	1064.98	**806.56**	1037.75	1066.88	959.35	860.99	1079.38	992.97	1119.74	951.24
DTS05	(10 × 5 × 2 × 6)	733.94	**541.00**	642.82	700.23	641.14	660.55	557.01	635.74	723.65	586.31
DTS06	(15 × 5 × 2 × 6)	1062.01	883.30	960.19	1113.51	1102.67	1087.96	916.27	1076.27	1005.49	**865.92**
DTS07	(15 × 5 × 2 × 6)	1151.27	**813.22**	1161.27	1216.01	1191.63	1114.00	1066.59	1085.14	1021.49	961.01
DTS08	(15 × 5 × 2 × 6)	1385.26	1174.17	1442.65	1339.30	1445.77	1149.88	1270.70	1375.56	1344.21	**1147.01**
DTS09	(15 × 5 × 2 × 6)	1253.24	1023.99	1190.72	1148.79	1175.86	1151.28	1096.48	1155.99	1133.82	**971.10**
DTS10	(15 × 5 × 2 × 6)	1054.21	**892.66**	1012.75	967.62	1081.33	1051.50	949.14	1022.06	1040.72	904.37
DTS11	(20 × 10 × 4 × 8)	1953.85	**1715.95**	1915.19	1957.85	2006.20	1883.46	1849.18	1966.94	1962.67	1736.41
DTS12	(20 × 10 × 4 × 8)	2494.22	2140.05	2505.62	2378.49	2549.42	2459.56	2176.28	2447.13	2281.03	**2058.08**
DTS13	(20 × 10 × 4 × 8)	2535.57	2315.86	2470.58	2842.95	2408.97	2396.38	2253.31	2556.40	2467.45	**2233.14**
DTS14	(20 × 10 × 4 × 8)	2418.38	**1911.06**	2232.89	2466.23	2466.74	2333.71	2213.34	2233.13	2428.07	2075.86
DTS15	(20 × 10 × 4 × 8)	2756.02	2432.80	2636.70	2672.80	2762.27	2725.48	2432.22	2662.01	2469.08	**2330.98**
DTS16	(30 × 10 × 4 × 8)	3191.56	2767.23	3019.43	3145.71	3173.97	2858.04	2890.06	3064.89	2956.48	**2710.78**
DTS17	(30 × 10 × 4 × 8)	2546.59	2407.14	2590.19	2532.12	2736.06	2789.14	2483.45	2552.98	2636.94	**2282.41**
DTS18	(30 × 10 × 4 × 8)	3123.97	2724.25	3023.06	3201.32	3044.44	3242.70	2701.11	3060.34	2993.41	**2665.75**
DTS19	(30 × 10 × 4 × 8)	2565.67	2191.83	2445.05	2569.51	2475.30	2384.91	2348.01	2443.58	2464.71	**2181.29**
DTS20	(30 × 10 × 4 × 8)	2502.25	2158.45	2324.29	2517.05	2445.94	2514.11	2227.46	2398.20	2412.91	**2074.92**
DTS21	(40 × 10 × 6 × 10)	2681.81	2569.60	2522.70	2654.32	2742.73	2729.60	2637.54	2655.97	2693.10	**2499.69**
DTS22	(40 × 10 × 6 × 10)	2653.47	2464.33	2714.39	2644.74	2691.51	2716.40	2460.68	2731.46	2672.32	**2446.56**
DTS23	(40 × 10 × 6 × 10)	3128.76	2716.77	2827.70	2875.99	2958.36	2880.92	2694.69	3016.58	2993.60	**2622.10**
DTS24	(40 × 10 × 6 × 10)	2855.68	**2394.92**	2676.84	2887.37	2977.64	2792.09	2663.91	2921.10	2855.45	2488.06
DTS25	(40 × 10 × 6 × 10)	2974.84	2808.39	3032.13	2835.49	3098.25	2924.71	2876.17	3184.68	3084.48	**2793.28**
DTS26	(60 × 10 × 6 × 10)	3874.44	3665.47	3930.54	3865.20	3957.28	4006.12	3680.22	4181.81	4131.68	**3650.88**
DTS27	(60 × 10 × 6 × 10)	3530.66	3149.57	3392.50	3188.14	3573.73	3422.59	3237.36	3372.05	3374.96	**3146.22**
DTS28	(60 × 10 × 6 × 10)	3475.73	3503.52	3522.61	3403.31	3609.90	3556.47	3251.06	3634.88	3578.61	**3054.25**
DTS29	(60 × 10 × 6 × 10)	3665.71	3442.74	3755.09	3751.04	3892.93	3620.14	3624.49	3799.18	3701.53	**3410.41**
DTS30	(60 × 10 × 6 × 10)	3535.31	3105.57	3333.97	3451.64	3669.49	3587.53	3263.70	3505.19	3327.97	**3066.11**

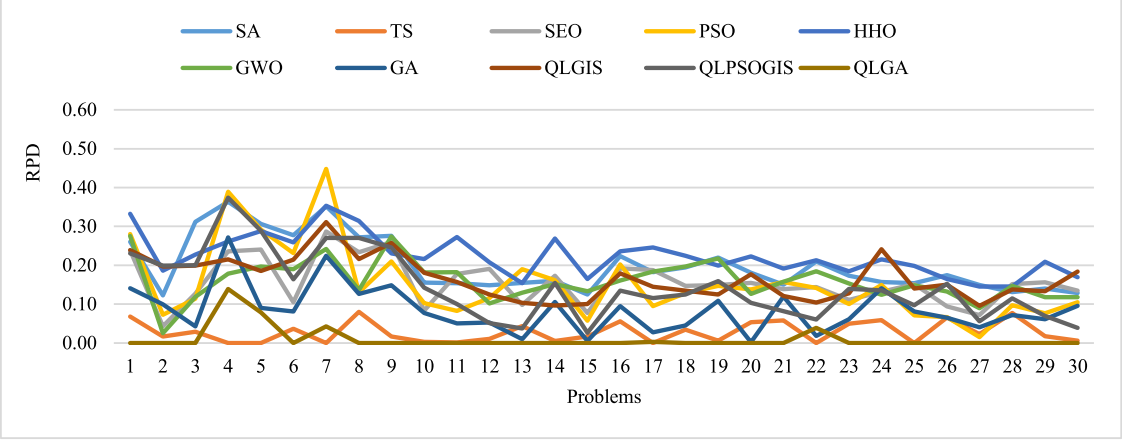


Fig. 19. The RPD values of static problems.

