Received: 8 June 2026 | Revised: 20 July 2026 | Accepted: 21 August 2026 | Published online: 22 September 2026

RESEARCH ARTICLE

Journal of Data Science and Intelligent Systems
2026, Vol. 00(00) 1–22
DOI: 10.47852/bonviewJDSIS620210735

BON VIEW PUBLISHING

Dynamic Multi-skill Project Scheduling Under Disruptions with Earliness–Tardiness Objectives

Hao Nguyen Thi[1,2], Loc Nguyen The[1] and Huu Dang Quoc[3,*]

[1] *School of Mathematics and Computer Science, Hanoi National University of Education, Vietnam*

[2] *Hung Vuong University, Vietnam*

[3] *Thuong Mai University, Vietnam*

Abstract: Existing studies on the Multi-Skill Resource-Constrained Project Scheduling Problem (MS-RCPSP) mostly assume static environments and focus solely on makespan minimization, which does not adequately reflect project scheduling under disruptions. In practice, disruptions may alter task durations during project execution, while completing projects either too early or too late relative to a deadline can incur substantial costs. To address this gap, this paper introduces the Dynamic Earliness–Tardiness Multi-Skill Resource-Constrained Project Scheduling Problem (DETMS-RCPSP), which jointly considers multi-skill resource allocation, task-duration disruptions, and earliness–tardiness optimization. We prove that DETMS-RCPSP is NP-hard and propose NGSA-DE, a self-adaptive differential evolution algorithm for reactive rescheduling. Experimental results on the iMOPSE benchmark show that NGSA-DE achieves better solution quality and stability than the baseline Differential Evolution and Particle Swarm Optimization algorithms across most tested instances. Unlike approaches that focus only on makespan, NGSA-DE also considers earliness and tardiness, which helps reduce the costs caused by projects finishing too early or too late when task durations are disrupted. The proposed approach can therefore improve reactive rescheduling and support workforce management decisions in dynamic project environments.

Keywords: MS-RCPSP, dynamic project scheduling, task-duration disruption, earliness–tardiness minimization, adaptive differential evolution

1. Introduction

In modern project management, resources often possess multiple skills, while tasks require specific skill combinations at defined proficiency levels. The Multi-Skill Resource-Constrained Project Scheduling Problem (MS-RCPSP) [1, 2] models this operational complexity, finding broad application in IT, engineering, and smart manufacturing. In these domain settings, balancing workforce flexibility against specialization directly dictates execution efficiency. Because MS-RCPSP is NP-hard [3, 4], researchers generally rely on metaheuristic algorithms to find high-quality schedules within practical time limits.

Most MS-RCPSP formulations [2, 3] rely on a static assumption: task durations remain fixed throughout execution. Yet real-world projects routinely face execution disruptions, such as duration shifts caused by site conditions [5] or unplanned resource unavailability [6], which quickly invalidate baseline schedules. Handling these uncertainties requires reactive scheduling approaches [5, 7] capable of adjusting allocations on the fly as conditions change.

Furthermore, standard MS-RCPSP formulations [1, 2, 8] focus almost exclusively on minimizing makespan or direct project cost, overlooking several key operational expenses. In practice, completing tasks early can generate storage, docking, or idle-waiting costs, whereas delays trigger contractual penalties and ripple through dependent projects [9, 10]. Evaluating schedules under an earliness–tardiness criterion [9] thus offers a far truer reflection of actual project performance.

Although previous studies [8, 11–13] have addressed dynamic environments, duration uncertainty, or earliness–tardiness objectives, few have integrated these factors within the MS-RCPSP framework. Existing models generally fall into three groups: those that examine multi-skilled workforces under static conditions, those that address dynamic scheduling without multi-skill constraints, and those that overlook the relationship between earliness–tardiness penalties and reactive rescheduling. A recent review of MS-RCPSP studies by Bahroun et al. [2] also points to the need for further research on this integrated setting. To address this gap, this paper proposes the Dynamic Earliness–Tardiness Multi-Skill Resource-Constrained Project Scheduling Problem (DETMS-RCPSP), which simultaneously incorporates multi-skill resource structures, dynamic changes in the durations of

***Corresponding author:** Huu Dang Quoc, Thuong Mai University, Vietnam. Email: huudq@tmu.edu.vn

selected tasks during execution, rescheduling mechanisms, and an earliness–tardiness optimization objective.

To solve the proposed problem, we develop the Neighborhood-Guided Self-Adaptive Differential Evolution (NGSA-DE) algorithm. Building upon the DE-based approach for MS-RCPSP introduced by Myszkowski et al. [14], the proposed algorithm integrates a neighborhood-guided mutation mechanism that selects the best individual in the neighborhood of the current individual to participate in the mutation process, thereby enhancing local exploitation performance. In addition, the crossover rate (CR) is self-adaptively updated based on the number of successful mutations, following modern parameter adaptation mechanisms [15, 16]. Benchmark testing demonstrates that our approach yields superior solution quality and stability compared to standard baselines.

This study makes three main contributions, as follows:

1) Formulate the new DETMS-RCPSP problem, a multi-skill resource-constrained project scheduling problem that incorporates disruptions in task execution times. The objective is to minimize the earliness–tardiness deviation of the project completion time from the prescribed completion time. The study proves that the proposed problem is NP-hard, thereby necessitating the use of evolutionary algorithms to obtain approximate solutions.
2) Propose the NGSA-DE algorithm for solving the DETMS-RCPSP. The algorithm is developed on the Differential Evolution (DE) framework with two major enhancements: a neighborhood-guided mutation operator that leverages elite local solutions to improve the search process, and an adaptive CR informed by historical performance feedback. These mechanisms enable the algorithm to effectively balance local exploitation and global exploration without requiring manual parameter tuning.
3) Conduct extensive experiments using the iMOPSE benchmark dataset to evaluate the proposed NGSA-DE algorithm for solving the DETMS-RCPSP. The experimental results are collected, aggregated, and analyzed by comparing NGSA-DE with the standard DE and Particle Swarm Optimization (PSO) algorithms. The results demonstrate that NGSA-DE achieves superior solution quality and search stability. The Friedman and Wilcoxon signed-rank tests are employed to assess the statistical significance of the results, while ablation experiments are conducted to evaluate the contribution of each component of the proposed algorithm.

The rest of this paper is organized as follows: Section 2 reviews related work, Section 3 presents the mathematical formulation, Section 4 describes the NGSA-DE algorithm, Section 5 presents the computational results, and Section 6 concludes the paper with possible directions for future research.

2. Related Work

The MS-RCPSP has attracted significant research interest, resulting in numerous high-quality publications presenting effective solution methods. This problem typically focuses on optimizing the makespan, cost, or both. In practice, projects are executed under economic contracts or agreements that specify predetermined product delivery deadlines. Consequently, developing a schedule that minimizes the earliness/tardiness deviation from these contractual deadlines is crucial for safeguarding the interests and reputations of all parties involved. This section reviews prior research regarding the four key aspects of the proposed approach: multi-skilled resource project scheduling, dynamic disruption management, earliness/tardiness optimization, and adaptive DE algorithms. This overview highlights the limitations of existing models while establishing the foundation and motivation for developing the DETMS-RCPSP model and the NGSA-DE algorithm.

2.1. Overview of MS-RCPSP and its variants

The RCPSP has been widely studied, resulting in many variants that address different real-world requirements. A recent comprehensive survey by Hartmann and Briskorn [17] shows that research on the RCPSP has developed in several directions, including resource configurations, optimization criteria, and execution environments. Among these extensions, MS-RCPSP is considered a significant extension, in which each resource may possess multiple skills, and each task requires a specific set of skills [2].

Early studies on MS-RCPSP primarily focused on exact methods. A notable example is the branch-and-bound method proposed by Bellenguez-Morineau and Néron [3] for solving the problem. However, due to high computational complexity, such methods are only applicable to small-scale instances. Consequently, heuristic and metaheuristic approaches rapidly emerged as the dominant research direction.

MS-RCPSP has since been extended in various directions to reflect practical characteristics better. Studies have examined multi-objective formulations with skill transfers [18], multi-project scheduling with shared resources [19], and practical constraints such as preemption, time windows, and resource leveling [8]. Previous studies by Snauwaert and Vanhoucke [4], as well as Myszkowski and Laszczyk [20], have also focused on problem classification and benchmark dataset generation to support algorithm evaluation and comparison. Despite these advances, most existing formulations still assume static conditions and overlook execution uncertainties. Recently, Vermeire and Vanhoucke [21] proposed a priority-rule-based heuristic for the MS-RCPSP that pairs a parallel scheduling scheme with a novel resource assignment procedure, achieving near-optimal solutions with high computational efficiency. Additionally, Snauwaert and Vanhoucke [22] introduced a solution framework for MS-RCPSP with hierarchical skills, in which the skill level of a resource directly influences task execution duration, thereby enriching the problem's modeling space.

2.2. Solution methods for MS-RCPSP

Given the NP-hard nature of the problem, metaheuristic and hybrid approaches have become the predominant approaches for solving large-scale MS-RCPSP instances. Representative methods include hybrid algorithms, such as DE combined with a greedy heuristic (DEGR) [14], efficient hybrid approaches, such as MEMINV [23], and swarm intelligence-based algorithms, such as R-PSO [24]. Quality-aware scheduling methods have also been proposed: Peng et al. [25] developed an MS-RCPSP model that incorporates a quality-propagation mechanism and network restructuring, enabling simultaneous optimization of project duration and quality risk. More recently, hybrid evolutionary algorithms have also been proposed for disruption-aware MS-RCPSP. Su et al. [26] developed a reactive multi-objective optimization model incorporating overtime, proactive preemption, and skill-dependent setup times, while Ghasemi et al. [27] introduced a reliability-aware bi-objective model with stochastic machine-service disruptions solved using a hybrid

NSGA-II algorithm. These studies demonstrate the effectiveness of evolutionary optimization in handling disruptions and uncertainty in multi-skilled project scheduling. In addition, multi-objective evolutionary methods with specialized operators [28] and iterated local search strategies [12] have been proposed to improve exploitation of the solution space.

Despite achieving promising results, these methods are primarily designed for static environments and generally lack effective adaptation mechanisms when problem conditions change during execution.

2.3. RCPSP under uncertainty and dynamic environments

In practice, projects are frequently affected by uncertainty and disruptions. Numerous studies have examined task-duration uncertainty [29], resource variability [6], and project execution disruptions [5, 7]. These approaches include both proactive and reactive strategies for dealing with dynamic environments. Machine learning techniques, particularly reinforcement learning, have also been employed to address scheduling problems under uncertainty [30–32]. From an analytical perspective, Chen et al. [33] developed a proactive–reactive scheduling framework based on time buffers for the RCPSP with uncertain task durations. Their findings show that combining proactive buffering with reactive adjustments improves schedule stability and robustness compared with relying on either strategy alone. Furthermore, Rahimifard et al. [34] proposed an integrated simulation framework that combines discrete-event simulation with a multi-agent system to simultaneously handle multiple sources of uncertainty in project scheduling, enabling more flexible and realistic modeling of dynamic feedback loops. Satic et al. [35] proposed an Approximate Dynamic Programming (ADP) algorithm combined with simulation for the dynamic-stochastic multi-project scheduling problem with uncertain task durations and stochastically arriving projects, though without incorporating multi-skill structures. These methods can learn how to adapt scheduling decisions as project conditions change. However, most existing studies focus on general RCPSP formulations and do not explicitly consider multi-skill constraints or reactive rescheduling.

2.4. Earliness–tardiness optimization criteria

In most existing studies, the predominant optimization objective is to minimize makespan. However, this criterion does not fully capture the real-world costs of projects. Several works have examined earliness–tardiness criteria in project scheduling [9, 10] and in multi-skill contexts with deadline constraints [11]. Nevertheless, the integration of this criterion within dynamic environments subject to disruptions remains limited.

2.5. Differential evolution and adaptive variants

DE is an effective population-based metaheuristic for solving complex optimization problems. Numerous studies have focused on enhancing DE's performance through adaptive parameter adjustment mechanisms and the dynamic selection of evolutionary operators [16, 36]. These research directions can be categorized into parameter adaptation, mutation strategy selection, and hybridization mechanisms [16]. Specifically, JADE [37] and L-SHADE [38] adjust the scaling factor (F) and CR based on the success history of candidate solutions, whereas other methods dynamically adjust CR based on fitness feedback [15]. Although these mechanisms improve the balance between exploration and exploitation, they typically rely on population-level fitness information or historical data without directly exploiting neighborhood relationships within the solution space. This is considered a limitation when solving scheduling problems, where the quality of a solution can be significantly influenced by solutions in its immediate neighborhood.

In the context of project scheduling, Vermeire and Vanhoucke [21] demonstrated that combining priority rules with adaptive resource management mechanisms can yield high-quality schedules while maintaining relatively low computational complexity. Their findings indicate that incorporating problem-specific adaptive mechanisms into the scheduling process allows for the effective exploitation of the inherent structural information of project scheduling problems. This research lays the groundwork for extending adaptive mechanisms—moving beyond conventional parameter adjustment—toward strategies that directly exploit local information within the solution space. Recent studies continue to affirm the effectiveness of adaptive and hybrid metaheuristic mechanisms for complex optimization problems, including the Resultant Vector Strategy [39], the Improved Mountain Gazelle Optimizer [40], the Discrete Puma Optimizer [41], and the Adaptive Elephant Optimizer [42]. Overall, these studies highlight the potential of combining adaptive control with problem-specific search mechanisms to enhance the balance between exploration and exploitation. However, the direct integration of neighborhood information and adaptive hybridization mechanisms into DE to solve the MS-RCPSP has not yet been fully investigated. Addressing this research gap, the present study incorporates a neighborhood-guided search strategy and an adaptive hybridization mechanism into the DE algorithm. The proposed mechanisms are designed to exploit local information within the solution space while flexibly adjusting the search process, thereby enhancing the DE algorithm's ability to handle the complex precedence, resource, and mode-selection constraints characteristic of this problem.

