ENGINEERING Management 2026, 13(1): 213–239
https://doi.org/10.1007/s42524-026-5124-4

RESEARCH ARTICLE

Feiran WANG, Jiawei CHEN, Yonghao DU, Yanjie SONG, Yingwu CHEN, Rammohan MALLIPEDDI, Witold PEDRYCZ

LLM-assisted adaptive large neighborhood search for agile earth observation satellite scheduling

Abstract The Agile Earth Observation Satellite Scheduling Problem (AEOSSP) is a complex NP-hard challenge that involves selecting, sequencing, and timing observation tasks to maximize imaging profits while adhering to various constraints. In our study, we developed a mixed-integer programming model for AEOSSP, incorporating key constraints related to visible time windows and time dependencies. To tackle this, we propose an Evolutionary Adaptive Large Neighborhood Search Algorithm (evALNS) enhanced by Large Language Models (LLMs). Our work pioneers the application of LLMs to ALNS by being the first to automatically develop and evolve its critical destroy heuristics. However, a naive application of LLMs is insufficient for such a complex domain. We therefore introduce a novel Dual-Population Co-Evolutionary Computing Framework (DPEC) to bridge the LLM's knowledge gap by synergizing LLM-generated heuristics with expert-designed ones. This co-evolution, guided by a Functional Natural Language Embedding (FNLE) strategy and customized prompts, significantly enhances the adaptability and efficiency of ALNS. Extensive numerical experiments demonstrated the superiority of the evALNS evolved under our framework, achieving an average profit improvement of 8.48% compared to the original ALNS with expert-designed destroy operators.

Keywords large language model, algorithm design, adaptive large neighborhood search, satellite scheduling

1 Introduction

Earth Observation Satellites (EOSs) are vital for monitoring the planet's environment, providing crucial data for applications such as environmental monitoring, navigation, and resource management (Du et al., 2022; Yang, 2021). Recent advancements in satellite technology have significantly improved maneuverability, leading to the development of Agile Earth Observation Satellites (AEOSs) (Yao et al., 2023). These new-generation AEOSs have rapidly become essential tools in space missions management, creating unprecedented development opportunities for next-generation remote sensing imaging technology due to their enhanced maneuverability and operational flexibility (Chen et al., 2024b; Chen et al., 2024c).

However, AEOSs introduce multidimensional complexity to scheduling due to their technical characteristics. Unlike traditional EOSs, AEOSs have three additional degrees of freedom—pitch (maneuvering forward or backward along the flight path), roll (tilting side-to-side), and yaw (rotating the satellite body)—allowing observations before and after overhead passage, thus extending visible time windows (VTWs). Fig. 1 illustrates this distinction, where tasks are represented by pink circles, their visible time windows (VTWs) by black boxes, and the actual observation periods by green strips.

Received May 20, 2025; revised Nov. 7, 2025; accepted Dec. 4, 2025

Feiran WANG, Jiawei CHEN, Yonghao DU (✉), Yingwu CHEN
College of Systems Engineering, National University of Defense Technology, Changsha 410073, China
E-mail: duyonghao15@163.com

Yanjie SONG
School of Information Science and Technology, Dalian Maritime University, Dalian 116026, China

Rammohan MALLIPEDDI
Department of Artificial Intelligence, School of Electronics Engineering, Kyungpook National University, Taegu 702-701, Republic of Korea

Witold PEDRYCZ
Department of Electrical and Computer Engineering, University of Alberta, Edmonton AB T6G 2J7, Canada; Institute of Systems Engineering, Macao University of Science and Technology, Macao SAR, China; Research Center of Performance and Productivity Analysis, Istinye University, Istanbul 34010, Türkiye

This work was supported by the National Natural Science Foundation of China (Grant No. 72201272 and 72501042), the Young Elite Scientists Sponsorship Program by CAST (Grant No. 2023-JCJQ-QT-042), and The Science and Technology Innovation Program of Hunan Province (Grant No. 2025RC3111).

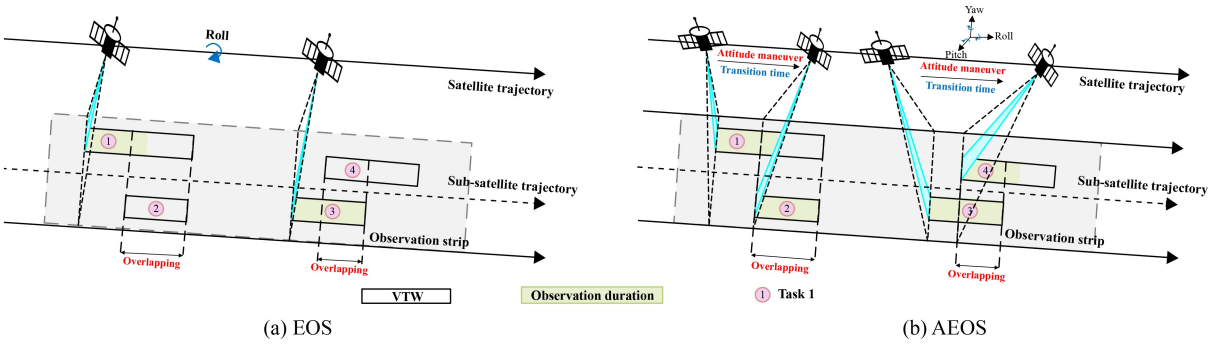

Fig. 1 Difference between EOS (a) and AEOS (b) observation modes. Pink circles represent tasks, with numbers indicating task indices (e.g., "1" corresponds to Task 1). Black boxes indicate the VTWs for each task, which represent the full window of opportunity for observation. Green strips represent the actual scheduled observation periods, whose start times are chosen within the corresponding VTWs.

A VTW represents the entire timeframe of geometric visibility, which is often much longer than the required observation duration; the scheduling problem thus involves selecting an optimal start time for the observation (green strip) within the VTW (black box). As shown in the figure, the VTWs for different tasks (e.g., tasks 1 and 2, and tasks 3 and 4) are determined independently and can overlap in time, creating scheduling conflicts. In EOS mode, such conflicts prevent the execution of both tasks because the satellite cannot be pointed at two different targets simultaneously. In contrast, AEOSs use their maneuverability to mitigate these conflicts. For example, an AEOS can observe task 1 ahead of time with a pitch maneuver, then observe task 2 after passing overhead, effectively resolving the VTW conflict.

This enhanced maneuverability introduces multidimensional, nonlinear system characteristics. Inter-task maneuvering creates time-dependent constraints (Du et al., 2020), while variable energy consumption and storage further complicate scheduling. These factors make AEOS scheduling problems NP-hard (Wang et al., 2021). Therefore, developing efficient AEOS scheduling approaches to optimize observational profits has significant practical importance and underscores the need for intelligent scheduling methods.

Research on AEOSSP algorithms shows a trend toward diversification and interdisciplinary integration. For reinforcement learning algorithms, (Herrmann and Schaub, 2023) formulated the AEOSSP as a markov decision process. They employed monte carlo tree search with supervised learning to train the agent, evaluating two backup strategies. Exact algorithms, such as adaptive-directional dynamic programming with decremental state space relaxation, effectively solve single-orbit scheduling problems and enhance computational efficiency for AEOSSP (Peng et al., 2020). Heuristic algorithms (HAs) are preferred in complex optimization problems for their simplicity, computational speed, and resource efficiency (Li et al., 2023, Li et al., 2025). Unlike reinforcement learning, which requires extensive training data, and exact algorithms, which face exponential time complexity, HAs provide practical solutions for complex scheduling problems including AEOSSP (Kandepi et al., 2024; Song et al., 2024; Han et al., 2023).

Among HAs, Adaptive Large Neighborhood Search (ALNS) has shown strong optimization performance in AEOS scheduling (Liu et al., 2017; He et al., 2018; Liu et al., 2024c; Wu et al., 2024a) and in applications like the vehicle routing problem (Liu et al., 2023b; Jia et al., 2023). ALNS incorporates various destroy and repair operators with an adaptive selection mechanism that dynamically modifies solutions, enhancing local search effectiveness. A critical component of ALNS is the heuristic neighborhood structure design, particularly the destroy heuristic. This operator removes specific tasks from current solutions according to predefined rules. By disrupting local optimality through task removal, it increases the likelihood of finding global optima and prevents premature convergence by encouraging exploration of new solutions. Without efficiently designed destroy heuristics, ALNS may produce lower quality solutions and become trapped in local optima. Therefore, improving destroy heuristic efficiency is crucial for enhancing ALNS's effectiveness in solving combinatorial optimization problems, particularly the complex AEOSSP.

To overcome HAs limitations while preserving efficiency, more advanced techniques have emerged. Methods integrating artificial intelligence (AI) with heuristic algorithms have been developed to address AEOS scheduling. Some approaches combine data-driven strategies with HAs to solve the AEOSSP (Du et al., 2019; Wu et al., 2022). For satellite range scheduling, methods utilizing deep reinforcement learning (DRL) integrated with heuristic scheduling techniques and a reinforcement learning-based genetic algorithm have been proposed (Ou et al., 2023; Song et al., 2023). A knowledge-assisted ALNS that combines integer programming and data mining effectively solves the satellite-ground link scheduling problem (Liu et al., 2024d; Chen et al., 2025a). The success of these hybrid methodologies has extended to other related scheduling domains (Pan et al.,

2024; Wu et al., 2024b; Wang et al., 2024; Pan et al., 2025). This demonstrates that AI-HA integration has become a promising approach for solving the AEOSSP.

However, the AI technologies mentioned above are infeasible for designing efficient heuristic rules, such as destroy operators in ALNS, as they lack text and code generation capabilities. Large language models (LLMs), as pre-trained, advanced deep learning models, have demonstrated powerful capabilities in natural language processing, code generation, artificial intelligence systems, and uncovering scientific patterns (Zhao et al., 2023; Liu et al., 2024a; Vaithilingam et al., 2022; Chen et al., 2025b; Kaddour et al., 2023; Kasneci et al., 2023; Thirunavukarasu et al., 2023). Concurrently, studies have highlighted LLMs' increasing use in optimization. For example, some researchers (Pluhacek et al., 2023) utilized GPT-4 to integrate Particle Swarm Optimization (PSO) with Cuckoo Search (CS), while others (Zhang et al., 2023) leveraged LLMs to enhance hyperparameter optimization by treating model code as a hyperparameter.

Furthermore, the integration of LLMs with evolutionary computation (EC) frameworks, known as LLM-EC, has emerged as a powerful paradigm for designing heuristics (Liu et al., 2024b; Liu et al., 2023a; Liu et al., 2024c; Stein and Bäck, 2025). This approach enables automatic generation of HAs without extensive human expertise or domain-specific knowledge. LLM-EC combines LLMs' generative abilities with evolutionary computation processes—such as selection, crossover, and mutation—guided by tailored prompts. Over successive iterations, the algorithm's fitness typically converges to yield an optimized heuristic. This fusion demonstrates LLMs' potential to empower intelligent and creative optimization processes, opening new avenues for enhancing heuristic performance.

However, existing LLM-EC approaches often perform well on general or toy problems but face significant hurdles when applied to complex, real-world scheduling problems like AEOSSP. This is because LLMs lack the intricate, domain-specific knowledge required to understand constraints such as time-dependent transitions and multi-orbit VTWs. A naive application of LLM-EC is therefore likely to produce syntactically correct but semantically flawed or inefficient heuristics

To address the inefficiency of destroy operators in ALNS and efficiently solve the AEOSSP, we propose an Evolutionary Adaptive Large Neighborhood Search Algorithm (evALNS) that leverages LLMs' powerful heuristic generation capabilities. The evALNS is an improved version of ALNS that incorporates creative and efficient destroy operators generated by LLMs. To the best of our knowledge, this is the first application of LLMs in designing ALNS destroy operators, significantly enhancing algorithm performance in the complex AEOS scheduling domain. While the concurrent study by (Ye et al., 2025) is highly relevant, our work differs fundamentally in its core challenge and methodology. Ye et al. focus on creating a self-adapting process for general-purpose optimization, where the LLM dynamically refines prompts for neighborhood selection using a scoring mechanism. In contrast, our research addresses the domain-specific knowledge gap of LLMs for the specialized AEOSSP (Adaptive Evolutionary Optimization for Scheduling Problems, if not defined earlier). We aim to enable LLMs to generate complete destroy heuristics, which extend beyond simple scoring rules. Our key contribution, the Dual-Population Co-Evolutionary (DPEC) framework, is designed for knowledge fusion, allowing the LLM to directly learn from and co-evolve with expert-designed heuristics.

First, to address LLMs' insufficiency in understanding the AEOSSP, we proposed integrating LLM-generated heuristics with expert-designed heuristics within a Dual-Population Co-Evolutionary Framework (DPEC). This innovative framework allows LLMs to better understand the AEOSSP by utilizing information from the expert population, enabling the creation of highly effective destroy operators. Second, we introduced the concept of Functional Natural Language Embedding (FNLE) within the evolved destroy operator code, co-evolving these functional descriptions alongside their codes. By integrating functionality directly into the corresponding code and evolving it in a single-threaded manner, we maintained LLMs' reasoning chain during the iteration process, facilitating efficient destroy operator design and deepening the model's understanding of AEOSSP, thereby reducing error rates in generated code. Finally, we have developed tailored prompting strategies for AEOSSP. These strategies encompass efficient initialization prompts for destroy operators, prompts for crossover mutation, and prompts based on the FNLE we proposed.

Our evALNS leverages LLMs for dynamic heuristic generation to enhance ALNS performance in the AEOSSP, reducing manual design efforts and effectively addressing complex scheduling challenges like the AEOSSP.

In summary, this paper presents the following key contributions:

1) Pioneered the application of LLMs for designing critical components of ALNS, specifically by developing a framework to automatically evolve high-performance destroy heuristics for the complex AEOSSP.

2) Proposed a novel Dual-Population Co-Evolutionary (DPEC) framework that effectively bridges the knowledge gap between general-purpose LLMs and specialized optimization domains. This framework, supported by our FNLE strategy and customized prompts, enables LLMs to generate creative and domain-aware heuristics.

3) Provided empirical evidence of the framework's effectiveness through extensive experiments. Our results not only show significant performance gains (an 8.48% improvement) but also, through ablation studies,

demonstrate that the DPEC and FNLE components are crucial for success.

The paper is structured as follows: Section 2 presents the mathematical model for the AEOSSP, detailing the key notations and problem formulation. Section 3 discusses the evALNS algorithm assisted by LLMs, including the DPEC framework, FNLE strategy, and our proposed customized prompt engineering. Section 4 introduces the computational experiments and results, covering the ablation study, sensitivity analysis, and cross-validation. Finally, Section 5 concludes the paper by summarizing the findings and contributions.

2 Mathematical model

For AEOSSP modeling, several core constraints must be considered. First, the Visible Time Window (VTW) represents the limited time frames during which a satellite can observe a task. Typically, the scheduling period includes 14 to 15 orbital cycles per day (Liu et al., 2017), with each task potentially appearing in multiple cycles. This results in multiple VTWs for each task, requiring the implementation of the **uniqueness constraint** to select the appropriate VTW. Since the VTWs for AEOSs are generally much longer than the observation duration, a start time must be chosen within the task's VTW to satisfy the specified time window limits, which are referred to as the **VTW constraint**. Additionally, AEOSs have time-dependent characteristics that necessitate sufficient time for the satellite's attitude adjustments between tasks, leading to the **transition time constraint**. Furthermore, additional constraints such as energy and storage must also be considered.