Rescheduling has been shown to be effective in optimizing scheduling performance by achieving an average improvement rate of 8.7%. Because of the severity of disruptions and the ability of the rescheduling mechanism to reallocate tasks efficiently, certain problem instances show a greater improvement in performance. As a result of these findings, it is evident that dynamic scheduling adjustments are critical to the success of CMg environments, which ensures a better utilization of resources and a reduction of delays.

5.7. Sensitivity analysis

A sensitivity analysis was conducted to determine the model's robustness and performance under varying operational conditions. Both static and dynamic problem scenarios are examined in this analysis to see how changes in logistics and processing times affect the maximum completion time. These parameters directly impacts the efficiency of the scheduling and rescheduling [103].

Sensitivity analysis results provide valuable insight into the system's responsiveness to changes in logistics and processing times. First, we examined the effect of varying logistics time. Fig. 22 shows the impact of processing time on the objective function of scheduling and rescheduling problems. There is a strong correlation between the processing time and the maximum completion time, both in static and dynamic problems. This relationship suggests that longer processing times result in longer completion times and greater resource consumption [104].

When processing time was reduced by 40%, the objective value decreased by 26.0%, with smaller but still significant reductions as it decreased. The maximum completion time also increased with an increase in processing time, with the highest increase of 15.2% when processing time was increased by 40%. Processing time directly

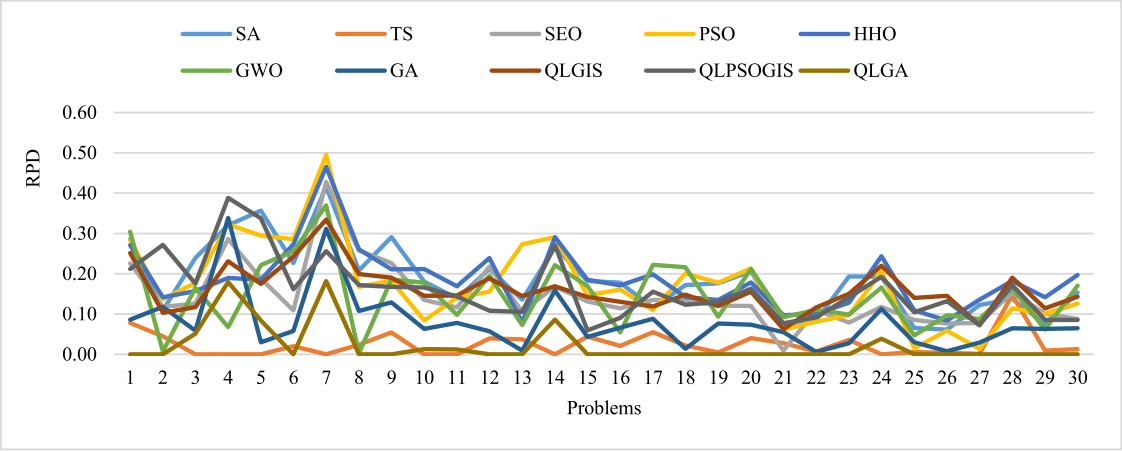


Fig. 20. The RPD values of dynamic problems.

Table 20
Performance comparison of QLGA/D and QLGA across different size benchmark problems.

Problems	Small size problems		Problems	Medium size problems		Problems	Large size problems	
	QLGA/D	QLGA		QLGA/D	QLGA		QLGA/D	QLGA
DTS01	903.08	**793.97**	DTS11	1925.12	**1836.28**	DTS21	2583.53	**2520.89**
DTS02	**877.71**	917.00	DTS12	**2243.15**	2283.50	DTS22	**2472.03**	2622.40
DTS03	**758.24**	800.72	DTS13	**2397.13**	2452.35	DTS23	**2662.57**	2734.47
DTS04	**976.44**	982.25	DTS14	2268.49	**2175.59**	DTS24	2551.00	**2528.19**
DTS05	623.92	**621.94**	DTS15	**2512.66**	2574.90	DTS25	2980.99	**2838.42**
DTS06	**914.06**	919.97	DTS16	2913.57	**2727.84**	DTS26	**3573.05**	3726.19
DTS07	1047.36	**971.56**	DTS17	2606.52	**2435.35**	DTS27	**3054.56**	3233.22
DTS08	1247.08	**1207.63**	DTS18	3018.05	**2843.24**	DTS28	3455.14	**3258.19**
DTS09	1090.04	**1030.04**	DTS19	2355.33	**2267.37**	DTS29	3648.91	**3500.85**
DTS10	954.79	**937.68**	DTS20	2256.64	**2252.70**	DTS30	3292.83	**3208.88**

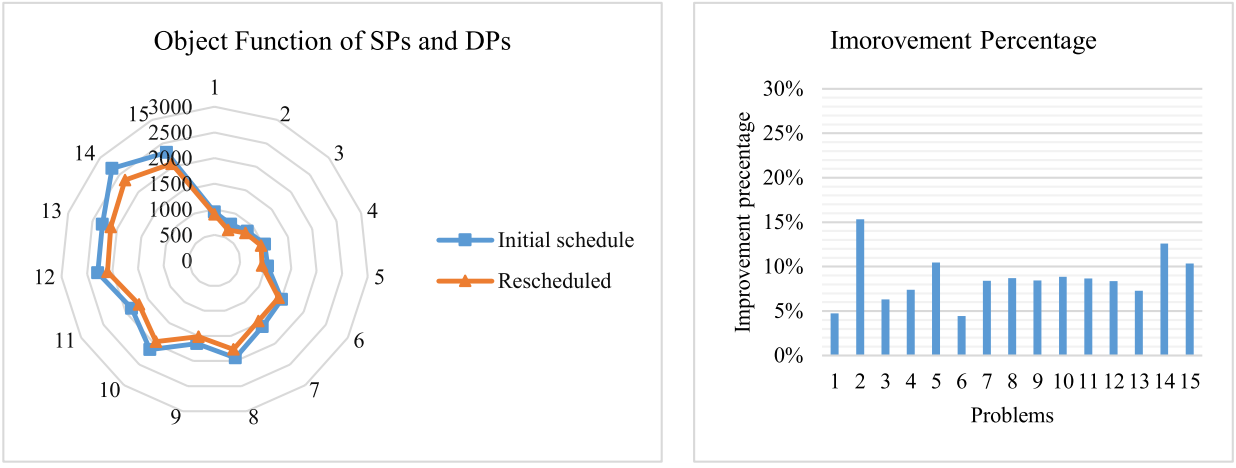


Fig. 21. The effect of rescheduling on initial schedule.

influences completion time, with significant improvements occurring when processing time is reduced. The dynamic problem was even more affected by processing time changes. The completion time decreased by 29.0% when processing time was reduced by 40%, and this reduction continued for smaller changes.