2.6. Comparative analysis of related work

Table 1 summarizes the main differences between the proposed approach and representative studies, focusing on multi-skilled resources, dynamic disruptions, earliness–tardiness objectives, and solution methods. As can be observed, the simultaneous integration of these aspects remains limited in existing studies.

In Table 1, RA denotes either reactive rescheduling in a dynamic context or adaptive search mechanisms within optimization algorithms. The comparison shows that existing studies usually address only part of the problem. None of the reviewed studies combines multi-skilled resources, disruption-driven dynamic scheduling, earliness–tardiness criteria, and adaptive mechanisms within the same framework.

The comparison points to three main research gaps:

1) Existing models rarely integrate multi-skilled workforce allocation, dynamic duration disruptions, and earliness–tardiness trade-offs simultaneously. For instance, Wang et al. [6] examined multi-skilled resource scheduling under dynamic availability, whereas Zhang et al. [32] compared several optimization algorithms, including policy-gradient-based reinforcement learning methods, for deterministic multi-skilled project scheduling. However, neither study considered earliness–tardiness scheduling objectives. In contrast, Pamay et al. [10] addressed earliness–tardiness objectives in a dynamic resource-constrained multi-project scheduling environment but did not incorporate multi-skilled resource structures.

Table 1
Comparison of related studies on key problem dimensions and solution approaches

Refs.	MS	DD	ET	RA	Method type
Exact methods and classification					
[3]	✓				Exact
[4]	✓				Classification
[9]			✓		Exact
Static MS-RCPSP					
[11]	✓		✓		Mathematical
[13]	✓				Exact/heuristic
[18]	✓				Evolutionary
[21]	✓				Priority rule heuristic
[22]	✓				Math. framework
[23]	✓				Hybrid
[24]	✓				PSO
[25]	✓				Metaheuristic
Dynamic RCPSP					
[5]		✓		✓	Reactive
[6]	✓	✓			Optimization
[7]		✓		✓	Reactive
[10]		✓	✓		Mathematical
[33]		✓		✓	Proactive-reactive
[34]		✓		✓	Simulation
[35]		✓			ADP + Simulation
Reinforcement learning (RL)-based					
[30]		✓		✓	RL
[31]		✓		✓	Deep RL
[32]	✓	✓		✓	RL
DE and adaptive variants					
[14]	✓				DE hybrid
[15]				✓	Adaptive DE
[37]				✓	JADE
[38]				✓	L-SHADE
This work	✓	✓	✓	✓	NGSA-DE

Note: ✓ = feature considered; blank = feature not considered. MS = multi-skill; DD = dynamic/disruption handling; ET = earliness–tardiness objective; RA = reactive rescheduling or adaptive search mechanism.

2) Few optimization methods combine local neighborhood information with global historical search feedback. Existing adaptive methods, such as JADE [37] and L-SHADE [38], mainly rely on historical success information and do not explicitly use neighborhood information.

3) Existing DE-based methods for the MS-RCPSP (e.g., [14]) do not include effective parameter adaptation mechanisms. Meanwhile, advanced adaptive DE variants [37, 38] have not yet been applied to multi-skill project scheduling under dynamic disruptions.

These gaps motivate the DETMS-RCPSP formulation and the NGSA-DE algorithm proposed in this study.

3. Problem Formulation

The DETMS-RCPSP problem is defined as follows. A project consists of a set of predefined tasks, each with a specific duration and requirements regarding the skill types and proficiency levels of the resources performing them. Precedence relationships exist among the tasks, such that a task can only commence after all its preceding tasks have been completed. The project is executed by a finite set of resources, each possessing one or more skill types with corresponding proficiency levels. A resource can perform only one task at any given time. During project execution, unexpected disruptions may alter the actual durations of the tasks. The objective is to determine a schedule that minimizes the total penalty cost associated with early completion or delays relative to the predetermined project deadline.

3.1. Notation

The sets, parameters, and decision variables used in the DETMS-RCPSP formulation are summarized in Table 2.

3.2. Dynamic model

Consider a project comprising a task set T^* with predefined precedence relations, resources, and skills. During project execution, a disruption is assumed to occur when $a\%$ of the tasks has already commenced, where $a \in (0, 100)$. At this point, the task set is partitioned into two subsets:

Tasks that have commenced: $J^{start} = \{T_i \in T^* \mid T_i \text{ has already started}\}$; $|J^{start}| = a\% \cdot tcount$

Tasks that have not yet commenced: $J^{not} = \{T^* \setminus J^{start}\}$

The number of affected tasks is modeled using a Beta–Binomial distribution. The Beta–Binomial distribution allows the proportion of affected tasks to vary across scenarios, thereby reflecting the non-uniform variability observed in practice. Disruption probability distribution: $p \sim \text{Beta}(\alpha, \beta)$, where α, β are two parameters controlling the proportion of affected tasks.

Number of changed tasks: $N^{chg} \sim \text{Binomial}\left(|J^{not}|, p\right)$

Set of affected tasks: $J^{chg} \subseteq J^{not}, \quad |J^{chg}| = N^{chg}$

The disruption status of each task follows a Bernoulli distribution: $y_i \sim \text{Bernoulli}(p), \quad \forall T_i \in J^{not}$

In project scheduling research, task durations are commonly modeled using the Program Evaluation and Review Technique (PERT), in which execution times follow a Beta distribution over a bounded interval, characterized by three parameters: optimistic, most likely, and pessimistic durations [43, 44]. PERT is widely used in project scheduling because it can represent uncertain activity durations using three estimates. In this study, when a disruption occurs, the original duration of the affected task is treated as the most likely value in the PERT model. The optimistic and pessimistic values are determined according to a predefined uncertainty level. This keeps the original task-duration data unchanged while allowing the duration to vary within the specified range. The updated duration of task T_i is determined as follows:

$$d_i' = \begin{cases} d_i \text{ if } T_i \in J^{start} \\ d_i(1+\epsilon_i) \quad \text{if } T_i \in J^{chg} \\ d_i \text{ if } T_i \in J^{not} \setminus J^{chg} \end{cases} \qquad \epsilon_i \sim \text{PERT}(-\delta_1, 0, \delta_2) \tag{1}$$

where δ_1 is the lower bound (the task can be shortened by at most $\delta_1 \times 100\%$), δ_2 is the upper bound (the task can be extended by at most $\delta_2 \times 100\%$), and 0 is the most likely value, corresponding to no change.

Table 2
List of notations for the DETMS-RCPSP formulation

Symbol	Description
R	Set of resources in the project
R_m	Resource with index m
T	Set of tasks in the project
T_i	Task with index i
R^i	Set of resources capable of executing task T_i; $R^i \in R$
$rcount$	Total number of resources in the project
T^m	Set of all tasks that can be executed by resource R_m, T^m
P_i	Set of predecessor tasks that must be completed before task T_i
$tcount$	Total number of tasks in the project
S	Set of all skills available across all resources
S^m	Set of skills that resource R_m may possess; $S^m \subseteq S$
S_j	Skill with index j
A^i	Set of all skill requirements needed to execute task T_i.
r_j	Proficiency level of skill S_j
g_j	Type of skill S_j
B_i, C_i	Start time and completion time of task T_i, respectively
d_i	Execution duration of task T_i.
$V^t_{m,i}$	Binary variable at time t indicating whether resource R_m is executing task T_i; $V^t_{m,i} = 1$ if task T_i is being executed by resource R_m at time t, and 0 otherwise
X	A feasible schedule for project execution
X_{all}	Set of all feasible schedules for project execution
$f(X)$	Objective function of schedule X
Ft, Lt	First and last tasks scheduled in schedule X, respectively
J^{start}	Set of tasks that have already commenced execution at the time under consideration
J^{not}	Set of tasks that have not yet commenced execution at the time under consideration
J^{chg}	Set of tasks whose execution durations have changed
N^{chg}	Number of tasks whose execution duration has changed
$M^{deadline}$	Project completion deadline

The completion time of each task is determined based on its start time and the updated execution duration: $C_i = B_i + d'_i, \quad \forall T_i \in T^*$. Tasks that have already commenced retain their execution schedule at the time of disruption. The remaining constraints are identical to those in the MS-RCPSP. The objective function of the DETMS-RCPSP is defined as:

$$f(X) = e \cdot \max\left(0, M^{deadline} - C_{Lt}\right) + l \cdot \max\left(0, C_{Lt} - M^{deadline}\right) \quad (2)$$

where e *and* l denote the penalty weights for earliness and tardiness, respectively. However, the present study focuses exclusively on minimizing the deviation between the actual project completion time and the specified deadline. Since a project can only be completed either early or late, the earliness and tardiness penalty terms in Equation (2) are mutually exclusive and never become active simultaneously. Furthermore, all experiments are conducted with $e = l = 1$, reducing the objective function to the absolute completion-time deviation, $f(X) = \left|C_{Lt} - M^{deadline}\right|$. The explicit e and l coefficients are retained only to preserve the generality of the formulation and facilitate future extensions involving asymmetric earliness and tardiness penalties, such as inventory holding costs or contractual delay penalties.

In the objective function, the first term penalizes project completion earlier than the deadline, whereas the second term penalizes tardy project completion. Therefore, $f(X) \geq 0$ for all feasible schedules. When the project is completed exactly on time ($C_{Lt} = M^{deadline}$), the objective function attains its optimal value $f(X) = 0$. Accordingly, the problem is to determine a schedule X that minimizes the objective function, that is, $f(X) \to min$.

3.3. Constraints of the DETMS-RCPSP formulation

The formulation described above leads to four main groups of constraints: task precedence, resource allocation, skill requirements, and schedule feasibility under dynamic disruptions. The corresponding constraints are summarized in Table 3.

Constraint (3): Each resource possesses at least one skill. Constraints (4, 5): The completion time of the task (T_i) is determined by its start time and its non-negative execution duration. Constraint (6): Task (T_i) can only start after its predecessor task (T_j) has been completed. Constraint (7): Resource R_m can execute task T_i only if it possesses the required skill type with a skill level greater than or equal to the required level. Constraint (8): Each task is assigned to exactly one resource. Constraint (9): At any given time, each resource can execute at most one task. Constraints (10, 11): When a% of the tasks have already commenced, the task set is partitioned into two subsets: J^{start}, representing the set of tasks that have already started, and J^{not}, representing the set of tasks that have not yet started. Constraints (12, 13): Determine the number and set of tasks with modified execution durations J^{chg} based on the Beta–Binomial and Bernoulli distributions. Constraint (14): Update the new execution durations d'_i for affected tasks using the PERT distribution. Constraint (15): Update the relationship between task start and completion times. Constraint (16): Preserve the original schedule for tasks that have already started, where $\overline{B_i}$ and $\overline{C_i}$ denote the original start and completion times of task T_i, respectively.

Example 1. Consider a project whose information is provided in Table 4, with a project deadline $M^{deadline} = 13$.

Based on the project information above, the task network can be represented in Figure 1, while a Gantt chart of a feasible project schedule is illustrated in Figure 2.

Assume that a disruption occurs after seven tasks have already commenced, that is, $J^{start} = \{T_1, T_2, T_3, T_4, T_5, T_6, T_8\}$, while the remaining three tasks, $J^{not} = \{T_7, T_9, T_{10}\}$, have not yet started. The disruption affects two tasks, resulting in $J^{chg} = \{T_7, T_{10}\}$, where the execution duration of T_7 increases from 3 to 4 time units, whereas the duration of T_{10} decreases from 3 to 2 time units. The updated Gantt chart is illustrated in Figure 3.

The project completion time corresponds to the completion time of task T_{10}, that is, $C_{T_{10}} = 12$. Therefore, the objective function value is $f(X) = \max(0,13\text{-}12) = 1$,

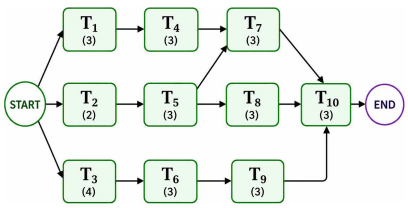
Table 3
Constraints of the DETMS-RCPSP

$S^j \neq \varnothing,\ \forall R_j \in R$	(3)
$d_i \geq 0,\ \forall T_i \in T$	(4)
$C_i \geq 0,\ \forall\ T_i \in T$	(5)
$C_{\mathrm{j}} \leq\ B_i, \forall T_i \in T, i \neq 1, T_j \in P_i$	(6)
$\forall R_m \in R^i, \exists S_j \in S^m : g_{S_j} = g_{A^i},\ r_{S_q} \geq r_{A^i}$	(7)
$\forall T_i \in T, \forall t \in (B_{Ft}, C_{Lt}),\ \exists! R_m \in R :\ V^t_{m,i} = 1$	(8)
$\forall R_m \in R, \forall t \in (B_{Ft}, C_{Lt}) :\ \sum_{i=1}^{tcount} V^t_{m,i} \leq 1$	(9)
$J^{start} = \{T_i \in T^* \mid T_i\ \text{has already started}\}, \lvert J^{start}\rvert = \lfloor a\%.tcount \rfloor$	(10)
$J^{not} = T^* \setminus J^{start}$	(11)
$p \sim \mathrm{Beta}(\alpha, \beta), N^{chg} \sim \mathrm{Binomial}(\lvert J^{not}\rvert, p)$	(12)
$y_i \sim \mathrm{Bernoulli}(p), \forall T_i \in J^{not}, y_i \in 0,1,\ J^{chg} = \{T_i\ \in J^{not} \lvert y_i = 1\},\ \lvert J^{chg}\rvert = N^{chg}$	(13)
$d'_i = \begin{cases} d_i & \text{if } T_i \in J^{start} \cup \left(J^{not} \setminus J^{chg}\right) \\ d_i(1+\epsilon_i) & \text{if } T_i \in J^{chg} \end{cases} \quad \epsilon_i \sim PERT(-\delta_1, 0, \delta_2)$	(14)
$C_i = B_i + d'_i, \quad \forall i \in T^*$	(15)
$B_i = \overline{B_i}, \quad C_i = \overline{C_i} \quad \forall i \in J^{start}$	(16)

Table 4
Project information

Task	Task name	Duration	Skill	Predecessor
T_1	Requirements analysis	3	(Dev, 2)	
T_2	System design	2	(Test, 1)	
T_3	Database design	4	(Design, 2)	
T_4	Module *A* development	3	(Dev, 3)	T_1
T_5	Test case development	3	(Test, 3)	T_2
T_6	Database implementation	3	(DB, 3)	T_3
T_7	Module *B* development	3	(Dev, 4)	T_4, T_5
T_8	Module *A* testing	3	(Test, 4)	T_5
T_9	System testing	3	(Design, 4)	T_6
T_{10}	Integration and deployment	3	(DB, 3)	T_7, T_8, T_9
Resource		Resource's skills		
	R_1	(Dev, 5), (Test, 2), (DB, 3)		
	R_2	(Test, 4), (Dev, 2), (Design, 1)		
	R_3	(Design, 5), (DB, 4), (Dev, 2)		

Figure 1
Task network of the project



with the corresponding project schedule represented as $X = \{R_1, R_2, R_3, R_1, R_2, R_3, R_1, R_2, R_3, R_3\}$.