Before modeling the AEOSSP, this paper introduces several key assumptions to simplify its complexity. Based on these assumptions, we define the notations used to describe the problem in this section. Clarifying these notations establishes a clear foundation for the subsequent model construction and analysis. Finally, we present the AEOSSP model, including the objective function and associated constraints.

2.1 Assumptions

To align with the operational workflows of actual satellite mission management and facilitate research, this study makes the following assumptions to simplify the AEOSSP:

1) For complex imaging requirements, such as large-area target coverage, methods like strip segmentation, as described by Chatterjee and Tharmarasa (2024), can transform these tasks into multiple meta-tasks. In this paper, we focus solely on meta-tasks.

2) Tasks are non-preemptive during execution; once initiated, a task cannot be interrupted.

3) Each task is executed no more than once and cannot be repeated.

4) Special constraints, such as emergency events and cloud cover, are not considered

2.2 Notations

We provide an overview of the parameters and variables utilized in this study. Let $\mathcal{T}=\{1,2,\ldots,i,\ldots m\}$ represent the set of tasks and $O=\{1,2,\ldots,j,\ldots n\}$ represent the set of orbits in a defined observation period, where m and n denote the number of tasks and orbit, respectively. For clarity, the variables and related parameters used in modeling the AEOSS problem are summarized in Table 1. Notably, we have highlighted the decision variables in bold.

2.3 Model

Based on the analysis of the core constraints associated with the AEOSSP, we present the model as follows:

$$\max \sum_{i\in\mathcal{T}}\sum_{k\in O} q_i \cdot z_i^k, \tag{1}$$

$$\text{s.t.} \sum_{k\in O} z_i^k \leqslant 1,\ \forall i\in\mathcal{T}, \tag{2}$$

$$\sum_{j\in\mathcal{T}\cup\{e\}, j\neq i} w_{ij}^k = \sum_{j\in\mathcal{T}\cup\{s\}, j\neq i} w_{ji}^k = z_i^k,\ \forall i\in\mathcal{T},\ k\in O, \tag{3}$$

$$\sum_{j\in\mathcal{T}\cup\{e\}} w_{sj}^k = 1,\ \forall k\in O, \tag{4}$$

$$\sum_{j\in\mathcal{T}\cup\{s\}} w_{je}^k = 1,\ \forall k\in O, \tag{5}$$

$$\sum_{i\in\mathcal{T}}\sum_{k\in O} z_i^k \cdot d_i \cdot cb_i \leqslant C,\ \forall i\in\mathcal{T}, \tag{6}$$

$$\sum_{i\in\mathcal{T}}\sum_{k\in O} z_i^k \cdot d_i \cdot em_i + \Delta t \cdot h^{pd} \leqslant E,\ \forall i\in\mathcal{T}, \tag{7}$$

$$u_i + d_i + \delta_{ij}^k - u_j \leqslant M\left(1-w_{ij}^k\right),\ \forall i,j\in\mathcal{T}, k\in O, \tag{8}$$

$$bu_i^k \leqslant u_i \leqslant u_i + d_i \leqslant eu_i^k,\ \forall i\in\mathcal{T}, k\in O, \tag{9}$$

$$z_i^k \leqslant a_i^k,\ \forall i\in\mathcal{T}, k\in O, \tag{10}$$

$$w_{ij}^k, z_i^k, y_i^l \in \{0,1\}, u_i\in\mathbb{Z}^+,\ \forall i,j\in\mathcal{T}\cup\{e,s\}, k\in O. \tag{11}$$

The objective function (1) maximize the total profits

Table 1 Notations

Notation	Description
z_i^k	Binary decision variable: $z_i^k = 1$ if task i is scheduled in orbit k, else $z_i^k = 0$
w_{ji}^k	Binary decision variable: $w_{ji}^k = 1$ means in orbit k, task i is scheduled before task j, else $w_{ji}^k = 0$
u_i	Integer decision variable: Start observation time of task i (seconds)
q_i	Profit from observing task i once
d_i	Observation duration of task i
cb_i	Storage required per second for observing task i
C	Maximum storage capacity of the satellite
em_i	Power consumption per second for observing task i
h^{pd}	Power consumption for a one-degree maneuver
E	Maximum energy capacity of the satellite
δ_{ij}^k	Transit time between tasks i and j in orbit k
M	A large positive number
bu_i^k	Start time of the VTW for task i in orbit k
eu_i^k	End time of the VTW for task i in orbit k
a_i^k	Intermediate binary variable indicating if there is a VTW for task i in orbit k
$\alpha^*, \beta^*, \gamma^*$	Roll, pitch, and yaw angles at time $*$
s	Starting dummy node
e	Ending dummy node

(An expert-defined nonnegative number describing the importance of the task) of all selected tasks. The detailed explanations of the constraints are outlined as follows:

1) Constraint (2) ensures that each task i can be selected in at most one orbit k.

2) Constraint (3) ensures flow balance for task i in orbit k, meaning the inflow and outflow of task i are equal to z_i^k.

3) Constraint (4) ensures that the total flow from the starting dummy node s is 1.

4) Constraint (5) ensures that the total flow reaching the ending dummy node e is 1.

5) Constraint (6) ensures that the total capacity consumed by all observed tasks cannot exceed the capacity limit C.

6) Constraint (7) ensures that the total energy consumed by executing the scheduling plan does not exceed the energy limit E.

7) Constraint (8) is the transit time constraint between tasks i and j in orbit k.

8) Constraint (9) is the VTW constraint, meaning the beginning and ending observation times cannot exceed the duration of the VTW.

9) Constraint (10) ensures the feasibility of task i in orbit k.

10) Constraint (11) ensures that w_{ij}^k and z_i^k are binary variables, and u_i is a positive integer variable.

In the model, δ_{ij}^k represents the transit time between tasks i and j, and can be calculated as follows:

$$\delta_{ij}^k = f(u_i, u_j) = \begin{cases} 11.66, & \Delta t \leqslant 10 \\ 5 + \dfrac{\Delta t}{a_1}, & 10 < \Delta t \leqslant 30 \\ 10 + \dfrac{\Delta t}{a_2}, & 30 < \Delta t \leqslant 60 \\ 16 + \dfrac{\Delta t}{a_3}, & 60 < \Delta t \leqslant 90 \\ 22 + \dfrac{\Delta t}{a_4}, & 90 < \Delta t. \end{cases} \tag{12}$$

The values in Eq. (12) are derived from reference (Wu et al., 2023). Δt is a measure of the satellite's attitude maneuvering angle, which can be calculated by

$$\Delta t = |\alpha^{v_i} - \alpha^{u_j}| + |\beta^{v_i} - \beta^{u_j}| + |\gamma^{v_i} - \gamma^{u_j}|, \ \forall i, j \in \mathcal{T}, \tag{13}$$

where α^{v_i}, β^{v_i} and γ^{v_i} represent the satellite's roll, pitch, and yaw angles at the end time of task i, and α^{u_j}, β^{u_j}, and γ^{u_j} are the corresponding angles at the start time of task j, and since $v_i = u_i + d_i$, then we obtain

$$\Delta t = \left|\alpha^{u_j + d_j} - \alpha^{u_j}\right| + \left|\beta^{u_j + d_j} - \beta^{u_j}\right| + \left|\gamma^{u_j + d_j} - \gamma^{u_j}\right|, \ \forall i, j \in \mathcal{T}. \tag{14}$$

Furthermore, since the observation duration of the tasks are a fixed value which are generally specified by the operations department, we can see that δ_{ij}^k is actually a bivariate function of the decision variables u_i and u_j, and can be defined as:

$$(u_i, u_j) \xrightarrow{f} \delta_{ij}^k. \tag{15}$$

In Eq. (12), $a_1,...,a_4$ represent the angular velocities corresponding to different attitude angle variations, respectively. This is because larger changes in attitude angles require correspondingly higher angular velocities to maintain efficient attitude maneuvers. Specifically, we refer to references (Liu et al., 2017; Wu et al., 2023) and set the parameters as follows: $a_1 = 1.5^\circ/s$, $a_2 = 2^\circ/s$, $a_3 = 2.5^\circ/s$, and $a_4 = 3^\circ/s$. It is also important to note that the VTW on different orbits must adhere to the transit time constraint. This is because a satellite will pass through an extended eclipse period when it changes orbits. Therefore, we have imposed transit time constraints only on different VTWs within the same orbit.

Considering the constraints presented, Constraint (8) is inherently nonlinear. This complexity renders the AEOSSP NP-hard (Liu et al., 2017), posing a significant challenge even for professional integer programming solvers. Research by (Liu et al., 2017) demonstrated that CPLEX becomes ineffective when handling more than 12 tasks. This inefficiency stems from the expanded solution space created when linearizing the nonlinear constraint. HAs, particularly ALNS, provide advantages over exact algorithms and reinforcement learning-based methods, including greater efficiency, flexibility, and robustness. This motivates our approach to leverage LLM-assisted heuristic algorithm design to enhance ALNS for efficiently solving the AEOSSP.

3 Evolutionary ALNS driven by large language models

In this section, we introduce our primary contribution: an evolutionary adaptive large neighborhood search algorithm driven by LLMs to effectively solve the complex AEOSSP. Recognizing that a naive application of LLMs lacks the domain-specific knowledge required for this problem, we have developed a novel Dual-Population Co-Evolutionary (DPEC) framework to guide the evolution of high-performance destroy operators. This section systematically unpacks our methodology. We begin by presenting the overall architecture of the LLM-DPEC framework (Section 3.1). We then detail its critical components, including our unique approach to individual representation using a functional natural language embedding strategy (Section 3.2), the initialization of both the LLM-generated and expert populations (Section 3.3), the design of tailored, prompt-driven evolutionary operations (Section 3.4), and finally, the fitness evaluation process (Section 3.5).

3.1 Dual-population cooperative evolution framework

In contrast to classic problems like the TSP and the bin packing problem, the training data for the AEOSSP is less abundant for LLMs. While solutions such as fine-tuning the model (Ding et al., 2023; Thirunavukarasu et al., 2023) exist to improve its suitability for the AEOSSP, this approach is labor-intensive, inefficient, and can result in significant waste of computational resources. Recently, retrieval augmented generation (Gao et al., 2023; Fan et al., 2024) has emerged as a promising alternative; However, it also encounters several challenges, including the need for considerable human and material resources to collect and organize data for constructing a vector database. A lightweight solution involves incorporating information from classic and efficient expert-designed destroy operators into the LLMs. This integration would enable the LLMs to develop a comprehensive understanding of the heuristic rules and design concepts relevant to the AEOSSP.

Fig. 2 illustrates the complete workflow of our proposed LLM-DPEC framework, which is designed to evolve high-performance destroy operators for our evALNS algorithm.

The process can be broken down into the following key stages:

1) Initialization: The process begins by creating two distinct populations of destroy operators. The first, the LLM Population, is generated from scratch using carefully designed prompts. The second, the Expert Population, is seeded with well-established, human-designed heuristics from existing literature.

2) Parallel evolution: Each population undergoes an independent evolutionary process for a set number of generations. Within each generation, new candidate operators (offspring) are created through LLM-guided crossover and mutation operations.

3) Fitness evaluation: Every newly generated destroy operator is evaluated by integrating it into the ALNS algorithm. This ALNS is then run on a specific AEOSSP scenario, and the resulting total profit serves as the operator's fitness score. A higher profit indicates a more effective operator.

4) Co-evolutionary interaction: After a predefined number of generations (the elite exchange cycle), the core co-evolutionary mechanism is triggered. The top-performing individuals (elites) from the LLM population are transferred to the expert population, and vice versa. This exchange allows the LLM to learn from proven, domain-specific strategies while potentially injecting novel, creative solutions into the expert pool.

5) Termination and selection: The cycle of parallel evolution and periodic interaction continues until the maximum number of generations is reached. Finally, the single best-performing destroy operator from either population is selected as the final output, which is then used in the evALNS for solving the AEOSSP.

The advantage of this framework is that it allows expert-designed heuristic information to assist the LLMs in better understanding the AEOSSP, while the heuristics generated by the LLMs may, in turn, help the expert

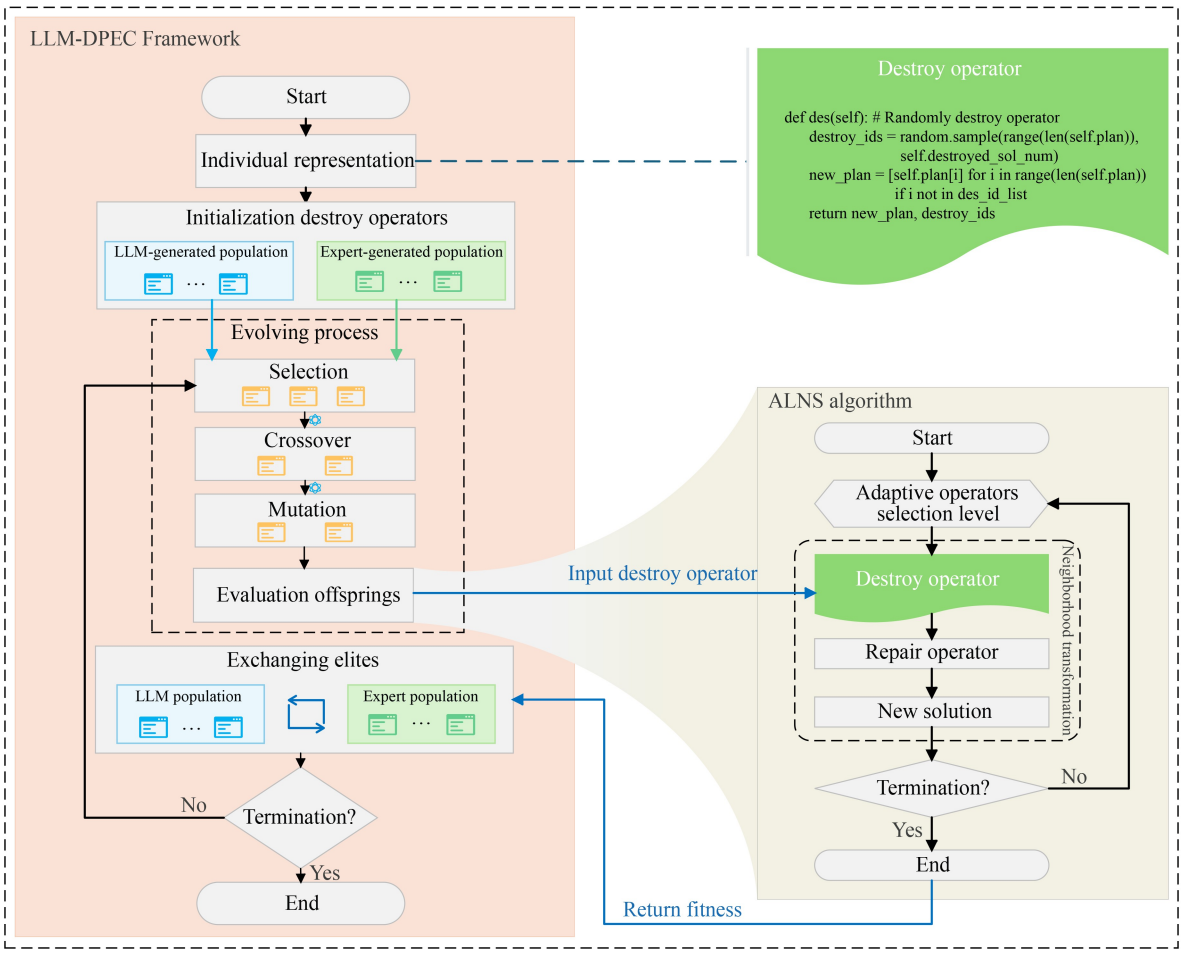

Fig. 2 Diagram of LLM-DPEC framework for evolving the destroy operator within the evolutionary ALNS (evALNS) to solve the AEOSSP.

population acquire new heuristic rules and both evolve more effectively. Algorithm 1 Illustrates the pseudo-code of the proposed LLM-DPEC framework.