A similar pattern emerged when examining the effect of changes in logistics time (see Fig. 23). Observing the static problem, we found that as logistics time decreased, the objective function also decreased significantly. For a 40% reduction in logistics time, 24.4% of the maximum completion time was reduced, followed by 12.7%, 11.1%, and 5.7% reductions for a 30% reduction in logistics time, 20% reduction in logistics time, and 10% reduction in logistics time, respectively. In the case of a static problem, an increase in logistics time led to an increase in the maximum completion time. For a dynamic problem, an increase in logistics time caused a reduction in maximum completion time, and the reductions continued similarly, but in a lesser magnitude.

In both static and dynamic scenarios, logistics and processing times significantly influence the maximum completion time. This emphasizes the importance of optimizing logistics and processing times. Managing logistics and processing time efficiently is crucial for optimizing overall operational efficiency in dynamic scheduling systems, where uncertain and variable factors are more prominent.

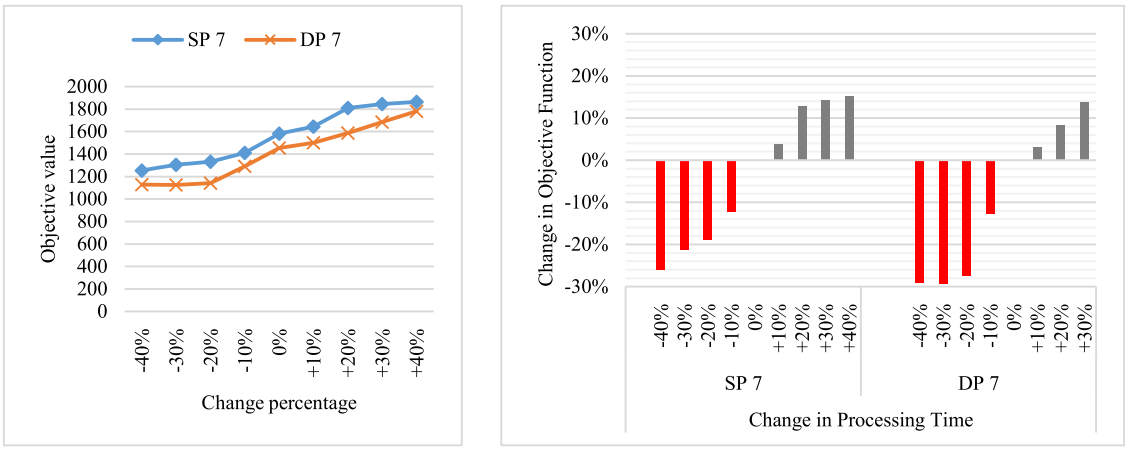


Fig. 22. The impact of processing time on maximum completion time.

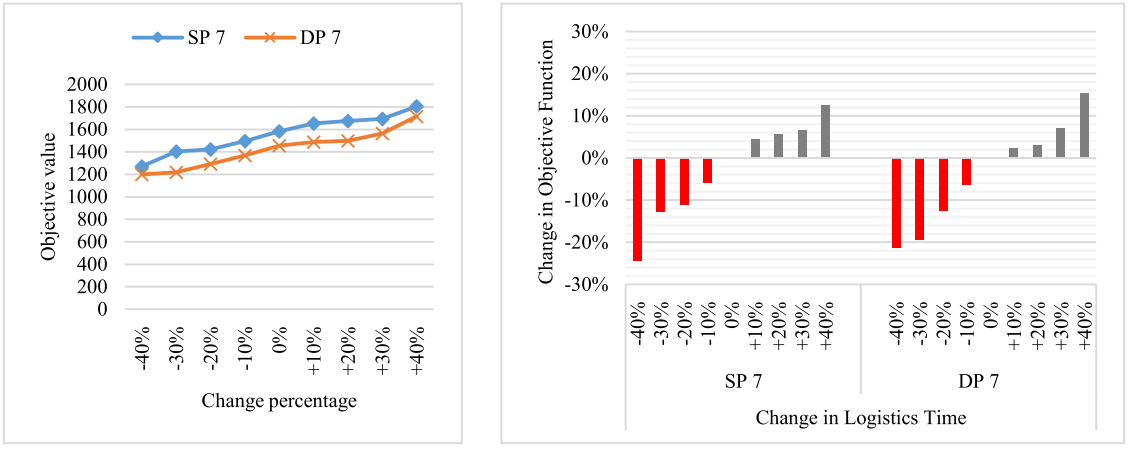


Fig. 23. The impact of logistic time on maximum completion time.

5.8. Managerial insights

Flexibility and adaptability are crucial components of dynamic scheduling environments, as demonstrated by the sensitivity analysis and rescheduling impact analysis. Managing disruptions in dynamic environments can be made easier with rescheduling, as demonstrated in the analysis. It is evident that rescheduling can provide substantial performance enhancements based on the 8.7% average improvement in maximum completion time achieved. In light of the significant role rescheduling plays, managers should implement real-time scheduling systems, which can continuously monitor and adjust schedules in real-time. As a result, resource utilization will be enhanced, while operations will run more smoothly and efficiently despite disruptions.

As a result of the sensitivity analysis, it was discovered that fluctuations in logistics and processing times can have a significant impact on maximum completion times. In industries like cloud manufacturing, logistics and processing times can be highly variable, so this finding has important managerial implications. Accordingly, managers should invest in strategies to reduce variability in logistics and processing times as well as rescheduling. Supply chains can be optimized, transportation can be improved, and production processes can be automated to reduce the impact of disruptions. In order to minimize the negative impact of such fluctuations on the system's performance, managers should prioritize robust scheduling systems that can accommodate such fluctuations.

As a final note, the findings emphasize the need to improve scheduling algorithms and operational processes continuously. Rescheduling can optimize completion times significantly, but its effectiveness can be greatly enhanced when paired with a systematic approach to reducing logistics and processing time variability. In order to anticipate potential disruptions and optimize scheduling adjustments ahead of time, managers should invest in predictive models and machine learning tools. Companies can improve operational efficiency and a competitive edge in the market by integrating these tools with rescheduling mechanisms, ultimately leading to an improved scheduling system and increased productivity.

6. Conclusion

A proactive-reactive approach was used to address the dynamic TSSA problem in response to task cancellation in the CMg system. First, a mixed-integer programming model was developed to minimize the maximum time required to complete tasks with a static status. To provide an event-driven rescheduling framework, this model was extended into a dynamic problem. The problem is NP-hard, which led to the development of a Q-learning-based metaheuristic algorithm (QLGA) for solving medium- and large-scale problems. A number of classical and recently developed metaheuristics were compared to the proposed algorithm through numerical and benchmark examples.

