Journal of Industrial and Management Optimization

JIMO, 22(3): 1464–1490.
DOI: 10.3934/jimo.2026054
Received: 03 November 2025
Revised: 28 January 2026
Accepted: 10 February 2026
Published: 27 February 2026

https://www.aimspress.com/journal/jimo

Case report

Multi-objective project portfolio scheduling with multi-skilled and inter-project dependency based on NSGA-II: Case study

Heng Zhang*, Yanfei Ma and Yixuan Qin

School of Mechanical Engineering, Guizhou University, Guiyang 550025, China

* **Correspondence:** Email: brother_heng@foxmail.com; Tel: +86-181-1473-7730.

Abstract: This study addresses the complex resource-constrained scheduling problem for software project portfolios, where baseline and customized projects with inter-dependencies are developed in parallel. We formulated a nonlinear integer programming model that simultaneously minimizes total duration, optimizes software quality, and balances engineer workload, explicitly incorporating multi-skilled human resources and cross-project dependencies. To solve this problem, we developed an improved Non-dominated Sorting Genetic Algorithm II (NSGA-II) algorithm featuring localized coding, heuristic population initialization, and Pareto-based local search. A real-world case study from an AI-powered voice technology enterprise demonstrates the method's efficacy: the final population significantly outperforms the initial one, and our algorithm surpasses classic NSGA-II, Ant Lion Optimization (ALO), and Simulated Annealing (SA) in convergence and diversity. Crucially, our approach achieves remarkable improvements—reducing duration by 16%, enhancing quality by 21.6%, and improving workload balance by 94.4% compared to skill-homogenized scenarios. Similarly, inter-project dependency-aware scheduling improves duration, quality, and workload balance by 5.2%, 3.9%, and 53.9%, respectively, compared to inter-project dependency-unaware scenarios. Managers can utilize the decoded Pareto optimal solutions to formulate detailed allocation plans, thereby achieving contextually optimized resource management.

Keywords: project portfolio scheduling; multi-skilled; inter-project dependency; NSGA-II; multi-objective optimization
Mathematics Subject Classification: 90B50, 68U35

1. Introduction

The burgeoning AI-powered voice technology sector is characterized by rapid innovation cycles and intense market competition. Companies in this area, such as our case study enterprise Company S, must concurrently manage the iterative development of a core product baseline and the delivery of multiple customized projects for diverse clients. This dual-track research and development (R&D) strategy, while essential for market responsiveness, precipitates a complex project portfolio scheduling problem. Traditional scheduling approaches, often manual and myopic, fail to systematically account for heterogeneous multi-skilled engineers, critical inter-project dependencies (e.g., the reuse of baseline components in custom projects), and the need to balance competing objectives of time-to-market, software quality, and employee workload equity. Project portfolio scheduling, by contrast, enables coordinated governance of interdependent projects through the synergistic integration of collaborative components, thereby optimizing resource allocation efficiency and organizational project management competencies to align with the strategic objectives of technology-driven enterprises [1]. This paper presents an in-depth case study addressing this pressing industrial challenge.

Adam Smith's seminal proposition in An Inquiry into the Nature and Causes of the Wealth of Nations unequivocally stated: "The greatest improvement in the productive powers of labor, and the greater part of the skill, dexterity, and judgment with which it is anywhere directed, or applied, seem to have been the effects of the division of labor" [2]. His paradigmatic pin-manufacturing illustration further substantiates how specialization-driven task partitioning enhances labor productivity. The contemporary paradigm of a multi-skilled technical workforce, capable of adapting to diverse roles, constitutes a sophisticated evolution of Smith's classical doctrine, reconciling specialization benefits with operational flexibility. A software project portfolio typically shares constrained resources encompassing human, material, and financial capital. Among these, multi-skilled human resources (HR) represent a distinctive category of flexible assets, characterized by employees' composite competencies, heterogeneous skill combination, and cross-functional task adaptability [3]. Unlike conventional industrial products reliant on mechanical repetition, as a highly customized knowledge-intensive product, software development fundamentally depends on engineers' cognitive labor and flexible skill deployment. Rational human resource allocation proves instrumental in ensuring project portfolio delivery compliance, driving corporate cost-performance optimization, and overcoming developmental bottlenecks. Conversely, arbitrary or unscientific scheduling practices not only undermine operational goal attainment and workforce creativity utilization but also detrimentally impact employee satisfaction and engagement. Consequently, optimizing the scheduling of scarce multi-skilled R&D HR constitutes a critical lever for enhancing project portfolio efficiency and quality.

Behrouz's comprehensive scientometric assessment [4] of multi-skilled scheduling literature (2000–2020) revealed a compound annual growth pattern commencing circa 2010, peaking at 17 peer-reviewed publications in 2019. This scholarly momentum aligned with Michael's empirical identification [5] of "competency-skill governance" and "constraint resource profiling" (particularly specialty-skilled personnel) as paramount success determinants in multi-project management under volatile, uncertain, complex, and ambiguous (VUCA) contexts. Pioneering studies further substantiate these insights: Hegazy's groundbreaking work [6] in construction management first formalized the multi-skilled resource-constrained scheduling problem, demonstrating through comparative analysis that multi-skilled strategies reduce project duration by 18%–22% while lowering costs by 12%–15% versus single-skill approaches. Arashpour's longitudinal study [7] of cross-training interventions

revealed that strategically developed multi-skilled workforce decreased project cycle times by 25%, increased output quality indices by 30%, and improved resource utilization rates by 40% in manufacturing environments. Addressing the computational complexity inherent in flexible workshop scheduling, Fekri [3] developed a bi-objective mixed-integer linear programming (MILP) model minimizing makespan and human resource idle time. Experimental comparisons between GA and SA yielded critical insights: GA solutions achieved 15%–20% superior objective function values, whereas SA exhibited 30% faster convergence. This dichotomy underscores the necessity for context-driven meta-heuristic selection frameworks.

The multi-skilled human resource scheduling problem in project portfolio management falls within the category of the resource-constrained multi-project scheduling problem (RCMPSP), which is a typical NP-hard problem with high solution difficulty. Current RCMPSP solution methods are primarily divided into exact algorithms, heuristic algorithms, and meta-heuristic algorithms (MAs). For medium and large-scale project scheduling problems, the former two methods perform poorly in terms of both solution effectiveness and efficiency. Existing RCMPSP research and models mainly focus on MAs for solution approaches. Kanchan Rajwar [8], in a review and survey of meta-heuristic algorithms and their variants based on literature statistics from Google Scholar up to the end of December 2022, found that the two most popular algorithms are particle swarm optimization (PSO) and GA. Hosseinian [9] investigated the multi-skilled resource-constrained project scheduling problem and designed an improved genetic algorithm to effectively explore the solution space, achieving optimal solutions in 70% of the test problems. James [10] developed a resource-constrained project scheduling model for large-scale manufacturing enterprises. He solved it using a genetic algorithm based on the heuristic labor allocation method, ultimately optimizing the total cost by 10.34%. In recent years, with the advancement of computational capabilities and optimization algorithms, some scholars have applied more novel meta-heuristic algorithms in their research on multi-skilled scheduling problems. Li [11] designed a multi-objective discrete Jaya algorithm (MODJaya) for the multi-skilled resource-constrained project scheduling problem in manufacturing systems, aiming to minimize both the total project duration and total cost, and validated its superiority on benchmark datasets in a smart multi-objective project scheduling environment. Ghamginzadeh [12] studied the multi-skilled project scheduling problem under fuzzy conditions, captured experts' cognitive information on time uncertainty using fuzzy sets, constructed a bi-objective model to minimize the total project duration and total labor allocation cost, and solved it using a multi-objective imperialist competitive algorithm (MOICA) with parameters optimized based on the Taguchi method.

The RCMPSP has been extensively studied [9,10]. Research has been conducted on multi-skilled resources [11,12] and even inter-project dependencies [13]. The multi-skilled nature of engineers means each activity has multiple potential performers, causing the combinatorial solution space for "activity-engineer" assignments to explode exponentially. Inter-project technical dependencies, such as the reuse of baseline components, break project independence, making any local scheduling decision potentially impact the entire system. This necessitates collaborative optimization, further elevating the problem's complexity to NP-hard.

However, these studies are predominantly situated in manufacturing or construction contexts or rely on standardized benchmark datasets. A significant gap remains in the literature concerning the integrated modeling and empirical validation of these complexities within the software customization industry. Specifically, the distinct nature of dependencies in "baseline-customization" scenarios and the human-centered objective of workload balance in knowledge-intensive R&D environments remain

underexplored. Additionally, existing multi-skilled scheduling models mainly focus on single-objective and bi-objective planning, with few studies considering three planning objectives. For example, Behrouz [4] found that 68.8% of multi-skilled scheduling research only considers a single objective. In modern management practice, some managers focus on timely completion, while others prioritize cost control or other goals. These different objectives often influence and even conflict with each other. For instance, when human resources are limited, if the expectation is to shorten the completion time by compressing the R&D cycle, the quality is often difficult to ensure. At the same time, minimizing makespan and maximizing software quality may cause workload inequity among engineers, leading to engineers with high proficiency levels to take on more workload than those with lower proficiency levels [14]. As human-centered HR systems, especially for technology-driven startups, it is necessary to mitigate the workload inequity and enhance engineers' satisfaction. Consequently, managers should balance both single and cross-objectives, optimizing the overall goals in a coordinated manner.

