Li *et al. Green Manuf Open* 2024;2:2
DOI: 10.20517/gmo.2023.091901

Green Manufacturing Open

Research Article

Open Access

Integrating meta-heuristics and a Sarsa algorithm for disassembly scheduling problems with cycle time and hazard coefficients

Dachao Li[1], Kaizhou Gao[1,2], Yaxian Ren[1], Ruixue Zhang[1], Yaping Fu[3]

[1]School of Computer, Liao Cheng University, Liaocheng 252000, Shandong, China.
[2]Macau Institute of Systems Engineering, Macau University of Science and Technology, Taipa 999078, Macao, China.
[3]School of Business, Qingdao University, Qingdao 266071, Shandong, China.

Correspondence to: Dr. Kaizhou Gao, School of Computer, Liao Cheng University, Huxi Street, Dongchangfu District, Liaocheng 252000, Shandong, China. E-mail: gaokaizh@aliyun.com

How to cite this article: Li D, Gao K, Ren Y, Zhang R, Fu Y. Integrating meta-heuristics and a Sarsa algorithm for disassembly scheduling problems with cycle time and hazard coefficients. *Green Manuf Open* 2024;2:2. https://dx.doi.org/10.20517/gmo.2023.091901

Received: 19 Sep 2023 **First Decision:** 4 Jan 2024 **Revised:** 9 Jan 2024 **Accepted:** 19 Jan 2024 **Published:** 29 Jan 2024

Academic Editor: Hongchao Zhang **Copy Editor:** Pei-Yun Wang **Production Editor:** Pei-Yun Wang

Abstract

End-of-life products recycling can reduce the waste of resources, and disassembly line scheduling planning can effectively improve the recycling efficiency and reduce the pollution of the environment. This work addresses a bi-objective disassembly line scheduling problem with considering time interference between tasks. The weighted sum of the cycle time and hazard coefficients is optimized. First, a mathematical model of the disassembly line scheduling problem is established under the constraints of priority and time interference relationships. Second, four meta-heuristics are improved to solve the concerned problems, including particle swarm optimization, artificial bee colony, genetic algorithm and variable neighborhood search. Ten objective-oriented local search operations are designed for improving meta-heuristics' performance. A reinforcement learning algorithm, Sarsa, is employed to guide task assignment among workstations and local search selection during iterations, respectively. Finally, experiments are carried out for 10 instances with different scales. The effectiveness of the improving strategies is verified; the meta-heuristics combined with Sarsa based task assignment and local search strategies has better robustness and stability than the classical ones. Comparisons and discussions show that the particle swarm optimization with improved strategies outperforms other algorithms.

Keywords: Disassembly line scheduling, meta-heuristics, Sarsa algorithm, bi-objective

INTRODUCTION

As economic and societal development progresses, governments are introducing regulations aimed at safeguarding the environment, while people's consciousness regarding environmental protection is steadily increasing. The emergence of recycling economy is playing a pivotal role in propelling the growth of the remanufacturing industry[1]. The rapid pace of product obsolescence, coupled with technological advancements, is considerably shortening the lifespan of products[2]. In contrast to methods such as incineration and landfill, the processes of disassembly and recycling offer notable economic and environmental benefits. Consequently, there is a growing demand for the proper treatment of a substantial volume of discarded products. In this context, the recycling of End-of-Life (EOL) products holds immense significance, yielding advantages for both environmental preservation and the principles underpinning the circular economy. Since the recycling trend is unavoidable, the disassembly of EOL products emerges as a pivotal undertaking, bearing considerable research importance[3,4].

To address the challenges posed by large-scale disassembly, Güngör and Gupta introduce the concept of disassembly lines, underscoring their vital role in product disassembly and recycling[5]. Disassembly sequence planning focuses on optimizing the disassembly sequence of EOL products' components, aiming to minimize disassembly costs, enhance recycling efficiency, and maintain a reasonable level of stability throughout the disassembly process[6]. Planning and scheduling EOL products' disassembly order is a challenge to enhance the efficiency and stability of disassembly lines. The disassembly line scheduling problem (DLSP) has been proven to be NP-hard[7]. Literature[8] introduces the concept of sequentially-dependent disassembly line balance problems (DLBPs) and formulates the corresponding mathematical models. Emrah[9] applies constraint programming for the first time to the disassembly line balancing problem, improving the optimal solution for several medium-sized benchmark instances. ÇİL presents a mixed-integer linear programming (MILP) model for multiplayer disassembly line balancing and develops constraint programming methods to solve large-scale instances[10]. Meng *et al.* present a mixed-integer linear model for solving disassembly planning and scheduling problems[11]. Some publications tackle DLSP using diverse traditional mathematical methods[12-14]. However, they are not suitable for the challenges from large-scale instances in real-life situations[15]. Compared to the traditional mathematical optimization methods, the meta-heuristics are suitable for a wider range of problem structures and can obtain high-quality solutions for large-scale problems quickly. However, meta-heuristics are usually only able to find near-optimal solutions. In recent years, many kinds of advanced meta-heuristics have been proposed for the exploration of decision-making problems in various domains. Chen *et al.* propose a self-adaptive fast fireworks algorithm (SF-FWA) that enables linear computational complexity in terms of problem dimensionality and makes the overall able to automatically adapt to a rich set of function landscapes[16]. Singh *et al.* solve the proposed multi-objective mathematical model for resource allocation through exact approaches and meta-heuristics[17]. Pasha *et al.* design a customized multi-objective hybrid meta-heuristic that directly considers problem-specific attributes to solve a multi-objective optimization model of the vehicle path problem in a box factory[18]. Singh *et al.* extend the research by proposing a novel ant-based generative structural hyper-heuristic to investigate how different pheromone graphs affect their performance[19]. Experiments demonstrate key differences in performance between two different pheromone spaces. Furthermore, researchers in various fields solve decision-making problems through meta-heuristics[20-27].

ZMeta-heuristics are gradually applied for solving disassembly scheduling problems due to their excellent performance in balancing the computation cost and solution quality[28-30], including particle swarm optimization (PSO)[31-36], artificial bee colony (ABC)[37-40], genetic algorithm (GA)[41-46], and variable neighborhood search (VNS)[40,47-50]. Kalayci *et al.* propose a PSO based on variable neighborhood mutation

operators to solve DLBPs[31]. A problem-specific swarm optimization method is designed to modify the adaptive parameters in disassembly sorting, thereby controlling the update mechanism of the disassembly process[34]. Wang *et al.* establish a DLBP model that balances the economy and environment, considering the precedence constraints, and develop a discrete multi-objective ABC algorithm to solve the problem[37]. For solving the DLSP, Zhang *et al.* use a hybrid graph to represent constraints and precedence relationships and propose a hybrid ABC algorithm[39]. Slama *et al.* propose a block-based GA to solve the DLSP[41]. The DLSP is efficiently addressed by Wu *et al.* through the proposal of a hybrid local search GA, along with the implementation of a four-layer encoding and decoding strategy[42]. A new genetic simulated annealing algorithm is proposed by Wang *et al.* to optimize the model[46]. According to the advantages of GA, two-point mapping crossover and single-point insertion mutation operations are constructed to guarantee the sequence's priority and disassembly constraints. To balance the disassembly line, Ren *et al.* combine variable local search algorithms with multi-criteria decision making to propose an improved general VNS (GVNS)[48]. Liu *et al.* formulate a disassembly planning model that revolves around the achievement rate of disassembly[50]. They introduce a comprehensive variable local search algorithm and integrate four distinct local search operators into their approach to mitigate the impact of unforeseeable variables.

In many publications, traditional mathematical methods and meta-heuristics are used to solve various scheduling and optimization problems, aiming to obtain optimal or approximate optimal solutions. In recent years, reinforcement learning algorithms have been employed separately or combined with meta-heuristics for addressing various scheduling problems, including disassembly scheduling problems. Tuncel *et al.* utilize a reinforcement learning algorithm based on Monte Carlo to address the disassembly line problem[51]. It can solve large-scale problems within a reasonable time frame. Zhao *et al.* employ ensemble reinforcement learning to tackle the challenge of structural uncertainty in EOL products, effectively adapting to the optimal disassembly sequence[52]. The literature[53] combines a brainstorming optimization algorithm with reinforcement learning and uses a reward feedback mechanism to guide the selection of four mutation strategies. By integrating metaheuristics and machine learning techniques, Karimi-Mamaghan *et al.* solve combinatorial optimization problems[54]. Furthermore, some researchers employ metaheuristics in conjunction with *Q*-learning strategies within the framework of reinforcement learning to effectively address scheduling problems[55-57].

In summary, most research has predominantly centered on optimizing the disassembly sequence or maximizing factors such as disassembly time, energy consumption, or profit across multiple objectives. Yet, comparatively less attention is directed toward assessing the impact of environmental protection pressures on disassembly time and costs, especially for potential harm from hazardous products. To address this gap, this study focuses on two objectives: cycle time (CT) and the hazard coefficients. The hazard coefficients are related to the position of the hazard tasks in a solution and the average disassembly time of all tasks. The contributions of this paper are given as follows:

(1) A mathematical model is developed for DLSP with optimizing the weighted sum of CT and hazard coefficients.

(2) For task assignment, two workstation allocation strategies are first designed, and Sarsa algorithm is used to select a premium one during iterations.

(3) Sarsa algorithm is employed to guide the selection of ten local search operators, which are designed for improving convergence of meta-heuristics.

The remaining sections of this study are structured as follows: Section "INTRODUCTION" presents a mathematical model for optimizing DLSP with CT and hazard coefficients. In Section "METHODS", the proposed algorithms are described. Experimental results and comparisons are provided in Section

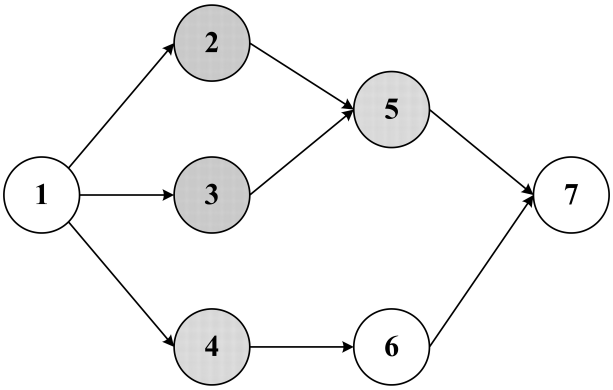

Figure 1. Task disassembly sequence diagram.

"RESULTS". Finally, Section "CONCLUSIONS AND FUTURE WORK" offers a summary of this work with several future directions.

METHODS

Problem description

Disassembly is a comprehensive manufacturing process that detaches a variety of components at multiple workstations. During a disassembly process, n tasks need to be allocated among m workstations. To streamline the complexity of the problem, we assume that all workstations can disassemble any task and are functionally identical[40]. A disassembly task involves a series of operations. Once a task is completed, the resulting components or sub-components can be further disassembled into their constituent parts until the core is entirely disassembled. In essence, the disassembly process is a progressive dismantling of the core into its individual components and subassemblies through a sequence of tasks. Upon satisfying the constraints, we allocate the disassembly tasks to specific workstations on the disassembly line, adhering to the disassembly order that meets the requirements. Once a task is assigned to a workstation, it becomes fixed and cannot be transferred to another workstation. It must be completed at the workstation. The number of available workstations remains constant, while different disassembly sequences can result in variations in disassembly time and hazard coefficients. The DLSP includes three sub-problems: (1) assigning tasks to workstations; (2) task sequencing; and (3) adjusting the task sequence to minimize the CT and hazard coefficients.