In the study of population-based algorithms, HHO and PSO performed similarly to each other, while GWO, GA, and QLGA performed well, with QLGA providing the best results. In terms of single-solution algorithms, TS outperforms the others, SEO is moderate, and SA is weak. The QLGA solution is recommended if achieving the highest level of quality is a primary objective. Also, if we consider the time, SA's

speed is ideal, and if we want to consider both solution quality and computational time, TS algorithm works best. Additionally, the results indicated that the maximum completion time of all problems was reduced after the rescheduling stage. This suggests that task cancellations can be managed more effectively with the rescheduling model. In addition, sensitivity analysis shows how logistics and processing time directly affect rescheduling and scheduling problems.

While this study focused on operator-level adaptation using Q-learning in response to task scheduling in cloud manufacturing, several directions remain open for future investigation. One possibility is to extend the current framework to incorporate sustainability criteria such as energy consumption during rescheduling. In addition, other dynamic events, such as rework operations, unexpected arrivals of new tasks, machine breakdowns, and material shortages, can be considered to better reflect real-world uncertainty. Moreover, the use of more advanced reinforcement learning techniques, such as deep Q-networks or other deep reinforcement learning methods, may offer greater flexibility and scalability for handling larger and more complex scheduling environments. Finally, to enhance the generalizability and reusability of the RL-based framework, future research could explore transfer learning approaches, domain adaptation techniques, or modular policy architectures.

Notation

The current study uses the ChatGPT AI tool for passage checking and grammatical revision. We guarantee the validity of the passing and human knowledge verifies its accuracy.

CRediT authorship contribution statement

Atefeh Rajabi-Kafshgar: Writing – original draft, Resources, Methodology. **Mostafa Hajiaghaei-Keshteli:** Writing – review & editing, Supervision, Methodology. **Mohammad Reza Mohammad Aliha:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.rcim.2025.103160.

Data availability

Data will be made available on request.

References

[1] V. Gharibvand, M.K. Kolamroudi, Q. Zeeshan, Z.M. Çınar, S. Sahmani, M. Asmael, B. Safaei, Cloud based manufacturing: A review of recent developments in architectures, technologies, infrastructures, platforms and associated challenges, The Int. J. Adv. Manuf. Technol. 131 (1) (2024) 93–123, https://doi.org/10.1007/s00170-024-12989-y.

[2] S. Chiappa, E. Videla, V. Viana-Céspedes, P. Piñeyro, D.A. Rossit, Cloud manufacturing architectures: State-of-art, research challenges and platforms description, J. Ind. Inf. Integr. 34 (2023) 100472, https://doi.org/10.1016/j.jii.2023.100472.

[3] M.S. Kavre, V.K. Sunnapwar, B.B. Gardas, Cloud manufacturing adoption: a comprehensive review, Inform. Syst. E-Business Manag. (2023) 1–71, https://doi.org/10.1007/s10257-023-00638-y.

[4] L. Guo, Y. He, C. Wan, Y. Li, L. Luo, From cloud manufacturing to cloud–edge collaborative manufacturing, Robot. Comput. Integr. Manuf. 90 (2024) 102790, https://doi.org/10.1016/j.rcim.2024.102790.

[5] T. Yang, Y. Ding, W. Chen, Trustworthy collaborative evaluation of multi-service subjects in the cloud manufacturing model, Alex. Eng. J. 113 (2025) 1–11, https://doi.org/10.1016/j.aej.2024.11.021.

[6] J. Ren, Y. Cheng, Y. Zhang, F. Tao, Benchmarking for platform-aggregated manufacturing service collaboration: methodology and implementation, Robot. Comput. Integr. Manuf. 91 (2025) 102853, https://doi.org/10.1016/j.rcim.2024.102853.

[7] Y. Liu, L. Wang, X.V. Wang, X. Xu, L. Zhang, Scheduling in cloud manufacturing: state-of-the-art and research challenges, Int. J. Prod. Res. 57 (15–16) (2019) 4854–4879, https://doi.org/10.1080/00207543.2018.1449978.

[8] M. Ghaleb, H. Zolfagharinia, S. Taghipour, Real-time production scheduling in the industry-4.0 context: addressing uncertainties in job arrivals and machine breakdowns, Computers and Operations Research 123 (2020) 105031, https://doi.org/10.1016/j.cor.2020.105031.

[9] N. Zhu, G. Gong, D. Lu, D. Huang, N. Peng, H. Qi, An effective reformative memetic algorithm for distributed flexible job-shop scheduling problem with order cancellation, Expert. Syst. Appl. 237 (2024) 121205, https://doi.org/10.1016/j.eswa.2023.121205.

[10] M. Khadivi, T. Charter, M. Yaghoubi, M. Jalayer, M. Ahang, A. Shojaeinasab, H. Najjaran, Deep reinforcement learning for machine scheduling: methodology, the state-of-the-art, and future directions, Comput. Ind. Eng. 200 (2025) 110856, https://doi.org/10.1016/j.cie.2025.110856.

[11] E.J. Ghomi, A.M. Rahmani, N.N. Qader, Cloud manufacturing: challenges, recent advances, open research issues, and future trends, The Int. J. Adv. Manuf. Technol. 102 (2019) 3613–3639, https://doi.org/10.1007/s00170-019-03398-7.

[12] R. Rashidifar, H. Bouzary, F.F. Chen, Resource scheduling in cloud-based manufacturing system: a comprehensive survey, The Int. J. Adv. Manuf. Technol. 122 (2022) 4201–4219, https://doi.org/10.1007/s00170-022-09873-y.

[13] A. Elgendy, J. Yan, M. Zhang, A parallel distributed genetic algorithm using Apache Spark for flexible scheduling of multitasks in a cloud manufacturing environment, Int. J. Comput. Integr. Manuf. 37 (5) (2024) 652–667, https://doi.org/10.1080/0951192X.2023.2228277.

[14] Y. Liu, X. Xu, L. Zhang, L. Wang, R.Y. Zhong, Workload-based multi-task scheduling in cloud manufacturing, Robot. Comput. Integr. Manuf. 45 (2017) 3–20, https://doi.org/10.1016/j.rcim.2016.09.008.

[15] H. Akbaripour, M. Houshmand, T. van Woensel, N. Mutlu, Cloud manufacturing service selection optimization and scheduling with transportation considerations: mixed-integer programming models, The Int. J. Adv. Manuf. Technol. 95 (2018) 43–70, https://doi.org/10.1007/s00170-017-1167-3.