3.4. NP-hard proof of the DETMS-RCPSP

In this section, we prove that the DETMS-RCPSP is NP-hard.

Theorem 1. The decision version of DETMS-RCPSP, denoted by DETMS-RCPSP-D, is NP-hard. Consequently, the optimization version of DETMS-RCPSP is also NP-hard.

To establish the NP-hard of the DETMS-RCPSP optimization problem, we show that no polynomial-time algorithm can solve the problem optimally in the general case unless $P = NP$. The proof is based on a sequence of polynomial-time reductions from known NP-hard problems:

$$\text{3-PARTITION} \leq_p \text{MS-RCPSP} \leq_p \text{DETMS-RCPSP} \quad (17)$$

Here, the notation $P1 \leq_p P2$ indicates that problem $P1$ can be transformed into problem $P2$ in polynomial time. Therefore, problem $P2$ is at least as difficult as problem $P1$. The decision version of DETMS-RCPSP, denoted by DETMS-RCPSP-D, is defined as follows.

Figure 2
Gantt chart of a project schedule

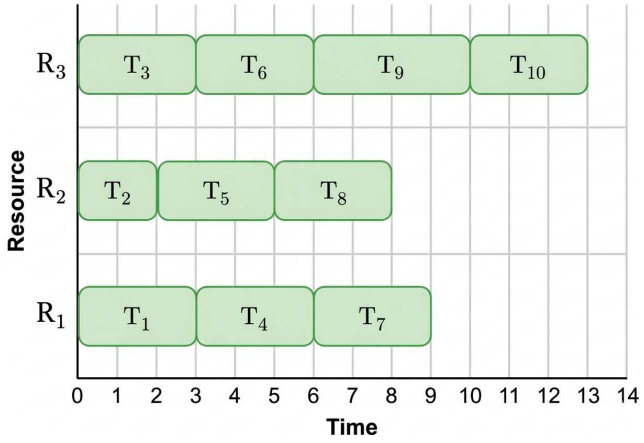
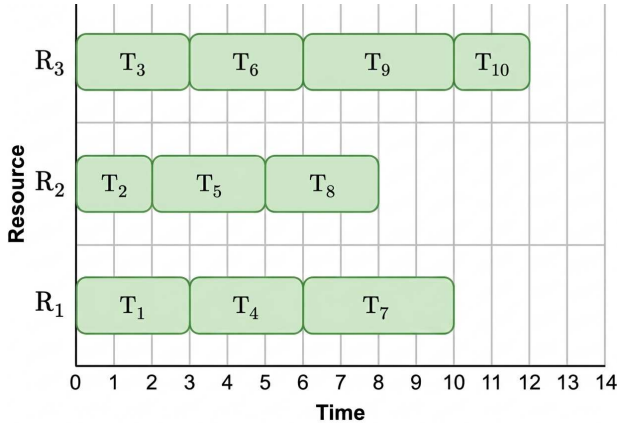


Figure 3
Gantt chart of the schedule after the disruption



Given an instance of DETMS-RCPSP consisting of a task set T, a resource set R, a skill set S, precedence constraints P, task durations d, skill requirements A, disruption parameters, penalty weights e and l, a project deadline $M^{deadline}$, and a threshold $K \geq 0$, determine whether there exists a feasible schedule X such that $f(X) \leq K$. The objective function $f(X)$ is defined in Equation (2). To prove that DETMS-RCPSP is NP-hard, it is sufficient to show that its decision version, DETMS-RCPSP-D, is NP-hard. The proof proceeds in two steps:

Step 1: Show that DETMS-RCPSP-D belongs to the class NP.

Step 2: Construct a polynomial-time reduction from MS-RCPSP to DETMS-RCPSP-D, that is, MS-RCPSP $\leq_p$ DETMS-RCPSP-D

3.4.1. Proof that DETMS-RCPSP-D belongs to the class NP

Consider a candidate schedule X. The feasibility of X can be verified in polynomial time by performing the following checks in Table 5.

Overall, the verification procedure requires polynomial time with respect to the input size, with total complexity bounded by: $O\left(|T|^2 + |T| \cdot |S| + |R| \cdot C_{Lt}\right)$.

Since every certificate (schedule) can be verified in polynomial time, the decision problem DETMS-RCPSP-D belongs to the class NP: DETMS-RCPSP-D $\in$ NP.

3.4.2. Proof by polynomial-time reduction: MS-RCPSP $\leq_p$ DETMS-RCPSP-D

To establish the NP-hard of DETMS-RCPSP-D, we construct a polynomial-time reduction from the known NP-hard problem MS-RCPSP.

Construction of the Reduction: Consider an arbitrary instance $I = (T, R, S, P, d, A)$ of MS-RCPSP together with a makespan bound C^*. We construct an instance I′ of DETMS-RCPSP-D as in Table 6.

Under these settings, the objective function becomes:

$$\begin{aligned} f(X) &= 0 \cdot max\,(C^* - C_{Lt}, 0) + 1 \cdot max\,(0, C_{Lt} - C^*) \\ &= max\,(0, C_{Lt} - C^*) \end{aligned} \tag{18}$$

The decision condition of DETMS-RCPSP-D is $f(X) \leq K$. Since $K = 0, \max(0, C_{Lt} - C^*) \leq 0$. This condition holds if and only if $C_{Lt} \leq C^*$. Because C_{Lt} represents the completion time of the final task, it is precisely the project makespan C_{max}. Therefore, $C_{Lt} \leq C^* \iff C_{max} \leq C^*$

This is exactly the decision question of MS-RCPSP: whether there exists a feasible schedule whose makespan does not exceed the threshold C^*.

Consequently, an MS-RCPSP instance I admits a feasible schedule with makespan at most C^* if and only if the constructed DETMS-RCPSP-D instance I' admits a feasible schedule satisfying $f(X) \leq 0$. The transformation from I to I' requires only copying the original instance and assigning constant parameter values. Therefore, the reduction can be performed in polynomial time: $O(|T| + |R| + |S|)$, where (|T|), (|R|), and (|S|) denote the numbers of tasks, resources, and skills, respectively. Since MS-RCPSP is NP-hard [3, 4], and MS-RCPSP $\leq_p$ DETMS-RCPSP-D, DETMS-RCPSP-D is also NP-hard.

Table 5
Verification steps and computational complexity

Step	Verification condition	Specific condition	Cost
1	Check precedence constraints	$\forall T_i, \forall T_m \in P_i:\ C_m \leq B_i$	$O\left(\lvert T\rvert^2\right)$
2	Check skill constraints	$\forall R_m \in R^i, \exists S_j \in S^m:\ g_{S_j} = g_{A^i},\ r_{S_j} \geq r_{A^i}$	$O(\lvert T\rvert . \lvert S\rvert)$
3	Check that each resource executes at most one task at any time	$\forall R_m, \forall t:\ \sum_{i=1}^{tcount} V_{m,i}^t \leq 1$	$O(\lvert R\rvert . C_{Lt})$
4	Check dynamic constraints (3.10)–(3.16): update d_i'	$C_i = B_i + d_i', \forall T_i \in T^*$	$O(\lvert T\rvert)$
5	Compute $f(X)$ and compare with K	$f(X) \leq K$?	$O(1)$

Table 6
Construction of a DETMS-RCPSP-D instance from an MS-RCPSP instance

Component in I'	Assigned value	Rationale/consequence
Task set, resource set, and skill set	Identical to I	Preserves the original project structure
Precedence constraints P'	Identical to P	Preserves task dependencies
Disruption parameter a	$a = 0\%$	Implies $J^{start} = \varnothing$
Beta-Binomial parameters	$\alpha \to \infty, \beta \to \infty$	Produces $p \approx 0$, hence $N^{chg} = 0$
Consequence	$d'_i = d_i$	$J^{chg} = \varnothing$
Deadline $M^{deadline}$	$M^{deadline} = C^*$	Corresponds to the makespan threshold
Earliness weight e	$e = 0$	Eliminates earliness penalties
Tardiness weight l	$l = 1$	Penalizes only tardiness
Threshold K	$K = 0$	Requires zero penalty

Combining the results from Sections 3.4.1 and 3.4.2, we obtain:

+ DETMS-RCPSP-D $\in$ NP, and

+ MS-RCPSP $\leq_p$ DETMS-RCPSP-D.

Since MS-RCPSP is NP-hard, DETMS-RCPSP-D is NP-hard. Consequently, the optimization version of DETMS-RCPSP is also NP-hard. Therefore, unless $P = NP$, no polynomial-time algorithm exists that can solve DETMS-RCPSP optimally in the general case.

4. Proposed Algorithm

This section presents the complete design of the NGSA-DE algorithm for the DETMS-RCPSP. The section is organized as follows: (1) identification of the not-yet-started task set; (2) schedule reconstruction; (3) fitness function design; (4) neighborhood-guided self-adaptive mutation mechanism (*SAdapt*); (5) complete pseudocode of the NGSA-DE algorithm; and (6) complexity analysis.

4.1. Identification of the set of not-yet-started tasks

When executing a schedule X under a $a\%$ disruption scenario, a disruption may occur, causing the processing times of certain tasks that have not yet started to deviate from their originally planned values, resulting in earlier or later completion than initially scheduled. To model this effect, it is necessary to identify the subset of tasks that have not yet commenced at the time of disruption.

Let X denote a feasible schedule of the DETMS-RCPSP. The objective of this procedure is to extract the set of not-yet-started tasks, denoted by J^{not}, which may be affected by subsequent dynamic changes.

Input: A feasible schedule X; Output: The set of not-yet-started tasks J^{not}. Identification procedure

- Step 1: Task sorting. Sort all tasks in the schedule X in ascending order of their expected completion times, resulting in a sequence denoted by *TaskSort*.

- Step 2: Reverse traversal and set construction. Traverse *TaskSort* from the last element to the first. For each task, perform the following operations:

+ Step 2.1: Let *iEnd* denote the expected completion time of the current task.

+ Step 2.2: Identify all tasks that have not yet started at the disruption time.

+ Step 2.3: Identify all tasks whose start times are strictly less than *iEnd*, forming the set J^{start}.

+ Step 2.4: Initialize the not-yet-started set $J^{not} = \varnothing$.

+ Step 2.5: Scan *TaskSort* from the end and add all tasks that have not yet started into J^{not}.

+ Step 2.6: Repeat Step 2.5 until $|J^{not}|$ reaches the predefined disruption threshold and then terminate and return J^{not}.

- Step 3: Termination condition. If no valid subset satisfying the disruption threshold is identified, return $J^{not} = \varnothing$.

Example 2: Based on the schedule provided in Example 1, the procedure is illustrated below.

- Step 1: Tasks are sorted in ascending order of their expected completion times to obtain *TaskSort*, as presented in Table 7. Table 7 can also be represented as a Gantt chart, as illustrated in Figure 4.

- Step 2: Reverse traversal of *TaskSort*

Table 7
Tasks sorted in ascending order of completion times in the schedule

No.	Task	Start time	Completion time
1	T_2	0	2
2	T_1	0	3
3	T_3	0	4
4	T_5	2	5
5	T_4	3	6
6	T_6	4	7
7	T_8	5	8
8	T_7	6	9
9	T_9	7	10
10	T_{10}	10	13

The traversal results are summarized in Table 8. At Iteration 6, when task T_4 is considered with $iEnd = 6$, the cardinality of J^{not} reaches the predefined disruption threshold. Consequently, the procedure terminates and returns $J^{not} = \{T_7, T_9, T_{10}\}$. This result indicates that tasks T_7, T_9, and T_{10} have not yet started at the disruption point and therefore constitute the set of disruption-sensitive tasks. According to the dynamic disruption model described in Section 3.2, only tasks belonging to J^{not} are eligible for duration modifications. Consequently, the processing times

Figure 4
Tasks arranged in ascending order of completion times

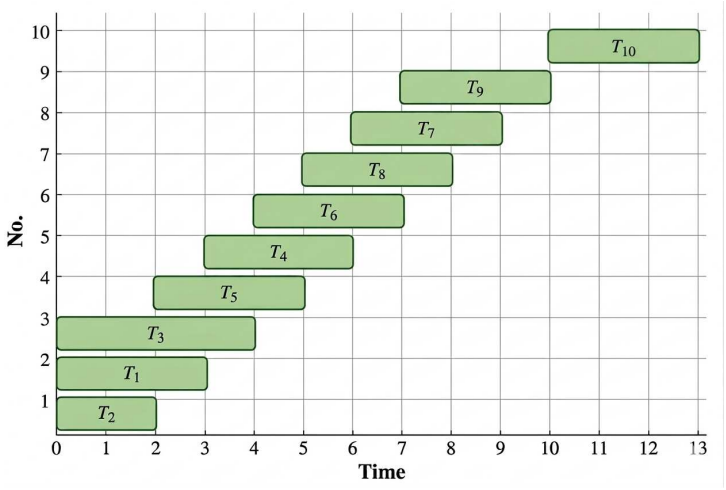


Table 8
Results of reverse traversal of $TaskSort$ in Example 2

Iteration	Current task	*iEnd*	J^{start}	J^{not}
1	T_{10}	13	$\{T_1, \ldots, T_{10}\}$	$\varnothing$
2	T_9	10	$\{T_1, \ldots, T_{10}\}$	$\varnothing$
3	T_7	9	$\{T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9\}$	$\{T_{10}\}$
4	T_8	8	$\{T_1, \ldots, T_9\}$	$\{T_{10}\}$
5	T_6	7	$\{T_1, \ldots, T_8\}$	$\{T_9, T_{10}\}$
6	T_4	6	$\{T_1, T_2, T_3, T_4, T_5, T_6, T_8\}$	$\{T_7, T_9, T_{10}\}$

of T_7, T_9, and T_{10} may be adjusted during the schedule recovery phase, whereas all previously started tasks retain their original durations.