In DLSP, there are priority and time interference relationships between disassembly tasks. An example is shown in Figure 1, and the detail data is reported in Table 1. The numbers inside the circle represent disassembly tasks, and the solid arrow represents the priority order between tasks. Task 1 is the predecessor of Tasks 2, 3, and 4, indicating that Task 1 must be disassembled before proceeding with Tasks 2, 3, and 4. Tasks with the same background lines have a time interference relationship between them. Table 1 provides a detailed description of the example. For example, if Task 4 is disassembled first, it will increase the disassembly time of Task 5 by two units, and the actual disassembly time of Task 5 is 3 + 2 = 5 units. Conversely, if Task 5 is disassembled first, it will also increase the actual disassembly time of Task 4. Figure 2 shows the Gantt chart of a solution for the example.

Mathematical models

In this section, a mathematical model of DLSP is developed with minimizing CT and hazard coefficients. The following notations s are used to express the concerned problems, and the detail data is reported in Table 2.

Table 1. The process of task disassembly

Workstation	Disassembly sequence							Run time	Idle time	CT	Hazard coefficients
	1	3	4	6	2	5	7				
1	5		3				5	13	8	21	23.14
2		5 + 1				3 + 2		11	5		
3				4	2			6	6		
	Actual disassembly time										

CT: Cycle time.

Table 2. Notations description

Notation	Description
Parameters	
i	Task index, $i \in \{1, 2, ..., N\}$.
j	Task index, $j \in \{1, 2, ..., N\}$.
k	Index of workstation, $k \in \{1, 2, ..., M\}$.
M	Workstation number.
N	Task number.
t_i	Disassembly time for task i.
t_a	The average processing time of all tasks.
sd_{ij}	The increasing disassembly time of task i, if task j interferes with task i, and j is disassembled before task i.
u_{ij}	If task i and task j have an interference relationship, $u_{ij} = 1$; otherwise, $u_{ij} = 0$.
$t_i^{'}$	The actual disassembly time of task i is the sum of the standard time and the interference time.
B_i	The start time for disassembly task i.
F_i	The completion time of task i.
T_k	The completion of workstation k.
G	An infinite positive number.
P_l	l^{th} element in a disassembly sequence, e.g., for sequence {1, 4, 2, 3, 6, 8, 7, 5}, $P_2 = 4$.
CT	Indicates the maximum completion time of all opening workstations.
D_{ij}	If task i can be disassembled before task j, $D_{ij} = 1$; otherwise, $D_{ij} = 0$.
Decision variables	
x_{ik}	If task i is assigned to workstation k, $x_{ik} = 1$, else $x_{ik} = 0$.
h_{P_l}	if the l^{th} element in a disassembly sequence is hazardous, $h_{P_l} = 1$; otherwise, $h_{P_l} = 0$.
w_{ij}	If task i is disassembled before task j, $w_{ij} = 1$, else $w_{ij} = 0$.
y_{ikj}	The disassembly sequential relationship between tasks i, j. If part i is a precursor part of j and i is assigned to workstation k, $y_{ikj} = 1$, else $y_{ikj} = 0$.

CT: Cycle time.

With defined notations, the mathematical representation of the DLSP can be formulated, as shown in Table 3.

Objective function (1) is calculated by assigning different weights to objective functions (2) and (3), where $\lambda + \omega = 1$. Objective function (2) is to maximize the actual completion time among the running workstations. Objective function (3) is the total hazard coefficients of all hazard tasks. Constraint (4) indicates that there is a precedence relationship between task i and task j in the disassembly process. Constraint (5) states that the actual disassembly time for a task is the sum of its standard time and the interference time. Constraint (6) means that the processing order of tasks needs to satisfy the disassembly priority relationship. Constraint (7) ensures that the task is executed only once. Constraint (8) calculates the

Table 3. The mathematical model of the DLSP

$\min f_1 = \lambda f_2 + \omega f_3$	(1)
$\min f_2 = CT = \max\{T_k \mid 1 \le k \le M\}, k \in \{1, 2, ..., M\}$	(2)
$\min f_3 = \sum_{i=1}^{N} (I \times h_{P_i} \times t_a)$	(3)
$w_{ij} + w_{ji} = 1, i \ne j, i, j \in \{1, 2, ..., N\}$	(4)
$t_i' \ge t_i + sd_{ji} \times w_{ij} \times u_{ij}, i, j \in \{0, 1, ..., N\}, i \ne j$	(5)
$w_{ij} \le D_{ij}, i, j \in \{0, 1, ..., N\}$	(6)
$w_{ii} = 0, i \in \{0, 1, ..., N\}$	(7)
$t_a = \sum_{i=1}^{N} t_i/N, i \in \{1, 2, ..., N\}$	(8)
$B_j \ge F_i - G \times (1 - w_{ij}), i, j \in \{0, 1, ..., N\}$	(9)
$F_i = B_i + t_i', i \in \{0, 1, ..., N\}$	(10)
$\sum_{k=1}^{M} x_{ik} = 1, i \in \{1, 2, ..., N\}$	(11)
$T_k = \max\{F_i \times x_{ik}\}, i \in \{1, 2, ..., N\}, k \in \{1, 2, ..., M\}$	(12)
$x_{ik} \in \{0, 1\}, i \in \{0, 1, ..., N\}, k \in \{1, 2, ..., M\}$	(13)
$w_{ij} \in \{0, 1\}, i, j \in \{0, 1, ..., N\}$	(14)
$y_{ikj} \in \{0, 1\}, i, j \in \{0, 1, ..., N\}, k \in \{1, 2, ..., M\}$	(15)
$t' \ge 0, B_i \ge 0, B_0 = 0, F_i \ge 0, T_k \ge 0, i, j \in \{0, 1, ..., N\}, k \in \{1, 2, ..., M\}$	(16)

DLSP: Disassembly line scheduling problem.

average processing time of all tasks. Constraint (9) enforces that the start time of a subsequent task should be greater than or equal to the completion time of the previous one. Constraint (10) states that the completion time of a task is equal to the sum of its start and the actual disassembly time. Constraint (11) defines that each task can be assigned to only one workstation. Constraint (12) represents the actual completion time of workstations. Constraints (13) to (16) give the sign constraints on the decision variables.

Solution representation

In DLSP, the order of disassembly tasks is optimized to minimize the CT and hazard coefficients, while ensuring the priority constraint of tasks. For example, there is an initial disassembly order, s = (7, 3, 4, 6, 2, 5, 1), representing the disassembly tasks in Figure 1. As depicted in Figure 1, s does not meet the priority of tasks. Consequently, by considering prioritization of tasks appropriately, s can be adjusted to s' as follows: s' = (1, 3, 4, 6, 2, 5, 7).

According to the characteristics of fixed workstations for this problem, two workstation assignment schemes are designed.

(1) The first step is to allocate the tasks among workstations randomly. Subsequently, based on task order and priority relationships, the tasks are sequentially disassembled on each workstation. According to the sequence s', seven tasks were assigned to three workstations with 2, 2, and 3 tasks, respectively. The workstation 1 is for tasks 1 and 3, while workstation 2 processes tasks 4 and 6. The remaining three tasks are disassembled on workstation 3, as depicted in Figure 3. The detailed data is reported in Table 4.

(2) The tasks in a feasible sequence are assigned to the workstation list one by one. In one round, each workstation has only one task. A new round starts if there are remaining tasks in the sequence. Finally, all tasks can be assigned to all workstations. For example, according to the sequence s' = (1, 3, 4, 6, 2, 5, 7), the total number of tasks is 7, and workstation number is 3. In the first round, task 1 is assigned to workstation 1, task 3 is assigned to workstation 2, and task 4 is assigned to workstation 3. In the next round, task 6 is assigned to workstation 1; the remaining tasks are assigned in the same way until all tasks are assigned. The

Table 4. Task disassembly process for random workstation allocation scheme

Workstation	Disassembly sequence							Run time	Idle time	CT	Hazard coefficients
	1	3	4	6	2	5	7				
1	5	5 + 1						11	6	17	23.14
2			3	4				7	10		
3					2	3 + 2	5	12	5		
	Actual disassembly time										

CT: Cycle time.

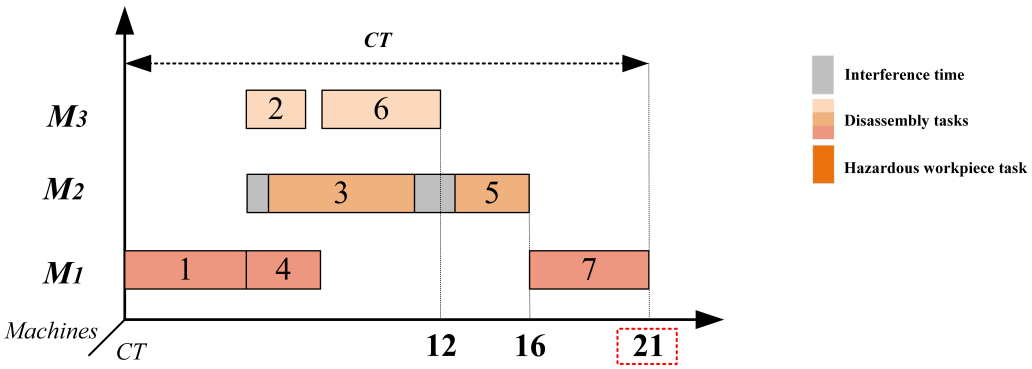

Figure 2. The Gantt chart for task 7.

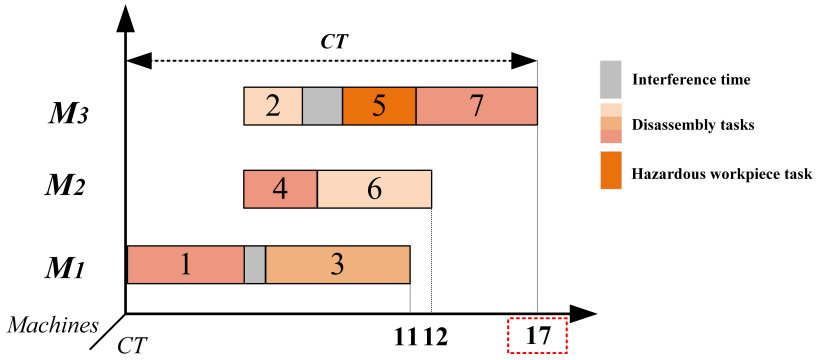

Figure 3. The Gantt chart for the random workstation assignment scheme. CT: Cycle time.

final Gantt chart is depicted in Figure 4. The detailed data reported is in Table 5.