This study aims to bridge this gap, contributing to three primary aspects. First, we formulate a novel integrated model that simultaneously captures the key intricacies of software project portfolio scheduling, including skill heterogeneity, cross-project precedence, and the triple objectives of minimizing duration, maximizing quality, and balancing workload metrics. Second, we propose a significantly enhanced NSGA-II variant featuring customized encoding, heuristic initialization, and local Pareto fine-tuning to efficiently solve this problem and present a comprehensive, real-world case study from an AI-powered voice technology company that validates the entire methodology. Finally, we decode the Pareto-optimal solutions into actionable Gantt charts and provide a quantitative analysis of the performance gains achieved through skill-aware and dependency-aware scheduling, offering valuable managerial insights for project managers.

The remainder of this paper is organized as follows: Chapter 2 describes the problem statement and definition, then introduces the establishment of the mathematical model. Chapter 3 details the adaptation of the NSGA-II and baseline algorithms. Chapter 4 presents the case data and experimental results. Finally, Chapter 5 summarizes the study and discusses potential directions for future research.

2. Model formulation

2.1. Problem description

Consider a scenario where a research cycle requires delivering N projects, including one baseline iterative project and N-1 customized projects. Through work breakdown structure (WBS) analysis, each project PR_i contains J_i activities. The workforce comprises E engineers collectively possessing K distinct skills to fulfill project requirements. Using the Program Evaluation and Review Technique (PERT), the standard duration for activity j in project PR_i is defined as t_{ij}, while the actual effort depends on both t_{ij} and the proficiency w_{ije} of the assigned engineer e. The scheduling process must satisfy constraints related to skill matching, activity precedence, and effort allocation. Essentially, the multi-skilled human resource scheduling problem in a project portfolio context involves searching for optimal project–activity–skill–engineer combinations guided by predefined objectives, as illustrated in Figure 1. As the dimensionality of objectives or the number of combinatorial factors increases, exhaustive search methods become computationally infeasible.

Given the multi-dimensional nature of project portfolio scheduling, the following assumptions are formulated to ensure model tractability and algorithmic consistency:

1. A unified scheduling mechanism governs all project resources, with transparent information sharing enabling cross-project collaboration. Although projects could theoretically be started at staggered intervals, this study focuses on resource allocation and dependency scheduling and, therefore, assumes that all projects commence within the same scheduling cycle.

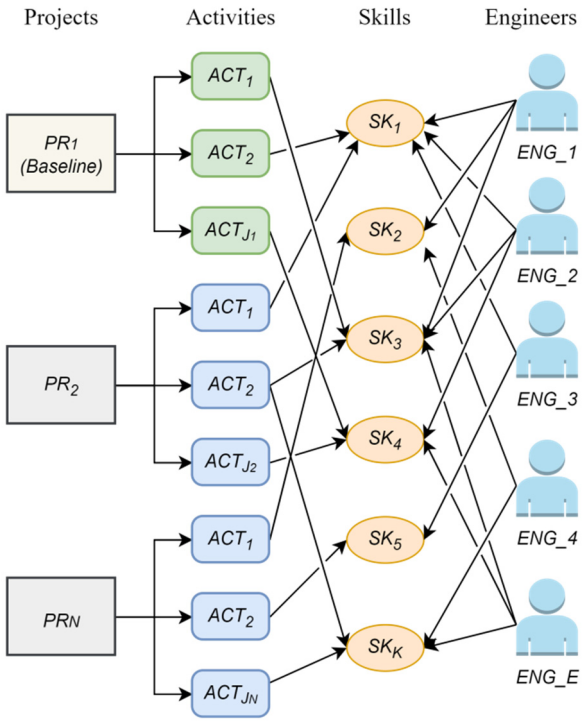

Figure 1. Prototype of the multi-skilled project portfolio scheduling problem.

2. No dynamic task insertion or preemption during scheduling. Engineers are assigned exclusively to single projects, with no inter-project transfers or mid-cycle personnel changes.
3. Each activity requires one skill, differing only in execution efficiency without feasibility constraints. Engineers apply one skill at a time, with no proficiency fluctuations during scheduling.
4. Based on component-based development practices, reusing validated modules can significantly reduce integration and testing time. This study simplifies it to zero duration to highlight dependency benefits.

2.2. Symbol and parameter definitions

The symbols and decision variables involved in the model are defined as shown in Table 1.

Table 1. Notations and descriptions.

Type	Notation	Description
Parameters	i	The index of project, $i=1,2,...,N$. N is the number of sub-projects in the project portfolio.
	j	The index of activities, $j=1,2,...,J_i$. J_i is the number of activities of sub-project i.
	k	The index of skills, $k=1,2,...,K$. K is the number of skill types.
	e	The index of engineers, $e=1,2,...,E$. E is the number of engineers.
	(i,j)	Symbols indicate activity j in sub-project i.
	(i',h)	(i',h) denotes the predecessor activity of (i,j), where $i'=i$ denotes predecessor activities from the same project, and $i'\neq i$ denotes inter-project dependencies.
	t_{ij}	Standard working hours for activity (i,j) in days. When t_{ij} is 0, it indicates that activity (i,j) reuses the output of its preceding activities without requiring additional development.
	w_{ije}	Skill proficiency of engineer e in activity (i,j), where 1 denotes the baseline proficiency level. Values below 1 indicate above-average competency, values above 1 signify below-average competency, and 0 indicates the absence of competency.
	θ	θ denotes an infinitesimal positive constant (e.g., 10^{-6}), enforcing the constraint that an engineer's skill proficiency must exceed zero when assigned to an activity.
	r_{ijk}	Quantity of skill resource k required for activity (i,j).
	R_k	Total amount of skill resource pool for skill k.
	W_{Qi}	Weight of software quality for project i, where the sum of all project weights must equal 1.
Set	A_t	Set of activities in operation within the time window t.
	P_{ij}^{intra}	Set of intra-project predecessor activities of (i,j).
	P_{ij}^{cross}	Set of inter-project predecessor activities of (i,j).
	P_{ij}	Set of predecessor activities for activity (i,j), such that there exists $(i',h)\in P_{ij}$ and $P_{ij} \leftarrow P_{ij}^{intra} \cup P_{ij}^{cross}$.
Decision variables	Y_{ie}	0–1 decision variable, equals 1 if engineer e is assigned to project i, and equals 0 otherwise.
	Z_{ije}	0–1 decision variable, equals 1 if engineer e is assigned to activity (i,j), and equals 0 otherwise.
Derived Variables	s_{ij}	Start time of activity (i,j).
	c_{ij}	Completion time of activity (i,j).
	t_{ij}^*	Actual working hours of activity (i,j).
	TP_i	Total working hours of project i.
	Q_i	Software quality indicator for project i.
	L_e	Total workload of engineer e.
	u	Average workload of the engineer team.

2.3. Objectives and constraints

The definitions of derived variables are given in Equations (1)–(7).

$$t_{ij}^* = t_{ij} \sum_{e=1}^{E} Y_{ie} Z_{ije} w_{ije}, \forall i,j \tag{1}$$

$$c_{ij} = s_{ij} + t_{ij}^{*} \tag{2}$$

$$s_{ij} \geq \max_{(i',h)\in P_{ij}} c_{i'h}, \forall i,j \tag{3}$$

$$TP_i = \sum_{j=1}^{J_i} t_{ij}^{*}, \forall i \tag{4}$$

$$Q_i = \sum_{j=1}^{J_i} \left(\sum_{e=1}^{E} Z_{ije} w_{ije} \frac{t_{ij}^{*}}{TP_i} \right), \forall i \tag{5}$$

$$L_e = \sum_{i=1}^{N} \sum_{j=1}^{J_i} Z_{ije} t_{ij} w_{ije}, \forall e \tag{6}$$

$$u = \frac{1}{E} \sum_{e=1}^{E} L_e \tag{7}$$

Equation (1) defines the actual working hours of an activity t_{ij}^{*} as the product of the standard working hours and the proficiency level. Equation (2) defines the completion time of activity *(i,j)* as the sum of its start time and actual working hours. Equation (3) indicates that the start of an activity must follow the completion of all its preceding activities, including the predecessor activities in the same project and those belonging to other projects. Equation (4) determines the sub-project's working hours as the sum of the actual working hours of each activity. Equation (5) defines the software quality metric of project i as the average skill proficiency of the project team: a smaller average skill proficiency indicates better software quality. Assuming that the quality level of an activity is regarded as the skill proficiency of the assigned engineer, then the quality of project i is the weighted average of the quality levels of each activity, with the weight being the proportion of the actual work hours of the activity (t_{ij}^{*} / TP_i). Equation (6) defines the total workload of engineer e, which is calculated by accumulating the actual working hours of all activities in which he participates. Equation (7) defines the average workload of the engineering team.

The definitions of the multi-objective functions are given in Equations (8)–(10).

$$\min \ T = \max_{i} \{c_{iJ_i}\} \tag{8}$$

$$\min \ Q = \sum_{i=1}^{N} Q_i W_{Q_i} \tag{9}$$

$$min \ \ V = \frac{1}{E}\sum_{e=1}^{E}\left(L_e - u\right)^2 \tag{10}$$

Equation (8) represents Objective Function 1, which aims to minimize the total project portfolio duration T. Here, T is defined as the maximum completion time of each sub-project's last activity (i,J_i). Equation (9) represents Objective Function 2, which aims to achieve the best software quality of the project portfolio, obtained through the weighted sum of the quality of each sub-project [15]. Equation (10) represents Objective Function 3, which aims to minimize the variance V of the engineers' workload or achieve the most balanced workload, obtained through the well-known variance calculation operator, such as the *var()* function in Matlab and Python.