Algorithm 1 details the execution flow of the proposed LLM-DPEC framework. The process begins with the initialization of the LLM and expert populations. Subsequently, for a predefined number of generations, the algorithm performs selection, crossover, mutation, and fitness evaluation within each population. After a specified number of generations, an elite exchange mechanism facilitates co-evolution by transferring top-performing individuals between the two populations. Upon completion, the framework returns the overall best destroy heuristic discovered. To explore the algorithm space X more effectively, we designed multiple crossover and mutation operators. Six prompt strategies were developed to create new destroy heuristics based on specific probabilities. When a strategy is selected in each generation, it is applied N times to produce N destroy heuristics. Each new heuristic is evaluated on AEOSSP scenario, and if feasible, added to the current population. Consequently, each generation can incorporate up to $6N$ new heuristic rules into the existing population. Finally, the best N destroy heuristics are selected to form the next generation.

3.2 Individual representation

In this section, we will introduce the individual representation within the LLM-DPEC framework, focusing primarily on the coding of individuals and our novel FNLE strategy applied to individual coding.

3.2.1 Individual coding

In the LLM-DPEC framework, each destroy operator functions as an individual. The destroy operator, coding in Python, is illustrated in Fig. 3. The code block consists of two main parts. The first part includes an explanatory section, which is further divided into three components:

1) Functionality block: The functionality block outlines the implementation logic and purpose of each destroy operator. This explanation aids the LLMs in understanding the algorithms within the population during each

Algorithm 1 Dual-population cooperative evolution algorithm for evolving the destroy operator in ALNS

Inputs: Population size n, Maximum generations T, Number of iterations per generation N, AEOSSP scenario s_i, Exchange cycle of elites T_e, Number of elites for exchanging N_e.

Outputs: Optimal destroy heuristic.

Initialize the LLM population P^0_{LLM} and expert population P^0_{expert} using LLM
for each generation t from 0 to $T-1$ **do**:
for each population P in $\{P_{\text{LLM}}, P_{\text{expert}}\}$ **do**:
$P_t^{(0)} \leftarrow P_t$
$O_{m,t} \leftarrow \varnothing$
for each iteration k from 1 to N **do**:
Select subset $P_t^{(k)}$ from $P_t^{(0)}$ using a specific selection mechanism.
Generate crossover offspring $O_{c,t}^{(k)}$ using pairs from $P_t^{(k)}$.
Apply mutations to obtain $O_{m,t}^{(k)}$ from $O_{c,t}^{(k)}$.
for each mutant a in $O_{m,t}^{(k)}$ **do**:
Evaluate its fitness f_a by applying a to ALNS for solving AEOSSP.
end for
Accumulate mutations in $O_{m,t}$ from $O_{m,t}^{(k)}$.
end for
Select next generation P_{t+1} based on fitness ranking from $O_{m,t} \cup P_t$
if termination condition met **then**:
break
end if
end for
if termination condition met **then**:
Perform elite exchange between P^{t+1}_{LLM} and P^{t+1}_{expert}, involving N_e elite individuals.
end if
end for
Find a^*_{LLM} as the best individual from P^T_{LLM}.
Find a^*_{expert} as the best individual from P^T_{expert}
Return the overall best destroy heuristic: $a^* \leftarrow Best(a^*_{\text{expert}}, P^T_{\text{expert}})$

iteration, ultimately allowing it to generate more effective algorithms.

2) Parameter block and return value block: The two blocks clarify the parameters of the algorithm and the output format, ensuring consistency throughout the implementation.

It is important to note that all destroy operators are encapsulated within the Destroy class in this implementation, indicating that the parameters for each destroy operator function are set to "self."

3.2.2 Natural language embedding for individuals' functionalities

The functionality of individuals often involves complex algorithmic logic and parameter settings. Using natural language to embed this information helps LLMs grasp the behavior and intent of individuals more intuitively. This understanding not only aids in generating new individuals but also provides feedback on existing ones during the optimization process. Furthermore, FNLE enhances algorithm design and implementation clarity by embedding algorithm functionalities and characteristics in natural language. This approach minimizes coding errors from misunderstandings or ambiguities and allows for effective identification and reuse of algorithm features, reducing redundant implementations and code duplication. Fig. 3 illustrates an example of an individual generated during the evolutionary process.

Fig. 4 demonstrates that each individual possesses a unique FNLE within the function code explanation block. As evolution advances, these FNLEs gradually converge into three main components: first, the indicators, which can be referred to as the characteristics of the problem scenarios utilized by the individual; second, an explanation of how the destroy operator selects tasks for destroy; and

Individual Representation in Python Code

```
def des(self): # destroy operator
    """
    Functionality:The basic idea of this function is to randomly select tasks and remove them.
    Parameter:
        - self
    Returns:
        - new_plan: new scheduling plan.
        - destroyed_task_ids: a List of indexes of the destroyed tasks.
    """
    # Main implementation
    destroyed_task_ids = random.sample(range(len(self.plan)), self.destroyed_task_number)
    new_plan = [task for i, task in enumerate(self.plan) if i not in destroyed_task_ids]
    return new_plan, list(destroyed_task_ids)
```

Fig. 3 An example of the individual representation for randomly destroy operator in Python code.

FNLE strategy

```
def des(self): # destroy operator
"""
```

Indicators

Functionality: This function integrates a hybrid approach combining VTW duration, attitude angle restrictiveness, task VTW flexibility, task priority, observation duration, the number of attitude angles per VTW, and a modified randomness factor to select tasks for removal.

Destroy mechanism

It prefers tasks with shorter VTW durations, more restrictive attitude angles, less flexible VTWs, lower priorities, shorter observation durations, fewer attitude angles per VTW, and introduces a modified randomness factor for better exploration.

Additional elements

Additionally, it incorporates an adaptive weight adjustment mechanism based on task index parity for improved robustness.

```
"""
<codes...>
```

Fig. 4 Demonstration of FNLE strategy.

third, additional elements, methods, or mechanisms that are currently integrated into this operator.

The application of the FNLE strategy has led to the design of a specific mutation prompt, as illustrated in Fig. 8 in Subsection 3.4.2, which depicts the crossover prompt C1. The purpose of prompt C1 is to enable selected parent individuals to integrate their functionalities. This fusion clearly benefits the LLMs by allowing them to better understand the functions of each individual through the natural language representation of each parent's FNLE. This improved understanding facilitates enhanced integration and crossover, ultimately resulting in more accurate code and improved destroy capabilities of leading offspring individual. Figure 5 provides an example of the process for generating offspring in a specific iteration using the C1 prompt in conjunction with the FNLE strategy.

Figure 5 shows that, guided by the C1 prompt, the offspring inherit indicators from the parents. For example, the yellow sections from Parent 1 and the green sections from Parent 2 are both passed on to the offspring. Additionally, several destroy mechanisms are integrated into the offspring. For instance, the adaptive weight adjustment mechanism from Parent 1 and the probabilistic removal mechanism from Parent 2 are combined into the offspring's mechanism of adaptive randomness and multi-criteria decision-making, as shown in the pink section.

In summary, by effectively utilizing FNLE, we can enhance the functional understanding of individuals, facilitating the resolution and optimization of the AEOSSPs. This approach not only provides rich contextual information to LLMs but also offers new insights for future research in heuristics evolution by using LLMs.

3.3 Initialization

In this section, we will describe how the two populations in the LLM-DPEC framework are generated during initialization.

3.3.1 LLM-generated population

First, the LLM-generated population is created using an initialization prompt to guide the LLM. Most research focuses on well-known problems like TSP, VRP, and bin packing, which likely

benefit from ample training data available to the LLM. In contrast, research on AEOSS is less prevalent with scarce resources, creating challenges for the LLM to comprehend these issues deeply. A well-structured initialization prompt is essential for generating efficient

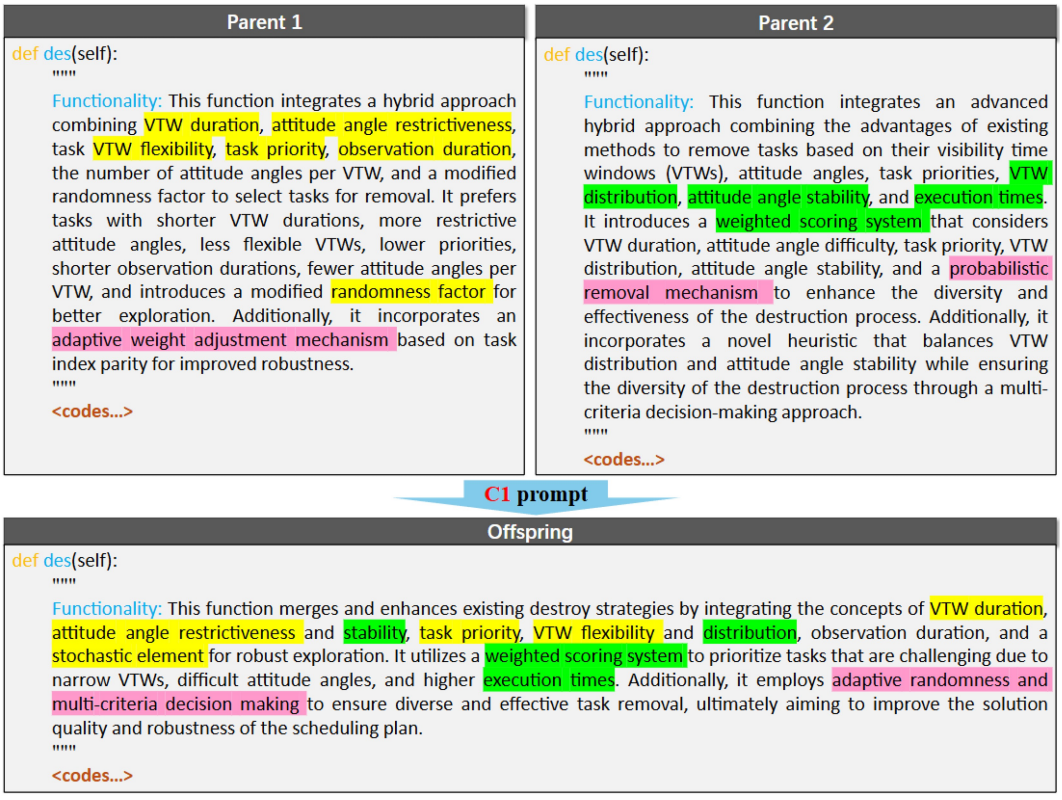

Fig. 5 Demonstration of the application of C1 crossover prompt in the FNLE strategy. The yellow sections in Parent 1 and the green sections in Parent 2 represent the heuristic indicators embedded in the destroy operators of the two parents. The magenta sections in both parents denote the respective destroy mechanisms.

initial destroy operators tailored to the AEOSSP. We organize the initialization prompts into several modules as shown in Fig. 6.

1) Problem description: This module clarifies the nature of the problem and our objectives, effectively informing the LLM of intended outcomes.

2) Example codes: Providing example code during initialization enhances the LLM's robustness and correctness while helping it understand our goals. Fig. 3 displays this example code module.

3) Supporting classes: Project-based approaches are preferred since we typically encapsulate VTW and attitude angle data before instantiation. Clear information about data types enables the LLM to design more efficient destroy operators while enhancing code correctness.

Detailed population initialization prompts can be found in Appendix A Fig. A1 It includes the following modules:

4) Variable explanations: Clear explanations reduce the risk of misusing variables in complex problems like AEOSS, minimizing errors in LLM outputs.

5) Clear task prompts: These help the model accurately understand intentions and expectations, ensuring generated responses align with user needs and leading to more effective destroy operator designs.

6) Requirements and output control: Clearly specifying the requirements for function parameters, return values, helper functions, third-party libraries, and output formats is crucial. Doing so can significantly reduce errors in code generation and simplify the management of strings produced by LLMs.

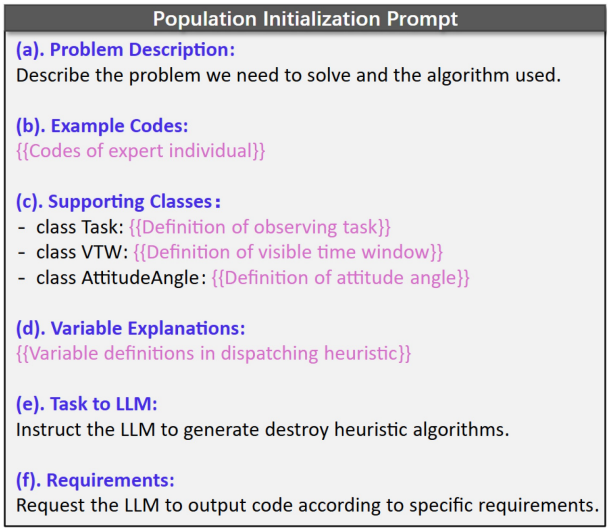

Fig. 6 Prompts for population initialization using LLMs.

3.3.2 Expert-generated population

Second, the expert-generated population is based on the efficient destroy operators developed in previous studies, such as those by (He et al., 2018; Wu et al., 2023). We select a subset of these operators to form the initial population within the expert-generated population in the DPEC framework. The selected expert individuals, along with their detailed introductions, are as follows:

1) Worst path destroy: This destroy operator evaluates the quality of a solution sequence of length pd by calculating its unit profit (i.e., the profit of pd tasks divided by the length of the corresponding time period) and removes the worst-performing sequence.

2) Priority destroy: This greedy heuristic aims to remove the N lowest-priority tasks from the current scheduling solution.

3) Opportunity destroy: Within a scheduling period, a satellite may have multiple time windows for a single observation task. This destroy operator will remove N tasks with the maximum number of VTWs.

4) Conflict destroy: The calculation of the conflict degree is a two-step process:

First, the conflict degree between the currently scheduled VTW and other unscheduled VTWs is defined as follows:

$$C(tw') = \frac{\sum_{tw'' \in Over(tw')} Duration(tw', tw'')}{|Over(tw')|}, \tag{16}$$

where tw' represents the currently scheduled VTW, $tw'' \in \mathcal{U}$ and $\mathcal{U}$ is the set of unscheduled VTWs, $Over(tw')$ represents the set of all VTWs that overlap with tw', and $Duration(tw', tw_{mn})$ indicates the overlap length. Therefore, the conflict degree of tw' is defined as the average overlapping length with tw'.