[16] M.M. Fazeli, Y. Farjami, M. Nickray, An ensemble optimisation approach to service composition in cloud manufacturing, Int. J. Comput. Integr. Manuf. 32 (1) (2019) 83–91, https://doi.org/10.1080/0951192X.2018.1550679.

[17] Y. Laili, S. Lin, D. Tang, Multi-phase integrated scheduling of hybrid tasks in cloud manufacturing environment, Robot. Comput. Integr. Manuf. 61 (2020) 101850, https://doi.org/10.1016/j.rcim.2019.101850.

[18] B. Vahedi-Nouri, R. Tavakkoli-Moghaddam, Z. Hanzálek, H. Arbabi, M. Rohaninejad, Incorporating order acceptance, pricing and equity considerations in the scheduling of cloud manufacturing systems: matheuristic methods, Int. J. Prod. Res. 59 (7) (2021) 2009–2027, https://doi.org/10.1080/00207543.2020.1806370.

[19] S. Valizadeh, O. Fatahi Valilai, M. Houshmand, Allocation and scheduling of digital dentistry services in a dental cloud manufacturing system, Int. J. Comput. Integr. Manuf. 35 (6) (2022) 645–661, https://doi.org/10.1080/0951192X.2021.1992668.

[20] M. Chen, J. Xu, W. Zhang, Z. Li, A new customer-oriented multi-task scheduling model for cloud manufacturing considering available periods of services using an improved hyper-heuristic algorithm, Expert. Syst. Appl. 269 (2025) 126419, https://doi.org/10.1016/j.eswa.2025.126419.

[21] G.X. Peng, Y.P. Wen, J.X. Liu, G.S. Kang, B.M. Zhang, M.H. Zhou, Energy-aware cloud manufacturing service selection and scheduling optimization, Int. J. Comput. Integr. Manuf. (2024) 1–26, https://doi.org/10.1080/0951192X.2024.2333024.

[22] Y. Ping, Y. Liu, L. Zhang, L. Wang, X. Xu, Sequence generation for multi-task scheduling in cloud manufacturing with deep reinforcement learning, J. Manuf. Syst. 67 (2023) 315–337, https://doi.org/10.1016/j.jmsy.2023.02.009.

[23] W. Zhang, J. Xiao, W. Liu, Y. Sui, Y. Li, S. Zhang, Individualized requirement-driven multi-task scheduling in cloud manufacturing using an extended multifactorial evolutionary algorithm, Comput. Ind. Eng. 179 (2023) 109178, https://doi.org/10.1016/j.cie.2023.109178.

[24] A. Salmasnia, Z. Kiapasha, M. Pashaeenejad, Subtasks scheduling of tasks with different structures in cloud manufacturing systems under maintenance policy and focusing on logistics, tardiness, and earliness aspects, Operational Research 24 (3) (2024) 48, https://doi.org/10.1007/s12351-024-00857-2.

[25] M. Wang, P. Zhang, G. Zhang, K. Sun, J. Zhang, M. Jin, A resilient scheduling framework for multi-robot multi-station welding flow shop scheduling against robot failures, Robot. Comput. Integr. Manuf. 91 (2025) 102835, https://doi.org/10.1016/j.rcim.2024.102835.

[26] C. Destouet, H. Tlahig, B. Bettayeb, B. Mazari, Flexible job shop scheduling problem under Industry 5.0: A survey on human reintegration, environmental consideration and resilience improvement, J. Manuf. Syst. 67 (2023) 155–173, https://doi.org/10.1016/j.jmsy.2023.01.004.

[27] X. Zhang, Y. Han, G. Królczyk, M. Rydel, R. Stanislawski, Z. Li, Rescheduling of distributed manufacturing system with machine breakdowns, Electronics. (Basel) 11 (2) (2022) 249, https://doi.org/10.3390/electronics11020249.

[28] L. Haghnegahdar, S.S. Joshi, N.B. Dahotre, From IoT-based cloud manufacturing approach to intelligent additive manufacturing: industrial internet of Things—An