The definitions of constraints are given in Equations (11)–(15).

$$\sum_{e=1}^{E} Z_{ije} = 1, \forall i, j \tag{11}$$

$$\sum_{(i,j)\in A_t}\sum_{e=1}^{E} Z_{ije} r_{ijk} \le R_k, \forall k, t \tag{12}$$

$$\sum_{i=1}^{N} Y_{ie} = 1, \forall e \tag{13}$$

$$Z_{ije} \le Y_{ie}, \forall i, j, e \tag{14}$$

$$w_{ije} \ge \theta * Z_{ije}, \forall i, j, e \tag{15}$$

Equation (11) ensures every activity is assigned to a designated assignee. Equation (12) defines that the number of activities dependent on skill k at time t cannot exceed the number of engineers who possess skill k. Equation (13) defines that each engineer can only be assigned to a single project. Equation (14) ensures that activity execution by engineer e occurs only after their assignment to project i. Equation (15) requires that an engineer must possess the skill required by an activity to be assigned to it.

2.4. Illustrative example

To elucidate the model's mechanics, consider a minimal example with two projects (Project 1: baseline; Project 2: customization), four activities, and two R&D personnel (E1, E2). The data is summarized in Table 2. Activity (2,2) has a mandatory technological dependency on activities (1,2) and (2,1), meaning it requires zero effort if their outputs are reused. A candidate solution assigns

engineer E1 to activities (1,1) and (1,2), and engineer E2 to activities (2,1) and (2,2). The model computes the objectives as follows:

Table 2. Minimal example description.

Activity *(i,j)*	Activity description	Predecessor (P_{ij})	Assignee	Skill req.	Proficiency (w_{ije})	Std. days (t_{ij})	Act. days (t_{ij}^*)
(1,1)	Audio recorder and player development	-	E1	J4	1.0	3	3
(1,2)	ASR neural network deployment on ARM Cortex-M55	(1,1)	E1	J10	0.9	8	7.2
(2,1)	BLE-based provisioning	-	E2	J6	1.2	3	3.6
(2,2)	Hybrid edge-cloud ASR and dialog management	(1,2), (2,1)	E2	J7	0.8	5	4.0

1. *Actual duration (Eq. 1):* The actual duration of activity (2,2) is $t_{22}^* = w_{222} \times t_{22} = 0.8 \times 5 = 4$. In a dependency-unaware scenario, engineer E2 should spend an extra 8.8 days to the development-duplicated feature of activity (1,2), as his proficiency in skill J10 is only 1.1.
2. *Precedence (Eq. 3):* The start time of activity (2,2) must satisfy:

   $s_{22} \geq max\{s_{12} + t_{12}^*, s_{21} + t_{21}^*\} = max\{3 + 7.2, 0 + 3.6\} = 10.2$ days.
3. *Sub-project total working hours (Eq. 4):* $TP_1 = t_{11}^* + t_{12}^* = 3 + 7.2 = 10.2$ days, $TP_2 = t_{21}^* + t_{22}^* = 3.6 + 4 = 7.6$ days.
4. *Project portfolio duration (T, Eq.8),* determined by the last finishing activity of all sub-projects: $T = max\{s_{12} + t_{12}^*, s_{22} + t_{22}^*\} = max\{3 + 7.2, 10.2 + 4\} = 14.2$.
5. *Quality (Q, Eq.9)* assumes that weights of software quality for projects are 0.6 and 0.4, respectively, that is $Q = 0.6 \times \left(1.0 \times \frac{3}{10.2} + 0.9 \times \frac{7.2}{10.2}\right) + 0.4 \times \left(1.2 \times \frac{3.6}{7.6} + 0.8 \times \frac{4}{7.6}\right) \approx 0.953$.
6. *Workload (V, Eq.10):* Since engineers E1 and E2 are each solely responsible for project 1 and project 2, respectively, their workloads are equivalent to the private project's total working hours. That is, $V = \frac{1}{2}\left[(10.2 - 8.9)^2 + (7.6 - 8.9)^2\right] = 1.69$, where 8.9 is the mean workload.

This example verifies that the model correctly enforces precedence and skill-matching constraints and computes the three objective functions based on the project–activity–skill–engineer assignment. However, this manually constructed solution is not necessarily optimal. Therefore, identifying high-quality solutions requires efficient meta-heuristics like the well-adapted NSGA-II presented in the next section.

3. NSGA-II-based algorithm design

The mathematical model formulated in Section 2 defines a highly discrete and combinatorial optimization problem. The core decision involves assigning a finite set of engineers to a finite set of

activities, which is intrinsically a combinatorial matching problem without inherent continuity. This discrete nature, characterized by a solution space populated with project–activity–engineer permutations, renders gradient-based optimization methods inapplicable and makes the problem NP-hard. Given this combinatorial landscape, population-based meta-heuristics are a natural fit. Among them, GA [16] and NSGA-II [17,18] variants are particularly renowned for their effectiveness in navigating discrete search spaces through operations like crossover and mutation that directly manipulate symbolic representations.

Given the need for extensive operator adaptation in this project portfolio human resource scheduling problem, the study adopts the well-established and widely validated NSGA-II as the foundational algorithm. Further, conventional genetic operators are not capable of fine-tuning high-quality solutions. For this reason, a local search operator based on Pareto front is introduced to fine-tune high-quality solutions and improve the local searching efficiency, which is complementary to the crossover and mutation operators [19,20].

3.1. NSGA-II main workflow

The main scheduling workflow, illustrated in Figure 2, is composed of project portfolio resource initialization, population initialization, and genetic iteration phases, and terminates when reaching predefined maximum iterations.

3.2. Coding scheme

The core challenge addressed in this study involves searching for optimal project–activity–skill–engineer combinations, guided by scheduling objectives. To this end, the study adopts a segmented natural number–encoding strategy that represents chromosomes from the perspective of projects and activities, as illustrated in Figure 3. Each chromosome segment, separated by delimiters, corresponds to the engineer–activity assignments for a specific project. Consider that a project portfolio requires 8 engineers to complete 3 sub-projects with 5, 3, and 5 activities, as shown in Figure 3, where symbol → means mapping of activity and engineer:

1. *Segment 1 (Project 1)*: Activities 1-2 → Engineer 1, Activity 3 → Engineer 4, Activities 4-5 → Engineer 5.

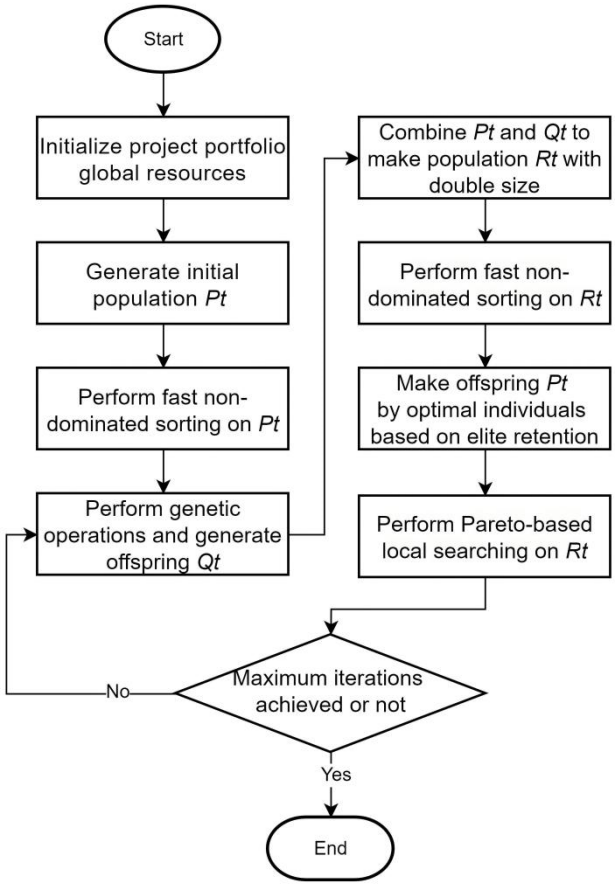

Figure 2. NSGA-II main workflow.

2. *Segment 2 (Project 2)*: Activities 1-2 → Engineer 3, Activity 3 → Engineer 6.
3. *Segment 3 (Project 3)*: Activities 1-3 → Engineer 2, Activities 4-5 → Engineers 7 and 8.

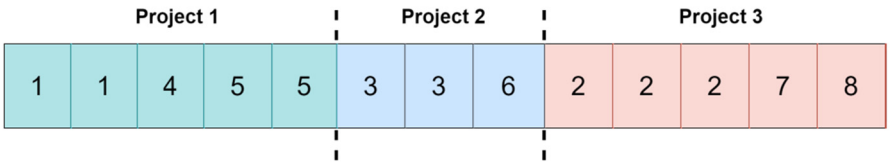

Figure 3. Coding scheme and example.