4.2. Schedule reconstruction procedure

For each project schedule, the following feasibility requirements must be satisfied: (i) every task must be assigned to a resource; (ii) the completion time of each task must equal its start time plus its processing duration; and (iii) all precedence constraints among tasks must be respected. Based on these requirements, the schedule reconstruction procedure is performed as follows.

Input: A feasible schedule X; Output: A reconstructed feasible schedule $\widehat{X}$.

- Step 1: Resource assignment. Determine the assignment of tasks to resources.

- Step 2: Initial schedule generation. Compute the start and completion times of all tasks on their assigned resources.

- Step 3: Precedence-feasibility adjustment. Update the start times of successor tasks according to the precedence constraints. Recalculate the start and completion times of all affected tasks until all precedence relations are satisfied.

- Step 4: Identification of not-yet-started tasks. Determine the set of not-yet-started tasks J^{not} using the procedure described in Section 4.1.

- Step 5: Determination of the number of affected tasks. Randomly determine the number of tasks whose processing times may deviate from their original values.

- Step 6: Selection of affected tasks. Randomly select tasks from J^{not} until the number of selected tasks reaches the quantity determined in Step 5.

- Step 7: Duration adjustment. For each selected task, randomly generate a new processing time according to the disruption model while preserving the predefined variability characteristics.

- Step 8: Local schedule update. Recompute the start and completion times of all tasks whose processing times have been modified on their assigned resources.

- Step 9: Global precedence-feasibility reconstruction. Based on the precedence constraints among the not-yet-started tasks, recalculate the start times of successor tasks. Then update the start and completion times of all affected tasks to ensure that all precedence constraints remain satisfied.

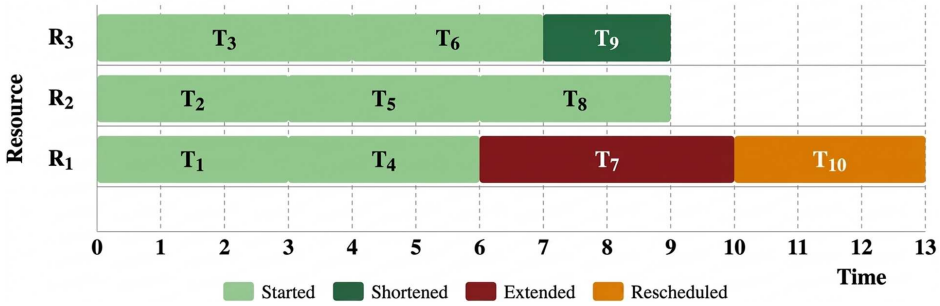
- Step 10: Schedule reconstruction completion. Obtain the reconstructed schedule $\widehat{X}$, which satisfies all resource, temporal, and precedence constraints of the DETMS-RCPSP.

Example 3. Continuing from Example 2, after identifying the set of not-yet-started tasks corresponding to a 30% disruption threshold, $J^{not} = \{T_7, T_9, T_{10}\}$, assume that tasks T_7 and T_9 undergo processing-time changes. The schedule reconstruction procedure described in Section 4.2 is then applied to all tasks in J^{not} to restore schedule feasibility. Figure 5 presents the reconstructed schedule.

4.3. Fitness function design

The DETMS-RCPSP aims to minimize the earliness–tardiness deviation with respect to a committed project deadline, denoted by $M^{deadline}$. When a disruption occurs during

Figure 5
Reconstructed project schedule after the disruption



project execution, the actual project completion time C_{Lt} may deviate from the planned deadline. Consequently, the objective is to identify a schedule that minimizes the total weighted deviation from $M^{deadline}$.

In practice, project managers typically establish project plans based on an initially optimized schedule and communicate the corresponding completion deadline to stakeholders. When a disruption occurs during project execution, the schedule is revised to keep the project completion time as close as possible to the original commitment. The project should not finish earlier than planned, as this may lead to unnecessary resource waiting or idle time, and it should not finish later, as this may result in schedule violations. Here, $M^{deadline}$ is determined from the initially optimized schedule and is used as the completion deadline for rescheduling after a disruption. Because no additional time buffer is included in the initial schedule, the algorithm must handle the disruption without relying on preallocated slack. Consequently, the algorithm must recover schedule feasibility and maintain deadline adherence under unfavorable conditions. Investigating the impact of alternative deadline settings on solution quality and convergence behavior represents an interesting direction for future research.

Based on the objective function of DETMS-RCPSP defined in Equation (2), the fitness value of an individual x in the DE population is computed as:

$$fitness(x) = e \cdot max\left(0, M^{deadline} - C_{Lt}(x)\right) + l \cdot max\left(0, C_{Lt}(x) - M^{deadline}\right) \tag{19}$$

where:

Symbol	Description
x	Candidate schedule individual in the DE population
$C_{Lt}(x)$	Actual completion time of the final task under schedule x after disruption
$e \geq 0$	Earliness penalty weight
$l \geq 0$	Tardiness penalty weight
$\text{fitness}(x) = 0$	Optimal value indicating completion exactly at the committed deadline

It can be observed that the fitness function is identical to the objective function of DETMS-RCPSP. Therefore, minimizing $fitness(x)$ is equivalent to minimizing the total weighted earliness–tardiness deviation from the committed project deadline. The optimal fitness value is achieved when the project is completed exactly at $M^{deadline}$, yielding $\text{fitness}(x) = 0$.

4.4. Neighborhood-guided mutation method

In the standard DE algorithm, the mutation vector is generated using the global best individual P_{best} and two randomly selected individuals from the population. This strategy improves convergence speed, but it uses only the global best solution and does not consider the local neighborhood of the current individual P_i. Since the earliness–tardiness objective in DETMS-RCPSP can have multiple local optima, this may lead to premature convergence and loss of population diversity.

We therefore introduce a neighborhood-guided mutation mechanism, denoted by *SAdapt*, which uses a star-shaped neighborhood constructed from fitness information. For a given individual P_i, the center of the star corresponds to P_i, while the neighboring vertices are formed by the w individuals whose fitness values are closest to that of P_i. Specifically, an individual P_j belongs to the neighborhood of P_i if:

$$\left|\text{fitness}\left(P_j\right) - \text{fitness}\left(P_i\right)\right| \leq D_i \tag{20}$$

where D_i is a fitness-distance threshold selected such that exactly (w) individuals are included in the neighborhood set P_S.

Defining neighborhoods according to fitness proximity offers two important advantages. First, it groups individuals with comparable solution quality, thereby identifying schedules that lie within the same region of the search space. Second, the resulting mutation directions become more locally informative than those obtained through purely random selection, leading to enhanced exploitation capability while preserving search diversity.

The *SAdapt* function returns the best individual within the neighborhood of P_i, defined as:

$$P^i_{best} = SAdapt\left(P_{all}, i, w\right) = arg \min_{P_j \in P_S} fitness\left(P_j\right) \tag{21}$$

where P_{all} denotes the current population, and P_S is the neighborhood set associated with P_i.

The neighborhood-best solution P'^{i}_{best} subsequently replaces the global best individual P_{best} in the mutation operator. Consequently, the search direction becomes adaptive to the local landscape surrounding P_i, enabling the algorithm to perform more effective local refinement while reducing the likelihood of premature convergence.

To balance exploration and exploitation throughout the evolutionary process, the neighborhood size w is decreased linearly from w_{max} to w_{min} over successive generations. At early stages, a larger neighborhood encourages broader exploration of the search space, whereas a smaller neighborhood in later generations promotes intensive exploitation around promising regions. This adaptive mechanism allows the algorithm to gradually shift from global exploration to local refinement as convergence progresses.

The detailed pseudocode of the *SAdapt* procedure is presented in Algorithm 1.

Algorithm 1: SAdapt

- Input: P_{all}: current population; i: index of the current individual; w: number of neighboring individuals to select
- Output: P^{i}_{best}: best individual in the neighborhood set P_S

Begin
1. $P_S \leftarrow \varnothing$
2. mp ← fitness(P_{all}[i])
   // fitness value of the current individual
3. $P^{sort}_{all} \leftarrow$ Sort(P_{all}, fitness)
   // ascending order of fitness values
4. $D_i \leftarrow$ fiDistance(P_{all}, i, w)
   // fitness-distance threshold ensuring that
   // exactly w neighboring individuals are selected
5. for each individual $P_j \in P^{sort}_{all}$ do
6. if $P_j \neq P_{all}$[i] then
7. if |fitness(P_j) − mp| ≤ D_i then
8. $P_S \leftarrow P_S \cup P_j$
9. end if
10. if $|P_S| = w$ then
11. break
12. end if
13. end if
14. end for
15. $P^{i}_{best} \leftarrow \arg\min_{P_j \in P_S} fitness(P_j)$
16. return P^{i}_{best}

End

Here *fiDistance*() computes the fitness-distance threshold centered at P_i such that exactly w neighboring individuals are included in the neighborhood set P_S.

4.5. NGSA-DE algorithm

The two key components of NGSA-DE, namely, the neighborhood-guided mutation mechanism SAdapt and the self-adaptive crossover-rate CR mechanism, are jointly designed based on the following observation. The effectiveness of neighborhood-guided mutation depends on a sufficiently large CR to transmit beneficial directional information from the mutation vector to the offspring. Conversely, the self-adaptive CR mechanism can generate meaningful feedback only when the mutation operator produces high-quality search directions. Consequently, the two mechanisms are intrinsically coupled and should be activated simultaneously throughout the evolutionary process. The mutation operator employed in NGSA-DE is defined as:

$$v_j = \left| P_{i,j} + F \cdot \left(P^{i}_{best} - P_{i,j} \right) + F \cdot \left(P_{k1,j} - P_{k2,j} \right) \right| \quad (22)$$

where P^{i}_{best} denotes the neighborhood-best individual identified by the *SAdapt* procedure. The adaptive parameter mCR is updated according to [37]:

$$mCR \leftarrow (1 - c) \cdot mCR + c \cdot mean(aScr) \quad (23)$$

Equation (23) updates the historical CR parameter using information collected from successful offspring. Combined with the neighborhood-guided mutation strategy, this adaptation mechanism enables NGSA-DE to dynamically balance exploration and exploitation throughout the search process.

Where:

best	Index of the neighborhood-best individual selected by *SAdapt*
CR_i	Crossover rate associated with individual i
F	Mutation scaling factor sampled from $U(0,1)$
aScr	Archive of successful crossover-rate values
c	Learning coefficient, $c \sim U(0,1)$.
$mean(aScr)$	Mean value of successful CR_i entries stored in *aScr*

Consequently, the algorithm can effectively exploit promising regions while maintaining sufficient population diversity to mitigate premature convergence.

The complete NGSA-DE procedure is presented in Algorithm 2.

Algorithm 2: NGSA-DE for DETMS-RCPSP

- Input: Dataset; N: population size; *maxGen*: maximum number of generations; e, l: earliness and tardiness penalty weights; w_{max}, w_{min}: neighborhood size bounds.
- Output: bestSolution, bestObj.