To clearly demonstrate the performance improvement of meta-heuristics combined with Sarsa, an example of 8 tasks is solved. Sarsa's reward feedback can be utilized to select the appropriate task assignment for the problem size and through better choosing local search to obtain higher quality solutions. From Figure 5 to Figure 7, it can be clearly seen that the results by the meta-heuristics with Sarsa strategies are better than the algorithm without Sarsa strategies.

Meta-heuristics

This section describes four meta-heuristics: PSO, ABC, GA, and VNS. They start from parameter and population initialization. Then, the initial solutions are evaluated. After that, new solutions are generated by

Table 5. Task disassembly process for the second workstation allocation scheme

Workstation	Disassembly sequence							Run time	Idle time	CT	Hazard coefficients
	1	3	4	6	2	5	7				
1	5			4			5	14	3	17	23.14
2		5			2 + 1			8	9		
3			3			3 + 2		8	9		
	Actual disassembly time										

CT: Cycle time.

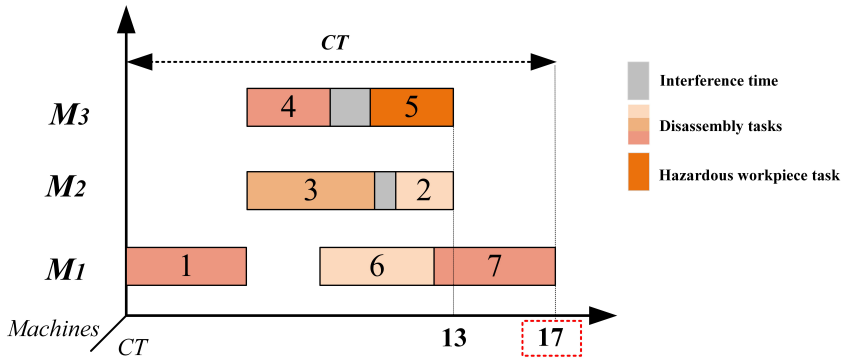

Figure 4. The Gantt chart for the second workstation assignment scheme. CT: Cycle time.

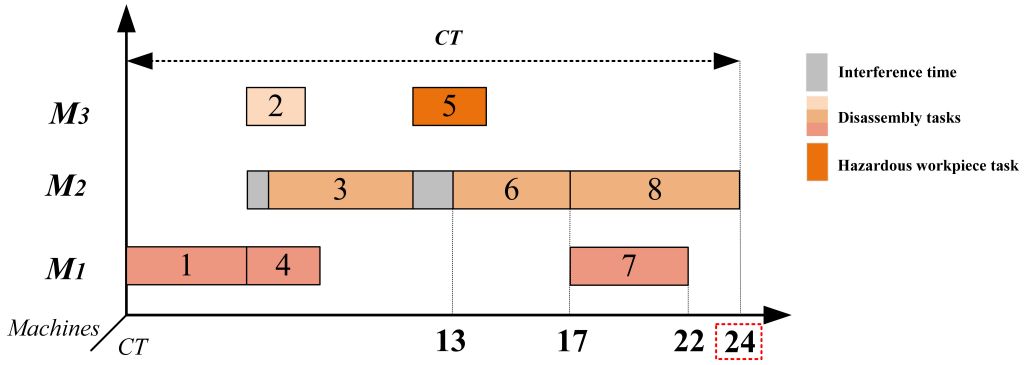

Figure 5. The Gantt chart for task 8. CT: Cycle time.

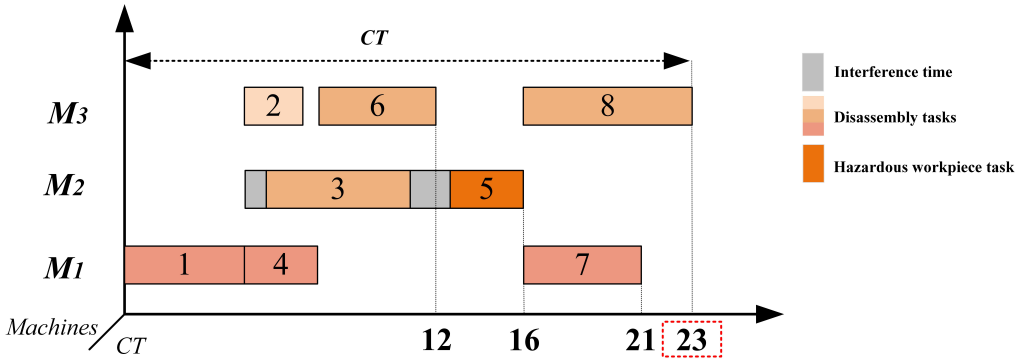

Figure 6. The Gantt chart for task 8 applying only meta-heuristics. CT: Cycle time.

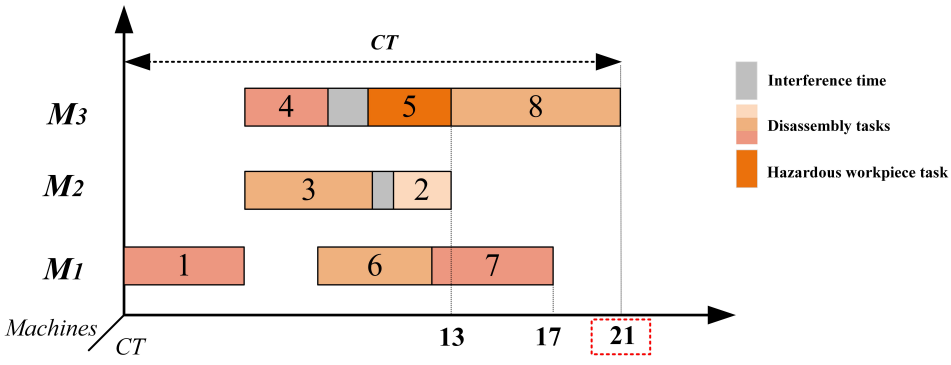

Figure 7. The Gantt chart for task 8 applying the combination of metaheuristics with Sarsa. CT: Cycle time.

algorithm-specific strategies to update population. These steps are repeated until the termination condition is satisfied. The unified framework of the four algorithms is presented in Figure 8.

Sarsa

As a reinforcement learning algorithm, Sarsa is introduced to improve the adaptability and performance of *Q*-learning[58]. The main difference between Sarsa and *Q*-learning is the strategy to update *Q*-value. The schematic representation of the Sarsa framework is depicted in Figure 9.

The updating formula of *Q*-value in Sarsa is expressed as Equation (17).

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)] \tag{17}$$

where $Q(S_t, A_t)$ is the *Q*-value of taking an action A_t at state S_t, α represents the learning rate, and R is the reward after executing action A_t. According to the problem of DLSP, the reward value formula is designed as Equation (18). γ is the discount factor, which takes values in the range [0,1].

$$R = \begin{cases} (Target^{current} - Target^{new})/10, & Target^{new} < Target^{current} \\ 1 & Target^{new} = Target^{current} \\ 0 & Target^{new} > Target^{current} \end{cases} \tag{18}$$

Different from *Q*-learning, Sarsa can select an action under state S_{t+1} according to different distributions of possible future returns for the current state, rather than directly selecting the largest expected *Q*-value to execute actions under the next state S_{t+1}. When updating the *Q*-table, Sarsa is still selecting the next action on the *Q*-table, which has not been updated yet. However, *Q*-learning chooses the next action on the updated *Q*-table.

For DLSP problems, we design a strategy π based on a probability distribution and set four values P = (0.7, 0.8, 0.9, 1), respectively. The action with the highest expected value is selected in the probability of P, and random selection is executed under the probability of (1 - P), as depicted in Equation (19).

$$A_{t+1} \sim \pi(\cdot | S_{t+1}) \tag{19}$$

Sarsa based task assignment

Two workstation assignment schemes are presented in Section "Solution representation". The Sarsa algorithm is used to select an appropriate workstation assignment scheme (action) during meta-heuristics' iterations. The initial *Q*-table is shown in Table 6. Each solution corresponds to a state S, while each action A represents a

Table 6. Initial Q-table

		a_1	a_2	a_3	...	a_n
$Q(s_t, a_t)$ =	s_1	0	0	0	...	0
	s_2	0	0	0	...	0
	s_3	0	0	0	...	0
	⋮	⋮	⋮	⋮	⋱	
	s_n	0	0	0	...	0

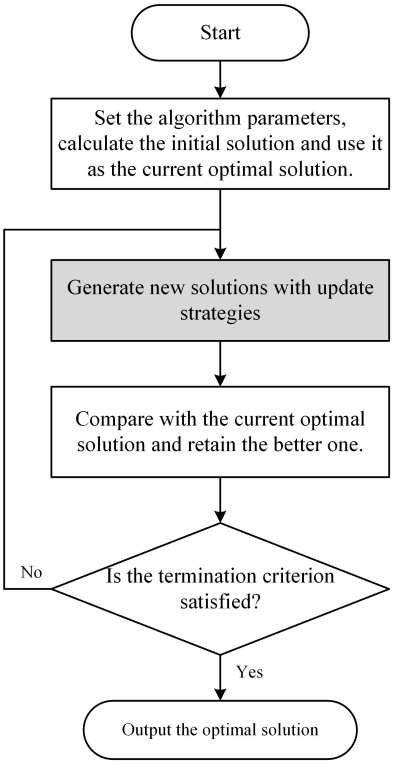

Figure 8. The unified framework of the four algorithms.

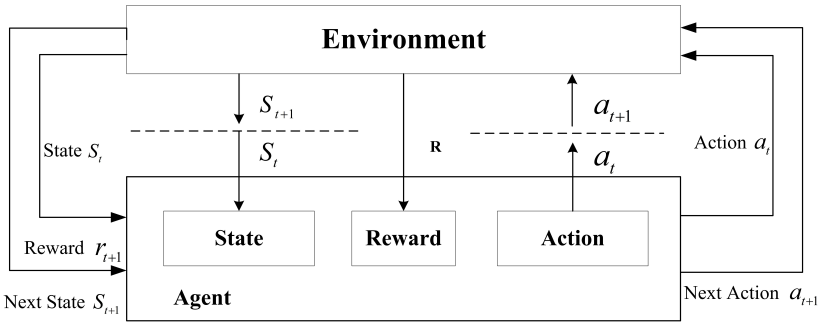

Figure 9. The framework of Sarsa.

workstation assignment scheme. If a solution is improved by executing a workstation allocation strategy, a reward is obtained and the corresponding *Q*-value is updated. The ratio of the workstation allocation

scheme being selected for the next iteration is increased. Conversely, if a solution is not improved, there is no reward and its selection probability in the next iteration decreases. The π strategy is employed for action selection. Through continuous exploration and feedback in the learning process, the algorithm can discern better task allocation methods tailored to different requirements. Algorithm 1 describes the steps of this task assignment strategy.

Algorithm 1. Sarsa based task assignment

Initialize : *Population size* (P_s), *Algorithm running time*(T), $Q - table$, *iteration number* $t = 0$

While $t < T$ ***do***

  Initialize S.

  for $S = 1$ *to* $P_s - 1$ ***do***

    Through π *strategy to choose* A_t *in* S_t.

    if $target^{new} < target^{current}$

      $target^{current} = target^{new}$

    end if

    Get the R *and* S_{t+1}

    Choose action a_{t+1} *with* π.