3.3. Population initialization

This study proposes random initialization to generate individuals' stochastic assignment, which is widely adopted to ensure solution space coverage. The initialization process assigns minimally sufficient engineers to ensure all activities are staffed with qualified personnel first, then invites additional engineers with complementary skills to diversify activity assignments. The last step ensures all activities are allocated and no engineers remain idle. The Pascal style pseudo-code for population initialization is provided below:

#Operator: Population initialization

Input:

Projects = {PR_1, PR_2, ..., PR_n}

Engineers = {E_1, E_2, ..., E_m}

Skills = {SK_1, SK_2, ..., SK_k}

Output:

Initialized population

Resource Initialization:

1 Construct *SK_OWNERS*:

1.1 **for** each skill *SK* **in** *Skills*:

SK_OWNERS[*SK*] ← { *E* ∈ *Engineers* | *E* owns *SK* }

1.2 Initialize activity pools:

UNASSIGNED_ACTS ← { *(i,j)* ∈ Activities of PR_i }

ASSIGNED_ACTS ← ∅

1.3 Build *ACT_ENG_MAP*:

for each activity *(i,j)*:

SK ← Required skill for *(i,j)*

ACT_ENG_MAP[*i,j*] ← *SK_OWNERS*[*SK*]

#Primary Assignment:

2. **while** *UNASSIGNED_ACTS* ≠ ∅:

2.1 *(i,j)* ← RandomSelect(*UNASSIGNED_ACTS*)

2.2 *SK* ← Required skill for *(i,j)*

2.3 *ENG_POOL* ← *SK_OWNERS*[*SK*]

2.4 **while** True:

E ← RandomSelect(*ENG_POOL*)

if SatisfyAssignmentConstraints(*E*, *i*):

2.4.1 *ASSIGNED_ENG*[*i*].add(*E.id*)

2.4.2 Set gene(i,j) ← *E.id*

2.4.3 Update activity pool :

UNASSIGNED_ACTS.remove(*(i,j)*)

ASSIGNED_ACTS.add(*(i,j)*)

break

#PR_i Activity Handling:

3. **while** *UNASSIGNED_ACTS*[*i*] ≠ ∅:

3.1 activity *j* ← RandomSelect(*UNASSIGNED_ACTS*[*i*])

3.2 *SK* ← Required skill for *(i,j)*

3.3 *ENG_POOL* ← *SK_OWNERS*[*SK*]

3.4 Candidates ← *ASSIGNED_ENG*[*i*] ∩ *ENG_POOL*

if Candidates ≠ ∅:

3.4.1 *E* ← RandomSelect(Candidates)

3.4.2 Set gene(i,j) ← *E.id*

else:

Choose new engineer, same as step 2.4

```
        Update pools as Step 2.4.3
#Post-processing and update SK_OWNERS:
4.1 for each E in ASSIGNED_ENG:
      Remove E from all SK_OWNERS[SK]
4.2 # All activities assigned with minimal staffing
4.3 while each SK_OWNERS[SK] ≠ ∅:
4.3.1  E ← RandomSelect(SK_OWNERS)
      #Make qualified activity list(QAL) for E
4.3.2  QAL ← { (i,j) | E masters SK_required(i,j) }
      if QAL = ∅: continue
4.3.3  (i,j) ← RandomSelect(QAL)
      current_owner ← gene(i,j)
4.3.4  if IsSoleActivity(current_owner, i):
        continue
4.3.5  else:
        Set gene(i,j) ← E.id
        Update ASSIGNED_ENG[i]
        Remove E from SK_OWNERS
```

3.4. Genetic operators

1. Roulette wheel selection: Well-known selection methods include Goldberg's tournament selection and Holland's roulette wheel selection. This design adopts the latter. Given that the model has three objectives, it is necessary to normalize the multi-dimension objectives, calculate the cumulative probability for each dimension, and then take the arithmetic mean to compare the objective values. The total number of selections is equal to the population size, and the selection process is repeated to select one individual for the intermediate population each time.
2. Multi-point crossover: Common crossover strategies include single-point crossover and multi-point crossover between two parent individuals, with other gene positions remaining unchanged or adjusted according to constraints. This design employs the latter. During crossover, two activity gene positions are randomly selected, and then the genes between these two points are exchanged.
3. Swapping mutation: Common mutation strategies include fine-tuning of activity start and end times, adjustment of resource types or quantities, and changes in the sequence of operations. This design adopts a resource reallocation and adjustment strategy. During mutation, to avoid generating illegal individuals, two activity gene positions are randomly selected under constraints, and then their genes are swapped.

3.5. Pareto-based local searching

When crossover and mutation operations adjust genes between projects, local searching fine-tunes genes within a single project by precisely perturbing current generation Pareto non-dominated solutions, thereby enhancing convergence and distribution. Additionally, through adaptive attenuation of perturbation strength, it achieves extensive exploration in the early stage and precise exploitation in

the later stage. In implementation, different local search strategies are selected based on the probability interval, including:

1. Critical path optimization: Identifies the project's critical path and replaces less efficient engineers with more skilled candidates to compress the project duration.
2. Engineer exchange: Swaps tasks between two engineers within the same project to avoid low human resource utilization or uneven task distribution.
3. Random perturbation: Makes minor adjustments to the assignment of engineers for random activities to introduce diversity and prevent getting stuck in local optima.

The private configurations of Pareto local searching were set as follows: starting iteration G_{ls} = 25, critical path fine-tuning probability threshold P_{ls_cp} = 0.5, and engineer exchanging fine-tuning probability threshold P_{ls_ee} = 0.8. When the real-time probability is less than P_{ls_cp}, the critical path optimization method is used. When the probability is within the [P_{ls_cp}, P_{ls_ee}] interval, the engineer exchange method is adopted. When the probability is greater than P_{ls_ee}, the random perturbation method is employed. If the individual obtained from local search has better solution quality, the original individual is replaced.

3.6. Baseline algorithms for comparison

To rigorously validate the performance of our adapted NSGA-II, we implemented two additional well-established meta-heuristics as baselines: multi-objective ALO [21] and multi-objective SA [3]. Crucially, both algorithms were carefully adapted to our specific scheduling problem to ensure fair comparison. The key adaptations include:

1. ALO adaptation

a) The Ant Lion positions employ the same multi-segment integer encoding scheme as described in Section 3.2.
b) The random walking operator is redefined to perform constraint-preserving neighborhood searches with the same operations mentioned in Section 3.5, ensuring solution feasibility throughout the optimization process.
c) The selection of guiding Ant Lions and archive maintenance follows a crowding distance-based elitism strategy similar to NSGA-II, with adaptive weight adjustment for balancing the three competing objectives.

2. SA adaptation

a) The solution representation uses identical discrete encoding with project and activity assignment layers.
b) The neighborhood generation operator is redefined to incorporate multiple problem-specific moves, including engineer swapping and activity reassignment, maintaining solution validity through embedded constraint checks.
c) The acceptance criterion employs an adaptive scalarizing function with dynamic weight adjustment, while Pareto archive maintenance follows crowding distance-based pruning similar to NSGA-II for preserving solution diversity.

To ensure a fair comparison, all algorithms, including our adapted NSGA-II, ALO, and SA, are evaluated on the same objective functions and constraints. While experimental comparison has focused on ALO and SA to demonstrate the efficacy of our proposed NSGA-II, it is worthwhile to contextualize our findings within the broader landscape of multi-objective optimizers. Algorithms such as the multi-

objective evolutionary algorithm based on decomposition (MOEA/D) and multi-objective PSO (MOPSO) are indeed prominent in the field. The choice of NSGA-II as our foundation was motivated by its well-known ability to handle non-convex Pareto fronts and its particular suitability for our highly combinatorial and constrained project–activity–skill–engineer matching problem. Although PSO is widely applied, its encoding and update mechanisms for discrete combinatorial optimization problems require more complex designs to avoid infeasible solutions. Preliminary tests indicated that its adaptability and efficiency in this problem were not advantageous, hence it was excluded from the final comparison. Future work could involve a broader benchmarking against a wider range of algorithms once they are similarly tailored to this problem's unique constraints.

4. Case study

As a technological innovation enterprise at the national level in China, Company S has been dedicated to the product implementation of integrated AI-powered voice interaction solutions in the AIoT (AI + IoT) industry, including applications for smart homes and healthcare. Established at the end of 2022, the R product team focuses on the full-chain AI-powered voice interaction solution, based on cloud-edge collaboration architecture, enabling the intelligence of household appliances. Based on the large-scale software customization theory, the company suggests and implements a dual-track R&D management strategy of product baseline and customized projects in order to meet the market's expectation for the quick delivery of personalized AI-powered voice products.

Prior to this study, the R product team relied primarily on manual scheduling by project managers, which was sub-optimal in the following aspects: First, it was time-consuming and unable to rapidly evaluate multiple hypothetical scenarios. Second, it frequently overlooked inter-project dependencies, leading to rework and delays, as seen when customized projects failed to properly account for baseline development progress. Additionally, it caused inequitable workload distribution, with senior engineers frequently becoming bottlenecks and experiencing burnout. Finally, it lacked a quantifiable measure of software quality based on team composition.

The R product team's success hinges on delivering four interconnected projects, including three customized projects and the baseline, within a 1.5-month window. The tight coupling between the baseline and custom projects, the latter having a mandatory technological dependency on the former, resulting in zero effort if reused, makes this a typical example of a modern software portfolio scheduling problem. The highly heterogeneous skill set of the eight engineers, expanding the search space of the scheduling problem, coupled with the diverse priorities of the projects (reflected in the quality weights W_{Qi}), renders intuitive or simplistic scheduling methods ineffective. Solving this problem is therefore not only of academic interest but also a critical business necessity for Company S to maintain its competitive edge. Additionally, the development process for these projects resembles a waterfall model, characterized by predefined activities and fixed precedence relationships, rather than an agile or iterative approach. This makes the problem particularly amenable to the optimization approach proposed herein.

Python 3 was used in development and case validation. The development environment included the Windows 11 operating system, Intel(R) Core(TM) i5-8250U CPU, 24 GB RAM, and PyCharm 2024.2 IDE with an Anaconda3-2023.03 engine.