overview, Int. J. Adv. Manuf. Technol. 119 (3) (2022) 1461–1478, https://doi.org/10.1007/s00170-021-08436-x.
[29] L. Zhou, L. Zhang, B.R. Sarker, Y. Laili, L. Ren, An event-triggered dynamic scheduling method for randomly arriving tasks in cloud manufacturing, Int. J. Comput. Integr. Manuf. 31 (3) (2018) 318–333, https://doi.org/10.1080/0951192X.2017.1413252.
[30] C. Yang, T. Peng, S. Lan, W. Shen, L. Wang, Towards IoT-enabled dynamic service optimal selection in multiple manufacturing clouds, J. Manuf. Syst. 56 (2020) 213–226, https://doi.org/10.1016/j.jmsy.2020.06.004.
[31] M. Mahmoodjanloo, R. Tavakkoli-Moghaddama, A. Baboli, A. Bozorgi-Amiri, Distributed job-shop rescheduling problem considering reconfigurability of machines: a self-adaptive hybrid equilibrium optimiser, Int. J. Prod. Res. 60 (16) (2022) 4973–4994, https://doi.org/10.1080/00207543.2021.1946193.
[32] K. Zhu, G. Gong, N. Peng, L. Zhang, D. Huang, Q. Luo, X. Li, Dynamic distributed flexible job-shop scheduling problem considering operation inspection, Expert. Syst. Appl. 224 (2023) 119840, https://doi.org/10.1016/j.eswa.2023.119840.
[33] W. Xiong, M.K. Lim, M.-L. Tseng, Y. Wang, An effective adaptive adjustment model of task scheduling and resource allocation based on multi-stakeholder interests in cloud manufacturing, Advanced Engineering Informatics 56 (2023) 101937, https://doi.org/10.1016/j.aei.2023.101937.
[34] J. Wang, Z. Wu, L. Yang, W. Hu, C. Song, Z. Zhu, X. Guo, P. Cao, Investigation on distributed rescheduling with cutting tool maintenance based on NSGA-III in large-scale panel furniture intelligent manufacturing, J. Manuf. Process. 112 (2024) 214–224, https://doi.org/10.1016/j.jmapro.2024.01.016.
[35] M. Sun, J. Ding, Z. Zhao, J. Chen, G.Q. Huang, L. Wang, Out-of-order execution enabled deep reinforcement learning for dynamic additive manufacturing scheduling, Robot. Comput. Integr. Manuf. 91 (2025) 102841, https://doi.org/10.1016/j.rcim.2024.102841.
[36] A. Salmasnia, Z. Kiapasha, Integration of sub-task scheduling and logistics in cloud manufacturing systems under setup time and different task arrival times, Int. J. Comput. Integr. Manuf. 36 (7) (2023) 985–1008, https://doi.org/10.1080/0951192X.2022.2162595.
[37] F. Li, T.W. Liao, L. Zhang, Two-level multi-task scheduling in a cloud manufacturing environment, Robot. Comput. Integr. Manuf. 56 (2019) 127–139, https://doi.org/10.1016/j.rcim.2018.09.002.
[38] Y. Hu, F. Zhu, L. Zhang, Y. Lui, Z. Wang, Scheduling of manufacturers based on chaos optimization algorithm in cloud manufacturing, Robot. Comput. Integr. Manuf. 58 (2019) 13–20, https://doi.org/10.1016/j.rcim.2019.01.010.
[39] T. Wang, P. Zhang, J. Liu, L. Gao, Multi-user-oriented manufacturing service scheduling with an improved NSGA-II approach in the cloud manufacturing system, Int. J. Prod. Res. 60 (8) (2022) 2425–2442, https://doi.org/10.1080/00207543.2021.1893851.
[40] H. Zhang, K. Li, C. Chu, Z. Jia, Parallel batch processing machines scheduling in cloud manufacturing for minimizing total service completion time, Comput. Oper. Res. 146 (2022) 105899, https://doi.org/10.1016/j.cor.2022.105899.
[41] Z. Chen, L. Zhang, X. Wang, K. Wang, Cloud–edge collaboration task scheduling in cloud manufacturing: an attention-based deep reinforcement learning approach, Comput. Ind. Eng. 177 (2023) 109053, https://doi.org/10.1016/j.cie.2023.109053.
[42] S. Liu, Q. Deng, X. Liu, Q. Luo, F. Li, C. Jiang, Dual-service integrated scheduling of manufacturing and logistics for multiple tasks in cloud manufacturing, Expert. Syst. Appl. 235 (2024) 121129, https://doi.org/10.1016/j.eswa.2023.121129.
[43] Z. Zhao, H. Zhou, W. Zheng, Collaborative optimization for scheduling manufacturing tasks and transport vehicles considering manufacturer's time availability in cloud manufacturing, Int. J. Comput. Integr. Manuf. (2024) 1–26, https://doi.org/10.1080/0951192X.2024.2382211.
[44] J. Xiao, Y. Cai, Y. Chen, Study on deep reinforcement learning for multi-task scheduling in cloud manufacturing, Int. J. Comput. Integr. Manuf. (2025) 1–18, https://doi.org/10.1080/0951192X.2025.2452981.
[45] J. Ding, Y. Wang, S. Zhang, W. Zhang, Z. Xiong, Robust and stable multi-task manufacturing scheduling with uncertainties using a two-stage extended genetic algorithm, Enterp. Inf. Syst. 13 (10) (2019) 1442–1470, https://doi.org/10.1080/17517575.2019.1656290.
[46] M. Yuan, X. Cai, Z. Zhou, C. Sun, W. Gu, J. Huang, Dynamic service resources scheduling method in cloud manufacturing environment, Int. J. Prod. Res. 59 (2) (2019) 542–559, https://doi.org/10.1080/00207543.2019.1697000.
[47] Z. Dai, Z. Zhang, M. Chen, Collaborative task scheduling with new task arrival in cloud manufacturing using improved multi-population biogeography-based optimization, Journal of Intelligent and Fuzzy Systems 41 (2) (2021) 3849–3872, https://doi.org/10.3233/JIFS-201066.
[48] X. Wang, L. Zhang, Y. Liu, F. Li, Z. Chen, C. Zhao, T. Bai, Dynamic scheduling of tasks in cloud manufacturing with multi-agent reinforcement learning, J. Manuf. Syst. 65 (2022) 130–145, https://doi.org/10.1016/j.jmsy.2022.08.004.
[49] X. Wang, L. Zhang, Y. Liu, Y. Laili, An improved deep reinforcement learning-based scheduling approach for dynamic task scheduling in cloud manufacturing, Int. J. Prod. Res. 62 (11) (2024) 4014–4030, https://doi.org/10.1080/00207543.2023.2253326.
[50] Y. Lei, Q. Deng, M. Liao, S. Gao, Deep reinforcement learning for dynamic distributed job shop scheduling problem with transfers, Expert. Syst. Appl. 251 (2024) 123970, https://doi.org/10.1016/j.eswa.2024.123970.
[51] F.B. Ozsoydan, A trajectory-based algorithm enhanced by Q-learning and cloud integration for hybrid flexible flowshop scheduling problem with sequence-dependent setup times: A case study, Computers and Operations Research 181 (2025) 107079, https://doi.org/10.1016/j.cor.2025.107079.