    Update $Q - value$ *according to Equation* (18)

  end for

end while

Update the $Q - table$;

Sarsa based local search

This article introduces ten local search operators for enhancing the performance of the meta-heuristics, which includes the adjustments for CT time and hazard coefficients. The detailed procedures of ten local searches are delineated below:

(1) Swap: Swap the positions of two random tasks in the sequence [Figure 10A].

(2) Double swap: Repeat the swap operation twice [Figure 10B].

(3) Insertion: Randomly take out a task from a sequence and insert it into a new position in the sequence [Figure 10C].

(4) Bind insertion: Randomly select two consecutive tasks from a task sequence and insert them at the position with the lowest desired target value [Figure 10D].

(5) Block insertion: Randomly select multiple consecutive tasks from a task sequence and insert them at the position with the lowest desired target value [Figure 10E].

(6) Insert sequentially: Randomly select multiple tasks in a task sequence and insert them into the sequence one by one with the lowest desired target value [Figure 10F].

(7) Inverse: Randomly select several consecutive tasks from a task sequence in their reverse order [Figure 10G].

(8) Sort: Adjust the hazard task to the next position that satisfies the priority relationship [Figure 10H].

(9) Random sort: Randomly adjust the hazardous task to the sequence position with satisfying the priority relationship [Figure 10I].

(10) Sort sequentially: Adjust the hazardous tasks in sequence to a new position with satisfying the priority relationship and the lowest target value [Figure 10J].

Sarsa is employed to select premium local search operators during iterations. The solutions are as states while ten local search operators are taken as actions. In the learning process, the π strategy is used to select actions for the next state. If the current action gets a better solution, there is a position reward, increasing its

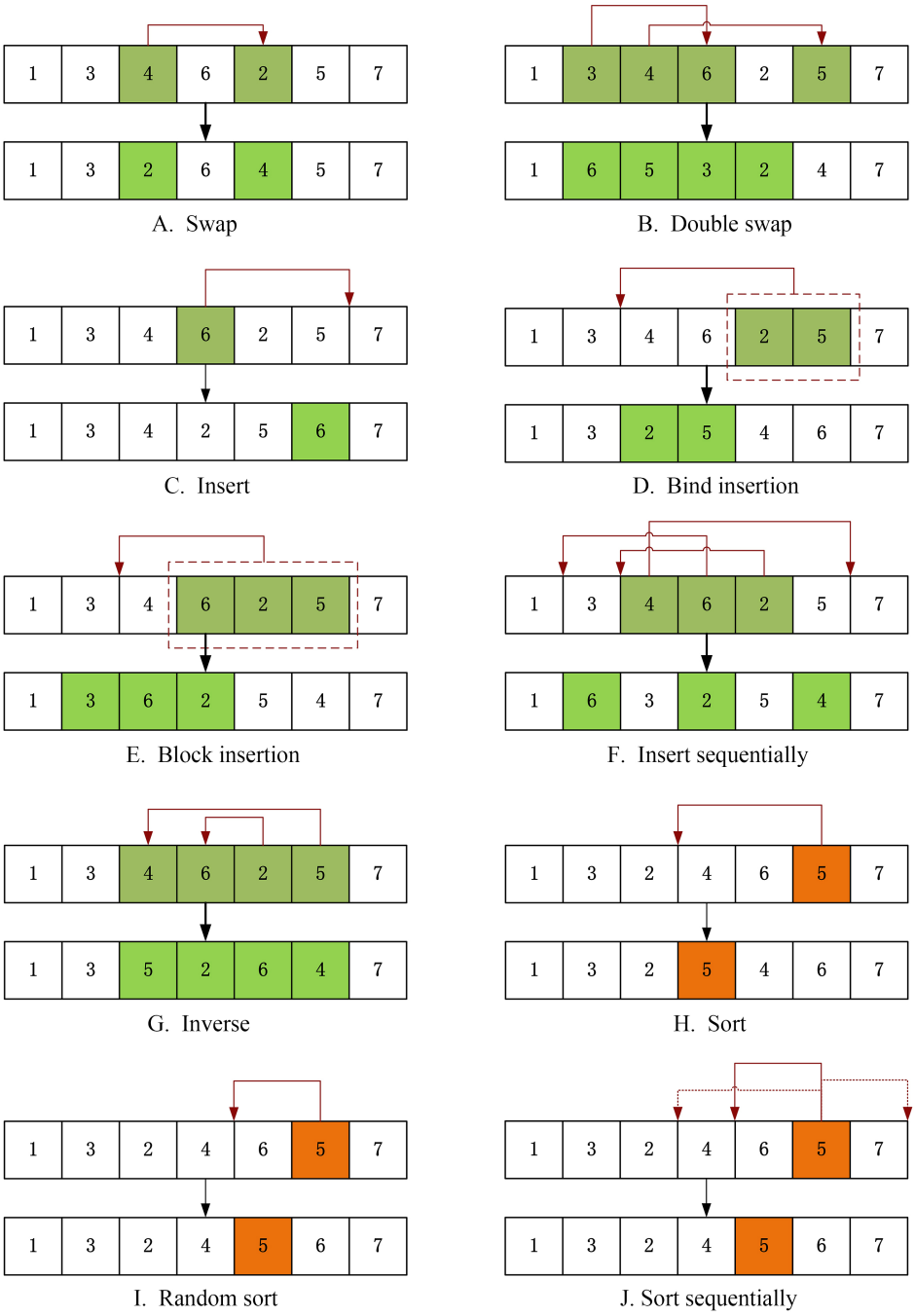

Figure 10. The framework of Sarsa.

probability of being selected in the next iteration. In contrast, the probability of being selected in the next iteration is decreased.

If the sequence obtained by local search does not satisfy the priority relationship, we need to update it to a feasible solution. The steps of Sarsa based local search are described in Algorithm 2.

Algorithm 2. Sarsa based local search

Initialize: *Population size* (P_s), *Algorithm running time*(T), $Q - table$, *iteration number* $t = 0$

While $t < T$ ***do***

  Initialize S.

  for $S = 1$ *to* $P_s - 1$ ***do***

    Through π *strategy to choose* A_t *in* S_t.

    if *sequence not statisfied priority relationship*

      Adjust sequence

    end if

    if $target^{new} < target^{current}$ ***then***

      $target^{current} = target^{new}$

    end if

    Get the R *and* S_{t+1}

    Choose action a_{t+1} *with* π.

    Update $Q - value$ *according to Equation* (18)

  end for

end while

Update the $Q - table$;

Framework of the proposed algorithms

The improved strategies are embedded in four meta-heuristics. The unified framework of them is shown in Figure 11. All algorithms start from an initial population and iteratively update population with their respective strategies. During iterations, two Sarsa based strategies are employed to improve the exploration and exploitation of meta-heuristics. Finally, the best result is output if the termination condition is met.

RESULTS

Experimental setup

To verify the effectiveness of the proposed strategies, ten instances with different scales are solved[59]. All the algorithms are compiled in C++; the running platform is a desktop computer with an Intel Core i7-10,700 CPU @ 2.90 GHz and 16 GB of RAM under Microsoft Windows 11.

To ensure a fair comparison, all algorithms are executed for the same termination time, four seconds, while their parameters are experimentally determined to achieve the optimal combination. The algorithm's performance is assessed by comparing its average value, minimum value, and the coefficient of variation (CV) in 20 runs. The smaller the CV value, the better the stability and robustness of the algorithm is.

The formula for calculating CV is as follows:

$$CV = \frac{SD}{AVE} \times 100\%$$

where AVE is the average of the target values obtained in 20 runs, and SD is the corresponding standard deviation.

Parameter setting

The orthogonal experiment design method is used to test the influence of parameters on the performance of the four combined algorithms. Taking the SPSO_SD as an example, there are four parameters, which are limited iterations (L), population size (Ps), learning rate (α), and discount rate (γ). Each parameter is set to four values, $L \in \{10, 20, 30, 40\}$, $Ps \in \{10, 20, 30, 40\}$, $\alpha \in \{0.2, 0.4, 0.6, 0.8\}$, and $\gamma \in \{0.05, 0.1, 0.15, 0.2\}$.

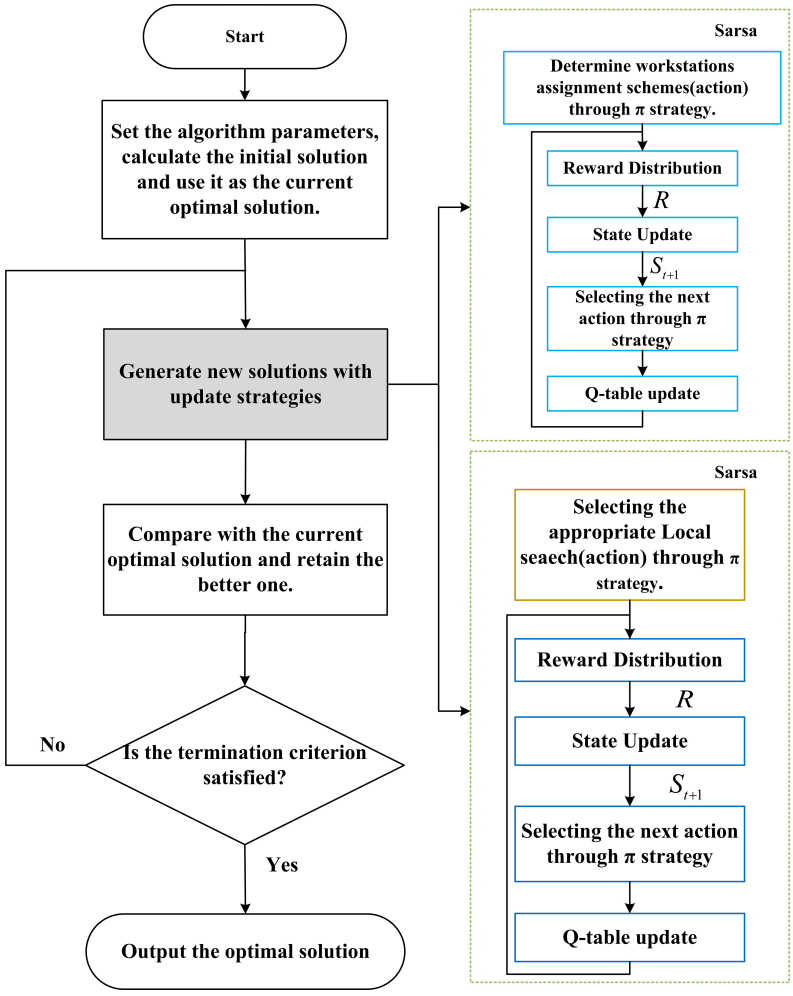

Figure 11. The framework of the proposed algorithms.

The orthogonal matrix L_{16} (4^4) is shown in Table 7. By solving each group of parameters in the SPSO_SD and analyzing the target values obtained from experimental results, a corresponding main effect diagram is constructed, as depicted in Figure 12. It is evident that the SPSO_SD has the best results under the parameter combination, L = 20, Ps = 20, α = 0.8, and γ = 0.1, which is adopted for further test and comparisons.