4.1. Activity info and network diagram

The project portfolio's double-code arrow network diagrams (AOA) info, the activity list of each project, the skill type required by each activity, the precedence and succession constraints between activities, and the standard working hours are shown in Table 3, being obtained through WBS and PERT. Since Project 2 is running on a new chipset with ARM Cortex-M55 architecture, and its Activity F (hybrid edge-cloud ASR and dialogue management) fully reuses the output of Activity D (ASR neural network deployment on ARM Cortex-M55 architecture) in the baseline, its working hours are set to 0, although it is not a dummy activity.

Table 3. Project portfolio activity info.

Project	AOA node	Skill req.	Std. days	Pre-node	Post-node
Project 1	A	10	6	(1, 1)	(1, 2)
(Baseline)	B	8	10	(1, 1)	(1, 3)
	C	2	8	(1, 2)	(1, 5)
	D	1	6	(1, 2)	(1, 4)
	E	7	0	(1, 4)	(1, 6)
	F	7	2	(1, 6)	(1, 7)
Project 2	A	1	2	(2, 1)	(2, 2)
	B	3	1	(2, 1)	(2, 3)
	C	3	1	(2, 1)	(2, 4)
	D	7	2	(2, 2)	(2, 5)
	E	2	5	(2, 5)	(2, 6)
	F	4	0	(1, 4), (2, 6)	(2, 7)
	G	5	5	(2, 7)	(2, 11)
	H	6	3	(2, 3)	(2, 8)
	I	6	4	(2, 8)	(2, 10)
	J	1	3	(2, 2)	(2, 11)
	K	7	4	(2, 4)	(2, 9)
	L	7	3	(2, 9)	(2, 10)
	M	7	4	(2, 10)	(2, 11)
	N	3	4	(2, 11)	(2, 12)
	O	8	4	(2, 11)	(2, 13)
Project 3	A	1	3	(3, 1)	(3, 2)
	B	3	1	(3, 1)	(3, 3)
	C	3	1	(3, 1)	(3, 4)
	D	7	3	(3, 2)	(3, 5)
	E	2	6	(3, 5)	(3, 6)
	F	5	4	(3, 6)	(3, 8)
	G	4	8	(3, 6)	(3, 7)
	H	5	4	(3, 8)	(3, 11)
	I	2	4	(3, 2)	(3, 3)

Continued on next page

Project	AOA node	Skill req.	Std. days	Pre-node	Post-node
Project 3	J	6	6	(3, 3)	(3, 9)
	K	7	3	(3, 9)	(3, 10)
	L	7	4	(3, 4)	(3, 10)
	M	7	4	(3, 10)	(3, 11)
	N	3	4	(3, 11)	(3, 12)
	O	8	3	(3, 12)	(3, 13)
Project 4	A	1	2	(4, 1)	(4, 2)
	B	3	1	(4, 1)	(4, 3)
	C	3	1	(4, 1)	(4, 4)
	D	3	1	(4, 1)	(4, 5)
	E	7	2	(4, 2)	(4, 6)
	F	2	4	(4, 6)	(4, 7)
	G	5	6	(4, 7)	(4, 8)
	H	5	4	(4, 8)	(4, 13)
	I	6	2	(4, 4)	(4, 11)
	J	6	2	(4, 11)	(4, 12)
	K	7	3	(4, 5)	(4, 12)
	L	9	6	(4, 3)	(4, 9)
	M	9	4	(4, 9)	(4, 10)
	N	2	2	(4, 2)	(4, 10)
	O	5	4	(4, 10)	(4, 12)
	P	7	3	(4, 12)	(4, 13)
	Q	3	5	(4, 13)	(4, 14)
	R	8	3	(4, 14)	(4, 15)

4.2. Skill proficiency and quality weight

The skill proficiency of engineers E1 to E8 is shown in Table 4.

Table 4. Engineer skill types and proficiency.

Skill code	Skill type	E1	E2	E3	E4	E5	E6	E7	E8
J1	AI-powered voice solution architecture design	0.7	1	1	1.1	1.2	1	1	1.2
J2	Linux/RTOS porting and maintaining	0.8	0.9	1.1	1.2	1.5	0.7	0.8	1.2
J3	Hardware driver development	0.6	0.8	1	1	1.1	1	1	1
J4	Audio framework development	1	0	1.5	2	2.2	0.6	0.6	1.8
J5	TCP/IP-based network application development	1.2	1.4	1.2	0	2	0.6	0.7	1.2
J6	BT and BLE-based application development	0.9	0.8	0.9	1	1	0.7	0.8	0.9
J7	Edge-cloud cooperative dialogue management	0.9	0.9	0.8	0.9	0.9	0.8	0.9	0.8
J8	Power management	1.2	1.5	1.4	2	2	0.5	1	1.3
J9	System profiling and exception back-trace	1.4	1.3	2	1.1	1	0.8	0.8	1.9
J10	Neural network deployment	1	2	1.2	0	0	0.8	0.8	1.3

Given the limited R&D resources, to maximize overall customer satisfaction, it is necessary to consider providing differentiated service commitments and setting different quality factor weights based on the characteristics of each sub-project and customer style. For example, the client of Project 2, as a leading manufacturer in the air-conditioner field, has strict acceptance standards and high product entry barriers, thus being assigned the highest weight of 0.35. The baseline, as a basic software solution, requires new functional modules to be modular and reusable, and its weight is set as the second highest at 0.3. The other two projects have similar customer requirements, with different weights set according to the order volume: 0.15 and 0.2. Furthermore, the practice of assigning dedicated engineers to a single project, as reflected in our model (Eq. 13), is a strategic approach for serving key account (KA) clients. It facilitates seamless communication and accountability, which are crucial for maintaining long-term partnerships and ensuring high-value project delivery.

4.3. Parameter sensitivity analysis and selection

To ensure the rationality of the proposed NSGA-II algorithm parameter selection, we employed the hypervolume (HV) metric to guide the selection of optimal NSGA-II parameter configurations. The HV metric represents the volume of the objective space enclosed by the Pareto front and the reference point. It simultaneously measures convergence and diversity, with higher HV values being preferable. HV calculation does not require prior knowledge of the true Pareto front, making it suitable for evaluating practical engineering problems. The reference point should be worse than all obtained solutions, meaning it must be inferior to the maximum value for each objective. We obtained the maximum values of multiple objectives from all solutions, then multiplied them by a coefficient of 1.1 to establish the reference point.

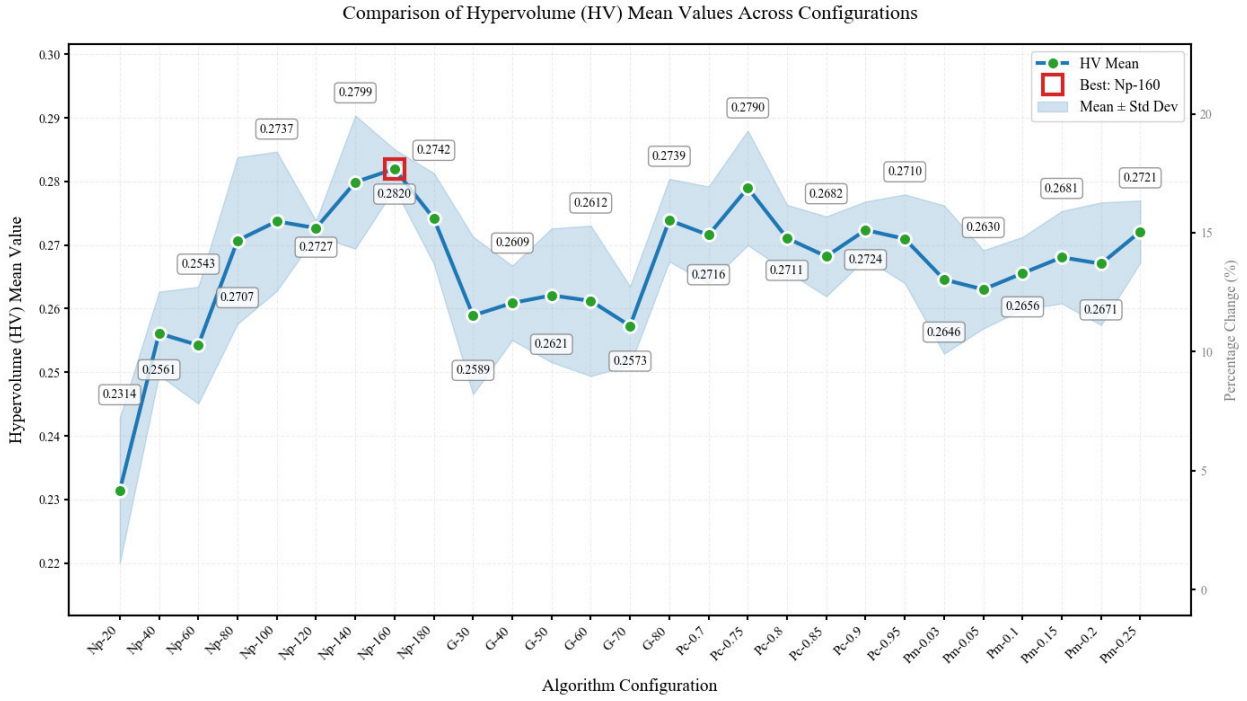

Figure 4. Comparison of HV values across NSGA-II configurations.

Table 5. Comparison of HV values for different NSGA-II configuration scenarios.