Second, the conflict degree of task T, denoted as $C(T)$, can be expressed as the sum of the conflict degrees of all its time windows. The formula is:

$$C(T) = \sum_{tw \in tw(T)} C(tw), \tag{17}$$

where $C(T)$ is the conflict degree of task T, $tw(T)$ is the set of all VTWs of task T, $C(tw)$ is the conflict degree of a single time window tw.

Finally, the operator identifies the N tasks with the highest total conflict degree $C(T)$ and removes them from the current schedule. This targeted removal aims to resolve the most significant bottlenecks, increasing the flexibility for the repair operator to improve the overall solution.

5) Transit-time destroy: In the current scheduling plan, we sequentially calculate the transition time from each task to the next and subsequently remove the N tasks with the longest transition times. This process is illustrated in Fig. 7. Let TT represent the set of transition times, defined as

$$TT = \{t_1, t_2, ..., t_s\}.$$

The destroy operator will then remove the N tasks corresponding to the maximum elements in TT.

3.4 Evolutionary operations

This section introduces the mechanism of individual selection in the evolutionary process and explains how crossover and mutation operations are implemented.

3.4.1 Selection operation

In the LLM-DPEC framework, similar to traditional EC, the individual selection mechanism determines how individuals are chosen to pass their genetic material to the next generation. This selection process typically relies on the fitness of each individual, which measures their performance or suitability to the AEOSSP scenario. Generally, individuals with higher fitness are more likely to be selected for reproduction, allowing their traits to spread through the population. Various strategies can implement this, such as the roulette wheel selection, where the probability of selection is proportional to fitness, or the tournament selection, where a random subset of individuals is chosen and the fittest among them is selected. We have chosen the roulette wheel method as the selection mechanism in the LLM-DPEC framework.

3.4.2 Crossover operation

Unlike traditional EC framework, in the LLM-DPEC framework, individuals are destroy HAs, which renders traditional crossover operations inapplicable. To facilitate the crossover of these algorithms, we have created specific prompts to guide LLMs in functionally integrating

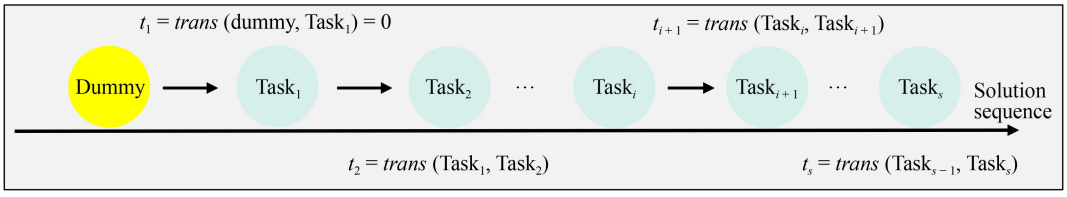

Fig. 7 Illustration of the transit time.

different individuals. For this purpose, we have designed two types of crossover prompts (C1, C2), each custom-designed to offer various strategies and directions for crossover destroy operators. The specific content is illustrated in Fig. 8. It is important to emphasize that this prompt-driven evolutionary mechanism is applied uniformly to both the LLM-generated and the expert-designed populations.

We will next explain the functions of these two crossover prompts separately.

C1: Combine functionalities: By combining existing destroy operators' capabilities, we create versatile operators with multiple functionalities, enhancing adaptability. Subsection 3.2.2 provides details on this FNLE strategy, which helps LLMs explore the solution space more effectively.

C2: Simulated crossover operation: Drawing from evolutionary computation, combining features of existing destroy operators through crossover operations produces efficient results while preserving genetic diversity. This approach guides LLMs to explore various crossover operators, improving the search for optimal algorithms.

3.4.3 Mutation operation

Similar to the crossover prompts, four types of mutation prompts (M1, M2, M3, M4) have been specifically designed to offer various strategies and directions for mutation destroy operators. The details are shown in Fig. 9.

Next, the functions of these four mutation prompts will be explained separately.

M1: In-depth understanding of variables: Prompt the LLM to analyze variable types (especially AttitudeAngle and VTW) to select a parent operator that guides the creation of a distinct operator. This deeper understanding enables generation of more innovative destroy operators.

M2: Randomness: Guide the LLM to incorporate randomness into the algorithmic logic when necessary, as randomness is crucial for maintaining robustness across scenarios.

M3: Use of more indicators: Encourage the LLM to integrate additional data indicators into the selected parent to design more complex destroy operators. Examples include profit rate per VTW, conflict degree within VTW, time intervals, and statistical measures (mean/variance of attitude angles).

M4: Parameter tuning: Direct the LLM to fine-tune the selected parent's parameters to optimize both efficiency and effectiveness.

3.5 Evaluation

Fig. 10 shows the simple demonstration for the evaluation of the individuals. As shown in Figure 10, within the evolution of the LLM-DPEC framework, new destroy operators created through crossover or mutation are integrated into the ALNS algorithm. The ALNS algorithm is then used to solve specific AEOSSP scenarios, returning a profit value, which is the sum of all task priorities in the scheduling plan. This profit value serves as the fitness value of the new destroy operator. A higher fitness value

Crossover Prompts

C1: Carefully read the functionality of the provided destroy functions step by step, and then integrate their functionalities to design a new destroy function.

C2: Simulate the crossover operation in evolutionary computation on the provided destroy operators and design a new and efficient one.

Fig. 8 Prompts for crossover operation.

Mutation Prompts

M1: Firstly, You must read the given Variable Explanations and Supporting Classes sections step by step and gain a thorough understanding of the variables' types. Secondly, Generate new 'destroy' function that are as much different as possible from the given 'destroy' function(s) by making full use of the information I provided, especially the information of AttitudeAngle and VTW.

M2: Add randomness to the provided function at an appropriate position in order to better explore the solution space. If you think it is unnecessary, do not add it.

M3: Firstly, You must read the given Variable Explanations and Supporting Classes sections step by step and gain a thorough understanding of the variables' types. Secondly, Using more indicators than the provided one to design a new 'des' function by making full use of the information I provided, especially the information of AttitudeAngle and VTW.

M4: Fine-tuning the parameters of the provided 'des' function to obtain a more effective one.

Fig. 9 Prompts for mutation operation.

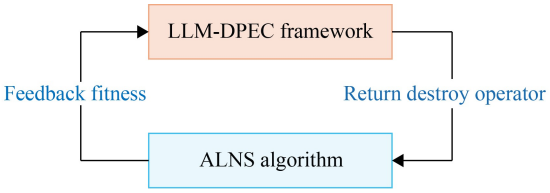

Fig. 10 A simple demonstration for the evaluation of the individual.

indicates better performance of the destroy operator.

4 Computational experiments

To verify the effectiveness of the proposed LLM-DPEC-assisted evALNS and its superiority over standard LLM-EC-assisted and expert-designed ALNS for AEOSSP, we conducted comprehensive experiments using randomly generated scenarios. This also includes ablation studies validating the effectiveness of dual populations and the FNLE strategy we proposed. All experiments were performed using Python 3.12 on a laptop with an i7-11700K processor (3.6 GHz) and 32 GB RAM.

4.1 Experiment setup

A significant challenge in AEOSSP research is the absence of universally recognized public benchmark data sets. This is largely due to the high problem specificity, where different satellite systems, orbital parameters, and mission objectives lead to variations in problem models and constraints across studies. Therefore, to ensure the reproducibility and relevance of our experiments, we employ a high-fidelity scenario generator described by (Chen et al., 2024a) The generation method is as follows:

The scheduling period spans from April 20, 2013, at 00:00:00 to April 21, 2013, at 00:00:00, covering one day and approximately 15 to 16 orbits. The satellite's orbital parameters include: semi-major axis of 7,141,701.7 m, orbital eccentricity of 0.000627, orbital inclination of 98.5964 degrees, argument of perigee of 95.5069 degrees, right ascension of the ascending node of 342.307 degrees, and mean anomaly of 125.2658 degrees. These parameters fully describe the satellite's orbital characteristics and position.

Tasks are uniformly distributed within a latitude range of 3° to 53° and a longitude range of 74° to 133°. The number of tasks ranges from 100 to 400, increasing in increments of 25, resulting in 13 different scenarios.

Each task in a scenario is defined by six attributes: task index, longitude, latitude, execution time, and priority. Execution time and priority are randomly selected integers ranging from 1 to 30 s and 1 to 10, respectively, where 1 is the lowest priority and 10 the highest. Latitude and longitude coordinates are generated uniformly within the given ranges. Based on the satellite's orbital parameters and the tasks' geographic coordinates, the required time-dependent attitude angles and the corresponding VTWs for each task are pre-calculated. These pre-calculated data serve as fixed inputs for our scheduling model. The core decision of our algorithm is then to select the optimal start time u_i for each scheduled task within its VTW, which in turn determines the specific angles used for calculating transition times. For the experiments, we refer to scenarios by the number of tasks, e.g., Scenario 100 contains 100 tasks.

The key algorithm parameters are listed in Table 2:

We selected two types of LLMs to evaluate their impact on the performance enhancement of evALNS. These are DeepSeek-v3 from DeepSeek and Qwen-Long from Alibaba. It is important to note that during the evaluation process, only one destroy operator generated by the LLM-DPEC or standard LLM-EC frameworks is chosen at a time. However, when comparing algorithms, we select the top five destroy operators identified after completing all iterations.

4.2 Experimental results

4.2.1 Results analysis

Next, we compare the expert-designed ALNS, the evALNS assisted by the LLM-EC framework (Liu et al., 2023a), and the evALNS generated by our proposed LLM-DPEC framework. We also consider ablation experiments on the dual population and FNLE strategy. The expert-designed ALNS refers to the ALNS with expert individuals described in Section 3.3.2. All comparison data represent average yields from 50 runs of each algorithm, where yield refers to the ratio of the current solution's profit to the total profit of the current scenario. We define "Gap" as the relative profit improvement of evALNS compared to expert-designed ALNS:

Table 2 Algorithm parameter configuration

Algorithm	Parameter	Value
LLM-DPEC	Maximum generations	20
	Iterations per generation	10
	Population size	10
	Crossover probability	1
	Mutation probability	0.5
	Number of parents for crossover	2
	Elite exchange frequency	Every 5 iterations
	Number of elite individuals exchanged	2
ALNS	Max iterations (for evaluation)	100
	Max iterations (for algorithm comparison)	1000
	Others	Per as (Liu et al., 2017)

$$Gap = \frac{P_{\text{evALNS}} - P_{\text{ALNS}}}{P_{\text{evALNS}}}, \quad (18)$$

where P_{evALNS} and P_{ALNS} represent the profits obtained by evALNS and ALNS algorithms, respectively.

Tables 3 and 4 display the average profits obtained from 30 iterations of expert-designed ALNS and evALNS assisted by the LLM-EC and LLM-DPEC frameworks. Specifically, Table 3 presents results generated by DeepSeek-v3, while Table 4 exhibits results from Qwen-Long. We emphasize the profit gap (Eq. (18)) of our proposed evALNS compared to expert-designed ALNS. Additionally, Fig. 11 illustrates the performance difference between the LLM-EC and LLM-DPEC frameworks when two distinct LLMs are used.

Table 3 Comparison of the basic ALNS, evALNS under the LLM-EC and LLM-DPEC frameworks when using DeepSeek-v3

Scenario	Profit			Gap	
	Basic	LLM-EC	LLM-DPEC	LLM-EC	LLM-DPEC
100	458.12	461.98	460.9	0.84%	0.60%
125	621.9	645.16	646.42	3.61%	3.79%
150	727.86	768.9	785	5.34%	7.28%
175	834.32	863.8	890.22	3.41%	6.28%
200	784.96	819.26	835.2	4.19%	6.02%
225	816.94	930.14	932.5	12.17%	12.39%
250	857.04	935.14	949.4	8.35%	9.73%
275	907.54	966.32	972.26	6.08%	6.66%
300	741.06	934.2	945.06	20.67%	21.59%
325	1010.02	1049.62	1109.14	3.77%	8.94%
350	860.66	970.31	972.28	11.30%	11.48%
375	806	863.1	880.52	6.59%	8.46%
400	992.3	1014.32	1068.1	2.17%	7.10%

Table 4 Comparison of the basic ALNS, evALNS under the LLM-EC and LLM-DPEC frameworks when using Qwen-Long

Scenario	Profit			Gap	
	Basic	LLM-EC	LLM-DPEC	LLM-EC	LLM-DPEC
100	458.12	460.74	460.44	0.57%	0.50%
125	621.9	639.58	647.1	2.76%	3.89%
150	727.86	748.64	780.96	2.78%	6.80%
175	834.32	861.96	846.76	3.21%	1.47%
200	784.96	794.68	834.7	1.22%	5.96%
225	816.94	843.84	926	3.19%	11.78%
250	857.04	914.56	929.06	6.29%	7.75%
275	907.54	966.86	967.78	6.14%	6.22%
300	741.06	814.26	938.54	8.99%	21.04%
325	1010.02	1081.42	1104.2	6.60%	8.53%
350	860.66	877.94	931.1	1.97%	7.57%
375	806	860.17	877.22	6.30%	8.12%
400	992.3	1005.72	1048.8	1.33%	5.39%

For clarity, LLM-EC and LLM-DPEC refer to the evolutionary ALNS (evALNS) algorithm implemented within the respective frameworks. "Basic" refers to the expert-designed ALNS. Thus, "Profit (LLM-EC)" indicates the total profit obtained by evALNS within the LLM-EC framework, while "Gap (EC)" signifies the percentage enhancement achieved by evALNS compared to expert-designed ALNS, as defined in Eq. (18).

Based on the provided data and Fig. 11, a comprehensive analysis shows that DPEC consistently outperforms EC in profit improvements over ALNS in both the DeepSeek-v3 and Qwen-Long models. In Table 3, LLM-DPEC exhibits an average gap of approximately 8.48%, compared to LLM-EC's 6.75%. Additionally, the gap enhancement under the LLM-DPEC framework is 20.4% relative to the LLM-EC framework. Similarly, in Table 4, DPEC maintains a higher average gap of 7.31%, compared to LLM-EC's 3.95%. The gap enhancement under the LLM-DPEC framework is 45.96% relative to the LLM-EC framework. This evidence indicates the superior performance of the LLM-DPEC framework in enhancing the profit across all scenarios.

The findings support the effectiveness and necessity of creating the heuristic function in ALNS using LLMs to solve the AEOSSP. Furthermore, they demonstrate that the integration of dual population and FNLE strategies into DPEC significantly enhances the effectiveness of evALNS.

Additionally, Table 3 generally shows larger gap values when compared to the data in Table 4, suggesting that the performance of the destroy operator generated by GPT is superior to those generated by Qwen-Long. While LLM-EC offers more stable and consistent gap improvements, LLM-DPEC exhibits greater variability, particularly in scenarios such as 300 and 325, but consistently delivers higher profit enhancements. This indicates that the DPEC framework has a significant advantage in breaking the ALNS algorithm's tendency to get trapped in local optima when solving the AEOSSP. This characteristic between stability and performance highlights LLM-DPEC's potential as a powerful optimization framework for solving the AEOSSP using evALNS. Overall, LLM-DPEC's advanced strategies make it a superior choice for evolving the destroy operator in ALNS, efficiently addressing the AEOSSP and demonstrating significant advantages regardless of the selected LLM.