Begin
1. P ← InitPopulation(N)
2. bestObj ← +∞
3. bestSolution ← null
4. for *gen* ← *1* to *maxGen* do
   // Step 1: Fitness evaluation
5. for *i* ← *0* to *N−1* do
6. dynamicMS ← EvaluateDynamic(P[i])
7. fitness[i] ← ComputeFitness(dynamicMS)
   // Equation (19)
8. if *fitness[i]* < *bestObj* then
9. bestObj ← fitness[i]
10. bestSolution ← P[i]
11. end if
12. end for
13. c ← U(0,1)
14. aScr ← ∅
   // Step 2: Evolutionary search
15. for *i* ← *0* to *N−1* do
   // Neighborhood identification
16. $P^{i}_{best} \leftarrow SAdapt(P, i, w)$
   // Parameter generation
17. $CR_i \leftarrow$ randnc(mCR, 0.1)
18. $F \leftarrow U(0,1)$
19. Select k1, k2 ∈ {0,...,N−1} such that $k1 \neq k2 \neq i$
20. $I_{rand} \leftarrow random(1, tcount)$
   // Mutation (Equation 4.4)
21. for *j* ← *1* to *tcount* do
22. $v_j \leftarrow |P_{i,j} + F\left(P^{i}_{best} - P_{i,j}\right) + F(P_{k1,j} - P_{k2,j})|$
23. end for
   // Binomial crossover
24. for *j* ← *1* to *tcount* do
25. if $U(0,1) \leq CR_i$ OR $j = I_{rand}$ then
26. $u_j \leftarrow v_j$
27. else

28. $u_j \leftarrow P_{i,j}$
29. end if
30. end for
// Constraint handling
31. $u \leftarrow simplebound(u)$
// Fitness evaluation
32. dynamicMS_u ← EvaluateDynamic(u)
33. z ← ComputeFitness(dynamicMS_u)
// Equation (19)
// Selection
34. if $z \leq fitness[i]$ then
35. $P[i] \leftarrow u$
36. $fitness[i] \leftarrow z$
37. $aScr \leftarrow aScr \cup CR_i$
38. end if
// Update global best
39. if $z < bestObj$ then
40. $bestObj \leftarrow z$
41. bestSolution ← u
42. end if
43. end for
// Step 3: Adaptive parameter update
44. if $aScr \neq \emptyset$ then
45. $mCR \leftarrow (1 - c)\,mCR + c \cdot mean(aScr)$
// Equation (23)
46. end if
47. $w \leftarrow w_{max} - (gen/maxGen)(w_{max} - w_{min})$
48. if $bestObj = 0$ then
49. break
50. end if
51. end for
52. return bestSolution, bestObj
End

where:

InitPopulation()	Generates the initial population
ComputeFitness()	Evaluates the fitness value according to Equation (19)
EvaluateDynamic()	Computes the actual project completion time C_{Lt} under the dynamic disruption environment, as described in detail in Algorithm 3
simplebound()	Repairs infeasible solutions and enforces variable bounds, as described in detail in Algorithm 4
SAdapt(P,i,w)	Identifies the neighborhood of the individual P_i and returns the neighborhood-best solution P^i_{best}

Algorithm 3: EvaluateDynamic()

Input: individual – candidate schedule
Output: objectiveValue
Begin
1. Determine J^{start} and J^{not} at the disruption point a (Section 3.2)
// Step 1: Generate disruption scenario
2. p ← Beta(α, β)
3. N^{chg} ← Binomial(|J^{not}|, p)
4. J^{chg} ← BernoulliSample(J^{not}, p, N^{chg})
// Step 2: Update affected task durations
5. for each $T_i \in J^{chg}$ do
6. Resample duration(T_i) from the PERT distribution
7. end for
// Step 3: Evaluate candidate schedule
8. Rebuild the schedule using the updated durations
9. objectiveValue ← f(individual)
10. return objectiveValue
End

Algorithm 4: simplebound()

- Input: *individual* – real-valued candidate vector (one gene per task)
- Output: repaired integer vector x (feasible resource assignment)
Begin
1. x ← empty vector of size |*individual*|
// Step 1: Enforce variable bounds (non-negativity and integrality)
2. for $i \leftarrow 0$ to |individual| − 1 do
3. x[i] ← round(|individual[i]|)
4. end for
// Step 2: Repair infeasible solutions (resource eligibility)
5. for $i \leftarrow 0$ to |individual| − 1 do
6. R_i ← set of eligible resources for task i
7. if $|R_i| = 1$ then
8. x[i] ← R_i[0]
9. else if $x[i] \notin R_i$ then
10. idx ← x[i] mod $|R_i|$
11. x[i] ← R_i[idx]
12. end if
13. end for
14. return x
End

4.6. Complexity analysis of the NGSA-DE algorithm

Based on the structure of Algorithm 2, the computational complexity of the major components of NGSA-DE is summarized in Table 9.

Table 9
Computational complexity of the main components of NGSA-DE

Component	Complexity
Population initialization	$(O(N \times tcount \times rcount))$
EvaluateDynamic(x)	$(O(tcount^2))$
Population evaluation	$(O(N \times tcount^2))$
SAdapt (per individual)	$(O(N \log N))$
SAdapt (entire population)	$(O(N^2 \log N))$
Mutation + Crossover	$(O(N \times tcount))$
One generation (overall)	$(O(N \times tcount^2 + N^2 \log N))$

Population initialization: Each individual is represented by a vector of *tcount* tasks. Generating N individuals and verifying feasibility with respect to resource-skill compatibility and precedence constraints requires
$O(N \times tcount \times rcount)$.

Schedule evaluation: For a given individual, the schedule reconstruction procedure repeatedly updates task start and completion times to satisfy precedence relations. In the worst case, multiple passes over the task set may be required, yielding a complexity of

$O(tcount^2)$. Evaluating the entire population, therefore, requires $O(N \times tcount^2)$.

SAdapt procedure: For each individual, *SAdapt* consists of two main operations: Sorting the population according to fitness values: $O(N \log N)$; Traversing the sorted population to identify neighboring individuals: $O(N)$. Therefore, the complexity of *SAdapt* for a single individual is $O(N \log N + N) = O(N \log N)$. Applying *SAdapt* to all N individuals results in $O(N^2 \log N)$.

Mutation and crossover: For each individual, the mutation and crossover operators process all (tcount) decision variables. Consequently, the complexity for the entire population is $O(N \times tcount)$.

Complexity per generation: Combining all dominant components yields the computational complexity of one generation: $O(N \times tcount^2 + N^2 \log N)$. Since the algorithm executes at most *maxGen* generations, the overall time complexity of NGSA-DE is

$O(maxGen \times N \times (tcount^2 + N \log N))$. Therefore, the dominant computational cost of NGSA-DE arises from schedule evaluation and neighborhood construction, corresponding to the terms $O(N \times tcount^2)$ and $O(N^2 \log N)$, respectively.

5. Performance Analysis

To demonstrate the effectiveness of the proposed algorithm in solving the DETMS-RCPSP, the NGSA-DE algorithm is implemented and experimentally evaluated. The experimental results are collected, aggregated, analyzed, and compared with those obtained by other algorithms. In addition, based on the experimental results, statistical analyses using the Friedman and Wilcoxon signed-rank tests are conducted to provide further insights into the performance of the proposed algorithm and to assess the statistical significance of the observed results.

5.1. Experimental setup

We evaluated NGSA-DE using 30 instances from the publicly available iMOPSE benchmark dataset [45]. Each instance describes a project with precedence relationships among activities, a multi-skilled workforce, and task-specific skill requirements. The selected instances were divided into two groups according to project size. The first group contains 15 instances with 100 tasks and 5, 10, or 20 resources. The second group contains 15 instances with 200 tasks and 10, 20, or 40 resources. The instances also differ in their precedence constraints, resource availability, and skill requirements.

All algorithms were implemented in C# and executed on the same hardware platform. For each benchmark instance, every algorithm was run independently 30 times to reduce the effect of random variation in the evolutionary process and obtain more stable performance estimates.

DETMS-RCPSP is a newly formulated problem that considers a dynamic execution environment and an earliness–tardiness objective. No existing method has been developed specifically for this problem. Earlier approaches, such as DEGR [14] and R-PSO [24], were developed for the static MS-RCPSP with a makespan objective. Applying these methods to DETMS-RCPSP would require changes to their objective evaluation and disruption-handling mechanisms. We therefore use DE as the main baseline, since NGSA-DE is developed from DE. PSO is also included as a representative swarm-based method.

The experimental study proceeds in two stages. The first compares NGSA-DE with DE and PSO to evaluate its performance against these two population-based methods. The second is an ablation study that examines the effects of the two components introduced in NGSA-DE. The first, *SAdapt*, is a mutation strategy that guides the search using information from high-quality neighboring individuals. The second, *Adaptive CR*, adjusts the crossover probability based on the success of recent mutations.

To measure what each component contributes, we build two intermediate variants within the same implementation, each with one component switched off. Keeping everything else fixed means any difference in performance can be traced to the component being tested, not to differences in implementation or setup. The disruption point a is a configurable input of the framework. Depending on the scenario, it can either be set by the user or drawn at random before optimization starts. In the experiments reported here, we set $a = 0.7$ so that all benchmark instances were evaluated under the same conditions.

The five algorithms compared in this study are as follows:

1) NGSA-DE. The proposed algorithm integrates both *SAdapt* and *Adaptive CR*. During mutation, search directions are guided toward the best neighboring individual identified by *SAdapt*, while the CR is dynamically adjusted based on feedback from successful mutations collected throughout the evolutionary process.
2) DE-SAdapt. An intermediate variant that preserves the *SAdapt*-based neighborhood-guided mutation strategy while employing a fixed CR. This variant is designed to evaluate the isolated contribution of neighborhood exploitation.
3) DE-ACR. An intermediate variant that employs the Adaptive CR mechanism but replaces neighborhood-guided mutation with the conventional global-best-oriented mutation strategy. This variant is used to evaluate the independent contribution of adaptive crossover control.
4) DE. The standard DE algorithm, without any adaptive or neighborhood-guided enhancements, serves as the fundamental reference method.
5) PSO. The standard PSO algorithm is included as a representative swarm intelligence method, providing a fundamentally different population-based search paradigm from DE.

The relationships among the five methods are summarized in Table 10, where each column corresponds to a specific component and the symbol ✓ indicates that the component is enabled.

Table 10
Mechanism configuration of the compared algorithms

Method	SAdapt	Adaptive CR
DE		
PSO		
DE-SAdapt	✓	
DE-ACR		✓
NGSA-DE	✓	✓

Note: ✓ indicates that the corresponding component is enabled; blank indicates that it is disabled.

To ensure a fair comparison, all algorithms were implemented within the same programming environment and executed under identical experimental conditions, including population initialization procedures, population size, maximum number of

generations, and the ET-based fitness evaluation function under dynamic disruptions. The parameter values used by the compared algorithms were selected according to recommendations from the original studies and preliminary calibration experiments. Table 11 summarizes the parameter settings adopted throughout the experimental evaluation.

Table 11
Parameter settings of the compared algorithms

Parameter	NGSA-DE	DE	PSO
Population size N	50	50	50
Max generations	1000	1000	1000
F	$U(0,1)$	$U(0,1)$	—
Initial mCR	0.5	$U(0,1)$	—
Initial neighborhood upper bound w_{max}	Random integer in [4, 10]	—	—
Initial neighborhood lower bound w_{min}			
Inertia weight ω	—	—	0.7
c^1, c^2	—	—	1.5

We evaluate NGSA-DE from five angles: how it compares against the other algorithms overall, how much each of its mechanisms contributes, how it converges, whether its advantages are statistically significant, and how robust it is. For the significance testing, we rely on two nonparametric tests: the Friedman test with Holm's post hoc procedure to compare overall rankings, and the Wilcoxon signed-rank test for pairwise comparisons.

5.2. Experimental results

This study analyzes the results from several angles to build a full picture of how NGSA-DE performs. We begin with an overall comparison against the competing algorithms across the iMOPSE instances. Next comes a contribution analysis, which separates what the *SAdapt* and *Adaptive CR* mechanisms each add and what they achieve together. We then turn to convergence, tracking how the algorithms behave over the course of the evolutionary process. A nonparametric statistical test follows, checking whether the performance gaps we observe are actually significant. Finally, we assess robustness through variability- and distribution-based measures, looking at how stable and consistent each algorithm stays across different instances.

5.2.1. Overall performance comparison

Experiments were conducted on the iMOPSE benchmark dataset, which consists of 30 instances. Among them, 15 instances contain 100 tasks with resource levels of 5, 10, and 20, while the remaining 15 instances contain 200 tasks with resource levels of 10, 20, and 40. The instances are denoted from iM1 to iM30.

Tables 12 and 13 present the statistical results, including the mean, standard deviation (Std), best, and worst values obtained by the compared algorithms for solving the DETMS-RCPSP. The experimental results reveal a clear separation between the two groups of algorithms. The first group, consisting of NGSA-DE (*mean* = 2.20) and DE-SAdapt (*mean* = 3.09), significantly outperforms the second group, which includes DE-ACR (*mean* = 17.42), DE (*mean* = 15.47), and PSO (*mean* = 31.64). The

Table 12
Mean ± standard deviation of the compared algorithms on the iMOPSE benchmark instances

Dataset	NGSA-DE	DE	PSO	DE-ACR	DE-SAdapt
iM1	2.10 ± 2.23	20.70 ± 39.73	15.80 ± 6.20	16.60 ± 31.61	7.80 ± 5.88
iM2	2.00 ± 2.26	0.80 ± 1.87	32.40 ± 36.24	1.20 ± 1.75	3.90 ± 4.58
iM3	2.30 ± 2.26	0.90 ± 1.10	5.00 ± 8.62	13.70 ± 34.17	0.70 ± 0.95
iM4	1.10 ± 1.20	8.00 ± 5.16	20.30 ± 8.93	23.10 ± 24.12	1.00 ± 1.56
iM5	2.11 ± 1.69	2.78 ± 3.03	56.33 ± 9.12	1.22 ± 1.56	3.67 ± 3.32
iM6	3.18 ± 1.60	2.36 ± 2.25	32.09 ± 8.48	13.18 ± 12.37	1.45 ± 1.44
iM7	0.90 ± 1.29	21.30 ± 31.15	23.30 ± 7.60	6.60 ± 10.71	3.90 ± 3.35
iM8	1.80 ± 2.39	5.60 ± 6.33	23.40 ± 4.22	12.90 ± 10.63	1.20 ± 1.62
iM9	2.90 ± 2.33	3.10 ± 2.69	38.80 ± 10.54	7.40 ± 13.34	2.10 ± 2.38
iM10	1.50 ± 1.27	3.80 ± 6.71	29.00 ± 7.10	19.30 ± 16.94	1.00 ± 1.33
iM11	2.10 ± 2.02	2.50 ± 2.42	6.80 ± 4.42	5.50 ± 5.91	1.80 ± 2.04
iM12	0.90 ± 1.29	7.70 ± 6.06	5.20 ± 4.26	10.50 ± 16.93	0.70 ± 1.06
iM13	1.30 ± 1.95	6.20 ± 10.62	26.70 ± 4.40	6.90 ± 10.15	1.40 ± 1.35
iM14	1.67 ± 1.22	2.67 ± 4.95	10.44 ± 3.47	67.00 ± 65.42	3.00 ± 2.69
iM15	2.55 ± 2.62	23.45 ± 37.14	20.45 ± 5.28	16.27 ± 40.69	2.55 ± 2.38
iM16	1.90 ± 2.42	17.30 ± 29.40	79.00 ± 7.57	15.60 ± 20.62	19.60 ± 10.35
iM17	3.30 ± 1.89	32.20 ± 58.04	65.00 ± 17.29	4.50 ± 4.84	1.60 ± 2.01
iM18	2.90 ± 3.41	6.90 ± 8.81	64.80 ± 14.94	7.50 ± 17.14	3.10 ± 2.81
iM19	2.00 ± 1.70	9.40 ± 8.53	64.70 ± 7.06	13.90 ± 15.88	2.60 ± 2.72
iM20	2.50 ± 2.17	2.90 ± 2.42	56.20 ± 4.10	46.80 ± 28.60	1.40 ± 1.51
iM21	1.60 ± 1.35	15.50 ± 22.17	44.60 ± 6.60	42.60 ± 29.23	1.40 ± 1.96
iM22	1.50 ± 1.78	3.70 ± 5.25	64.60 ± 7.07	25.70 ± 17.44	7.30 ± 5.56