Parameter	Test scope	Fixed configuration	HV		
			Mean ± Std	Min	Max
Population size N_p	20	$G = 50$	0.2314 ± 0.0114	0.2147	0.2501
	40	$P_c = 0.9$	0.2561 ± 0.0065	0.2472	0.2637
	60	$P_m = 0.02$	0.2542 ± 0.0091	0.2397	0.2658
	80		0.2707 ± 0.0131	0.2518	0.2903
	100		0.2737 ± 0.0109	0.2589	0.2889
	120		0.2726 ± 0.0011	0.2712	0.2744
	140		0.2798 ± 0.0104	0.2611	0.2924
	160		0.2819 ± 0.0030	0.2765	0.2859
	180		0.2741 ± 0.0071	0.2621	0.2844
Max iterations G	30	$N_p = 120$	0.2589 ± 0.0124	0.2406	0.2744
	40	$P_c = 0.9$	0.2608 ± 0.0058	0.2517	0.2696
	50	$P_m = 0.02$	0.2620 ± 0.0105	0.2454	0.2751
	60		0.2612 ± 0.0118	0.2412	0.2769
	70		0.2572 ± 0.0062	0.2472	0.2659
	80		0.2739 ± 0.0065	0.2645	0.2806
Crossover probability P_c	0.7	$N_p = 120$	0.2715 ± 0.0076	0.2603	0.2818
	0.75	$G = 50$	0.2789 ± 0.0090	0.2665	0.2929
	0.8	$P_m = 0.02$	0.2710 ± 0.0052	0.2657	0.2809
	0.85		0.2682 ± 0.0062	0.2561	0.2739
	0.9		0.2723 ± 0.0044	0.2651	0.2785
	0.95		0.2709 ± 0.0069	0.2619	0.2826
Mutation probability P_m	0.03	$N_p = 120$	0.2645 ± 0.0116	0.2508	0.2807
	0.05	$G = 50$	0.2630 ± 0.0061	0.2546	0.2709
	0.1	$P_c = 0.9$	0.2655 ± 0.0057	0.2573	0.2718
	0.15		0.2680 ± 0.0072	0.2571	0.2799
	0.2		0.2670 ± 0.0096	0.2555	0.2796
	0.25		0.2721 ± 0.0048	0.2665	0.2808

We separately tested the impact of key parameters on the HV metric, including population size N_p, number of iterations G, crossover probability P_c, and mutation probability P_m. Based on the single-variable method, we fixed all other parameters while altering only one parameter per trial (e.g., changing N_p from 80 to 160 while keeping iteration count, crossover, and mutation probabilities constant). After changing one parameter, each test was replicated 10 times, with HV metric variations calculated and observed. The design and results of the sensitivity analysis experiment are shown in Table 5. The distribution of HV metrics across different configuration combinations is illustrated in Figure 4.

As shown in Table 5 and Figure 4, the HV value of the NSGA-II exhibits the highest sensitivity to population size, followed by iteration count, while showing moderate sensitivity to crossover and mutation probabilities. In the population size sensitivity experiment, HV peaked at $N_p = 160$; in the iteration count sensitivity experiment, HV peaked at $G = 80$; in the crossover probability sensitivity

experiment, HV peaked at $P_c = 0.75$; and in the mutation probability sensitivity experiment, HV peaked at $P_m = 0.25$. Furthermore, when $N_p = 160$, $G = 50$, $P_c = 0.9$, and $P_m = 0.02$ (referred to as Configuration Combination A), we achieved the highest mean HV of 0.281984 and the second-lowest standard deviation of 0.003062, indicating superior and more stable algorithm performance under this configuration.

Based on the univariate analysis results, we further integrated the optimal configurations of parameters from the univariate tests, namely $N_p = 160$, $G = 80$, $P_c = 0.75$, and $P_m = 0.25$, designated as Configuration Combination B, and performed the same repeatability tests. Experimental results show that Configuration Combination B achieved an HV mean of 0.277647 and a standard deviation of 0.005387, worse than Configuration Combination A's performance, while also requiring longer runtime than Combination A. Therefore, we selected Configuration Combination A as the final parameter configuration for NSGA-II.

Employing the same parameter selection method, we also determined the key parameters for the SA and ALO algorithms. The values of key parameters for the four algorithms are shown in Table 6.

Table 6. Key parameter values of four algorithms.

Algorithm	Parameter	Value
NSGA-II variants	Population size	160
	Max iterations	50
	Crossover probability	0.9
	Mutation probability	0.02
SA	Initial temperature	2000
	Max iterations	1500
	Cooling rate	0.98
ALO	Population size	100
	Max iterations	80
	Max elite Antlion number	100

4.4. Evolutionary efficacy analysis

The experimental case was solved based on the proposed NSGA-II algorithm first. As all three optimization objectives evolve toward minimizing objective values, solutions positioned closer to the lower region (lower left edge) of the 3D coordinate system exhibit superior performance. Figure 5 demonstrates that the final-generation population (blue points) achieves significantly better performance than the initial population (red points), confirming the algorithm's evolutionary efficacy. The population evolves from the upper right to the lower left, as indicated by the black arrow.

Figure 4 further reveals two critical trade-offs. Improved software quality correlates with degraded workload balance when the project duration is fixed. Similarly, reduced project duration exacerbates workload imbalance when software quality is fixed. These observations align with practical project management principles: high-efficiency and robust deliveries often require greater involvement of senior engineers, whereas increasing junior/mid-level participation may elevate risks of delays or defects [14]. Project managers must strategically balance staff satisfaction, workload equity, and delivery performance based on contextual priorities such as deadline urgency and client quality standards [22].

Additionally, the optimized solutions of the final population yield project duration ranging from 40 to 57 days, software quality metrics between 0.883 and 0.952, and workload balance indices spanning 1.5E-4 to 7.2E-3. Meanwhile, the hypervolume metric reached 27 times that of the initial population, as shown in Table 7.

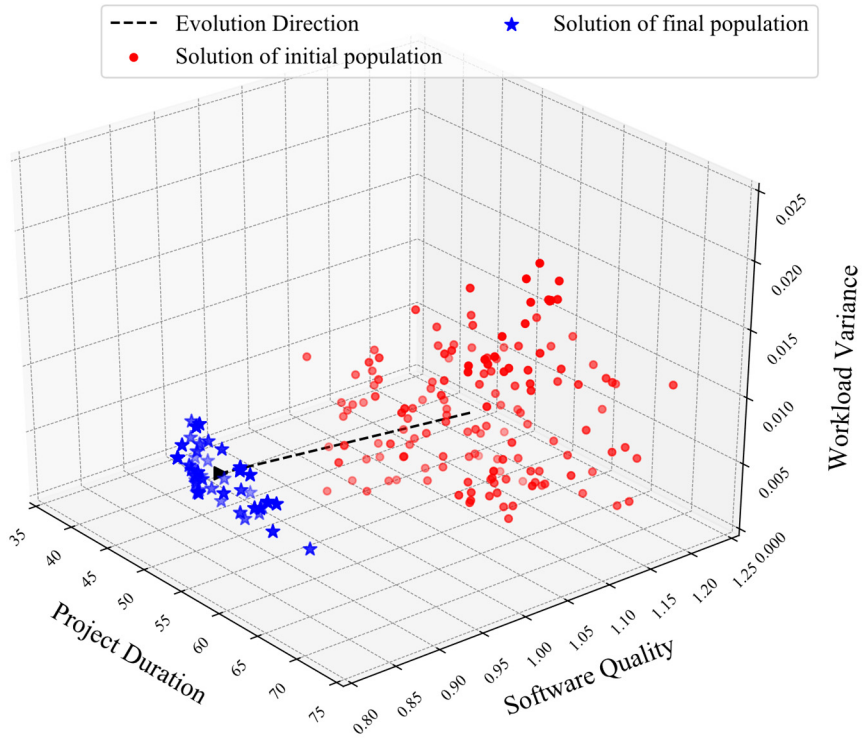

Figure 5. Solutions comparison between the initial and final population.

Table 7. Performance comparison of the initial and final populations.

Population	HV	Project duration T (days)		Software quality Q		Workload variance V	
		mean	min	mean	min	mean	min
Initial	1.902	62.5	49	1.063	0.942	9.3E-3	6.3E-4
Final	51.819	45	40	0.904	0.883	2.9E-3	1.5E-4

4.5. Comparison experiments

Considering the effect of pseudo-random numbers, for more reliable performance estimates, classical and improved NSGA-II variants were executed over 5 independent trials, with the same key algorithm configurations, as shown in Table 6. Additionally, the proposed algorithm was further compared with both ALO and SA meta-heuristic algorithms. The baseline algorithms were run repeatedly 10 and 5 times, respectively, to ensure that the total solution number in the iteration process was not less than NSGA-II variants. Post-execution, non-dominated solutions from the final generation were aggregated for each variant. Both solution sets underwent non-dominated sorting, and elite solutions were extracted to compare algorithmic efficacy.

Figure 6 demonstrates the Pareto front's diversity and convergence precision for four MAs, while Table 8 shows a detailed performance metric comparison. Compared with the classic version, the improved NSGA-II achieved a superior hypervolume metric. Although the other minimum, average statistical indicators of the classic NSGA-II were slightly inferior to those of the improved NSGA-II, at least they were still within the same order of magnitude. This also revealed its shortcomings in refining local optimal solutions.