Figure 12 illustrates the convergence curves of fitness by using the selected LLMs within the LLM-DPEC framework for scenario 400.

The convergence curves for fitness (scenario profit) in scenario 400 across different LLMs reveal a rapid increase in fitness values during the initial generations, followed by gradual stabilization. The maximum,

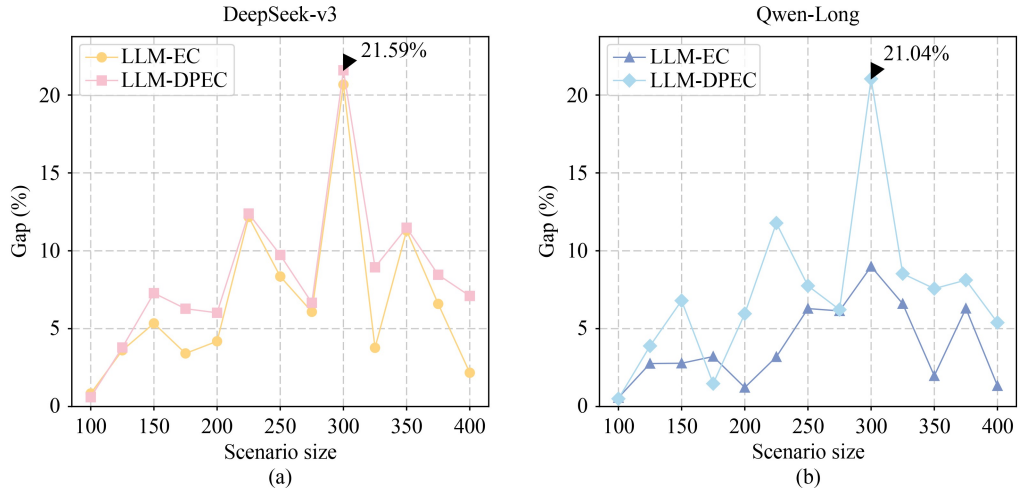

Fig. 11 Gaps for the evALNS under the LLM-EC and LLM-DPEC frameworks by using DeepSeek-v3 (a) and Qwen-Long (b) models.

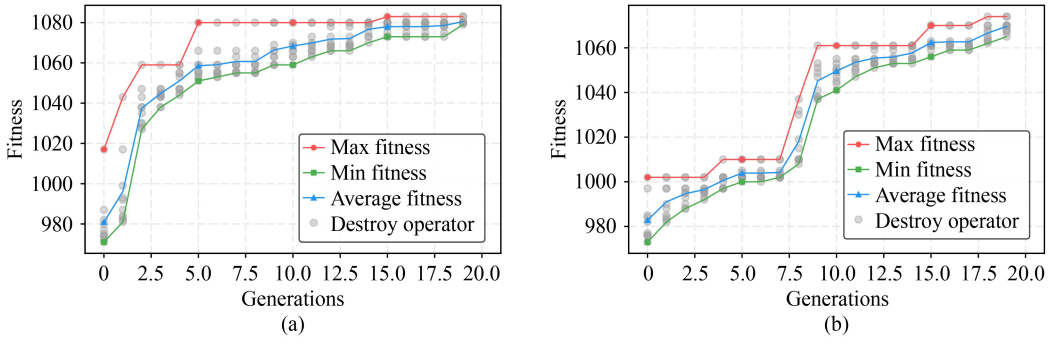

Fig. 12 The converging curves of fitness under the LLM-DPEC framework for scenario 400 when using different LLMs: (a) DeepSeek-v3 and (b) Qwen-Long.

minimum, and average fitness values demonstrate significant growth early on, indicating a swift improvement in population fitness. This growth eventually levels off, showcasing the algorithm's strong convergence and stability. Initially, the Destroy Operator promotes diversity within the population, displaying a wide range of fitness values. As generations progress, the fitness values cluster more tightly, indicating convergence toward higher fitness solutions. This trend highlights the algorithm's shift from exploration in the early stages to exploitation and optimization in later stages, effectively enhancing overall population fitness. Overall, the curves demonstrate the algorithm's effectiveness in improving fitness in the given scenario.

4.2.2 Analysis of computational cost

A key consideration for our LLM-DPEC framework is the computational cost of evolving destroy operators offline. To quantify this, we estimate that evolving the operators for a single 200-task scenario takes approximately 1.34 h (4810 s), as shown in Table 5. This calculation includes the total time for LLM API calls (for initialization, crossover, and mutation) and the many ALNS fitness evaluations (100 iterations per evaluation) throughout the entire 20-generation process.

This one-time, offline investment is justified by the significant and recurring gains it delivers in online performance. The investment produces a qualitatively superior algorithm; unlike the expert-designed ALNS, which faces diminishing returns with extended runtimes, evALNS uses evolved operators to unlock more profitable solution spaces, a foundational improvement reflected in the 8.48% performance uplift. Crucially, our cross-validation study (Section 4.4) confirms that these operators are not overfitted and generalize effectively to unseen problems. Therefore, for practical applications such as daily satellite scheduling, the initial time investment is rapidly amortized. Each subsequent operational run benefits from the superior heuristics, translating the upfront training cost into consistent and substantial long-term profit gains.

4.3 Ablation study

To assess the impact of the dual population and FNLE

Table 5 Computational cost estimation for Heuristic evolution on Scenario 200

Phase	Component	Calculation details	Estimated time (s)
Phase 1: initialization	LLM population Init.	(10 individuals × 1s/LLM call) + (10 individuals × 10 s/ALNS eval)	110
	Expert population Init.	10 individuals × 10 s/ALNS eval	100
	Subtotal		210
Phase 2: evolution	Per generation (for both populations)		
	LLM population (Crossover & Mutation)	10 crossover calls (10 s) + 5 mutation calls (5 s) = 15 LLM calls	15
	Expert Population (Crossover & Mutation)	10 crossover calls (10 s) + 5 mutation calls (5 s) = 15 LLM calls	15
	Fitness evaluation (for new individuals)	(10 LLM pop + 10 Expert pop) × 10 s/ALNS eval	200
	Cost per generation	15s + 15s + 200 s	230
	Total for 20 generations	20 generations × 230 s/generation	4,600
Total estimated cost	Phase 1 + Phase 2	210 s + 4600 s	4810

strategies on the LLM-EC framework, we will conduct several ablation experiments in this section. First, we will remove the FNLE strategy from the LLM-EC framework, specifically eliminating the natural language embedding strategy during the evolution of the destroy operators. We will then compare the gap in this scenario using data for the expert-designed ALNS consistent with Tables 3 and 4. It is important to note that when we remove the FNLE strategy for ablation study, the crossover prompt C1 we designed also becomes ineffective. To ensure fairness, we employ a crossover prompt strategy as outlined in Reference (Liu, et al., 2023a), as its effectiveness has already been validated in heuristic design. The details of this prompt are shown as follows:

New C1 prompt: Please identify the common elements of the ideas presented in the provided functions, and then, based on these common elements, create a different function that incorporates these ideas by adding some new components.

Table 6 presents the results of the ablation study by using DeepSeek-v3. The symbol "LLM-EC-1" refers to the evALNS algorithm evolved by the LLM-EC framework that incorporates the FNLE strategy, while the symbol "LLM-EC-0" refers to the evALNS algorithm evolved by the LLM-EC framework without the FNLE strategy. Similar to the previous data tables, the first two columns of the table represent the profit values for the corresponding scenarios, while the last two columns indicate the relative performance improvement gap percentages.

Figure 13 shows the optimization gap percentages between LLM-EC with and without the FNLE strategy in various scenarios. The size and color of the bubbles represent the magnitude and direction of the performance differences.

Data from Table 4 and Fig. 13 demonstrates that the FNLE strategy generally enhances evALNS algorithm performance across most scenarios, evidenced by higher profit values and gap percentages compared to the variant without FNLE. Although performance gains in scenarios 300 and 350 are less significant, Tables 3, 4, and 5 show consistently high performance improvements for scenarios 300 and 325. This indicates these scenarios are particularly responsive to destroy operator evolution in the LLM-EC framework, suggesting the new C1 prompt performs effectively in these cases.

Table 6 Comparison of the evALNS under the LLM-EC framework with and without the FNLE strategy

Scenario	Profit		Gap	
	LLM-EC-1	LLM-EC-0	LLM-EC-1	LLM-EC-0
100	461.98	462	0.84%	0.84%
125	645.16	642.52	3.61%	3.21%
150	768.9	760.4	5.34%	4.28%
175	863.8	843.66	3.41%	1.11%
200	819.26	812.54	4.19%	3.39%
225	930.14	921.9	12.17%	11.39%
250	935.14	925.64	8.35%	7.41%
275	966.32	958.92	6.08%	5.36%
300	934.2	971.5	20.67%	23.72%
325	1049.62	1030.54	3.77%	1.99%
350	970.31	991.6	11.30%	13.20%
375	863.1	856.72	6.59%	5.92%
400	1014.32	1008.6	2.17%	1.62%

4.4 Sensitivity analysis

In this section, we will conduct a sensitivity analysis of the parameters within the LLM-DPEC framework. Our analysis will focus on two key aspects: first, we will keep all other parameter settings constant while setting the evolutionary generation to 30. Second, we will maintain the other parameters and adjust the mutation rate to 0.2. Subsequently, we will re-evolve the destroy operators and apply them to ALNS to develop the new evALNS. This modified approach will be used to solve

the AEOSSP and calculate the corresponding profit for

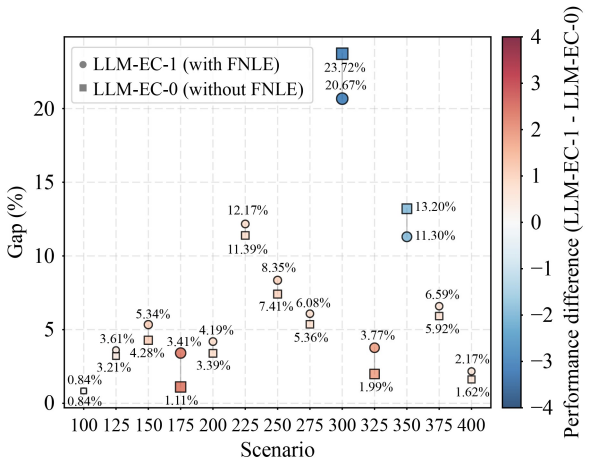

Fig. 13 Comparison of gap percentages with and without FNLE strategy under LLM-EC framework by using DeepSeek-v3.

each scenario. Finally, we will compare the resulting gaps. The table below presents the gap percentage data for evALNS across three frameworks: the basic LLM-DPEC, the LLM-DPEC with the number of iterations set to 30 (referred to as iter30), and the LLM-DPEC with the mutation rate adjusted to 0.2 (referred to as mut0.2). Table 7 details the data of sensitivity analysis.

According to the data in Table 7, the average gap is 10.26% for the basic configuration, 10.30% for iter30, and 10.26% for mut0.2. These results show that while increasing iterations (iter30) slightly improves evALNS performance, the difference is very small. Similarly, adjusting the mutation rate to 0.2 (mut0.2) has almost no effect. This suggests that parameter tuning only marginally enhances the LLM-DPEC framework's performance in solving AEOSSPs. Given the small differences, it is important to carefully select parameter settings based on specific scenarios to balance performance and computational costs, such as token usage.

4.5 Cross validation

In the LLM-DPEC framework, each scenario ultimately evolves a population of 10 destroy operators, which are interchangeable across different scenarios. While the previous sections demonstrated the effectiveness of our framework for specific instances, this section aims to verify the generalization performance of these operators across various scenarios. To this end, we conduct a cross-validation experiment by employing the K-fold cross validation method (Wong and Yeh, 2020). The specific cross-validation process is outlined in Algorithm 2 below.

In Algorithm 2, we first subdivide the set of scenarios into 5-folds: [100, 125, 150], [175, 200, 225], [250, 275, 300], [325, 350], and [375, 400]. We then iterate through these 5-folds to access the full scenario validation. In each iteration, we designate the current fold as the target fold and randomly select five scenarios from the remaining scenarios to form the resource fold. Next, we extract the evolved destroy operators from each scenario within the resource fold and integrate these operators into evALNS to address the scenarios in the target fold. For instance, if the target fold is [100, 125, 150], we can randomly select five scenarios from [175, 200, 225, 250, 275, 300, 325, 350, 375, 400], such as [200, 300, 325, 225, 400]. We then extract the most effective evolved destroy operators from these scenarios for use in evALNS and solve the three scenarios in the target fold [100, 125, 150] 50 times to obtain the corresponding average profits. Finally, we conclude the cross-validation process for all scenarios, thereby outlining the complete methodology of 5-fold cross-validation.

Table 7 Comparison of profits and gaps under different scenarios for LLM-DPEC_iter30, LLM-DPEC_mut0.2, and LLM-DPEC with basic configurations

Scenario	Profit		Gap		
	iter30	mut0.2	iter30	mut0.2	basic
200	813.84	815.42	6.12%	3.74%	6.02%
225	930.82	920.76	12.23%	11.28%	12.39%
250	952.3	951.5	10.00%	9.93%	9.73%
275	971.52	983.2	6.59%	7.70%	6.66%
300	946.38	979.76	21.70%	24.36%	21.59%
325	1110.62	1108	9.06%	8.84%	8.94%
350	970.52	946.54	11.32%	9.07%	11.48%
375	878.7	895.92	8.27%	10.04%	8.46%
400	1072.02	1071.36	7.44%	7.38%	7.10%

Table 8 shows the gap percentages data of the cross validation results for all scenarios under the LLM-DPEC framework by using DeepSeek-v3.

Figure 14 shows that the gap percentages for the evALNS under the original LLM-DPEC, the LLM-EC Framework and the LLM-DPEC for cross validation.

The results presented in Tables 3 and 8, as well as in Fig. 14, show that the gap percentages obtained through cross-validation are generally comparable to those from the original LLM-DPEC framework, and often surpass those of the LLM-EC framework. In particular, for instances such as 225, 300, and 375, the cross-validation gaps achieved by the LLM-DPEC framework are slightly higher than those of the original framework. Furthermore, the average gap of evALNS under cross-validation is 8.29%, close to the original LLM-DPEC average of 8.48%, whereas the evALNS gap under the LLM-EC framework is noticeably lower at 6.75%.