(*Continued*)

Table 12
(*Continued*)

Dataset	NGSA-DE	DE	PSO	DE-ACR	DE-SAdapt
iM23	1.30 ± 1.16	23.80 ± 32.15	37.40 ± 5.52	5.20 ± 5.87	2.40 ± 2.88
iM24	2.50 ± 1.72	61.10 ± 81.13	48.90 ± 10.35	11.00 ± 15.99	2.60 ± 2.63
iM25	1.40 ± 1.17	21.40 ± 57.89	35.20 ± 5.85	23.80 ± 26.67	4.10 ± 2.38
iM26	4.90 ± 3.51	23.70 ± 22.10	8.30 ± 3.74	18.90 ± 27.89	3.00 ± 4.57
iM27	2.60 ± 3.03	61.90 ± 77.29	22.90 ± 3.73	3.40 ± 5.36	2.10 ± 2.60
iM28	3.80 ± 2.25	67.30 ± 98.97	3.00 ± 2.16	49.20 ± 76.27	1.70 ± 1.64
iM29	3.00 ± 2.49	0.30 ± 0.67	1.70 ± 1.64	34.70 ± 58.26	1.80 ± 2.20
iM30	2.10 ± 1.20	2.70 ± 3.68	8.20 ± 3.71	2.40 ± 2.22	2.10 ± 2.51
Overall	2.20 ± 2.15	15.47 ± 37.86	31.64 ± 23.85	17.42 ± 30.84	3.09 ± 4.72

Table 13
Best/worst values of the compared algorithms on the iMOPSE benchmark instances

Dataset	NGSA-DE	DE	PSO	DE-ACR	DE-SAdapt
iM1	0/7	0/96	9/28	0/79	0/16
iM2	0/6	0/6	0/89	0/4	0/16
iM3	1/7	0/3	0/27	0/110	0/3
iM4	0/3	2/18	7/32	0/65	0/5
iM5	0/5	0/8	43/70	0/5	1/11
iM6	1/7	0/6	20/51	0/36	0/4
iM7	0/4	1/76	8/38	0/36	0/10
iM8	0/8	0/22	15/30	0/34	0/5
iM9	1/9	0/9	21/54	1/45	0/7
iM10	0/4	0/22	20/42	4/55	0/3
iM11	0/5	0/7	0/14	0/18	0/6
iM12	0/4	0/17	0/12	0/48	0/3
iM13	0/6	0/35	17/32	0/29	0/4
iM14	0/4	0/15	4/14	2/149	0/8
iM15	0/9	0/105	10/27	0/138	0/7
iM16	0/7	0/78	65/88	1/59	4/32
iM17	1/6	0/143	25/81	0/10	0/6
iM18	0/10	0/24	51/100	0/56	0/9
iM19	0/5	1/29	56/78	0/52	0/8
iM20	0/7	1/8	50/63	24/121	0/4
iM21	0/4	0/61	34/56	5/94	0/6
iM22	0/6	0/13	55/77	2/47	0/14
iM23	0/3	0/91	25/44	0/19	0/7
iM24	0/5	1/177	32/60	2/54	0/9
iM25	0/3	0/186	24/44	2/85	1/8
iM26	0/10	3/67	1/13	1/88	0/13
iM27	0/9	1/168	17/28	0/18	0/8
iM28	0/7	7/326	0/8	1/206	0/5
iM29	0/6	0/2	0/5	1/159	0/7
iM30	0/3	0/12	2/13	0/7	0/8
Overall	0/10	0/326	0/100	0/206	0/32

performance gap between these two groups is substantial: the average mean objective value of the first group (2.64) is approximately eight times lower than that of the second group (21.51), indicating that the improvements introduced by NGSA-DE and DE-SAdapt represent substantial enhancements in search effectiveness rather than marginal refinements.

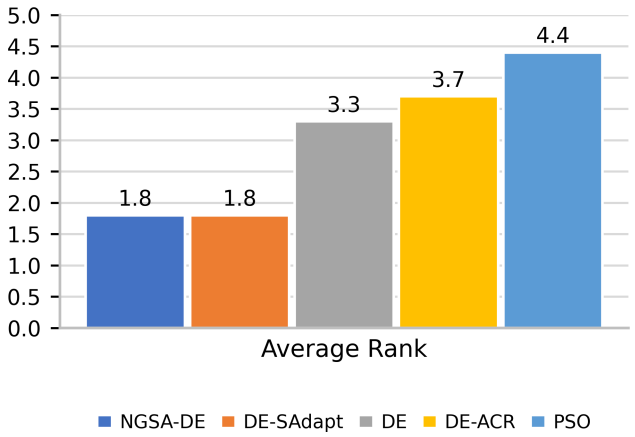
To facilitate a clearer overall comparison, the detailed results reported in Tables 12 and 13 are summarized using aggregate performance indicators, including average rank and overall mean performance statistics across the 30 benchmark instances.

As shown in Table 14 and Figure 6, NGSA-DE achieves the best overall performance, obtaining the lowest overall mean objective value (2.2) and sharing the best average rank (1.8) together with DE-SAdapt. DE-SAdapt attains a slightly higher mean (3.09) but ties with NGSA-DE in average rank, and both algorithms remain substantially superior to DE, DE-ACR, and PSO. The

Table 14
Overall performance summary

Algorithm	Overall mean	Overall Std	Average rank
NGSA-DE	2.2	2.15	1.8
DE-SAdapt	3.09	4.72	1.8
DE	15.47	37.86	3.3
DE-ACR	17.42	30.84	3.7
PSO	31.64	23.85	4.4

Figure 6
Average rank comparison



near-identical average ranks of NGSA-DE and DE-SAdapt suggest that neighborhood-guided mutation accounts for most of the performance improvement, while the additional gain from NGSA-DE is mainly reflected in lower variability (Std = 2.15 vs 4.72), indicating improved robustness and more consistent worst-case behavior. In contrast, DE, DE-ACR, and PSO consistently produce much larger objective values, indicating inferior search effectiveness on the DETMS-RCPSP benchmark.

Across the 15 small-scale instances (100 tasks, iM1–iM15), NGSA-DE achieves a mean objective value of 1.89, compared with 2.49 for the 15 large-scale instances (200 tasks, iM16–iM30), indicating that the algorithm maintains consistently strong performance as problem scale increases, with only a marginal increase in mean deviation.

To further investigate the source of the observed performance improvements, the contribution of the proposed neighborhood search (*SAdapt*) and *Adaptive CR* mechanisms is analyzed in the next subsection.

5.2.2. Contribution analysis

Since DE, DE-SAdapt, DE-ACR, and NGSA-DE correspond to different enhancement configurations derived from the same evolutionary framework, the experimental design naturally provides a structured basis for quantifying both the independent and combined contributions of the proposed improvement mechanisms. Table 15 and Figure 7 summarize the contribution analysis of each component incorporated into NGSA-DE.

Comparing DE with DE-ACR shows that adding the adaptive CR mechanism alone does not improve average performance; the mean objective value increases slightly from 15.47 to 17.42. This does not necessarily mean that *Adaptive CR* is ineffective. Rather, the result suggests that the benefit of CR adaptation depends on the quality of the mutation vectors generated during evolution. When mutation follows the conventional DE strategy, adjusting the crossover probability alone does not provide enough diversification to improve solution quality. Even so, DE-ACR reduces variability, with the *CV* decreasing from 2.45 to 1.77 and the worst-case result improving from 326 to 206.

Among the individual components, *SAdapt* has the largest effect. Compared with standard DE, DE-SAdapt reduces the mean objective value from 15.47 to 3.09, corresponding to an approximately fivefold improvement. The result indicates that neighborhood-guided mutation helps the algorithm exploit promising regions of the search space, escape local optima, and converge faster toward good solutions. Its worst-case value, however, remains relatively high at 32, suggesting that performance can still be unstable on some difficult instances without adaptive parameter control.

NGSA-DE combines neighborhood-guided mutation with adaptive CR control in a single evolutionary framework. The two mechanisms complement each other: neighborhood search provides better mutation vectors, while *Adaptive CR* learns from successful mutations and gradually adjusts the recombination probability to the problem characteristics. As shown in Table 15, NGSA-DE achieves a mean objective value of 2.20, about 29% lower than that of DE-SAdapt (3.09). The worst-case value also decreases from 32 to 10, a reduction of about 69%, while the coefficient of variation falls from 1.53 to 0.98. These improvements are larger than those obtained from either component alone, suggesting a synergistic effect between the two mechanisms. In particular, *Adaptive CR* appears to benefit more when the mutation vectors are generated using neighborhood information, which supports the rationale for combining the two mechanisms in NGSA-DE.

Overall, the results identify neighborhood-guided mutation as the main contributor to the improvement in solution quality. *Adaptive CR* mainly contributes to robustness by reducing variability and limiting poor worst-case outcomes. When combined, the two mechanisms give NGSA-DE the best balance between solution quality and search stability, supporting the proposed co-design strategy.

5.2.3. Convergence analysis

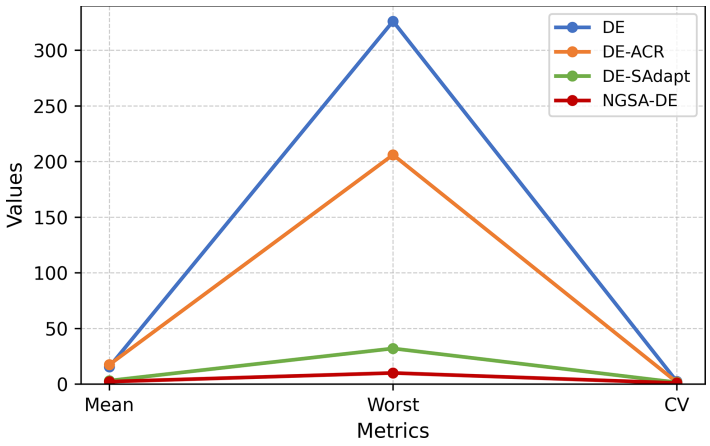
Final solution quality is not the only thing that matters; how quickly an algorithm converges also says a lot about how well

Table 15
Contribution analysis of the proposed mechanisms in NGSA-DE

Algorithm	Adaptive CR	Neighborhood search	Mean	Worst	CV
DE			15.47	326	2.45
DE-ACR	✓		17.42	206	1.77
DE-SAdapt		✓	3.09	32	1.53
NGSA-DE	✓	✓	2.20	10	0.98

Note: ✓ indicates that the corresponding mechanism is enabled. *CV* denotes the coefficient of variation (*Std*/*mean*).

Figure 7
Contribution analysis of the proposed mechanisms based on mean, worst, and CV

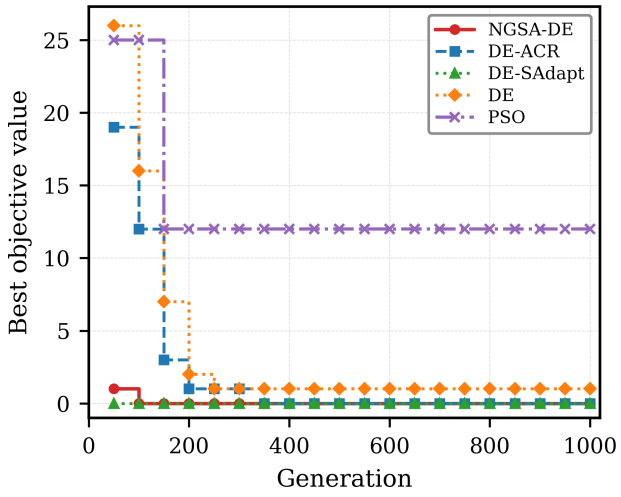


its search strategy works. To see what the neighborhood-guided mutation and adaptive crossover mechanisms contribute here, we recorded convergence curves for two representative iMOPSE instances of differing complexity.

Figures 8 and 9 trace the best objective value found by NGSA-DE, DE-SAdapt, DE-ACR, DE, and PSO across generations. The curves let us compare how fast each algorithm converges and how steadily it keeps making progress as the run continues.

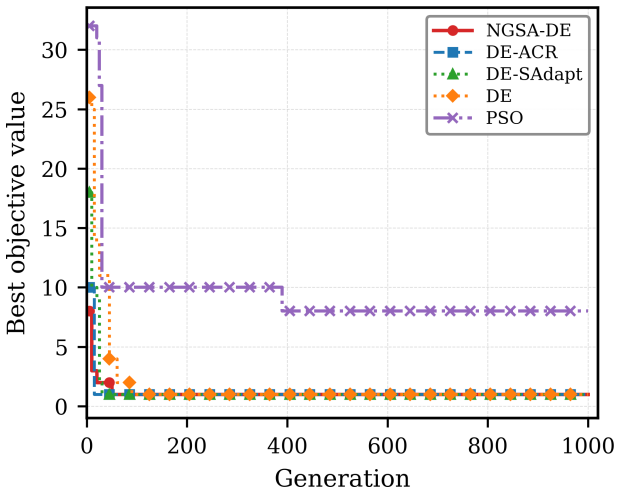
Figure 8 shows that NGSA-DE and DE-SAdapt converge substantially faster than the remaining algorithms, with DE-SAdapt reaching the optimal value from the earliest generations and NGSA-DE achieving it within the first 100 generations. DE-ACR exhibits steady improvement and reaches a competitive solution by generation 350. DE converges more slowly and retains a residual objective value throughout the entire search. PSO stagnates early at a substantially higher objective value and fails to improve further, indicating premature convergence. These results show that neighborhood-guided mutation is central to why the algorithm converges faster and searches more effectively.