[52] H. Yu, K.Z. Gao, Z.F. Ma, Y.X. Pan, Improved meta-heuristics with Q-learning for solving distributed assembly permutation flowshop scheduling problems, Swarm. Evol. Comput. 80 (2023) 101335, https://doi.org/10.1016/j.swevo.2023.101335.
[53] Y. Hou, H. Wang, X. Huang, A Q-learning-based multi-objective evolutionary algorithm for integrated green production and distribution scheduling problems, Eng. Appl. Artif. Intell. 127 (2024) 107434, https://doi.org/10.1016/j.engappai.2023.107434.
[54] B. Lu, K. Gao, Y. Ren, D. Li, A. Slowik, Combining meta-heuristics and Q-learning for scheduling lot-streaming hybrid flow shops with consistent sublots, Swarm. Evol. Comput. 91 (August) (2024) 1–18, https://doi.org/10.1016/j.swevo.2024.101731.
[55] F.B. Ozsoydan, Reinforcement learning enhanced swarm intelligence and trajectory-based algorithms for parallel machine scheduling problems, Computers and Industrial Engineering 203 (2025) 110948, https://doi.org/10.1016/j.cie.2025.110948.
[56] B. Qiu, K. Gao, H. Yu, A. Sadollah, Modelling and scheduling distributed assembly permutation flow-shops using reinforcement learning-based evolutionary algorithms, Eng. Appl. Artif. Intell. 142 (2025) 109851, https://doi.org/10.1016/j.engappai.2024.109851.
[57] L. Cheng, Q. Tang, L. Zhang, C. Yu, Computers & Industrial Engineering scheduling flexible manufacturing cell with no-idle flow-lines and job-shop via Q-learning-based genetic algorithm, Comput. Ind. Eng. 169 (2022) 108293, https://doi.org/10.1016/j.cie.2022.108293.
[58] Z. Liang, R. Yang, J. Wang, L. Liu, X. Ma, Z. Zhu, Dynamic constrained evolutionary optimization based on deep Q-network, Expert. Syst. Appl. 249 (2024) 123592, https://doi.org/10.1016/j.eswa.2024.123592.
[59] H. Tang, Y. Xiao, W. Zhang, D. Lei, J. Wang, T. Xu, A DQL-NSGA-III algorithm for solving the flexible job shop dynamic scheduling problem, Expert. Syst. Appl. 237 (2024) 121723, https://doi.org/10.1016/j.eswa.2023.121723.
[60] M. Karimi-Mamaghan, M. Mohammadi, B. Pasdeloup, P. Meyer, Learning to select operators in meta-heuristics: an integration of Q-learning into the iterated greedy algorithm for the permutation flowshop scheduling problem, Eur J Oper Res 304 (3) (2023) 1296–1330, https://doi.org/10.1016/j.ejor.2022.03.054.
[61] Y. Ren, K. Gao, Y. Fu, H. Sang, D. Li, Z. Luo, A novel Q-learning based variable neighborhood iterative search algorithm for solving disassembly line scheduling problems, Swarm. Evol. Comput. 80 (November 2022) (2023) 101338, https://doi.org/10.1016/j.swevo.2023.101338.
[62] W. Zhang, H. Geng, C. Li, M. Gen, G. Zhang, M. Deng, Q-learning-based multi-objective particle swarm optimization with local search within factories for energy-efficient distributed flow-shop scheduling problem, J. Intell. Manuf. (2023), https://doi.org/10.1007/s10845-023-02227-9.
[63] Y. Hu, L. Zhang, Z. Zhang, Z. Li, Q. Tang, Engineering applications of Artificial Intelligence matheuristic and learning-oriented multi-objective artificial bee colony algorithm for energy-aware flexible assembly job shop scheduling problem, Eng. Appl. Artif. Intell. 133 (PF) (2024) 108634, https://doi.org/10.1016/j.engappai.2024.108634.
[64] Z. Zhang, Z. Shao, W. Shao, J. Chen, D. Pi, MRLM: A meta-reinforcement learning-based metaheuristic for hybrid flow-shop scheduling problem with learning and forgetting effects, Swarm. Evol. Comput. 85 (August 2023) (2024), https://doi.org/10.1016/j.swevo.2024.101479.
[65] Y. Yao, X. Li, L. Gao, A DQN-based memetic algorithm for energy-efficient job shop scheduling problem with integrated limited AGVs, Swarm. Evol. Comput. 87 (2024) 101544, https://doi.org/10.1016/j.swevo.2024.101544.
[66] X. Chang, X. Jia, J. Ren, A reinforcement learning enhanced memetic algorithm for multi-objective flexible job shop scheduling toward Industry 5.0, Int. J. Prod. Res. 63 (1) (2025) 119–147, https://doi.org/10.1080/00207543.2024.2357740.
[67] G.A. Rolim, C.P. Tomazella, M.S. Nagano, On the integration of reinforcement learning and simulated annealing for the parallel batch scheduling problem with setups, Eur J Oper Res 326 (2) (2025) 220–233, https://doi.org/10.1016/j.ejor.2025.04.042.
[68] F. Tao, D. Zhao, Y. Hu, Z. Zhou, Resource service composition and its optimal-selection based on particle swarm optimization in manufacturing grid system, IEEe Trans. Industr. Inform. 4 (4) (2008) 315–327, https://doi.org/10.1109/TII.2008.2009533.
[69] E. Ahmadi, M. Zandieh, M. Farrokh, S.M. Emami, A multi objective optimization approach for flexible job shop scheduling problem under random machine breakdown by evolutionary algorithms, Comput. Oper. Res. 73 (2016) 56–66, https://doi.org/10.1016/j.cor.2016.03.009.
[70] H. Tong, J. Zhu, A novel method for customer-oriented scheduling with available manufacturing time windows in Cloud Manufacturing, Robot. Comput. Integr. Manuf. 75 (2022) 102303, https://doi.org/10.1016/j.rcim.2021.102303.
[71] Y. Li, X. Li, L. Gao, Z. Lu, Multi-agent deep reinforcement learning for dynamic reconfigurable shop scheduling considering batch processing and worker cooperation, Robot. Comput. Integr. Manuf. 91 (2025) 102834, https://doi.org/10.1016/j.rcim.2024.102834.
[72] V. Ramakurthi, V.K. Manupati, J. Machado, L. Varela, S. Babu, An innovative approach for resource sharing and scheduling in a sustainable distributed manufacturing system, Advanced Engineering Informatics 52 (2022) 101620, https://doi.org/10.1016/j.aei.2022.101620.
[73] M. Hajiaghaei-Keshteli, M. Aminnayeri, S.M.T. Fatemi Ghomi, Integrated scheduling of production and rail transportation, Computers and Industrial Engineering 74 (1) (2014) 240–256, https://doi.org/10.1016/j.cie.2014.05.026.
[74] M. Hajiaghaei-Keshteli, M. Aminnayeri, Solving the integrated scheduling of production and rail transportation problem by Keshtel algorithm, Appl. Soft. Comput. 25 (2014) 184–203, https://doi.org/10.1016/j.asoc.2014.09.034.
[75] G. Rahmanifar, M. Mohammadi, A. Sherafat, M. Hajiaghaei-Keshteli, G. Fusco, C. Colombaroni, Heuristic approaches to address vehicle routing problem in the