These findings lead to two main insights. First, the evolved destroy operators within the LLM-DPEC framework exhibit strong generalization capabilities; using them across different scenarios yields consistently

Algorithm 2 5-fold cross vlidation for the destroy operators of different scenarios

Inputs: Scenarios set D, all the evolved destroy operators associated with various scenarios

Outputs: Cross-validation profits

Divide D into 5 folds: F_1, F_2, F_3, F_4, F_5, such that $D = \bigcup_{k=1}^{5} F_k$ and $F_i \cap F_j = \varnothing$ $(\forall i, j \in \{1,2,...,5\}, i \neq j)$

for $k = 1$ to 5 **do:**

Step 1: Determine Target and Source Scenarios

$T_k \leftarrow F_k$ // Target scenarios

$S_k \leftarrow$ Randomly select 5 scenarios from $D \setminus F_k$ // Source scenarios

Step 2: Evaluate

Extract the best destroy operator from each scenario in S_k

Apply the extracted destroy operators in evALNS to solve T_k scenarios 50 times

Apply mutations to obtain $O_{m,t}^{(k)}$ from $O_{c,t}^{(k)}$

Step 3: Record Profits

Obtain the average profits P_k for scenarios in T_k in iteration k

end for

Step 4: Collect All Profits

Get the profits set $P \leftarrow \bigcup_{k=1}^{5} P_k$

Return P

Table 8 Cross validation profits and gaps comparison for different scenarios

Scenario	Profit	Gap		
	Cross-validation	Cross-validation	LLM-EC	LLM-DPEC
100	462	0.84%	0.84%	0.60%
125	644.06	3.44%	3.61%	3.79%
150	777.42	6.37%	5.34%	7.28%
175	878.82	5.06%	3.41%	6.26%
200	835.51	6.05%	3.39%	6.02%
225	951.33	14.13%	12.17%	12.39%
250	944.94	9.30%	8.35%	9.73%
275	974.12	6.83%	6.08%	6.66%
300	967.8	23.43%	20.67%	21.59%
325	1099.34	8.12%	3.77%	8.94%
350	952.95	9.68%	11.30%	11.48%
375	906.04	11.04%	6.62%	8.46%
400	1027.46	3.42%	2.17%	7.10%

competitive results, although slightly below those of the original setting. Second, both the cross-validated LLM-DPEC framework and the original LLM-DPEC consistently outperform the LLM-EC framework. Taken together, these results confirm the effectiveness and versatility of the proposed LLM-DPEC framework.

4.6 Algorithm comparison

To address the need for a more comprehensive performance evaluation, we conducted an extensive comparative study to benchmark our LLM-assisted approach against several state-of-the-art (SOTA) and baseline algorithms. This section presents the setup, results, and analysis of this comparison, providing convincing evidence of our algorithm's superiority.

4.6.1 Competitor algorithms and parameters setting

We selected three widely recognized algorithms to serve as competitors, representing different levels of heuristic sophistication. Furthermore, to demonstrate the power of our LLM-evolved operators, we tested them within both the standard ALNS framework and a more advanced hybrid ALNS framework. The algorithms are defined as follows:

Genetic Algorithm (GA) (Xhafa et al., 2012): A classic population-based metaheuristic. We implemented a standard GA with tournament selection, single-point crossover, and random mutation operators, which serves as a baseline to demonstrate the problem's complexity for traditional evolutionary methods.

Hill climbing with Expert Operators (HC) (Yao et al., 2023): A simple but effective local search algorithm. This version uses the same set of five expert-designed destroy operators as our baseline ALNS.

ALNS with Tabu search (ALNS/TFP) (He et al., 2020): This represents a strong SOTA competitor, inspired by the successful hybrid approach in (He et al., 2020). It enhances the expert-designed ALNS with a Tabu Search mechanism to avoid cycling, making it a

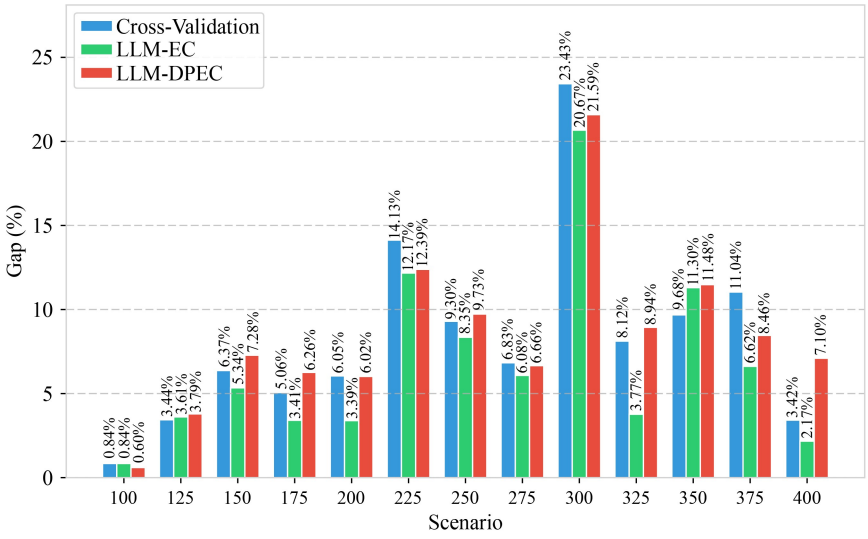

Fig. 14 Gap percentages for the evALNS under the original LLM-DPEC framework and the LLM-DPEC framework for cross validation.

highly robust and powerful heuristic for the AEOSSP.

evALNS (Ours): This is our proposed method as described in the paper, which uses the standard ALNS framework but replaces the expert-designed destroy operators with the superior ones evolved by our LLM-DPEC framework.

evALNS-TFP (Ours): This is a hybrid algorithm designed to test the ultimate potential of our evolved operators. We replaced the human-designed operators in the strong ALNS/TFP framework with our LLM-evolved operators. A superior performance here would demonstrate that our framework cannot only create a good standalone algorithm but can also significantly enhance existing SOTA methods.

The parameters for the comparative experiments were set to ensure a fair evaluation. All local search-based algorithms were executed for a maximum of 1,000 iterations, with their other parameters configured as specified in their original references. The Genetic Algorithm (GA) was configured with a population size of 100 for 500 generations, utilizing a crossover and mutation rate of 1.0 for both.

4.6.2 Comparative results and analysis

All algorithms were run 30 times on the same 13 scenarios described in Section 4.1, and the average profit values are reported. The experimental results are summarized in Table 9 and visualized in Fig. 15.

The results presented in Table 9 and Fig. 15 lead to several key insights:

1) The Genetic Algorithm (GA) consistently yields the lowest profit, confirming that the complex, constrained nature of the AEOSSP is challenging for standard population-based methods without specialized operators. The

Table 9 Comparative results of different algorithms across all scenarios

Scenario	GA	HC	ALNS/TFP	evALNS	evALNS-TFP
100	258.1	455	454.2	460.9	456.5
125	375.3	614.6	626.9	646.4	638.1
150	400.4	744.9	759.3	785	783.1
175	517.3	839.9	877.1	890.2	900.7
200	496.4	812.9	855.9	835.2	880.5
225	527.1	944.6	958.4	932.5	1010.6
250	564.6	988.2	990.5	949.4	1037.9
275	544.8	970.9	990.6	972.3	1087.4
300	534.2	1027.6	1024	945.1	1083.9
325	610.4	1087	1094	1109.1	1256.5
350	544.3	1026.7	1066	972.3	1115.6
375	565.1	1048.5	1066	880.5	1115.3
400	668	1131.4	1152	1068.1	1289.8

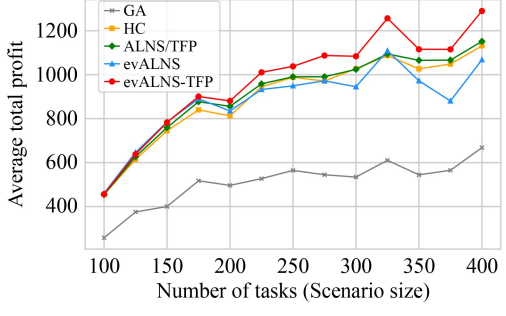

Fig. 15 Performance comparison of comparing algorithms across different problem scales.

Hill Climbing (HC) algorithm performs reasonably well, but is clearly outperformed by the more sophisticated

ALNS-based methods, highlighting the benefit of an adaptive search framework.

2) Our proposed evALNS demonstrates competitive performance against the strong SOTA baseline, ALNS/TFP. In several smaller to mid-sized instances (e.g., 125, 150, 175, 325 tasks), evALNS achieves higher profits, validating that the heuristics discovered by our LLM-DPEC framework are highly effective on their own.

3) The most significant finding is the outstanding performance of the evALNS-TFP algorithm. It consistently outperforms all other competitors, including the strong ALNS/TFP baseline, across nearly all scenarios. This result is crucial because it proves that our LLM-evolved destroy operators are not just effective, but are demonstrably superior to the set of carefully crafted, expert-designed operators used in the SOTA framework.

4) The performance gap between evALNS-TFP and ALNS/TFP widens as the problem size increases (from task 225 onwards). For the largest 400-task scenario, evALNS-TFP achieves an average profit of 1289.8, which is a 12.0% improvement over the 1151.7 achieved by ALNS/TFP. This indicates that the heuristics generated by our framework are more adept at navigating the larger and more complex search spaces of difficult scenarios. This synergy—combining a SOTA framework with our superior, LLM-generated operators—unlocks a new level of performance that neither component could achieve alone.

In summary, this comprehensive comparison provides clear and convincing evidence of the value of our proposed approach. The LLM-DPEC framework is capable of automatically discovering destroy heuristics that are powerful enough to not only compete with but also significantly enhance state-of-the-art algorithms for the complex AEOSSP.

4.7 Demonstration of the evolved destroy operator

To crystallize the novelty of the evolved heuristics, we present a detailed breakdown of the seven distinct criteria employed by the optimal destroy operator evolved for Scenario 375. Unlike traditional expert-designed operators, which typically focus on a single metric (e.g., removing tasks with the lowest priority or longest transition time), our LLM-evolved operator performs a sophisticated, multi-criteria evaluation. As detailed in Table 10, these criteria span multiple dimensions of the problem space, from task value and temporal constraints to complex satellite attitude dynamics.

The true innovation lies not only in the identification of novel criteria like Attitude Stability and VTW Feasibility but in the operator's ability to synthesize these seven distinct signals. It uses a dynamic weighting scheme (as detailed in Appendix A Fig. A2) to aggregate these scores, creating an adaptive policy that is far more powerful than any single, static rule. This demonstrates that our LLM-DPEC framework does not merely recombine existing ideas but discovers fundamentally new and more effective heuristic structures for complex optimization problems.

Additionally, we extracted the functional descriptions of all destroy operators generated by the LLM-DPEC framework using DeepSeek-v3 for 14 scenarios and created a word cloud, as shown in Fig. 16. The figure below highlights several key points. High-frequency vocabulary, including "attitude," "duration," and "priority," indicates that the final evolved destroy operators effectively utilize data related to attitude angles, task execution duration, and task priority, thereby enhancing the effectiveness of scheduling solutions. The terms "dynamic," "random," "scoring," and "weighted" suggest that the optimal destroy operator involves complex processes and mechanisms for dynamic adjustment and

Table 10 Multi-criteria evaluation framework of the LLM-Evolved destroy operator on Scenario 375

Criterion	Description	Focus area	Comparison to expert Heuristics
Priority score	A decaying function of the task's priority.	Task value	An enhancement of the basic Priority Destroy, integrated within a broader scoring system.
VTW overlap score	Measures the degree of conflict between a task's VTWs and those of other scheduled tasks.	Temporal conflict	A more integrated version of Conflict Destroy, considering overlaps within the current solution, not just with unscheduled tasks.
Observation duration	The task's required execution time.	Task requirement	A fundamental parameter, but here used as one of many factors rather than a standalone rule.
Attitude angle score	The average attitude maneuver required to transition from other tasks to the current task.	Inter-task maneuver	A sophisticated evolution of Transit-time Destroy, quantifying the average "cost" of including this task in a sequence.
Attitude stability score	Measures the total attitude variation required within the task's own VTWs.	Intra-task maneuver	Novel Concept: This metric, which assesses internal maneuver difficulty, has no direct parallel in expert heuristics.
VTW duration score	The total duration of all available VTWs for a task.	Temporal opportunity	A more nuanced version of Opportunity Destroy, measuring total available time rather than just the count of windows.
VTW feasibility score	The ratio of total available VTW duration to the required observation time.	Task feasibility	Novel Concept: A powerful derived metric that quantifies scheduling flexibility, a feature not explicitly used by expert rules.

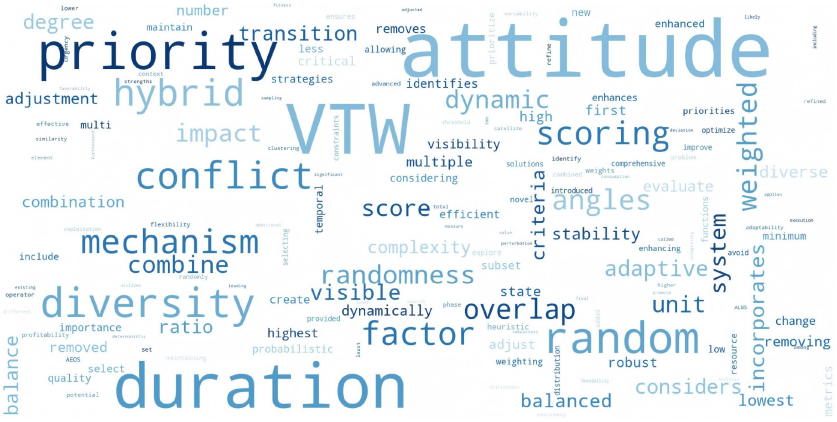

Fig. 16 Word cloud of the functional descriptions of all created destroy operators.

optimization. These operators incorporate elements of dynamism and randomness to adapt to changing scheduling requirements; while scoring and weighting are used to evaluate and optimize scheduling solutions based on specific criteria. Terms such as “balance,” “impact,” and “transition” indicate that the most effective destroy operators balance multiple factors. This reflects the role of the proposed prompt M3, which aims to introduce additional indicators. The term “transition” underscores that the operators ultimately incorporate the concept of transit time, which is derived from the information introduced in the expert population.

5 Conclusions

Our research demonstrates that incorporating LLMs into ALNS has led to the development of innovative destroy operators specifically for the AEOSSP. In all experimental scenarios, the evALNS with LLM-generated destroy operators achieved an average profit increase of 8.48% compared to the ALNS with manually designed destroy operators. This integration underscores the transformative potential of LLMs in designing HAs, indicating that evALNS not only enhances traditional ALNS but also uses LLMs to innovate and advance HAs for the AEOSSP. The success of this methodology highlights the critical role of adaptive design in improving algorithmic performance and the potential that LLMs have in broadly revolutionizing heuristic algorithm applications.

Furthermore, developing a comprehensive framework like the proposed LLM-DPEC with FNLE, which demonstrates robust generalization capabilities across various scenarios, is a pivotal goal for future research. To guide this direction, it is important to consider its broader applicability and limitations. Our methodology is best suited for complex combinatorial optimization problems that are amenable to metaheuristic approaches with decomposable operators (e.g., VRP, JSS), possess a well-defined solution structure, and feature intricate constraints. The co-evolutionary nature of the DPEC framework performs optimally when a baseline of expert-designed heuristics is available to guide the LLM. This positions our knowledge-fusion approach as a distinct alternative to other emerging paradigms, such as the self-adaptive prompt evolution proposed by (Ye et al., 2025), particularly for domains where expert knowledge is crucial but not inherently present in the LLM’s pre-training data.

However, realizing this potential requires addressing inherent limitations, including the complexity of prompt engineering, the need for a robust sandbox to validate LLM-generated code, the computational overhead of the evolutionary discovery process, and the reliance on pre-existing domain knowledge to seed the expert population.