Figure 8
Convergence behavior of the compared algorithms on the 100-task/5-resource instance



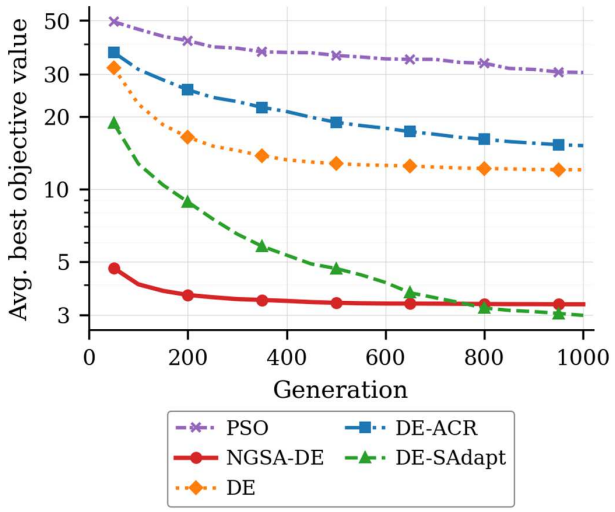
Similar convergence patterns are observed in Figure 9 for the larger, more constrained instance. NGSA-DE, DE-SAdapt, and DE-ACR all converge rapidly within the first 30 generations and reach comparable near-optimal values, demonstrating robust performance across all three algorithms. DE converges more slowly due to the increased problem scale but eventually stabilizes at a competitive value. PSO again exhibits premature stagnation at a substantially higher objective value throughout the search. These results confirm that the proposed search mechanisms maintain their effectiveness as problem complexity increases.

Figure 9
Convergence behavior of the compared algorithms on the 200-task/40-resource instance



To provide a more comprehensive evaluation of convergence behavior across the entire benchmark suite, Figure 10 presents the average convergence curves of the five algorithms, obtained by averaging the best objective value at each generation over all 30 benchmark instances. The aggregated results confirm the trends observed in Figures 8 and 9: In the early generations, both NGSA-DE and DE-SAdapt pull ahead of the other algorithms, converging much faster. Only NGSA-DE holds that pace, though, settling within the first 100–150 generations. DE-SAdapt keeps improving, but far more slowly and over a much longer stretch, before finally arriving at a slightly lower average objective value. DE and DE-ACR are slower off the mark and need many more generations to stabilize, while PSO converges prematurely and

Figure 10
Average convergence curve (best objective value vs generation) across all 30 benchmark instances for the five compared algorithms (log scale)



stays stuck at much higher objective values. Table 16 puts numbers to these observations across all 30 instances. Unlike Table 14, which reports the best-so-far (elite) solution, Table 16 reflects the entire population at the terminal generation (generation 1000): "Final Mean ± Std" is the population's mean objective value at that point, "AUC" (Area Under the Convergence Curve) jointly captures convergence speed and solution quality across the full search trajectory, and "Avg. gen to 95% of final value" is the average generation at which this mean first reaches within 5% of its terminal value. Since the terminal population still contains some inferior individuals, final mean values are higher than the overall mean in Table 14 for most algorithms (e.g., NGSA-DE: 3.33 vs 2.20), reflecting population diversity rather than a data inconsistency. While DE-SAdapt attains the lowest final population mean, NGSA-DE converges over six times faster (123 vs 764 generations) and has the lowest CV (0.42), indicating the highest robustness across instances.

Taken together, Figures 8–10 and Table 16 point to the neighborhood-guided mutation strategy as the main reason for the faster convergence: both NGSA-DE and DE-SAdapt make quick progress in the early generations. When this mutation strategy is paired with the adaptive crossover mechanism, NGSA-DE strikes the best balance overall, converging quickly, reaching good solutions, and holding up well across instances of different sizes and complexity.

5.2.4. Statistical analysis

To check whether the observed performance differences are statistically significant, we ran nonparametric tests across the 30 benchmark instances. The Friedman test assessed the overall differences among the competing algorithms, and we then applied the Holm post hoc procedure, with NGSA-DE as the control, to pinpoint which pairwise differences were significant. We also used the Wilcoxon signed-rank test for a closer pairwise comparison and to measure effect sizes. Tables 17–19 report the results.

Table 17
Friedman test results

Statistic	Value
Number of algorithms (k)	5
Number of datasets (N)	30
Friedman χ^2	65.258
p-value	2.27×10^{-13}
Iman-Davenport F	34.57
p-value	< 0.0001

The Friedman test revealed statistically significant differences among the five compared algorithms ($\chi^2 = 65.258, p < 0.0001$; Iman–Davenport $F = 34.57$, $p < 0.0001$). The average-rank analysis ranked NGSA-DE and DE-SAdapt exhibited statistically equivalent performance according to both the Friedman ranking and Wilcoxon signed-rank tests (*average rank = 1.80*), followed by DE (3.30), DE-ACR (3.70), and PSO (4.40).

To further identify pairwise differences, the Holm post hoc procedure was applied using NGSA-DE as the control algorithm. The results indicate that NGSA-DE significantly outperforms DE, DE-ACR, and PSO after Holm correction, whereas no statistically significant difference is detected between NGSA-DE and DE-SAdapt. Specifically, the adjusted p-values for all three comparisons fall below their respective Holm thresholds: NGSA-DE vs PSO ($p = 1.91 \times 10^{-10} < \alpha = 0.0125$), NGSA-DE vs DE-ACR ($p = 3.26 \times 10^{-6} < \alpha = 0.0167$), and NGSA-DE vs DE ($p = 2.39 \times 10^{-4} < \alpha = 0.025$), confirming statistically significant superiority in all three cases. No statistically significant difference is detected between NGSA-DE and DE-SAdapt ($p = 1.000$, $z = 0$).

The Wilcoxon signed-rank test provides complementary evidence. Against DE, DE-ACR, and PSO, NGSA-DE wins between 86.67% and 93.33% of the time, and every one of these comparisons is highly significant ($p < 0.0001$) with a large effect size ($r > 0.74$). In contrast, the comparison with DE-SAdapt does not reveal a statistically significant difference ($p = 0.3792$, $r = 0.162$), suggesting comparable average performance between the two algorithms.

This pattern indicates that neighborhood-guided mutation is what drives most of the improvement. NGSA-DE and DE-SAdapt reach comparable average solution quality, but NGSA-DE is the more robust of the two: it brings the

Table 16
Summary of convergence behavior across the 30 benchmark instances

Algorithm	Final mean ± Std	CV	AUC (mean ± Std)	Avg. gen to 95% of final value
NGSA-DE	3.33 ± 1.40	0.42	3.49 ± 1.43	123
DE-SAdapt	2.99 ± 3.14	1.05	5.75 ± 4.97	764
DE-ACR	15.14 ± 15.35	1.01	20.51 ± 17.45	602
DE	12.03 ± 11.92	0.99	14.32 ± 12.17	457
PSO	30.46 ± 22.62	0.74	36.50 ± 22.52	608

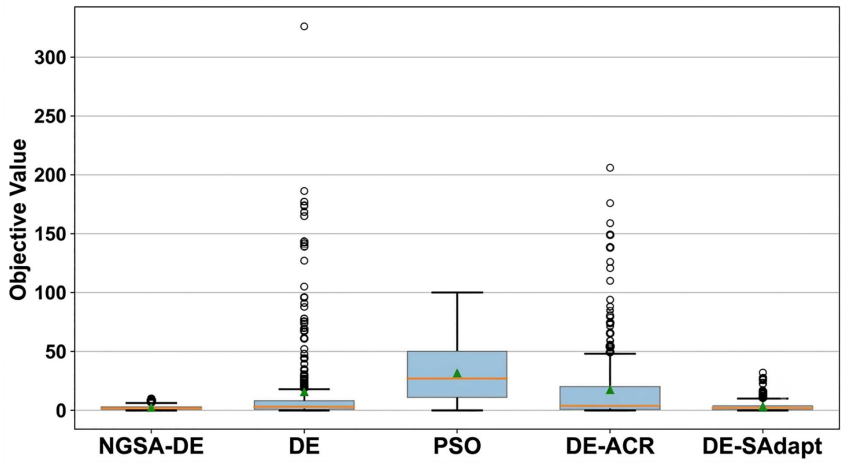
Table 18
Holm post hoc test (Control = NGSA-DE)

Comparison	Avg. rank difference	z-value	p-value	Holm α
NGSA-DE vs PSO	2.6	6.369	1.91×10^{-10}	0.0125
NGSA-DE vs DE-ACR	1.9	4.654	3.26×10^{-6}	0.0167
NGSA-DE vs DE	1.5	3.674	2.39×10^{-4}	0.025
NGSA-DE vs DE-SAdapt	0	0	1	0.05

Table 19
Wilcoxon signed-rank test

Comparison	Win/tie/loss	Win rate (%)	W	p-value	Effect r
NGSA-DE vs DE	26/0/4	86.67	35	< 0.0001	0.742
NGSA-DE vs PSO	28/0/2	93.33	3	< 0.0001	0.862
NGSA-DE vs DE-ACR	28/0/2	93.33	7	< 0.0001	0.847
NGSA-DE vs DE-SAdapt	13/2/15	43.33	189.5	0.3792	0.162

Figure 11
Boxplot comparison of the algorithms



worst-case objective value down from 32 to 10 and the coefficient of variation from 1.53 to 0.98. This indicates that the adaptive crossover mechanism mainly contributes to search stability and reliability, complementing the exploitation capability of the neighborhood-guided mutation strategy.

5.2.5. Robustness analysis

The robustness characteristics of the compared algorithms are illustrated in Figure 11 and summarized in Table 20.

Table 20
Robustness metrics

Algorithm	Mean	Median	IQR	CV
NGSA-DE	2.20	2.00	2.00	0.98
DE-SAdapt	3.09	2.00	4.00	1.53
DE	15.47	3.00	7.00	2.45
DE-ACR	17.42	4.00	19.00	1.77
PSO	31.64	27.00	39.25	0.75

NGSA-DE achieves the lowest overall mean objective value (2.20) and the smallest interquartile range (IQR = 2.00), suggesting high consistency across different problem instances and independent runs. Its median value is also among the lowest (2.00), confirming that this performance is not driven by a few exceptionally good runs but is maintained consistently throughout the benchmark set. However, a Wilcoxon signed-rank test comparing NGSA-DE and DE-SAdapt across the 30 instances yields a win/tie/loss record of 13/2/15, with $p = 0.379$ and a small effect size ($r = 0.162$), indicating that the two algorithms are not statistically distinguishable in terms of average solution quality.

NGSA-DE does show a clear advantage in robustness: its IQR (2.00) and coefficient of variation (CV = 0.98) are both substantially lower than those of DE-SAdapt (IQR = 4.00, CV = 1.53), indicating a narrower and more predictable spread of outcomes and fewer poor-quality (worst-case) results. This suggests that the neighborhood-guided mutation mechanism shared by both algorithms is the primary driver of search stability, while the additional gain from NGSA-DE's adaptive crossover lies not in improving average solution quality but in further tightening the distribution of outcomes and improving convergence reliability.

DE and DE-ACR, by contrast, have much larger IQR values (7.00 and 19.00) and much higher mean objective values, so their performance varies a great deal. PSO is a more subtle case: it posts the lowest CV (0.75), but its mean and median objective values are far worse than those of the DE-based methods. The low CV here is misleading; it reflects PSO settling consistently on

poor solutions rather than any real robustness, as its high median (27.00) and mean (31.64) make clear.

Overall, these results suggest that NGSA-DE strikes the best balance between solution quality and stability. The integration of the neighborhood-guided mutation and adaptive crossover mechanisms not only improves average performance but also reduces performance dispersion, leading to more reliable behavior across diverse DETMS-RCPSP instances.

6. Conclusion and Future Work

This study makes three primary contributions. First, it formulates a new problem, DETMS-RCPSP, aimed at optimizing the deviation of project completion time—whether early or late—relative to a predetermined deadline. A key characteristic of this problem is that task execution is susceptible to interruptions caused by unforeseen events during the process. The problem has been proven to be NP-hard. Second, it proposes the NGSA-DE algorithm, which integrates two adaptive mechanisms into the standard DE framework: a neighborhood-guided mutation strategy (leveraging information from high-quality neighboring solutions) and a self-adaptive crossover mechanism (adjusting the CR based on recent search outcomes). These mechanisms enhance the balance between exploration and exploitation during the optimization process. Third, large-scale experiments conducted on 30 standard iMOPSE datasets demonstrate the effectiveness and stability of NGSA-DE. The proposed method achieved the best overall performance, with an average objective value of 2.20; it outperformed DE, DE-ACR, and PSO in the majority of test cases and demonstrated performance competitive with DE-SAdapt. Statistical tests confirmed significant improvements over DE, DE-ACR, and PSO, evidenced by high win rates and large effect sizes. Further analysis reveals that neighborhood-guided mutation is the primary driver of performance improvement, while the adaptive crossover mechanism enhances the stability and robustness of the search process. NGSA-DE also exhibits faster convergence, mitigates premature convergence, and maintains more consistent performance across repeated experimental runs.

The study has certain limitations: the evaluation is restricted to the standard iMOPSE dataset and considers only task-duration disruptions, while other uncertainties—such as resource failures or unplanned activities—remain unmodeled. Furthermore, parameter sensitivity has not been systematically analyzed, and the current framework employs a reactive rescheduling strategy rather than a predictive-proactive one. Consequently, future research will focus on integrating multiple sources of uncertainty, developing a hybrid proactive-reactive scheduling approach that accounts for disruption forecasting, investigating adaptive and hybrid search mechanisms, and incorporating deep reinforcement learning to guide the dynamic search process. Extending the method to multi-project environments and validating it against diverse standard datasets and real-world industrial data will facilitate a more in-depth assessment of the method's generalizability and practical applicability.