In ABC, there are three parameters, employed bees (Ep), onlooker bees (Op), and scout bees (Sp), Op = 1 - Ep - Sp. By the trend of the parameter hierarchy of Figure 13, we choose the combination Ep = 0.5, Op = 0.2, and Sp = 0.3. In GA, there are two important parameters: crossover probability and mutation probability. After parameter tuning experiments, we choose Pc = 0.7 and Pm = 0.3, as depicted in Figure 14. In VNS, the maximum number of iterations and the number of domain operations have an important impact on the performance, and experimentally, the algorithm performs best when L = 20 and M = 30, as depicted in Figure 15.

We design a strategy π based on probability distribution. To choose a better probability setting, the largest instance is solved under six probabilities (P = 0.5, 0.6, 0.7, 0.8, 0.9, 1) with 20 runs independently. The first 20 optimal values under six probabilities are selected to judge the number of distributions in each

Table 7. Orthogonal experiments for parameter setting

No.	*L*	*Ps*	α	γ	Target
1	10	10	0.2	0.05	6,564
2	20	10	0.4	0.10	7,525
3	30	10	0.6	0.15	6,455
4	40	10	0.8	0.20	5,433
5	10	20	0.4	0.15	4,386
6	20	20	0.2	0.20	2,765
7	30	20	0.8	0.05	7,188
8	40	20	0.6	0.10	8,576
9	10	30	0.6	0.20	8,421
10	20	30	0.8	0.15	6,144
11	30	30	0.2	0.10	5,053
12	40	30	0.4	0.05	7,799
13	10	40	0.8	0.10	3,402
14	20	40	0.6	0.05	4,316
15	30	40	0.4	0.20	8,608
16	40	40	0.2	0.15	8,584

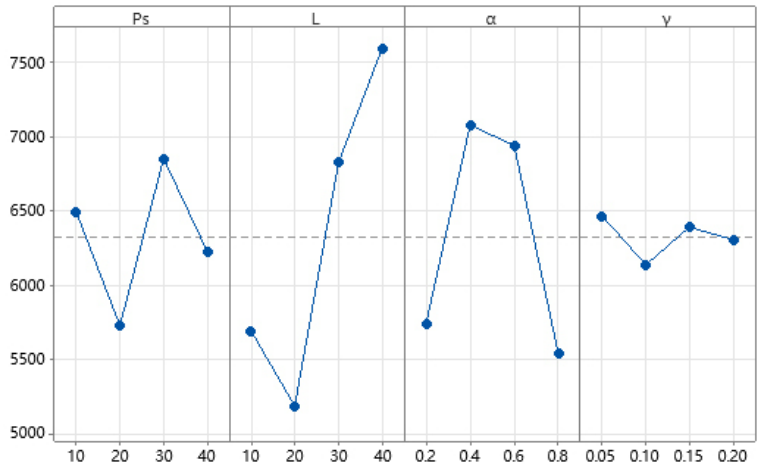

Figure 12. The main effects plot for SPSO_SD.

probability, as shown in Figure 16. The best four probability values are selected, (P = 0.7, 0.8, 0.9, 1).

Verifying the effectiveness of improved strategies

To verify the effectiveness of the proposed strategies, four meta-heuristics are compared to their respective variants. For all cases, the average (Ave) and minimum (Min) values of results are reported in Tables 8-11. From Table 8, SPSO_SD has the best results, with the best results for all cases and the smallest mean values. As shown in Table 9, SABC_SD has the smallest mean values for all cases and gets the best results in seven examples. The SABC_RD gets the best results for one case, while ABC_SD gets the best results in four cases. As reported in Table 10, SGA_SD obtains the smallest mean values for all cases and achieves the best results for five cases. The SGA_RD achieves the best results for only 1 case, while the GA_SD gets the best results for five cases. In Table 11, SVNS_SD has the smallest mean values for all cases and achieves the best result for seven cases. The SVNS_RD achieves the best result for one case. The VNS_SD achieves the best result

Table 8. Internal comparison of PSO algorithm

Tasks	PSO		SPSO_RD		PSO_SD		SPSO_SD	
	Ave	Min	Ave	Min	Ave	Min	Ave	Min
25	96.70	69.00	48.60	35.00	66.60	52.00	**47.65**	**32.00**
27	68.95	36.00	58.35	35.00	67.50	47.00	**48.90**	**27.00**
36	130.75	55.00	96.50	65.00	82.05	47.00	**79.90**	**32.00**
47	406.95	357.00	368.30	343.00	235.20	427.00	**358.85**	**241.00**
51	19,936.95	16,538.00	15,067.15	12,411.00	15,790.50	8,716.00	**13,557.10**	**8,082.00**
55	12,709.25	4,439.00	8,992.80	6,892.00	10,276.60	4,095.00	**7,074.25**	**3,431.00**
66	185.70	70.00	79.15	54.00	73.35	47.00	**59.00**	**41.00**
70	1,417.25	619.00	514.15	443.00	1,104.15	778.00	**406.65**	**338.00**
79	8,933.20	4,452.00	3,669.45	2,995.00	6,384.45	4,003.00	**2,765.60**	**2,173.00**
83	42,163.30	37,139.00	25,487.15	23,013.00	38,521.55	30,081.00	**23,561.05**	**17,316.00**

PSO: Particle swarm optimization; Ave: the average; Min: minimum.

Table 9. Internal comparison of ABC algorithm

Tasks	ABC		SABC_RD		ABC_SD		SABC_SD	
	Ave	Min	Ave	Min	Ave	Min	Ave	Min
25	78.95	44.00	77.05	49.00	67.40	44.00	**50.30**	**32.00**
27	65.35	45.00	61.20	36.00	64.55	41.00	**46.30**	**28.00**
36	114.55	64.00	106.15	66.00	88.85	34.00	**78.05**	**32.00**
47	325.65	160.00	400.85	244.00	314.40	145.00	**252.85**	**142.00**
51	20,681.80	13,558.00	19,207.40	13,500.00	19,485.65	**10,062.00**	**15,820.65**	12,411.00
55	13,492.05	4,383.00	8,385.50	**3,431.00**	9,209.25	5,088.00	**8,056.40**	**3,431.00**
66	150.55	82.00	124.85	50.00	102.95	**43.00**	**79.45**	**43.00**
70	1,456.90	784.00	780.95	331.00	824.40	**323.00**	**502.40**	331.00
79	7,894.50	3,759.00	5,679.85	2,489.00	5,489.55	**2,157.00**	**4,316.85**	2,022.00
83	39,651.25	33,843.00	33,654.65	27,724.00	36,514.05	32,823.00	**30,152.20**	**18,980.00**

ABC: Artificial bee colony; Ave: the average; Min: minimum.

for two cases while the VNS does not obtain the best mean value and minimum value for any case. It can be concluded that the meta-heuristics combined with Sarsa based task assignment and local search strategies have better robustness and stability than the compared ones.

The CV is employed as a metric to assess the influence of enhancement strategies on algorithms' stability and robustness. Tables 12-15 show the comparisons of CV among the four classical meta-heuristics and their variants, respectively. In Table 12, SPSO_SD receives the best Min values for eight out of ten cases. As shown in Table 13, the SABC_SD obtains the minimum values for eight cases, while the ABC_SD obtains the minimum values for two cases. As reported in Table 14, the SGA_SD obtains the minimum values for seven out of ten cases, while GA_SD gets the minimum values for two cases. In Table 15, SVNS_SD obtains the minimum values for eight cases, while the VNS_SD and SVNS_RD obtain the minimum value for one case. In each group, the algorithm with two Sarsa based strategies has smaller CV values for most instances. This means that the Sarsa based strategies can improve the stability and robustness of four meta-heuristics.

Table 10. Internal comparison of GA algorithm

Tasks	GA		SGA_RD		GA_SD		SGA_SD	
	Ave	Min	Ave	Min	Ave	Min	Ave	Min
25	99.25	46.00	97.75	70.00	78.20	44.00	**76.45**	**41.00**
27	79.55	38.00	81.85	47.00	56.20	**26.00**	**56.50**	27.00
36	126.15	54.00	125.45	86.00	115.05	**38.00**	**109.60**	42.00
47	430.50	250.00	512.45	323.00	408.10	**100.00**	**355.45**	145.00
51	21,068.85	8,514.00	20,795.40	14,013.00	23,868.20	**6,804.00**	**18,098.50**	8,158.00
55	13,754.80	6,701.00	11,878.70	8,340.00	12,657.45	6,587.00	**10,375.50**	**3,770.00**
66	171.30	75.00	151.25	**45.00**	164.35	52.00	**99.35**	**45.00**
70	1,542.10	830.00	1,175.40	345.00	1,056.60	**298.00**	**869.10**	345.00
79	10,334.10	5,658.00	7,799.50	2,922.00	7,925.75	4,057.00	**5,816.80**	**2,658.00**
83	48,563.35	42,158.00	34,561.05	32,401.00	38,771.85	36,486.00	**32,511.60**	**26,868.00**

GA: Genetic algorithm; Ave: the average; Min: minimum.

Table 11. Internal comparison of VNS algorithm

Tasks	VNS		SVNS_RD		VNS_SD		SVNS_SD	
	Ave	Min	Ave	Min	Ave	Min	Ave	Min
25	110.35	78.00	72.25	44.00	79.75	54.00	**50.20**	**32.00**
27	64.75	38.00	64.20	39.00	52.85	**26.00**	**55.35**	29.00
36	136.05	91.00	89.85	64.00	104.35	63.00	**74.10**	**32.00**
47	539.95	361.00	344.90	234.00	338.80	206.00	**325.30**	**159.00**
51	27,138.80	19,890.00	19,278.10	8,496.00	17,015.30	15,880.00	**14,763.85**	**8,082.00**
55	15,715.95	10,327.00	7,821.25	7,182.00	10,730.45	3,888.00	**7,046.95**	**3,534.00**
66	167.95	103.00	125.25	49.00	130.80	**43.00**	**63.10**	47.00
70	1,490.85	867.00	961.95	333.00	999.55	418.00	**438.20**	**311.00**
79	11,858.90	7,249.00	6,547.10	**2,030.00**	9,293.40	3,123.00	**3,402.20**	2,444.00
83	34,658.30	29,843.00	29,021.35	25,637.00	36,524.55	31,450.00	**27,563.35**	**19,765.00**

VNS: Variable neighborhood search; Ave: the average; Min: minimum.

Table 12. Comparison of CV values for the PSO variant algorithm

No.	Tasks	PSO	PSO_SD	SPSO_RD	SPSO_SD
1	25	19.69%	**3.67%**	6.64%	7.50%
2	27	10.87%	7.32%	4.45%	**3.89%**
3	36	18.14%	9.90%	10.86%	**3.96%**
4	47	10.90%	6.31%	5.09%	**4.96%**
5	51	17.89%	8.99%	6.76%	**5.42%**
6	55	19.36%	**4.44%**	5.42%	6.96%
7	66	12.75%	8.59%	9.38%	**2.61%**
8	71	14.54%	9.20%	7.68%	**2.42%**
9	79	15.19%	2.39%	8.01%	**1.04%**
10	83	19.08%	8.83%	11.41%	**7.32%**

CV: Coefficient of variation; PSO: particle swarm optimization.