Our approach emphasizes the importance of incorporating advanced AI techniques to significantly improve optimization outcomes. By focusing on the creation of specialized destroy operators, we aim to extend the capabilities of ALNS, ultimately facilitating more effective solutions in complex scheduling problem domains. Future work could explore reducing the dependency on expert knowledge and automating prompt refinement to enhance the framework’s autonomy and broaden its applicability to a wider range of optimization challenges.

Appendix A

A.1 Details of initialization prompt

Figure A1 displays the detailed prompt we designed for generating the initial population

We selected one of the most effective destroy operators evolved by our LLM-DPEC framework for a detailed analysis. This heuristic demonstrates a sophisticated, multi-faceted logic that surpasses typical human-designed operators. The full Python code for this heuristic is provided in Fig. A2.

The procedure of this evolved destroy operator is formally outlined in Algorithm A1.

LLM Population Initialization Prompt

```
# Problem Description:
The provided Python function with explanations implements the 'destroy' function for the Adaptive
Large Neighborhood Search (ALNS) algorithm which is specifically designed to tackle the Agile Earth
Observing Satellite (AEOS) scheduling problem. It heuristically removes a part of the current scheduling
plan which defines a tasks sequence , referred to as 'self.plan'.
# Example Destroy Function:
{{codes}
# Supporting Classes:
class VTW:
    """
    Represents a Visible Time Window for a 'task'.
    """
    def __init__(self, bt, et, dt, attitude_angle_list):
        self.begin_time = bt  # Begin time (list): [year, month, day, hour, minute, second]
        self.end_time = et    # End time (list): [year, month, day, hour, minute, second]
        self.vtw_dtime = dt   # Duration of the VTW (int)
        self.attitude_angle_list = attitude_angle_list  # (list): A list of 'AttitudeAngle' objects
class AttitudeAngle:
    """
    Represents a specific satellite attitude (orientation) at a given time point (in seconds).
    """
    def __init__(self, time_point, pitch, roll, yaw):
        self.time_point = time_point  # Time point (list): [year, month, day, hour, minute, second]
        self.pitch = pitch          # Pitch angle (float): Degree, range: [-45.0, 45.0]
        self.roll = roll          # Roll angle (float): Degree, range: [-45.0, 45.0]
        self.yaw = yaw            # Yaw angle (float): Degree, set to 0.0.
# Variable Explanations:
- self.plan:
  - A nested list representing the current scheduling plan.
  - Each element in 'self.plan' represents a 'task'.
  - Each 'task' is a list containing the following items, in order:
    1. Task Index (int): An integer from 0 to (number_of_tasks - 1) uniquely identifying the task.
    2. Visible Time Windows (list): A list of 'VTW' objects, each representing a time window
       when the task can be scheduled.
    3. Priority (int): An integer from 1 to 10 indicating the task's priority (10 being the highest).
    4. Observation Duration (int): The required duration of the task's observation, measured in
       time unit (seconds).
    5. Starting Observation Time (AttitudeAngle): An 'AttitudeAngle' object representing the
       desired starting time and orientation for the observation.
    6. Ending Observation Time (AttitudeAngle): An 'AttitudeAngle' object representing the
       desired ending time and orientation for the observation.
  - self.destroyed_task_number (int): The pre-defined number of tasks that need to be removed from
    the current scheduling plan.
# What You Need To Do:
- You must read the given Variable Explanations and Supporting Classes sections step by step and
  gain a thorough understanding of the variables' types.
- Your task is to assist me in designing a new and innovative destroy function by making full use of
  the information I provided, especially the information of AttitudeAngle and VTW.
# Requirements To You:
- Your Python function must retain the same stylistic approach, function name, variable names,
  parameter and returns as the provided destroy functions.
- You MUST replace your innovative ideas in the Functionality Part in your code like the provided one(s).
- Encapsulate your newly designed destroy function as shown in the provided 'destroy' function(s),
  ensuring that the imported packages remain unchanged and avoiding any additional outputs.
- If you define helper functions, make sure they are defined inside class Destroy.
- Do not redefine the Supporting Classes.
- All the output code must be complete; Do not omit any code.
```

Fig. A1 Detailed prompts for population initialization using LLMs.

Definitions of the relevant symbols used in the algorithm and analysis are provided in Table A1.

The heuristic's logic is structured into three main phases: multi-criteria evaluation, score aggregation with dynamic weighting, and probabilistic selection.

Phase 1: Multi-criteria evaluation

For each task $\tau_i \in \Pi$, the operator computes a vector of seven scores. Each score function f_c quantifies a specific aspect of the task:

1) Priority Score (f_{p}): A simple decaying priority.

$$f_{\mathrm{p}}(\tau_i) = p_i \cdot \delta, \text{ where } \delta = 0.9.$$

2) VTW Overlap Score (f_{o}): Measures how much a task's VTWs conflict with others.

Python codes for the optimal heuristic for scenario 375

Part One

```python
def des(self):  # destroy operator
    """
    Functionality: This function integrates and enhances the features from both provided
    destroy methods to create a more robust and adaptive destroy operator. It considers the
    priority, VTW overlap, observation duration, attitude angle differences, VTW duration,
    and attitude stability to determine the probability of a task being destroyed. It introduces
    a dynamic weighting system that adjusts the importance of each factor based on the
    current state of the plan. Additionally, it incorporates a decay factor for the priority score
    to reduce the impact of high-priority tasks over time, and a normalization factor for the
    overlap score to ensure fair comparison across different tasks. The function also
    includes a smoothing factor to avoid extreme probabilities and a threshold to ensure a
    minimum number of tasks are always considered for destruction.
    Parameter:
        - self
    Returns:
        - new_plan: new scheduling plan.
        - destroyed_task_ids: a List of indexes of the destroyed tasks.
    """

    def calculate_overlap_score(task):
        """
        Helper function to calculate the overlap score for each task.
        """
        vtw_list = task[1]  # Get the list of VTWs for the task
        overlap_score = 0
        for vtw in vtw_list:
            for other_task in self.plan:
                if task != other_task:
                    for other_vtw in other_task[1]:
                        if max(vtw.begin_time, other_vtw.begin_time) <
                                min(vtw.end_time, other_vtw.end_time):
                            overlap_score += 1
        return overlap_score

    def calculate_priority_score(task):
        """
        Helper function to calculate the priority score for each task.
        """
        priority = task[2]  # Priority is the third element in the task list
        decay_factor = 0.9  # Decay factor to reduce the impact of high-priority tasks over time
        return priority * decay_factor

    def calculate_observation_duration(task):
        """
        Helper function to calculate the observation duration for each task.
        """
        return task[3]  # Observation duration is the fourth element in the task list

    def calculate_attitude_angle_difference(task1, task2):
        """
        Helper function to calculate the attitude angle difference between two tasks.
        """
        start_angle1 = task1[4]
        start_angle2 = task2[4]
        end_angle1 = task1[5]
        end_angle2 = task2[5]

        pitch_diff_start = abs(start_angle1.pitch - start_angle2.pitch)
        roll_diff_start = abs(start_angle1.roll - start_angle2.roll)
        yaw_diff_start = abs(start_angle1.yaw - start_angle2.yaw)

        pitch_diff_end = abs(end_angle1.pitch - end_angle2.pitch)
        roll_diff_end = abs(end_angle1.roll - end_angle2.roll)
        yaw_diff_end = abs(end_angle1.yaw - end_angle2.yaw)

        total_diff = (pitch_diff_start + roll_diff_start + yaw_diff_start + pitch_diff_end
                      + roll_diff_end + yaw_diff_end) / 6
        return total_diff

    def calculate_attitude_angle_score(task):
        """
        Helper function to calculate the attitude angle score for each task.
        """
        angle_score = 0
        for other_task in self.plan:
            if task != other_task:
                angle_score += calculate_attitude_angle_difference(task, other_task)
        return angle_score

    def calculate_vtw_duration_score(task):
        """
        Helper function to calculate the VTW duration score for each task.
        """
        vtw_list = task[1]  # Get the list of VTWs for the task
        total_duration = sum(vtw.vtw_dtime for vtw in vtw_list)
        return total_duration
```

Part Two

```python
    def calculate_attitude_stability_score(task):
        """
        Helper function to calculate the attitude angle stability score for each task.
        """
        vtw_list = task[1]  # Get the list of VTWs for the task
        stability_score = 0
        for vtw in vtw_list:
            pitch_diff = max([angle.pitch for angle in vtw.attitude_angle_list]) -
                         min([angle.pitch for angle in vtw.attitude_angle_list])
            roll_diff = max([angle.roll for angle in vtw.attitude_angle_list]) -
                        min([angle.roll for angle in vtw.attitude_angle_list])
            stability_score += pitch_diff + roll_diff
        return stability_score

    def calculate_vtw_feasibility(task):
        """
        Helper function to calculate the VTW feasibility score for each task.
        """
        total_vtw_duration = sum(vtw.vtw_dtime for vtw in task[1])
        return total_vtw_duration / task[3]

    def calculate_dynamic_weights():
        """
        Helper function to calculate dynamic weights for each factor
        based on the current state of the plan.
        """
        # Example dynamic weights (can be adjusted based on the current state of the plan)
        weights = {
            'priority': 1.0 + (0.1 * len(self.plan)),
            'overlap': 0.8 + (0.1 * len(self.plan)),
            'observation_duration': 0.5 + (0.1 * len(self.plan)),
            'attitude_angle': 0.7 + (0.1 * len(self.plan)),
            'vtw_duration': 0.6 + (0.1 * len(self.plan)),
            'attitude_stability': 0.5 + (0.1 * len(self.plan)),
            'vtw_feasibility': 1.2 + (0.1 * len(self.plan))
        }
        return weights

    def calculate_destroy_probability(task):
        """
        Helper function to calculate the probability of a task being destroyed.
        """
        overlap_score = calculate_overlap_score(task)
        priority_score = calculate_priority_score(task)
        observation_duration = calculate_observation_duration(task)
        attitude_angle_score = calculate_attitude_angle_score(task)
        vtw_duration_score = calculate_vtw_duration_score(task)
        attitude_stability_score = calculate_attitude_stability_score(task)
        vtw_feasibility = calculate_vtw_feasibility(task)

        weights = calculate_dynamic_weights()

        # Normalize overlap score to ensure fair comparison
        max_overlap_score = max([calculate_overlap_score(t) for t in self.plan])
        normalized_overlap_score = overlap_score / max_overlap_score
                                   if max_overlap_score > 0 else 0

        total_score = (weights['priority'] * (10 - priority_score)) + \
                      (weights['overlap'] * normalized_overlap_score) + \
                      (weights['observation_duration'] * (observation_duration / 1000)) + \
                      (weights['attitude_angle'] * (attitude_angle_score / 10)) + \
                      (weights['vtw_duration'] * (vtw_duration_score / 1000)) - \
                      (weights['attitude_stability'] * (attitude_stability_score / 10)) - \
                      (weights['vtw_feasibility'] * vtw_feasibility)

        # Smoothing factor to avoid extreme probabilities
        smoothing_factor = 0.01
        return 1 / (1 + total_score + smoothing_factor)

    # Calculate the probabilities for all tasks
    probabilities = [calculate_destroy_probability(task) for task in self.plan]

    # Normalize the probabilities
    total_probability = sum(probabilities)
    normalized_probabilities = [p / total_probability for p in probabilities]

    # Ensure a minimum number of tasks are always considered for destruction
    min_destroyed_tasks = max(1, int(len(self.plan) * 0.1))
    destroyed_task_ids = random.choices(range(len(self.plan)), weights=normalized_probabilities,
                                        k=max(min_destroyed_tasks, self.destroyed_task_number))

    # Create the new plan by excluding the destroyed tasks
    new_plan = [self.plan[i] for i in range(len(self.plan)) if i not in destroyed_task_ids]

    return new_plan, destroyed_task_ids
```

Fig. A2 Demonstration of an evolved optimal destroy operator scenario 375.

$$f_{\mathrm{o}}(\tau_i,\Pi)=\sum_{\tau_j\in\Pi,j\neq i}\sum_{W\in\mathcal{W}_i}\sum_{W'\in\mathcal{W}_j}\mathbb{I}(W\cap W'\neq\varnothing)$$

3) Observation Duration (f_{od}): The task's observation duration.

$$f_{\mathrm{od}}(\tau_i)=d_i$$

4) Attitude Angle Score (f_{a}): The average attitude maneuver required from other tasks to τ_i. Let $\Delta A(\tau_i,\tau_j)$ be the L_1 norm of the angle difference.

$$f_{\mathrm{a}}(\tau_i,\Pi)=\frac{1}{|\Pi|-1}\sum_{\tau_j\in\Pi,j\neq i}\Delta A(\tau_j,\tau_i)$$

5) VTW Duration Score (f_{vd}): The total available time for the task.

Algorithm A.1 LLM-evolved multi-Criteria destroy operator

Inputs: Current scheduling plan Π, number of tasks to destroy d

Outputs: A new plan Π' and the set of destroyed tasks Π_d

for each task $\tau_i \in \Pi$ **do:**

Calculate a set of seven criterion scores: $\{f_c(\tau_i, \Pi) | c \in C\}$ where $C = \{\text{priority}, \ldots, \text{feasibility}\}$

Calculate dynamic weights $w_c(|\Pi|)$ for each criterion $c \in C$

Normalize scores where necessary, yielding $f'_c(\tau_i, \Pi)$

Compute the composite score $S(\tau_i)$ as a weighted sum: $S(\tau_i) = \sum_{c \in C_+} w_c(|\Pi|) f'_c(\tau_i) - \sum_{c \in C_-} w_c(|\Pi|) f'_c(\tau_i)$

Calculate the destroy probability $P(\tau_i)$ using an inverse relationship: $P(\tau_i) = (1 + S(\tau_i) + \varepsilon)^{-1}$

end for

Normalize the probabilities for all tasks in the plan: $P_{\text{norm}}(\tau_i) = \frac{P(\tau_i)}{\sum_{\tau_j \in \Pi} P(\tau_j)}$

Determine the number of tasks to destroy: $k = \max(1, \lceil 0.1 \cdot |\Pi| \rceil, d)$

Sample k tasks to form the destroyed set Π_d from Π according to the distribution P_{norm}

Construct the new plan: $\Pi' \leftarrow \Pi \setminus \Pi_d$

Return Π', Π_d

Table A1 Mathematical notation for the destroy operator

Symbol	Description
Π	The current scheduling plan, a set of tasks $\tau_1, \tau_2, \ldots$
τ_i	The i-th task in the plan
p_i	The intrinsic priority of task τ_i
d_i	The required observation duration for task τ_i
$\mathcal{W}_i$	The set of Visible Time Windows (VTWs) for task τ_i. A single VTW is $W \in \mathcal{W}_i$
$\mathbf{A}_i(t)$	The attitude angle vector (pitch, roll, yaw) for task τ_i at time t
C	The set of evaluation criteria. $C+$ and $C-$ are subsets of positive and negative criteria
$f_c(\tau_i, \Pi)$	The score of task τ_i for criterion c
w_c	The dynamic weight of $c \in C$
$S(\tau_i)$	The final composite score of task τ_i, indicating its resistance to destruction
ε	A small positive smoothing factor
$P(\tau_i)$	The unnormalized probability of destroying task τ_i
$\mathbb{I}(\cdot)$	The indicator function, which is 1 if the condition is true, and 0 otherwise

$$f_{\text{vd}}(\tau_i) = \sum_{W \in \mathcal{W}_i} |W|$$

6) Attitude Stability Score (f_s): Measures attitude variation within a task's VTWs. Let $\mathbf{A}_i^W$ be the set of attitude angles within a VTW W.

$$f_s(\tau_i) = \sum_{W \in \mathcal{W}_i} \left(\max_{\mathbf{a} \in \mathbf{A}_i^W} a_{\text{pitch}} - \min_{\mathbf{a} \in \mathbf{A}_i^W} a_{\text{pitch}} + \max_{\mathbf{a} \in \mathbf{A}_i^W} a_{\text{roll}} - \min_{\mathbf{a} \in \mathbf{A}_i^W} a_{\text{roll}} \right)$$

7) VTW Feasibility Score (f_v): *The ratio of available VTW duration to required observation time.

$$f_v(\tau_i) = \frac{f_{vd}(\tau_i)}{d_i}$$

Phase 2: Score aggregation with dynamic weighting

This phase combines the individual scores into a single composite score $S(\tau_i)$. A key feature is the dynamic weighting scheme, where the weight w_c for each criterion c is a linear function of the current plan size $|\Pi|$:

$$w_c(|\Pi|) = a_c + b_c \cdot |\Pi|$$

where a_c and b_c are constants evolved by the LLM. The final score $S(\tau_i)$ is a linear combination, with f'_c representing the normalized score:

$$S(\tau_i) = \sum_{c \in C_+} w_c(|\Pi|) f'_c(\tau_i) - \sum_{c \in C_-} w_c(|\Pi|) f'_c(\tau_i)$$

Here, criteria in C_- (e.g., feasibility f_v) increase a task's destruction likelihood, while those in C_+ increase its

preservation value.