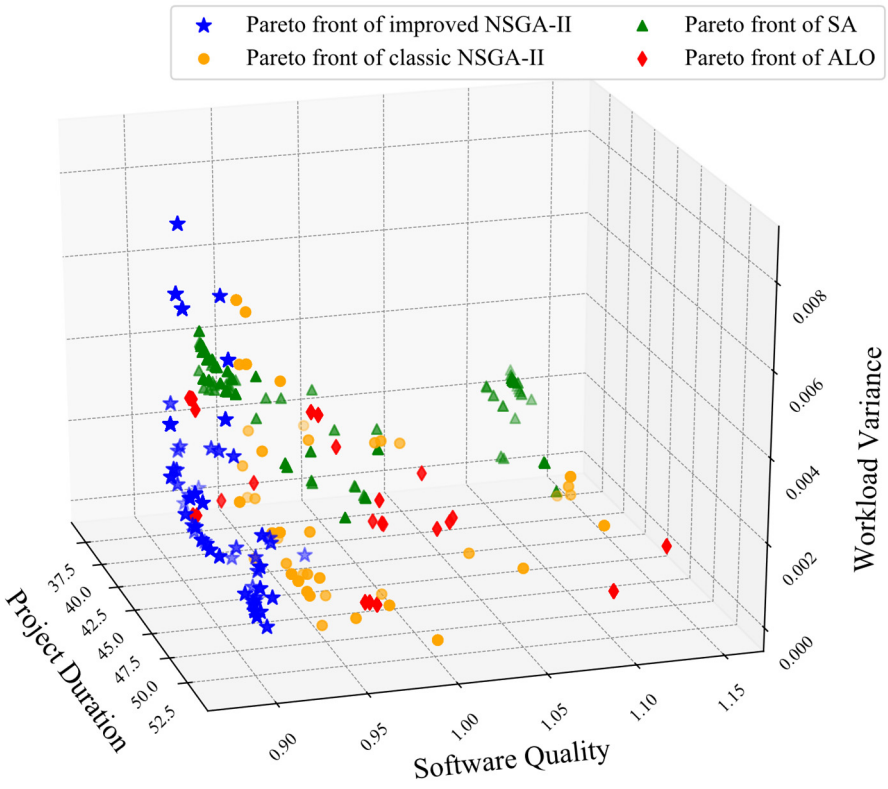

Figure 6. Pareto front comparison of four algorithms.

Table 8. Performance comparison of four algorithms.

Algorithm	HV	Total time (min)	Project duration T (days)		Software quality Q		Workload variance V	
			min	mean	min	mean	min	mean
Classic NSGA-II	2.6043	9.6	37.5	46	0.912	0.988	9.54E-5	2.56E-3
Improved NSGA-II	6.9914	10.5	37	44.5	0.882	0.906	6.51E-5	2.37E-3
ALO	5.4945	22.6	38.5	44.5	0.904	0.988	5.39E-4	1.41E-3
SA	3.1367	2.5	37	39.5	0.909	1.009	1.57E-3	3.36E-3

The SA ran fastest, with an average time of 2.5 minutes, much shorter than that of others. SA achieved much better values on statistical indicators of T but worse values on Q and V indicators. In particular, the minimum of the V indicator is two orders of magnitude worse than the improved NSGA-II algorithm. Its Pareto frontier closely resembled a structure composed of several parallel curves. On one hand, despite having strong neighborhood search capabilities, the single chain searching mode of SA tended to trap it in local optima. On the other hand, the SA, while competitive, struggled with the

discrete nature of the project portfolio scheduling problem, and multiple constraints of project–activity–skill–engineer combinations resulted in many neighborhood operations producing infeasible solutions, making it difficult to maintain the diversity of Pareto frontiers.

The ALO ran slowest, with an average time of 22.6 minutes. Its performance most closely approximates that of the improved NSGA-II, with advantages and disadvantages in different indicators, although several frontier points lie between the two NSGA-II algorithms. It expanded the search space through Ants' random walking, and maintained high-quality solutions through Ant Lion elites, contributing to the global searching ability and population diversity maintenance. Meanwhile, it conducted a neighborhood search guided by Ant Lion info, showing the potential for further optimization of the local optimal solution, but still slightly inferior to that of Pareto-based local fine-tuning of the improved NSGA-II variant, which might be enhanced by fine-tuning the random walking step size.

To quantify the benefits of heterogeneous skill allocation, a homogenized scenario where all engineers' skills were standardized to team averages is simulated. This simulation configuration resulted in stagnant software quality (fixed at 1.11), underscoring the necessity of skill diversity. Table 9 compares the optimal values under heterogeneous versus homogeneous skill conditions, in which project duration is improved by 16%, software quality gets an extra 21.6% enhancement, and workload balance achieves an extra 94.4% optimization.

Furthermore, to quantify the benefits of inter-project dependency-aware scheduling, keeping all NSGA-II configurations and constraints the same based on multi-skill aware scheduling but disconnecting activity dependencies between Project 1 (baseline project) and Project 2, leads to an extra component development for activity F of Project 2. As shown in Table 9, benefiting from multi-skilled and inter-project dependency-aware scheduling, project duration achieves a 5.2% reduction, while software quality and workload variance improve by 3.9% and 53.9%, respectively. These findings emphasize the criticality of modeling both engineer skill heterogeneity and inter-project dependencies to generate efficient scheduling schemes.

Table 9. Comparison of optimal objective values for different scheduling scenarios.

Scheduling scenarios objective	Project duration T (days)	Software quality Q	Workload variance V
Multi-skilled and inter-project dependency-aware	37	0.882	6.51E-5
Homogeneous skill	44	1.11	1.15E-3
Inter-project dependency-unaware	39	0.917	1.41E-4

4.6. Scheduling plan decoding

To streamline decision-making, a hybrid approach combining an analytic hierarchy process (AHP) and hierarchical filtering is proposed: First, assign weights to objectives via AHP based on project-specific priorities. Second, filter Pareto solutions sequentially by the highest-weighted objective. If there are multiple Pareto solutions, filter them again according to the next highest weight objective until the decision-maker can select a satisfactory solution.

A representative solution from the final-generation Pareto front (objective values: [39, 1.03, 8.967E-4]) is decoded into a comprehensive project–activity–skill–engineer scheduling plan. Figure 7 displays the corresponding Gantt chart, detailing the activity schedule and resource allocation across

projects. Each color represents the activity allocation for a specific project, while each row represents an engineer's activity allocation within that project. The letters A through Z within each row denote different activities within that project.

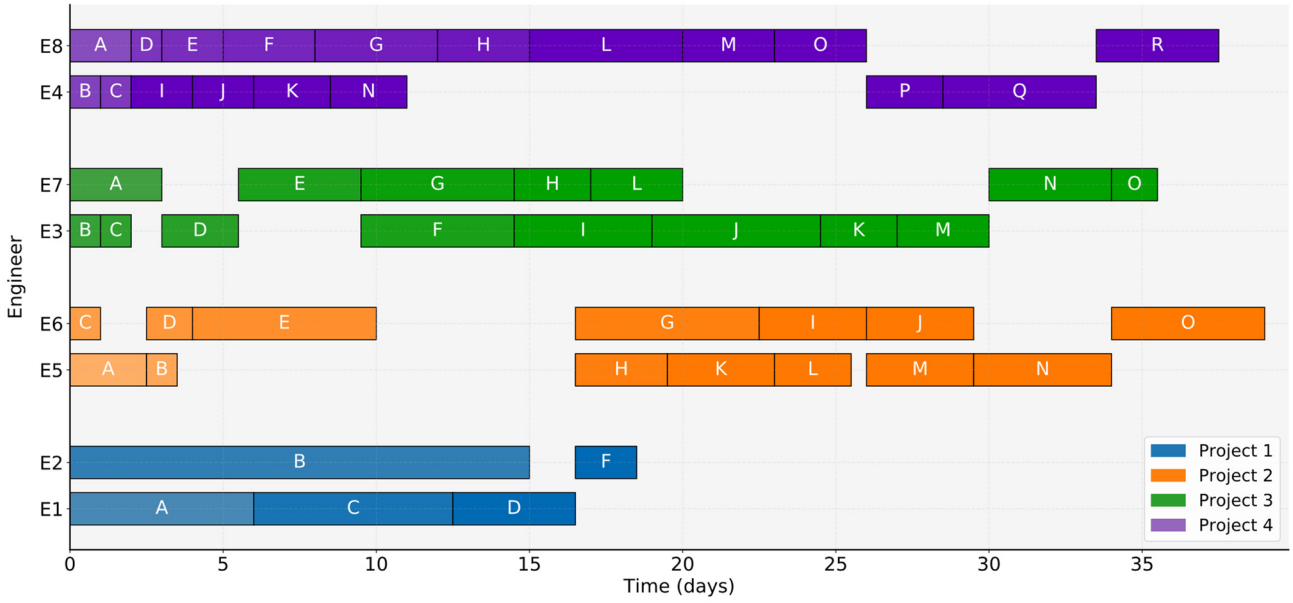

Figure 7. Gantt chart of the project portfolio.