Ethical Statement

This study does not contain any studies with human or animal subjects performed by any of the authors.

Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

Data Availability Statement

The data that support the findings of this study are openly available in iMOPSE at https://github.com/imopse/iMOPSE/tree/main/configurations/problems/MSRCPSP/d36.

Author Contribution Statement

Hao Nguyen Thi: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing – original draft, Writing – review & editing. **Loc Nguyen The:** Conceptualization, Methodology, Investigation, Writing – original draft, Writing – review & editing. **Huu Dang Quoc:** Conceptualization, Software, Validation, Formal analysis, Writing – original draft, Writing – review & editing.

References

[1] Wiest-Goyeneche, J. R., Solano-Charris, E. L., & Vega-Mejía, C. A. (2025). Multi-skill resource-constrained project scheduling problem: A case study in an open-pit mining process. *Engineering Optimization*, *57*(7), 1890–1912. https://doi.org/10.1080/0305215X.2024.2376852

[2] Bahroun, Z., As'ad, R., Tanash, M., & Athamneh, R. (2024). The multi-skilled resource-constrained project scheduling problem: A systematic review and an exploration of future landscapes. *Management Systems in Production Engineering*, *32*(1), 108–132. https://doi.org/10.2478/mspe-2024-0012

[3] Bellenguez-Morineau, O., & Néron, E. (2007). A branch-and-bound method for solving multi-skill project scheduling problem. *RAIRO—Operations Research*, *41*(2), 155–170. https://doi.org/10.1051/ro:2007015

[4] Snauwaert, J., & Vanhoucke, M. (2023). A classification and new benchmark instances for the multi-skilled resource-constrained project scheduling problem. *European Journal of Operational Research*, *307*(1), 1–19. https://doi.org/10.1016/j.ejor.2022.05.049

[5] Khoshsirat, M., & Mousavi, S. M. (2024). A new proactive and reactive approach for resource-constrained project scheduling problem under activity and resource disruption: A scenario-based robust optimization approach. *Annals of Operations Research*, *338*(1), 597–643. https://doi.org/10.1007/s10479-024-05895-9

[6] Wang, M., Liu, G., & Lin, X. (2022). Dynamic optimization of the multi-skilled resource-constrained project scheduling problem with uncertainty in resource availability. *Mathematics*, *10*(17), 3070. https://doi.org/10.3390/math10173070

[7] Rahman, M. H. F., Chakrabortty, R. K., & Ryan, M. J. (2021). Managing uncertainty and disruptions in resource constrained project scheduling problems: A real-time reactive approach. *IEEE Access*, *9*, 45562–45586. https://doi.org/10.1109/ACCESS.2021.3063766

[8] Hatami-Moghaddam, L., Khalilzadeh, M., Shahsavari-Pour, N., & Sajadi, S. M. (2024). Developing a robust multi-skill, multi-mode resource-constrained project scheduling model with partial preemption, resource leveling, and time windows.

Mathematics, *12*(19), 3129. https://doi.org/10.3390/math12193129

[9] Vanhoucke, M., Demeulemeester, E., & Herroelen, W. (2001). An exact procedure for the resource-constrained weighted earliness–tardiness project scheduling problem. *Annals of Operations Research*, *102*(1), 179–196. https://doi.org/10.1023/A:1010958200070

[10] Pamay, M. B., Bülbül, K., & Ulusoy, G. (2014). Dynamic resource constrained multi-project scheduling problem with weighted earliness/tardiness costs. In P. S. Pulat, S. C. Sarin, & R. Uzsoy (Eds.), *Essays in production, project planning and scheduling: A festschrift in honor of Salah Elmaghraby* (pp. 219–247). Springer. https://doi.org/10.1007/978-1-4614-9056-2_10

[11] Maghsoudlou, H., Afshar-Nadjafi, B., & Niaki, S. T. A. (2019). Preemptive multi-skilled resource constrained project scheduling problem with hard/soft interval due dates. *RAIRO—Operations Research*, *53*(5), 1877–1898. https://doi.org/10.1051/ro/2018103

[12] Ramos, A. S., Miranda-Gonzalez, P. A., Nucamendi-Guillén, S., & Olivares-Benitez, E. (2023). A formulation for the stochastic multi-mode resource-constrained project scheduling problem solved with a multi-start iterated local search metaheuristic. *Mathematics*, *11*(2), 337. https://doi.org/10.3390/math11020337

[13] Torba, R., Dauzère-Pérès, S., Yugma, C., Gallais, C., & Pouzet, J. (2024). Solving a real-life multi-skill resource-constrained multi-project scheduling problem. *Annals of Operations Research*, *338*(1), 69–114. https://doi.org/10.1007/s10479-023-05784-7

[14] Myszkowski, P. B., Olech, Ł. P., Laszczyk, M., & Skowroński, M. E. (2018). Hybrid differential evolution and greedy algorithm (DEGR) for solving multi-skill resource-constrained project scheduling problem. *Applied Soft Computing*, *62*, 1–14. https://doi.org/10.1016/j.asoc.2017.10.014

[15] Cheng, L., Zhou, J.-X., Hu, X., Mohamed, A. W., & Liu, Y. (2024). Adaptive differential evolution with fitness-based crossover rate for global numerical optimization. *Complex & Intelligent Systems*, *10*(1), 551–576. https://doi.org/10.1007/s40747-023-01159-4

[16] Ye, X., Li, J., Wang, P., & Suganthan, P. N. (2025). A comprehensive survey of adaptive strategies in differential evolutionary algorithms. *Swarm and Evolutionary Computation*, *98*, 102081. https://doi.org/10.1016/j.swevo.2025.102081

[17] Hartmann, S., & Briskorn, D. (2022). An updated survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, *297*(1), 1–14. https://doi.org/10.1016/j.ejor.2021.05.004

[18] Tian, Y., Xiong, T., Liu, Z., Mei, Y., & Wan, L. (2022). Multi-objective multi-skill resource-constrained project scheduling problem with skill switches: Model and evolutionary approaches. *Computers & Industrial Engineering*, *167*, 107897. https://doi.org/10.1016/j.cie.2021.107897

[19] Haroune, M., Dhib, C., Neron, E., Soukhal, A., Mohamed Babou, H., & Nanne, M. F. (2023). Multi-project scheduling problem under shared multi-skill resource constraints. *Transactions in Operations Research*, *31*(1), 194–235. https://doi.org/10.1007/s11750-022-00633-5

[20] Myszkowski, P. B., & Laszczyk, M. (2022). Investigation of benchmark dataset for many-objective multi-skill resource constrained project scheduling problem. *Applied Soft Computing*, *127*, 109253. https://doi.org/10.1016/j.asoc.2022.109253

[21] Vermeire, G., & Vanhoucke, M. (2025). A priority rule heuristic for the multi-skilled resource-constrained project scheduling problem. *Applied Soft Computing*, *171*, 112776. https://doi.org/10.1016/j.asoc.2025.112776

[22] Snauwaert, J., & Vanhoucke, M. (2025). A solution framework for multi-skilled project scheduling problems with hierarchical skills. *Journal of Scheduling*, *28*(3), 289–310. https://doi.org/10.1007/s10951-025-00836-1

[23] Quoc, H. D. (2023). MEMINV: A hybrid efficient approximation method solving the multi skill-resource constrained project scheduling problem. *Mathematical Biosciences and Engineering*, *20*(8), 15407–15430. https://doi.org/10.3934/mbe.2023688

[24] Quoc, H. D. (2021). The R-PSO algorithm solving multi-skill resource-constrained project scheduling problem. *Journal of Military Science and Technology,* (CSCE5), 71–82. https://doi.org/10.54939/1859-1043.j.mst.CSCE5.2021.71-82

[25] Peng, J., Su, Z., & Liu, X. (2025). Multi skill project scheduling optimization based on quality transmission and rework network reconstruction. *Scientific Reports*, *15*(1), 13545. https://doi.org/10.1038/s41598-025-92342-9

[26] Su, Y., Xu, Z., & Liu, D. (2025). A model and algorithm for reactive multi-objective multi-skilled project scheduling under resource disruptions. *Computers & Industrial Engineering*, *203*, 111043. https://doi.org/10.1016/j.cie.2025.111043

[27] Ghasemi, M., Nazari, A., Thiruvady, D., Tavakkoli-Moghaddam, R., Shahabi-Shahmiri, R., & Mirnezami, S.-A. (2026). A multi-skilled resource-constrained project scheduling with reliability constraints. *Computers & Industrial Engineering*, *219*, 112154. https://doi.org/10.1016/j.cie.2026.112154

[28] Antkiewicz, M., Myszkowski, P. B., Gmyrek, K., Krzeminski, A., & Calvo-Rolle, J. L. (2024). Efficiency of specialized genetic operators in non-dominated tournament genetic algorithm (NTGA2) applied to multi-objective multi-skill resource constrained project scheduling problem. In *Advances in Computational Collective Intelligence: 16th International Conference*, 97–110. https://doi.org/10.1007/978-3-031-70259-4_8

[29] Li, H., Zhu, H., Zheng, L., & Xie, F. (2024). Software project scheduling under activity duration uncertainty. *Annals of Operations Research*, *338*(1), 477–512. https://doi.org/10.1007/s10479-023-05343-0

[30] Sallam, K. M., Chakrabortty, R. K., & Ryan, M. J. (2021). A reinforcement learning based multi-method approach for stochastic resource constrained project scheduling problems. *Expert Systems with Applications*, *169*, 114479. https://doi.org/10.1016/j.eswa.2020.114479

[31] Cai, H., Bian, Y., & Liu, L. (2024). Deep reinforcement learning for solving resource constrained project scheduling problems with resource disruptions. *Robotics and Computer-Integrated Manufacturing*, *85*, 102628. https://doi.org/10.1016/j.rcim.2023.102628

[32] Zhang, Y., Chang, R., Omrany, H., Zuo, J., Burry, J., & Gu, N. (2025). Policy-gradient scheduling optimisation under multi-skill constraints: A comparative study on computational algorithms. *Journal of Building Design and Environment*, *3*(3), *202571*. https://doi.org/10.70401/jbde.2025.0017

[33] Chen, Y., Demeulemeester, E., & Matuschke, J. (2026). A purely buffer-based approach to the proactive and reactive

resource-constrained project scheduling problem. *Annals of Operations Research*, *362*(1), 555. https://doi.org/10.1007/s10479-025-06688-4

[34] Rahimifard, A., Nakhai-Kamalabadi, I., Khalili-Damghani, K., & Raissi, S. (2025). Simulation-based framework for stochastic multi-mode resource-constrained project scheduling. *MethodsX*, *15*, 103496. https://doi.org/10.1016/j.mex.2025.103496

[35] Satic, U., Jacko, P., & Kirkbride, C. (2024). A simulation-based approximate dynamic programming approach to dynamic and stochastic resource-constrained multi-project scheduling problem. *European Journal of Operational Research*, *315*(2), 454–469. https://doi.org/10.1016/j.ejor.2023.10.046

[36] Zhang, Y., Chen, G., Cheng, L., Wang, Q., & Li, Q. (2023). Methods to balance the exploration and exploitation in differential evolution from different scales: A survey. *Neurocomputing*, *561*, 126899. https://doi.org/10.1016/j.neucom.2023.126899

[37] Zhang, J., & Sanderson, A. C. (2009). JADE: Adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation*, *13*(5), 945–958. https://doi.org/10.1109/TEVC.2009.2014613

[38] Piotrowski, A. P., Piotrowska, A. E., & Napiorkowski, J. J. (2026). Experimental survey of L-SHADE and SHADE-based adaptive differential evolution algorithms. *Swarm and Evolutionary Computation*, *101*, 102286. https://doi.org/10.1016/j.swevo.2026.102286

[39] Rishakan, K. F., & Gharehchopogh, F. S. (2026). Resultant vector strategy: A new strategy for improved performance of metaheuristic algorithm. *Knowledge-Based Systems*, *342*, 115833. https://doi.org/10.1016/j.knosys.2026.115833

[40] Gharehchopogh, F. S., & Rishakan, K. F. (2026). Several improved models of the mountain gazelle optimizer for solving optimization problems. *Computer Modeling in Engineering & Sciences*, *146*(1), 24. https://doi.org/10.32604/cmes.2025.073808

[41] Anka, F., Soleimanian Gharehchopogh, F., Mousavirad, S. J., & Tejani, G. G. (2026). Discrete puma optimizer to solve combinatorial optimization problems. *International Journal of Computational Intelligence Systems*, *19*(1), 73. https://doi.org/10.1007/s44196-026-01159-5

[42] Hosseinzadeh, M., Tanveer, J., Rahmani, A. M., Abbaszadi, R., Gharehchopogh, F. S., Porntaveetus, T., & Lee, S.-W. (2026). A comprehensive survey inspired by elephant optimization algorithms: Comprehensive analysis, scrutinizing analysis, and future research directions. *Archives of Computational Methods in Engineering*, *33*(1), 139–186. https://doi.org/10.1007/s11831-025-10303-x

[43] Golenko-Ginzburg, D. (1988). On the distribution of activity time in PERT. *Journal of the Operational Research Society*, *39*(8), 767–771. https://doi.org/10.1057/jors.1988.132

[44] Premachandra, I. M. (2001). An approximation of the activity duration distribution in PERT. *Computers & Operations Research*, *28*(5), 443–452. https://doi.org/10.1016/S0305-0548(99)00129-X

[45] Myszkowski, P. B., Laszczyk, M., Nikulin, I., & Skowroński, M. (2019). iMOPSE: A library for bicriteria optimization in multi-skill resource-constrained project scheduling problem. *Soft Computing*, *23*(10), 3397–3410. https://doi.org/10.1007/s00500-017-2997-5

How to Cite: Thi, H. N., The, L. N., & Quoc, H. D. (2026). Dynamic Multi-skill Project Scheduling Under Disruptions with Earliness–Tardiness Objectives. *Journal of Data Science and Intelligent Systems*. https://doi.org/10.47852/bonviewJDSIS620210735