Table 13. Comparison of CV values for the ABC variant algorithm

No.	Tasks	ABC	ABC_SD	SABC_RD	SABC_SD
1	25	12.47%	9.87%	9.00%	**3.48%**
2	27	15.86%	5.24%	8.77%	**2.65%**
3	36	15.25%	4.51%	3.82%	**2.73%**
4	47	18.74%	**5.39%**	9.59%	6.76%
5	51	17.69%	5.47%	7.33%	**6.84%**
6	55	18.41%	8.99%	**8.79%**	**8.79%**
7	66	16.51%	8.28%	7.33%	**6.96%**
8	71	18.64%	**6.12%**	6.52%	8.08%
9	79	11.42%	7.70%	8.84%	**6.67%**
10	83	18.76%	12.47%	16.29%	**7.76%**

CV: Coefficient of variation; ABC: artificial bee colony.

Table 14. Comparison of CV values for the GA variant algorithm

No.	Tasks	GA	GA_SD	SGA_RD	SGA_SD
1	25	11.13%	8.11%	9.03%	**5.05%**
2	27	16.44%	7.49%	9.42%	**5.48%**
3	36	17.59%	8.34%	**7.75%**	8.37%
4	47	14.05%	9.63%	6.53%	**4.11%**
5	51	19.88%	**7.39%**	8.29%	7.40%
6	55	14.67%	9.74%	12.77%	**6.51%**
7	66	16.34%	**5.19%**	10.62%	6.73%
8	71	15.00%	8.44%	13.17%	**6.60%**
9	79	17.12%	7.22%	18.05%	**7.17%**
10	83	14.56%	10.28%	12.77%	**9.03%**

CV: Coefficient of variation; GA: genetic algorithm.

Table 15. Comparison of CV values for the VNS variant algorithm

No.	Tasks	VNS	VNS_SD	SVNS_RD	SVNS_SD
1	25	12.82%	9.26%	8.99%	**4.67%**
2	27	14.76%	8.25%	8.28%	**4.36%**
3	36	12.14%	6.76%	6.12%	**4.49%**
4	47	17.31%	6.76%	7.70%	**3.72%**
5	51	12.48%	9.95%	6.61%	**5.01%**
6	55	19.13%	7.34%	**7.33%**	7.64%
7	66	12.61%	9.03%	7.18%	**6.35%**
8	71	10.60%	3.96%	9.00%	**3.59%**
9	79	18.63%	**3.48%**	8.84%	4.32%
10	83	19.13%	10.49%	9.95%	**6.32%**

CV: Coefficient of variation; VNS: variable neighborhood search.

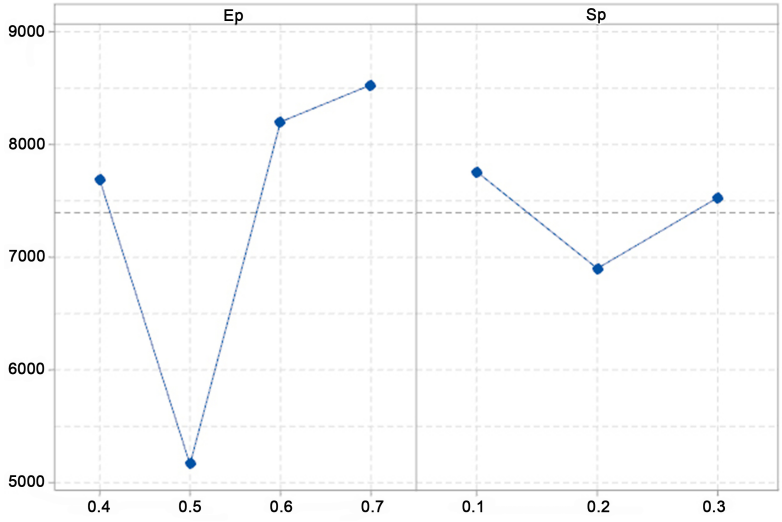

Figure 13. Parameter level trend of ABC. ABC: Artificial bee colony.

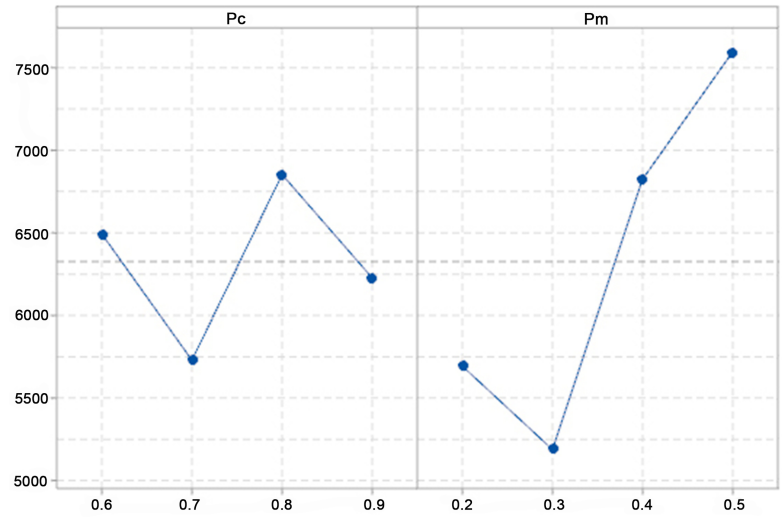

Figure 14. Parameter level trend of GA. GA: Genetic algorithm.

Further comparisons and discussions

To further verify the performance difference among four algorithms with Sarsa based strategies, the Friedman test and Nemenyi post-test are executed. The results are represented in Table 16, Figures 17 and 18. In Table 16, the obtained asymptotic significance (Asymp.Sig) is far less than 0.05, indicating noteworthy disparities among the four algorithms. It implies that there exist significant variations in their performance. Figure 17 shows the mean rank of algorithms by Nemenyi post-test. It is obvious that the SPSO_SD has the smallest rank value (1.70). Figure 18 visually presents the algorithm ranking distribution for all cases. It can be seen from Figure 18 that the SPSO_SD ranks first for six out of ten instances and second for two cases. The SABC_SD and SVNS_SD have ranked first for 2 out of 10 instances, respectively. The SGA_SD has a bad rank for most instances. Hence, the SPSO_SD outperforms its peers in solving the concerned problems.

Table 16. The results of Friedman test

N	10
Chi-Square	17.760
Df	3
Asymp.Sig.	.000

Asymp.Sig: Asymptotic significance.

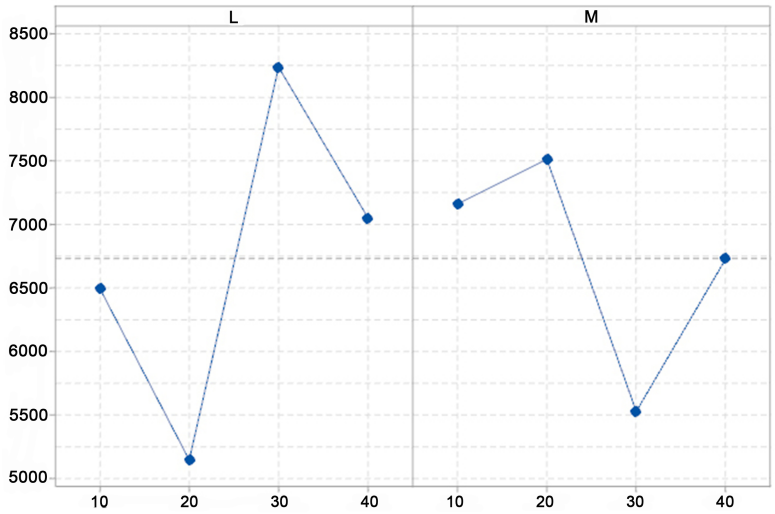

Figure 15. Parameter level trend of VNS. VNS: Variable neighborhood search.

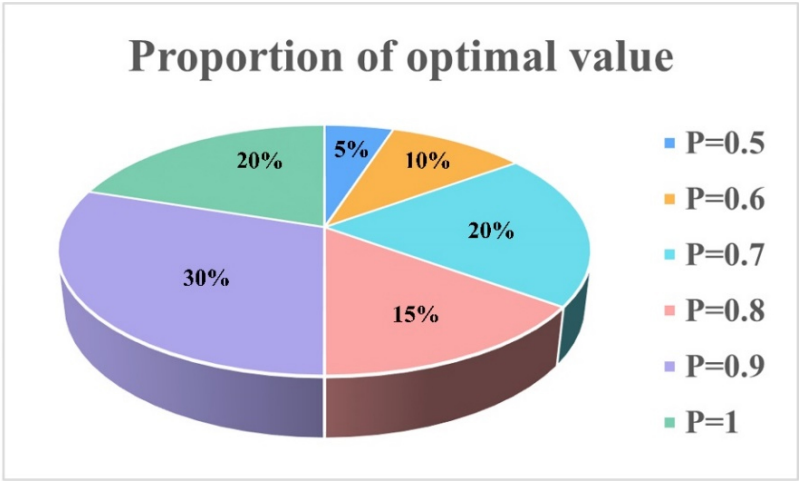

Figure 16. The number proportion of the optimal values of six probabilities.

Convergence analysis

In this section, we present curves depicting the convergence of four classical meta-heuristics. The results of cases with 47 tasks and 79 tasks are used, as shown in Figures 19 and 20. It is evident that the algorithms converge rapidly at the beginning and flatten out with the increasing computational time. SGA_GD algorithm gives the worst results. SVNS_SD has medium performance and more stability. SABC_SD fluctuates a lot during the convergence process. Compared with these algorithms, SPSO_SD has faster

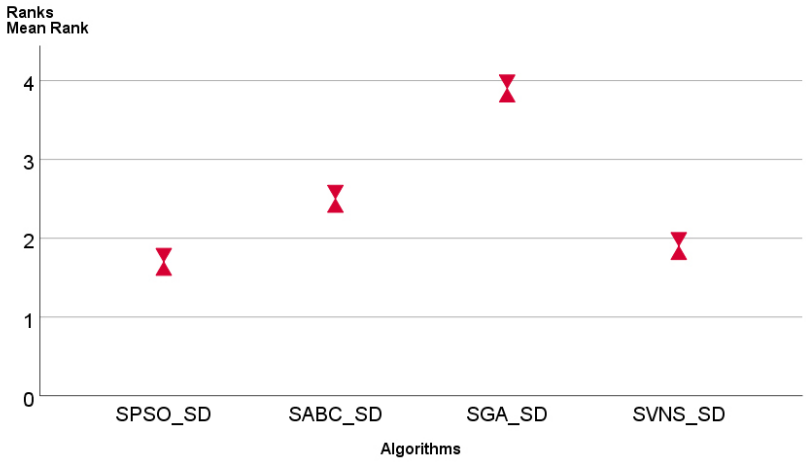

Figure 17. The rank of Nemenyi post-hoc test.