iot-based waste management system, Expert. Syst. Appl. 220 (2023) 119708, https://doi.org/10.1016/j.eswa.2023.119708.
[76] M. Hajiaghaei-Keshteli, Keshtel algorithm, A new optimization algorithm inspired by Keshtel's feeding, in: Proceeding in IEEE Conference on Industrial Engineering and Management Systems, 2013, pp. 2249–2253.
[77] M. Hajiaghaei-Keshteli, G. Rahmanifar, M. Mohammadi, F. Gholian-Jouybari, J. J. Klemeš, S. Zahmatkesh, A. Bokhari, G. Fusco, C. Colombaroni, Designing a multi-period dynamic electric vehicle production-routing problem in a supply chain considering energy consumption, J. Clean. Prod. 421 (2023) 138471, https://doi.org/10.1016/j.jclepro.2023.138471.
[78] M. Hajiaghaei-Keshteli, The allocation of customers to potential distribution centers in supply chain networks: GA and AIA approaches, Appl. Soft. Comput. 11 (2) (2011) 2069–2078, https://doi.org/10.1016/j.asoc.2010.07.004.
[79] F.F. Amir Mohammad, F. Gholian-Jouybari, M.P. Mohammad, H.-K. Mostafa, A Bi-objective stochastic closed-loop supply chain network design problem considering downside risk, Industrial Engineering & Management Systems 16 (3) (2017) 342–362, https://doi.org/10.7232/iems.2017.16.3.342.
[80] T. Borgonjon, B. Maenhout, A genetic algorithm for the personnel task rescheduling problem with time preemption, Expert. Syst. Appl. 238 (2024) 121868, https://doi.org/10.1016/j.eswa.2023.121868.
[81] F. Gholian-Jouybari, M. Hajiaghaei-Keshteli, A. Bavar, A. Bavar, B. Mosallanezhad, A design of a circular closed-loop agri-food supply chain network—A case study of the soybean industry, J. Ind. Inf. Integr. 36 (2023) 100530, https://doi.org/10.1016/j.jii.2023.100530.
[82] M.R. Delavar, M. Hajiaghaei-Keshteli, S. Molla-Alizadeh-Zavardehi, Genetic algorithms for coordinated scheduling of production and air transportation, Expert. Syst. Appl. 37 (12) (2010) 8255–8266, https://doi.org/10.1016/j.eswa.2010.05.060.
[83] F. Glover, Future paths for integer programming and links to artificial intelligence, Comput. Oper. Res. 13 (5) (1986) 533–549, https://doi.org/10.1016/0305-0548(86)90048-1.
[84] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science (1979) 220 (4598) (1983) 671–680, https://doi.org/10.1126/science.220.4598.671.
[85] J.H. Holland, Genetic algorithms, Sci. Am. 267 (1) (1992) 66–72, https://doi.org/10.1038/scientificamerican0792-66.
[86] J. Kennedy, R. Eberhart, Particle swarm optimization, Proceedings of ICNN'95 - International Conference on Neural Networks 4 (1995) 1942–1948, https://doi.org/10.1109/ICNN.1995.488968.
[87] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey Wolf optimizer, Advances in Engineering Software 69 (2014) 46–61, https://doi.org/10.1016/j.advengsoft.2013.12.007.
[88] A.A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, H. Chen, Harris hawks optimization: algorithm and applications, Future Generation Computer Systems 97 (2019) 849–872, https://doi.org/10.1016/j.future.2019.02.028.
[89] A.M. Fathollahi-Fard, M. Hajiaghaei-Keshteli, R. Tavakkoli-Moghaddam, The Social engineering Optimizer (SEO), Eng. Appl. Artif. Intell. 72 (2018) 267–293, https://doi.org/10.1016/j.engappai.2018.04.009.
[90] M. Mohammadi, G. Rahmanifar, M. Hajiaghaei-Keshteli, G. Fusco, C. Colombaroni, A. Sherafat, A dynamic approach for the multi-compartment vehicle routing problem in waste management, Renewable and Sustainable Energy Reviews 184 (2023) 113526, https://doi.org/10.1016/j.rser.2023.113526.
[91] A. Rajabi-Kafshgar, I. Seyedi, E.B. Tirkolaee, Circular closed-loop supply chain network design considering 3D printing and PET bottle waste, Environ. Dev. Sustain. (2024) 1–37, https://doi.org/10.1007/s10668-024-04767-3.
[92] S. Golmohamadi, R. Tavakkoli-Moghaddam, M. Hajiaghaei-Keshteli, Solving a fuzzy fixed charge solid transportation problem using batch transferring by new approaches in meta-heuristic, Electronic Notes in Discrete Mathematics 58 (2017) 143–150, https://doi.org/10.1016/j.endm.2017.03.019.
[93] A. Uzunoglu, C. Gahm, A. Tuma, Machine learning based algorithm selection and genetic algorithms for serial-batch scheduling, Comput. Oper. Res. 173 (2025) 106827, https://doi.org/10.1016/j.cor.2024.106827.
[94] K. Mostafaei, M. Yousefi, O. Kreuzer, M.N. Kianpour, Simulation-based mineral prospectivity modeling and Gray Wolf optimization algorithm for delimiting exploration targets, Ore Geol. Rev. 177 (2025) 106458, https://doi.org/10.1016/j.oregeorev.2025.106458.
[95] A. Rajabi-Kafshgar, F. Gholian-Jouybari, I. Seyedi, M. Hajiaghaei-Keshteli, Utilizing hybrid metaheuristic approach to design an agricultural closed-loop supply chain network. *Expert Systems with Applications* (Vol. 217), Elsevier Ltd, 2023, https://doi.org/10.1016/j.eswa.2023.119504.
[96] A. Seyyedabbasi, A reinforcement learning-based metaheuristic algorithm for solving global optimization problems, Advances in Engineering Software 178 (2023) 103411, https://doi.org/10.1016/j.advengsoft.2023.103411.
[97] T. Wauters, K. Verbeeck, P. De Causmaecker, G. Vanden Berghe, Boosting metaheuristic search using reinforcement learning, Studies in Computational Intelligence 434 (2013) 433–452, https://doi.org/10.1007/978-3-642-30671-6_17.
[98] Q. Luo, Q. Deng, G. Gong, X. Guo, X. Liu, A distributed flexible job shop scheduling problem considering worker arrangement using an improved memetic algorithm, Expert. Syst. Appl. 207 (2022) 117984, https://doi.org/10.1016/j.eswa.2022.117984.
[99] J. Hurink, B. Jurisch, M. Thole, Tabu search for the job-shop scheduling problem with multi-purpose machines, Operations-Research-Spektrum 15 (4) (1994) 205–215, https://doi.org/10.1007/BF01719451.
[100] D. Li, L. Meng, J. Li, K. Lu, Y. Yang, Domain adaptive state representation alignment for reinforcement learning, Inf Sci (Ny) 609 (2022) 1353–1368, https://doi.org/10.1016/j.ins.2022.07.156.
[101] F. Shoeleh, M. Asadpour, Skill based transfer learning with domain adaptation for continuous reinforcement learning domains, Applied Intelligence 50 (2) (2020) 502–518, https://doi.org/10.1007/s10489-019-01527-z.
[102] B. Wang, Domain Adaptation in Reinforcement Learning: approaches, limitations, and future directions, Journal of The Institution of Engineers (India): Series B 105 (5) (2024) 1223–1240, https://doi.org/10.1007/s40031-024-01049-4.
[103] A. Goli, A. Ala, M. Hajiaghaei-Keshteli, Efficient multi-objective meta-heuristic algorithms for energy-aware non-permutation flow-shop scheduling problem, Expert. Syst. Appl. 213 (2023) 119077, https://doi.org/10.1016/j.eswa.2022.119077.
[104] K. Tafakkori, R. Tavakkoli-Moghaddam, A. Siadat, Sustainable negotiation-based nesting and scheduling in additive manufacturing systems: A case study and multi-objective meta-heuristic algorithms, Eng. Appl. Artif. Intell. 112 (2022) 104836, https://doi.org/10.1016/j.engappai.2022.104836.