Phase 3: Probabilistic Selection

The composite score $S(\tau_i)$ is transformed into a destruction probability $P(\tau_i)$ via an inverse relationship, ensuring high-scoring tasks are less likely to be removed:

$$P(\tau_i|S(\tau_i)) = (1 + S(\tau_i) + \varepsilon)^{-1}.$$

After computing this for all tasks, the probabilities are normalized to form a valid probability distribution:

$$P_{\text{norm}}(\tau_i) = \frac{P(\tau_i)}{\sum_{\tau_j \in \Pi} P(\tau_j)}.$$

Finally, k tasks are sampled from Π according to the distribution defined by P_{norm}.

In summary, the mathematical properties of this heuristic are profound. It moves beyond simple rules by implementing:

1) High-Dimensional Feature Space: It operates in a seven-dimensional feature space $\{f_c\}$ to evaluate each task.

2) Adaptive Policy: The decision logic, parameterized by $w_c(|\Pi|)$, is not static but adapts to the current state of the solution.

3) Sophisticated Trade-offs: The structure of $S(\tau_i)$ embodies complex trade-offs, such as penalizing tasks with high feasibility to create more promising search opportunities.

4) Controlled Stochasticity: The final selection, governed by the distribution P_{norm}, balances deterministic evaluation with stochastic exploration.

A human engineer would be highly unlikely to manually derive this specific set of seven features, their complex interactions in the $S(\tau_i)$ formula, and the state-dependent nature of the weights $w_c(|\Pi|)$. This operator exemplifies the power of our LLM-DPEC framework to discover non-obvious, mathematically sophisticated, and highly effective heuristics for complex combinatorial optimization problems like the AEOSSP.

Competing Interests The authors declare that they have no competing interests.

References

Chatterjee A, Tharmarasa R (2024). Multi-stage optimization framework of satellite scheduling for large areas of interest. Advances in Space Research, 73(3): 2024–2039

Chen C, Li L, Du Y, Yao F, Xing L (2025a). A hybrid learning-assisted multi-parallel algorithm for a large-scale satellite-ground networking optimization problem. Engineering Management, 12(4): 1157–1174

Chen M, Du Y, Tang K, Xing L, Chen Y, Chen Y (2024a). Learning to construct a solution for the agile satellite scheduling problem with time-dependent transition times. IEEE Transactions on Systems, Man, and Cybernetics. Systems, 54(10): 5949–5963

Chen X, Gao L, Zhang M, Chen C, Yan S (2024b). Spectral-spatial adversarial multi-domain synthesis network for cross-scene hyperspectral image classification. IEEE Transactions on Geoscience and Remote Sensing, 62: 1–16

Chen X, Zhang M, Liu Y (2024c). Target detection with spectral graph contrast clustering assignment and spectral graph transformer in hyperspectral imagery. IEEE Transactions on Geoscience and Remote Sensing, 62: 1–16

Chen Y, Zhang H, Li C, Chi B, Chen X, Wu J (2025b). Large language model empowered smart city mobility. Engineering Management, 12(1): 201–207

Ding N, Qin Y, Yang G, Wei F, Yang Z, Su Y, Hu S, Chen Y, Chan C M, Chen W, Yi J, Zhao W, Wang X, Liu Z, Zheng H T, Chen J, Liu Y, Tang J, Li J, Sun M (2023). Parameter-efficient fine-tuning of large-scale pre-trained language models. Nature Machine Intelligence, 5(3): 220–235

Du Y, Wang L, Xing L, Yan J, Cai M (2019). Data-driven heuristic assisted memetic algorithm for efficient inter-satellite link scheduling in the BeiDou navigation satellite system. IEEE/CAA Journal of Automatica Sinica, 8(11): 1800–1816

Du Y, Wang T, Xin B, Wang L, Chen Y, Xing L (2020). A data-driven parallel scheduling approach for multiple agile earth observation satellites. IEEE Transactions on Evolutionary Computation, 24(4): 679–693

Du Y, Xing L, Chen Y (2022). Satellite scheduling engine: The intelligent solver for future multi-satellite management. Frontiers of Engineering Management, 9(4): 683–688

Fan W, Ding Y, Ning L, Wang S, Li H, Yin D, Chua T S, Li Q (2024). A survey on rag meeting llms: Towards retrieval-augmented large language models. In: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining: 6491–6501

Gao Y, Xiong Y, Gao X, Jia K, Pan J, Bi Y, Dai Y, Sun J, Wang H (2023). Retrieval-augmented generation for large language models: A survey. Preprint at arXiv. arXiv:2312.10997

Han C, Gu Y, Wu G, Wang X (2023). Simulated annealing-based heuristic for multiple agile satellites scheduling under cloud coverage uncertainty. IEEE Transactions on Systems, Man, and Cybernetics. Systems, 53(5): 2863–2874

He L, De Weerdt M, Yorke-Smith N (2020). Time/sequence-dependent scheduling: the design and evaluation of a general purpose tabu-based adaptive large neighbourhood search algorithm. Journal of Intelligent Manufacturing, 31(4): 1051–1078

He L, Liu X, Laporte G, Chen Y, Chen Y (2018). An improved adaptive large neighborhood search algorithm for multiple agile satellites scheduling. Computers & Operations Research, 100: 12–25

Herrmann A, Schaub H (2023). Reinforcement learning for the agile earth-observing satellite scheduling problem. IEEE Transactions on Aerospace and Electronic Systems, 59(5): 5235–5247

Jia S, Deng L, Zhao Q, Chen Y (2023). An adaptive large neighborhood search heuristic for multi-commodity two-echelon vehicle routing problem with satellite synchronization. Journal of Industrial and Management Optimization, 19(2): 1187–1210

Kaddour J, Harris J, Mozes M, Bradley H, Raileanu R, McHardy R (2023). Challenges and applications of large language models.

Preprint at arXiv. arXiv:2307.10169

Kasneci E, Sessler K, Küchemann S, Bannert M, Dementieva D, Fischer F, Gasser U, Groh G, Günnemann S, Hüllermeier E, Krusche S, Kutyniok G, Michaeli T, Nerdel C, Pfeffer J, Poquet O, Sailer M, Schmidt A, Seidel T, Stadler M, Weller J, Kuhn J, Kasneci G (2023). ChatGPT for good? On opportunities and challenges of large language models for education. Learning and Individual Differences, 103: 102274

Li R, Gong W, Wang L, Lu C, Zhuang X (2023). Surprisingly popular-based adaptive memetic algorithm for energy-efficient distributed flexible job shop scheduling. IEEE Transactions on Cybernetics, 53(12): 8013–8023

Li R, Wang L, Gong W, Ming F (2025). An evolutionary multitasking memetic algorithm for multiobjective distributed heterogeneous welding flow shop scheduling. IEEE Transactions on Evolutionary Computation, 29(6): 2287–2298

Liu F, Liu Y, Shi L, Huang H, Wang R, Yang Z, Zhang L, Li Z, Ma Y (2024a). Exploring and evaluating hallucinations in LLM-powered code generation. Preprint at arXiv. arXiv:2404.00971

Liu F, Tong X, Yuan M, Lin X, Luo F, Wang Z, Lu Z, Zhang Q (2024b). Evolution of heuristics: towards efficient automatic algorithm design using large language lodel. In: Forty-first International Conference on Machine Learning. PMLR 235, 32201–32223

Liu F, Tong X, Yuan M, Zhang Q (2023a). Algorithm evolution using large language model. Preprint at arXiv. arXiv:2311.15249

Liu S, Chen C, Qu X, Tang K, Ong Y S (2024c). Large language models as evolutionary optimizers. In: 2024 IEEE Congress on Evolutionary Computation (CEC), 1–8

Liu X, Laporte G, Chen Y, He R (2017). An adaptive large neighborhood search metaheuristic for agile satellite scheduling with time-dependent transition time. Computers & Operations Research, 86: 41–53

Liu Y, Roberto B, Zhou J, Yu Y, Zhang Y, Sun W (2023b). Efficient feasibility checks and an adaptive large neighborhood search algorithm for the time-dependent green vehicle routing problem with time windows. European Journal of Operational Research, 310(1): 133–155

Liu Z, Liu J, Liu X, Yang W, Wu J, Chen Y (2024d). Knowledge-assisted adaptive large neighbourhood search algorithm for the satellite–ground link scheduling problem. Computers & Industrial Engineering, 192: 110219

Ou J, Xing L, Yao F, Li M, Lv J, He Y, Song Y, Wu J, Zhang G (2023). Deep reinforcement learning method for satellite range scheduling problem. Swarm and Evolutionary Computation, 77: 101233

Pan Z, Wang L, Dong C, Chen J F (2024). A knowledge-guided end-to-end optimization framework based on reinforcement learning for flow shop scheduling. IEEE Transactions on Industrial Informatics, 20(2): 1853–1861

Pan Z, Wang L, Wang J, Zhang Q (2025). A bi-learning evolutionary algorithm for transportation-constrained and distributed energy-efficient flexible scheduling. IEEE Transactions on Evolutionary Computation, 29(1): 232–246

Peng G, Song G, Xing L, Gunawan A, Vansteenwegen P (2020). An exact algorithm for agile earth observation satellite scheduling with time-dependent profits. Computers & Operations Research, 120: 104946

Pluhacek M, Kazikova A, Kadavy T, Viktorin A, Senkerik R (2023). Leveraging large language models for the generation of novel metaheuristic optimization algorithms. In: Proceedings of the Companion Conference on Genetic and Evolutionary Computation, 1812–1820

Kandepi R, Saini H, George R K, Konduri S, Karidhal R (2024). Agile earth observation satellite constellations scheduling for large area target imaging using heuristic search. Acta Astronautica, 219: 670–677

Song Y, Ou J, Suganthan P N, Pedrycz W, Yang Q, Xing L (2023). Learning adaptive genetic algorithm for earth electromagnetic satellite scheduling. IEEE Transactions on Aerospace and Electronic Systems, 59(6): 9010–9025

Song Y, Suganthan P N, Pedrycz W, Yan R, Fan D, Zhang Y (2024). Energy-efficient satellite range scheduling using a reinforcement learning-based memetic algorithm. IEEE Transactions on Aerospace and Electronic Systems, 60(4): 4073–4087

Thirunavukarasu A J, Ting D S J, Elangovan K, Gutierrez L, Tan T F, Ting D S W (2023). Large language models in medicine. Nature Medicine, 29(8): 1930–1940

Vaithilingam P, Zhang T, Glassman E L (2022). Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. In: Chi Conference on Human Factors in Computing Systems, Extended Abstracts. New Orleans: ACM, 2022: 1-7

Stein N, Bäck T (2025). Llamea: A large language model evolutionary algorithm for automatically generating metaheuristics. IEEE Transactions on Evolutionary Computation, 29(2): 331–345

Wang X, Wang L, Dong C, Ren H, Xing K (2024). Reinforcement learning-based dynamic order recommendation for on-demand food delivery. Tsinghua Science and Technology, 29(2): 356–367

Wang X, Wu G, Xing L, Pedrycz W (2021). Agile earth observation satellite scheduling over 20 years: Formulations, methods, and future directions. IEEE Systems Journal, 15(3): 3881–3892

Wong T T, Yeh P Y (2020). Reliable accuracy estimates from k-Fold cross validation. IEEE Transactions on Knowledge and Data Engineering, 32(8): 1586–1594

Wu G, Xiang Z, Wang Y, Gu Y, Pedrycz W (2024a). Improved adaptiv large neighborhood search algorithm based on the two-stage framework for scheduling multiple super-agile satellites. IEEE Transactions on Aerospace and Electronic Systems, 60(5): 7185–7200

Wu J, Song B, Zhang G, Ou J, Chen Y, Yao F, He L, Xing L (2022). A data-driven improved genetic algorithm for agile earth observation satellite scheduling with time-dependent transition time. Computers & Industrial Engineering, 174: 108823

Wu J, Yao F, Song Y, He L, Lu F, Du Y, Yan J, Chen Y, Xing L, Ou J (2023). Frequent pattern-based parallel search approach for time-dependent agile earth observation satellite scheduling. Information Sciences, 636: 118924

Wu Y, Wang L, Chen J, Zheng J, Pan Z (2024b). A reinforcement learning driven two-stage evolutionary optimisation for hybrid seru system scheduling with worker transfer. International Journal of Production Research, 62(11): 3952–3971

Xhafa F, Sun J, Barolli A, Biberaj A, Barolli L (2012). Genetic algorithms for satellite scheduling problems. Mobile Information Systems,

8(4): 351–377

Yang C (2021). Innovation and development of BeiDou Navigation Satellite System (BDS) project management mode. Engineering Management, 8(2): 312–320

Yao F, Du Y, Li L, Xing L, Chen Y (2023). General modeling and optimization technique for real-world earth observation satellite scheduling. Engineering Management, 10(4): 695–709

Ye H, Xu H, Yan A, Cheng Y (2025). Large language model-driven large neighborhood search for large-scale MILP problems. In: 42nd International Conference on Machine Learning, Vienna, Austria, July 14-20, 2025

Zhang M R, Desai N, Bae J, Lorraine J, Ba J (2023). Using large language models for hyperparameter optimization. In: NeurIPS 2023 Foundation Models for Decision Making, New Orleans, LA, USA, December 10-16, 2023

Zhao W X, Zhou K, Li J, Tang T, Wang X, Hou Y, Min Y, Zhang B, Zhang J, Dong Z, Du Y, Yang C, Chen Y, Chen Z, Jiang J, Ren R, Li Y, Tang X, Liu Z, Liu P, Nie J Y, Wen J R (2023). A survey of large language models. Preprint at arXiv. arXiv:2303.18223