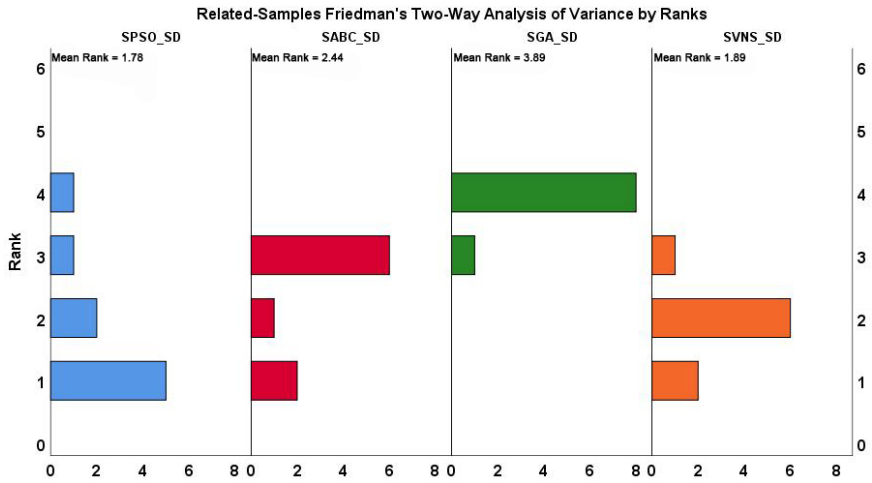

Figure 18. Friedman's binary rank variance analysis.

convergence and higher quality solutions. It can be observed that SPSO_SD has better efficiency and effectiveness than its peers.

Case study: aircraft engine disassembly

In the experimental part, the arithmetic instances include self-generated cases and some actual cases such as discarded cell phones for the case with 27 tasks, LCD TVs for the case with 36 tasks, laptop computers for the case with 47 tasks, and aircraft engines for the case with 51 tasks[59]. The case with 51 tasks is a specific case provided by a domestic airport. Figure 21 is the disassembly priority diagram of the aircraft engine disassembly case in our project.

CONCLUSIONS AND FUTURE WORK

A mathematical model for DLSP is established, considering time interference and priority relationships. The main objective is to minimize both CT and hazard coefficients. According to the problem's characteristics, a

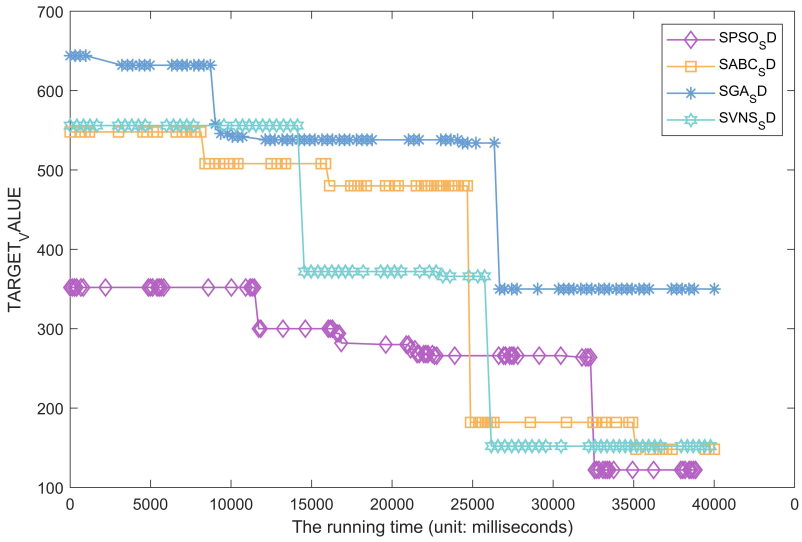

Figure 19. Convergence curves of all compared algorithms for task 47.

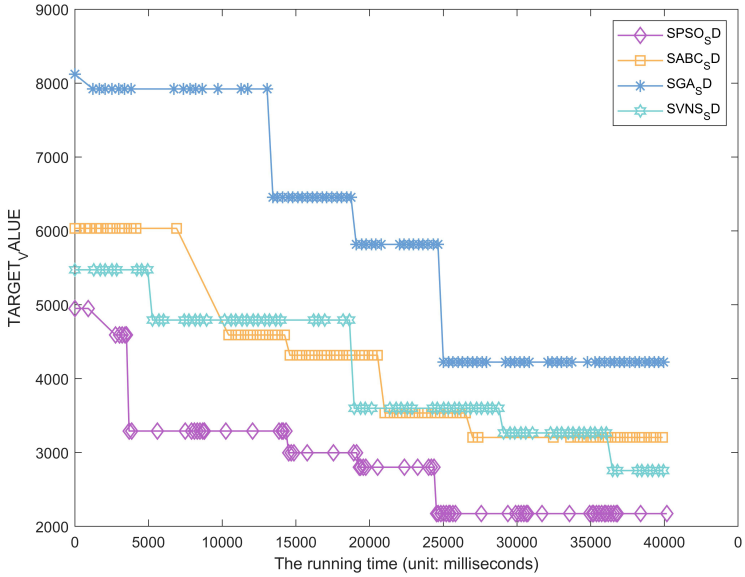

Figure 20. Convergence curves of all compared algorithms for task 79.

Sarsa algorithm with a reward feedback mechanism is employed in two stages for task allocation and guiding local search operators' selection. The experiments solve ten instances of varying scales. Through analysis and discussion of the results, it is proven that the PSO algorithm combined with two Sarsa strategies exhibits strong competitiveness for solving DLSP problems.

In the future, the following directions will be addressed: (1) building upon existing challenges, consider more practical constraints, such as incomplete disassembly, multilateral disassembly, and other related disassembly issues; (2) try to combine more reinforcement learning algorithms to meta-heuristics for solve disassembly problems; (3) attempt to apply the combination of reinforcement learning and meta-heuristics to other scheduling problems[60-62].

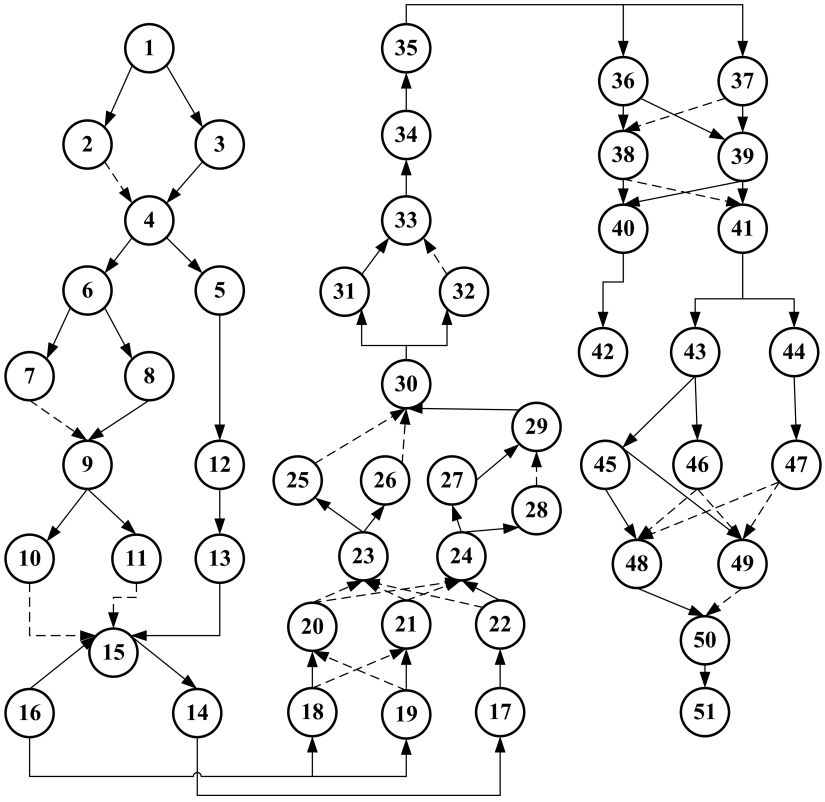

Figure 21. Precedence graph of the aircraft engine disassembly case.

DECLARATIONS

Authors' contributions

Software, validation, conceptualization, methodology, writing- original draft preparation: Li D
Supervision, writing - review and editing, validation, formal analysis: Gao K
Methodology, investigation, instance collection: Ren Y, Fu Y
Data curation, investigation, methodology: Zhang R

Availability of data and materials

Not applicable.

Financial support and sponsorship

This study is partially supported by the National Natural Science Foundation of China under Grant 62173356, the Science and Technology Development Fund (FDCT), Macau SAR, under Grant 0019/2021/A, Zhuhai Industry-University-Research Project with Hongkong and Macao under Grant ZH22017002210014PWC, the Guangdong Basic and Applied Basic Research Foundation (2023A1515011531), Research on the Key Technologies for Scheduling and Optimization of Complex Distributed Manufacturing Systems (22JR10KA007).

Conflicts of interest

All authors declared that there are no conflicts of interest.