5. Conclusion

This study builds a resource-constrained project portfolio scheduling model that addresses scenarios with heterogeneous multi-skilled human resources and parallel development of baseline and customized projects. The model aims to achieve three objectives by minimizing total project duration, maximizing software quality optimization, and balancing workload distribution. The proposed approach utilizes the improved NSGA-II algorithm to ensure timely project delivery during peak demand periods while enhancing employee satisfaction and aligning corporate survival with staff development needs. Key findings are summarized as follows:

1. By redesigning the coding scheme, heuristic population initialization, and Pareto-based local searching, the improved NSGA-II is aligned with multi-skilled project portfolio scheduling requirements. Computational experiments confirm the algorithm's evolutionary efficacy, in which the final-generation population displays significantly improved performance compared to the initial population. Notably, the hypervolume metric reached 27 times that of the initial population.
2. Compared with the classic NSGA-II, the improved NSGA-II achieved a superior hypervolume metric, while performing better in terms of minimum and average statistical indicators. Compared with ALO and SA, the improved NSGA-II not only possesses strong global search ability but also has excellent local search and refinement level for optimal solutions, demonstrating better model matching and solution effectiveness. In particular, workload variance objective gains orders of magnitude.
3. Scheduling multi-skilled engineers elevates team skill proficiency, positively influencing both project duration and software quality. Skills-aware scheduling achieves additional optimizations

of 16% in duration, 21.6% in quality, and 94.4% in workload balance over homogenized skill scenarios. Further, integrating a self-developed baseline with client-driven customized projects reveals inherent dependencies beyond resource competition. Reusable modular components from baseline projects significantly reduce redundant development in customized projects. Inter-project dependency-aware scheduling improves duration, quality, and workload balance by 5.2%, 3.9%, and 53.9%, respectively, compared to inter-project dependency-unaware scenarios. These findings emphasize the criticality of modeling both engineer skill heterogeneity and inter-project dependencies to generate efficient scheduling schemes.

4. The evolutionary trends reveal inherent trade-offs among the three objectives and empirically align with practical project management principles, while validating that a single aggregated objective function would be insufficient to capture the complex decision-making landscape faced by managers. Decision-makers must strategically balance time, quality, and employee equity objectives based on different contextual priorities, such as project deadlines and quality standards. Decoding optimal solutions generates detailed project–activity–skill–engineer allocation plans, visualized via Gantt charts.

In future research, the primary focus should be on breaking through the assumptions of the model to make it more reflective of uncertain enterprise operations. For instance, allowing the insertion of high-priority projects and dynamically updating the scheduling plan, as well as enabling the reallocation of engineers within and across different projects, would enhance the model's flexibility and compatibility. Meanwhile, more extensive numerical experiments should be conducted using multiple randomly generated instances of varying sizes. Additionally, further improvements to the NSGA-II algorithm could be explored, such as combining it with other meta-heuristic algorithms to develop more efficient search capabilities and operational performance.

Author contributions

H.Z. designed the framework and structure of the study, H.Z. wrote the main manuscript text, YF.M and YX.Q participated in implementing and debugging MAs scheduling software. All authors reviewed the manuscript.

Use of Generative-AI tools declaration

The authors declare they have used AI tools in the creation of this article.

AI tools used: DeepL Translator and Baidu AI LLM Translator.

How were the AI tools used? The aforementioned AI tools were employed to translate selected paragraphs from the original Chinese manuscript into an English draft. We rigorously reviewed all AI-generated translations through manual proofreading, content verification, academic terminology correction, and linguistic refinement, ultimately confirming the scientific accuracy of all content.

Where in the article is the information located? The initial English drafts of the Abstract, second paragraph of the Introduction, first two paragraphs of the Case Study, and Conclusion sections.

Acknowledgments

We would like to thank the anonymous referees for their valuable comments, which significantly improved the presentation of this article.

Conflict of interest

All authors agree to publish and declare no competing interests.

References

1. J. Chen, S. Tong, H. Xie, Y. Nie, J. Zhang, Model and Algorithm for Human Resource-Constrained R\&D Program Scheduling Optimization, *Discrete Dyn. Nat. Soc.*, **2019** (2019), 1–13. https://doi.org/10.1155/2019/2320632
2. H. Hu, Adam Smith from the Perspective of Modernization, *J. Hebei Univ. Econ. Bu.*, **44** (2023), 21–25. https://doi.org/10.14178/j.cnki.issn1007-2101.2023.05.003
3. M. Fekri, M. Heydari, M. M. Mazdeh, Bi-objective optimization of flexible flow shop scheduling problem with multi-skilled human resources, *Eng. Appl. Artif. Intel.*, **133** (2024), 108094. https://doi.org/10.1016/j.engappai.2024.108094
4. B. Afshar-Nadjafi, Multi-skilling in scheduling problems: A review on models, methods and applications, *Comput. Ind. Eng.*, **151** (2021), 107004. https://doi.org/10.1016/j.cie.2020.107004
5. M. Riesener, M. Kuhn, A. Keuper, G. Schuh, A literature analysis on success factors and their corresponding scientific approaches in multi-project management, *Procedia Cirp*, **119** (2023), 1176–1181. https://doi.org/10.1016/j.procir.2023.03.157
6. T. Hegazy, A. K. Shabeeb, E. Elbeltagi, T. Cheema, Algorithm for Scheduling with Multiskilled Constrained Resources, *J. Constr. Eng. M.*, **126** (2000), 414–421. https://doi.org/10.1061/(ASCE)0733-9364(2000)126:6(414)
7. M. Arashpour, R. Wakefield, N. Blismas, J. Minas, Optimization of process integration and multi-skilled resource utilization in off-site construction, *Automat. Constr.*, **50** (2015), 72–80. https://doi.org/10.1016/j.autcon.2014.12.002
8. K. Rajwar, K. Deep, S. Das, An exhaustive review of the metaheuristic algorithms for search and optimization: taxonomy, applications,and open challenges, *Artif. Intell. Rev.*, **56** (2023), 13187–13257. https://doi.org/10.1007/s10462-023-10470-y
9. A. H. Hosseinian, V. Baradaran, An Evolutionary Algorithm Based on a Hybrid Multi-Attribute Decision Making Method for the Multi-Mode Multi-Skilled Resource-constrained Project Scheduling Problem, *J. Optim. Ind. Eng.*, **12** (2019), 155–178. https://doi.org/10.22094/JOIE.2018.556347.1531
10. J. C. Chen, Y. Chen, T. Chen, Y. Lin, Multi-project scheduling with multi-skilled workforce assignment considering uncertainty and learning effect for large-scale equipment manufacturer, *Comput. Ind. Eng.*, **169** (2022), 108240. https://doi.org/10.1016/j.cie.2022.108240
11. Y. Li, L. Jian, W. Jing, Multi-skill resource constrained project scheduling using a multi-objective discrete Jaya algorithm, *Appl. Intell.*, **52** (2022), 5718–5738. https://doi.org/10.1007/s10489-021-02608-8

12. A. Ghamginzadeh, A. A. Najafi, M. Khalilzadeh, Multi-Objective Multi-Skill Resource-Constrained Project Scheduling Problem Under Time Uncertainty, *Int. J. Fuzzy Syst*, **23** (2021), 518–534. https://doi.org/10.1007/s40815-020-00984-w
13. E. Afruzi, A. Aghaie, A. Najafi, Robust Optimization for the Resource Constrained Multi-Project Scheduling Problem with Uncertain Activity Durations, *Sci. Iran.*, **2018** (2018), 875–908. https://doi.org/10.24200/sci.2018.20801
14. Y. Wu, S. Zeng, Y. Yu, Modelling and a hybrid genetic algorithm for the equity-oriented worker assignment problem in *seru* production systems, *J. Ind. Manag. Optim.*, **20** (2024), 36–58. https://doi.org/10.3934/jimo.2023068
15. R. Chen, C. Liang, D. Gu, H. Zhao, A competence-time-quality scheduling model of multi-skilled staff for IT project portfolio, *Comput. Ind. Eng.*, **139** (2020), 106183. https://doi.org/10.1016/j.cie.2019.106183
16. K. Ded, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, *Ieee T. Evolut. Comput.*, **6** (2002), 182–197. https://doi.org/10.1109/4235.996017
17. Y. Wang, S. Ling, Z. Cai, L. Fu, X. Jin, NSGA-II algorithm and application for multi-objective flexible workshop scheduling, *J. Algorithms Comput*, **14** (2020), 1–9. https://doi.org/10.1177/1748302620942467
18. H. Ma, Y. Zhang, S. Sun, T. Liu, Y. Shan, A comprehensive survey on NSGA-II for multi-objective optimization and applications, *Artif. Intell. Rev.*, **56** (2023), 15217–15270. https://doi.org/10.1007/s10462-023-10526-z
19. Y. Xu, Y. Chen, C. Wang, Y. Peng, Improving NSGA-III Algorithm for Solving High-dimensional Many-objective Green Flexible Job Shop Scheduling Problem, *J. Syst. Simul.*, **36** (2024), 2314. https://doi.org/10.16182/j.issn1004731x.joss.23-0694
20. Y. Xu, B. Hao, T. Gao, Q. Zhang, H. Lu, B. Zeng, Research On Frequency Assignment Of Air Navigation Station Based On Multi-objective Genetic Local Search Algorithm, *Sci. Technol. Eng.*, **25** (2025), 6530. https://doi.org/10.12404/j.issn.1671-1815.2405282
21. S. Mirjalili, P. Jangir, S. Saremi, Multi-objective ant lion optimizer: a multi-objective optimization algorithm for solving engineering problems, *Appl. Intell.*, **46** (2017), 79–95. https://doi.org/10.1007/s10489-016-0825-8
22. S. Verma, M. Pant, V. Snasel, A Comprehensive Review on NSGA-II for Multi-Objective Combinatorial Optimization Problems, *IEEE Access*, **9** (2021), 57757–57791. https://doi.org/10.1109/ACCESS.2021.3070634

AIMS Press