Ethical approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

REFERENCES

1. Liu J, Wang S. Balancing disassembly line in product recovery to promote the coordinated development of economy and environment. *Sustainability* 2017;9:309. DOI
2. Gungor A, Gupta SM. Issues in environmentally conscious manufacturing and product recovery: a survey. *Comput Ind Eng* 1999;36:811-53. DOI
3. Özceylan E, Kalayci CB, Güngör A, Gupta SM. Disassembly line balancing problem: a review of the state of the art and future directions. *Int J Prod Res* 2019;57:4805-27. DOI
4. Paterson DAP, Ijomah WL, Windmill JFC. End-of-life decision tool with emphasis on remanufacturing. *J Clean Prod* 2017;148:653-64. DOI
5. Güngör A, Gupta SM. Disassembly line in product recovery. *Int J Prod Res* 2002;40:2569-89. DOI
6. Fu Y, Zhou M, Guo X, Qi L, Sedraoui K. Multiverse optimization algorithm for stochastic biobjective disassembly sequence planning subject to operation failures. *IEEE Trans Syst Man Cybern Syst* 2022;52:1041-51. DOI
7. Lambert AJD. Disassembly sequencing: a survey. *Int J Prod Res* 2003;41:3721-59. DOI
8. Kalayci CB, Gupta SM. Artificial bee colony algorithm for solving sequence-dependent disassembly line balancing problem. *Expert Syst Appl* 2013;40:7231-41. DOI
9. Edis EB. Constraint programming approaches to disassembly line balancing problem with sequencing decisions. *Comput Oper Res* 2021;126:105111. DOI
10. ÇİL ZA. An exact solution method for multi-manned disassembly line design with AND/OR precedence relations. *Appl Math Model* 2021;99:785-803. DOI
11. Meng L, Zhang B, Ren Y, Sang H, Gao K, Zhang C. Mathematical formulations for asynchronous parallel disassembly planning of end-of-life products. *Mathematics* 2022;10:3854. DOI
12. Bentaha ML, Battaïa O, Dolgui A. An exact solution approach for disassembly line balancing problem under uncertainty of the task processing times. *Int J Prod Res* 2015;53:1807-18. DOI
13. Altekin FT, Akkan C. Task-failure-driven rebalancing of disassembly lines. *Int J Prod Res* 2012;50:4955-76. DOI
14. He J, Chu F, Dolgui A, Zheng F, Liu M. Integrated stochastic disassembly line balancing and planning problem with machine specificity. *Int J Prod Res* 2022;60:1688-708. DOI
15. McGovern SM, Gupta SM. Uninformed and probabilistic distributed agent combinatorial searches for the unary NP-complete disassembly line balancing problem. In: Proceedings Volume Environmentally Conscious Manufacturing V. 2005. DOI
16. Chen M, Tan Y. SF-FWA: a self-adaptive fast fireworks algorithm for effective large-scale optimization. *Swarm Evol Comput* 2023;80:101314. DOI
17. Singh P, Pasha J, Moses R, Sobanjo J, Ozguven EE, Dulebenets MA. Development of exact and heuristic optimization methods for safety improvement projects at level crossings under conflicting objectives. *Reliab Eng Syst Saf* 2022;220:108296. DOI
18. Pasha J, Nwodu AL, Fathollahi-fard AM, et al. Exact and metaheuristic algorithms for the vehicle routing problem with a factory-in-a-box in multi-objective settings. *Adv Eng Inform* 2022;52:101623. DOI
19. Singh E, Pillay N. A study of ant-based pheromone spaces for generation constructive hyper-heuristics. *Swarm Evol Comput* 2022;72:101095. DOI
20. Fu Y, Ma X, Gao K, Li Z, Dong H. Multi-objective home health care routing and scheduling with sharing service via a problem-specific knowledge-based artificial bee colony algorithm. *IEEE Trans Intell Transport Syst* 2023:1-14. DOI
21. Fu Y, Tian G, Li Z, Wang Z. Parallel machine scheduling with dynamic resource allocation via a master-slave genetic algorithm. *IEEJ Trans Electr Electron Eng* 2018;13:748-56. DOI
22. Fu Y, Wang H, Huang M, Wang J. A decomposition based multiobjective genetic algorithm with adaptive multipopulation strategy for flowshop scheduling problem. *Nat Comput* 2019;18:757-68. DOI
23. Ma X, Fu Y, Gao K, Zhu L, Sadollah A. A multi-objective scheduling and routing problem for home health care services via brain storm optimization. *Complex Syst Model Simul* 2023;3:32-46. DOI
24. Liang P, Fu Y, Ni S, Zheng B. Modeling and optimization for noise-aversion and energy-awareness disassembly sequence planning problems in reverse supply chain. *Environ Sci Pollut Res Int* 2021. DOI
25. Wang Y, Han Y, Wang Y, Li J, Gao K, Liu Y. An effective two-stage iterated greedy algorithm for distributed flowshop group scheduling problem with setup time. *Expert Syst Appl* 2023;233:120909. DOI

26. Wang Y, Han Y, Wang Y, Tasgetiren MF, Li J, Gao K. Intelligent optimization under the makespan constraint: rapid evaluation mechanisms based on the critical machine for the distributed flowshop group scheduling problem. *Eur J Oper Res* 2023;311:816-32. DOI
27. Wang Y, Han Y, Wang Y, Pan Q, Wang L. Sustainable scheduling of distributed flow shop group: a collaborative multi-objective evolutionary algorithm driven by indicators. *IEEE Trans Evol Comput* 2023. DOI
28. Chen S, Pan QK, Gao L, Sang H. A population-based iterated greedy algorithm to minimize total flowtime for the distributed blocking flowshop scheduling problem. *Eng Appl Artif Intell* 2021;104:104375. DOI
29. Huang YY, Pan QK, Gao L, Miao ZH, Peng C. A two-phase evolutionary algorithm for multi-objective distributed assembly permutation flowshop scheduling problem. *Swarm Evol Comput* 2022;74:101128. DOI
30. Wang ZY, Pan QK, Gao L, Wang YL. An effective two-stage iterated greedy algorithm to minimize total tardiness for the distributed flowshop group scheduling problem. *Swarm Evol Comput* 2022;74:101143. DOI
31. Kalayci CB, Gupta SM. A particle swarm optimization algorithm with neighborhood-based mutation for sequence-dependent disassembly line balancing problem. *Int J Adv Manuf Technol* 2013;69:197-209. DOI
32. Tseng YJ, Yu FY, Huang FY. A green assembly sequence planning model with a closed-loop assembly and disassembly sequence planning using a particle swarm optimization method. *Int J Adv Manuf Technol* 2011;57:1183-97. DOI
33. Bouazza S, Hassine H, Barkallah M, Amari S, Haddar M. Disassembly sequence optimization for profit and energy consumption using petri nets and particle swarm optimization. In: Ben Amar M, Bouguecha A, Ghorbel E, El Mahi A, Chaari F, Haddar M, editors. Advances in materials, mechanics and manufacturing II. Cham: Springer International Publishing; 2022. pp. 267-76. DOI
34. Yeh WC. Optimization of the disassembly sequencing problem on the basis of self-adaptive simplified swarm optimization. *IEEE Trans Syst Man Cybern A* 2012;42:250-61. DOI
35. Yeh WC. Simplified swarm optimization in disassembly sequencing problems with learning effects. *Comput Oper Res* 2012;39:2168-77. DOI
36. Li WD, Xia K, Gao L, Chao KM. Selective disassembly planning for waste electrical and electronic equipment with case studies on liquid crystaldisplays. *Robot Comput Integr Manuf* 2013;29:248-60. DOI
37. Wang K, Li X, Gao L, Li P, Sutherland JW. A discrete artificial bee colony algorithm for multiobjective disassembly line balancing of end-of-life products. *IEEE Trans Cybern* 2022;52:7415-26. DOI
38. Guo H, Zhang L, Ren Y, Li Y, Zhou Z, Wu J. Optimizing a stochastic disassembly line balancing problem with task failure via a hybrid variable neighborhood descent-artificial bee colony algorithm. *Int J Prod Res* 2023;61:2307-21. DOI
39. Zhang L, Wu Y, Zhao X, et al. A multi-objective two-sided disassembly line balancing optimization based on artificial bee colony algorithm: a case study of an automotive engine. *Int J Pr Eng Man GT* 2022;9:1329-47. DOI
40. Ren Y, Gao K, Fu Y, Sang H, Li D, Luo Z. A novel Q-learning based variable neighborhood iterative search algorithm for solving disassembly line scheduling problems. *Swarm Evol Comput* 2023;80:101338. DOI
41. Slama I, Ben-Ammar O, Dolgui A, Masmoudi F. Genetic algorithm and Monte Carlo simulation for a stochastic capacitated disassembly lot-sizing problem under random lead times. *Comput Ind Eng* 2021;159:107468. DOI
42. Wu T, Zhang Z, Zeng Y, Zhang Y. Mixed-integer programming model and hybrid local search genetic algorithm for human-robot collaborative disassembly line balancing problem. *Int J Prod Res* 2023;62:1758-82. DOI
43. Tseng HE, Chang CC, Lee SC, Huang YM. A block-based genetic algorithm for disassembly sequence planning. *Expert Syst Appl* 2018;96:492-505. DOI
44. Go TF, Wahab DA, Rahman MNA, Ramli R, Hussain A. Genetically optimised disassembly sequence for automotive component reuse. *Expert Syst Appl* 2012;39:5409-17. DOI
45. Alshibli M, El Sayed A, Kongar E, Sobh TM, Gupta SM. Disassembly sequencing using tabu search. *J Intell Robot Syst* 2016;82:69-79. DOI
46. Wang K, Li X, Gao L, Li P, Gupta SM. A genetic simulated annealing algorithm for parallel partial disassembly line balancing problem. *Appl Soft Comput* 2021;107:107404. DOI
47. Kalayci CB, Polat O, Gupta SM. A variable neighbourhood search algorithm for disassembly lines. *J Manuf Technol Manag* 2015;26:182-94. DOI
48. Ren Y, Zhang C, Zhao F, Triebe MJ, Meng L. An MCDM-based multiobjective general variable neighborhood search approach for disassembly line balancing problem. *IEEE Trans Syst Man Cybern Syst* 2020;50:3770-83. DOI
49. Akbay MA, Kalayci CB, Polat O. A parallel variable neighborhood search algorithm with quadratic programming for cardinality constrained portfolio optimization. *Knowl Based Syst* 2020;198:105944. DOI
50. Liu H, Zhang L. Optimizing a disassembly sequence planning with success rates of disassembly operations via a variable neighborhood search algorithm. *IEEE Access* 2021;9:157540-9. DOI
51. Tuncel E, Zeid A, Kamarthi S. Solving large scale disassembly line balancing problem with uncertainty using reinforcement learning. *J Intell Manuf* 2014;25:647-59. DOI
52. Zhao X, Li C, Tang Y, Cui J. Reinforcement learning-based selective disassembly sequence planning for the end-of-life products with structure uncertainty. *IEEE Robot Autom Lett* 2021;6:7807-14. DOI
53. Zhao F, Hu X, Wang L, Zhao J, Tang J, Jonrinaldi. A reinforcement learning brain storm optimization algorithm (BSO) with learning mechanism. *Knowl Based Syst* 2022;235:107645. DOI
54. Karimi-Mamaghan M, Mohammadi M, Pasdeloup B, Meyer P. 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* 2023;304:1296-330. DOI

55. Wang L, Gao K, Lin Z, Huang W, Suganthan PN. Problem feature based meta-heuristics with Q-learning for solving urban traffic light scheduling problems. *Appl Soft Comput* 2023;147:110714. DOI
56. Lin Z, Gao K, Wu N, Suganthan PN. Scheduling eight-phase urban traffic light problems via ensemble meta-heuristics and Q-learning based local search. *IEEE Trans Intell Transport Syst* 2023;24:14415-26. DOI
57. Yu H, Gao KZ, Ma ZF, Pan YX. Improved meta-heuristics with Q-learning for solving distributed assembly permutation flowshop scheduling problems. *Swarm Evol Comput* 2023;80:101335. DOI
58. Rummery GA, Niranjan M. On-line Q-learning using connectionist systems. Available from: https://www.researchgate.net/publication/2500611_On-Line_Q-Learning_Using_Connectionist_Systems. [Last accessed on 24 Jan 2024].
59. Li D, Gao K, Ren Y, Zhang R, Fu Y. The supplementary file of the paper "Integrating meta-heuristics and a Sarsa algorithm for disassembly scheduling problems with cycle time and hazard coefficients". Available from: https://www.researchgate.net/publication/377446410_The_supplementary_file_of_the_paper_Integrating_Metaheuristics_and_a_Sarsa_Algorithm_for_Disassembly_Scheduling_Problems_with_Cycle_Time_and_Hazard_Coefficients. [Last accessed on 24 Jan 2024].
60. Li H, Gao K, Duan PY, Li JQ, Zhang L. An improved artificial bee colony algorithm with Q-learning for solving permutation flowshop scheduling problems. *IEEE Trans Syst Man Cybern Syst* 2023;53:2684-93. DOI
61. Zhang Z, Liu H, Zhou M, Wang J. Solving dynamic traveling salesman problems with deep reinforcement learning. *IEEE Trans Neural Netw Learn Syst* 2023;34:2119-32. DOI
62. Zhao F, Di S, Wang L. A hyperheuristic with Q-learning for the multiobjective energy-efficient distributed blocking flow shop scheduling problem. *IEEE Trans Cybern* 2023;53:3337-50. DOI PubMed