Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Coevolutionary genetic programming for large-scale dynamic multi-aircraft task allocation*

Ce YU†, Xianbin CAO, Bo ZHANG, Wenbo DU, Tong GUO†‡

School of Electronic and Information Engineering, State Key Laboratory of CNS/ATM, Beihang University, Beijing 100191, China

†E-mail: yuce@csaa.org.cn; guotong1997@buaa.edu.cn

Received July 30, 2025; Revision accepted Nov. 10, 2025; Crosschecked Nov. 26, 2025

Abstract: Multi-aircraft task allocation (MATA) plays a vital role in improving mission efficiency under dynamic conditions. This paper proposes a novel coevolutionary genetic programming (CoGP) framework that automatically designs high-performance reactive heuristics for dynamic MATA problems. Unlike conventional single-tree genetic programming (GP) methods, CoGP jointly develops two interacting populations, i.e., task prioritizing heuristics and aircraft selection heuristics, to explicitly model the coupling between these two interdependent decision phases. A comprehensive terminal set is constructed to represent the dynamic states of aircraft and tasks, whereas a low-level heuristic template translates developed trees into executable allocation strategies. Extensive experiments on public benchmark instances simulating post-disaster emergency delivery demonstrate that CoGP achieves superior performance compared with state-of-the-art GP and heuristic methods, exhibiting strong adaptability, scalability, and real-time responsiveness in complex and dynamic rescue environments.

Key words: Task allocation; Genetic programming (GP); Hyperheuristic; Combinatorial optimization; Learn-to-optimize

https://doi.org/10.1631/FITEE.2500540 **CLC number:** TP301.6

1 Introduction

Aircraft, including unmanned aerial vehicles (UAVs) and helicopters, play a pivotal role in modern emergency response and disaster relief operations (Guo et al., 2021; Liu SY et al., 2025). Their inherent advantages, such as rapid mobility, wide area coverage, and operational flexibility, make them indispensable in scenarios including search and rescue, firefighting, medical evacuation, and post-disaster logistics support (Recchiuto and Sgorbissa, 2018; Xing et al., 2024). The effective deployment and utilization of these aerial platforms are critical to improving response efficiency, minimizing casualties, and reducing economic losses (Yu et al., 2025).

The core challenge of effectively deploying multiple aircraft lies in the multi-aircraft task allocation (MATA) problem, which involves determining the optimal assignment relationships between the aircraft and service points, as well as the sequence in which these points are serviced (Hou et al., 2025). MATA can be regarded as variant models of vehicle routing problem (VRP), which has been proven NP-hard (Guo et al., 2025b, 2025d). Efficient allocation directly impacts mission completion time and operational effectiveness. Existing task allocation methods are broadly categorized into proactive and reactive approaches. Proactive approaches typically assume knowledge or predictions about the probability distribution of service requests and use methodologies such as robust optimization (Sengupta et al., 2024),

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (No. U2333218)

ORCID: Tong GUO, https://orcid.org/0000-0001-7514-7742

chance-constrained optimization (Ponda et al., 2012; Yang and Chakraborty, 2020), or stochastic programming to derive statistically optimal service plans (van Steenbergen et al., 2025). In contrast, reactive approaches (Gao and Xin, 2019; Jia et al., 2024; Hu et al., 2025) dynamically allocate tasks to aircraft based on real-time situational information, without relying on predefined distributions. Given the inherent unpredictability and rapidly changing environments encountered in emergency scenarios, it is typically impractical to accurately anticipate or quantify service-point distributions beforehand, making reactive approaches more suitable and practical (Peng et al., 2024).

However, traditional effective reactive methods rely heavily on heuristics crafted by human experts, such as shortest-path or shortest-service-time rules (Gao et al., 2022). While these heuristics are intuitively appealing, they may fail to fully exploit the underlying structural characteristics of the problem or to adapt adequately to dynamic and evolving conditions. Genetic programming (GP), as an automated heuristic-generation method, offers the potential to automatically discover efficient and adaptive heuristics without relying on explicit expert knowledge (Xu et al., 2024b). GP methods have demonstrated significant promise in automatically generating effective heuristics for a range of complex dynamic optimization problems, including dynamic job shop scheduling (Nguyen et al., 2017; Zhang et al., 2023; Xu et al., 2024a), arc routing (Ardeh et al., 2022; Wang et al., 2022, 2023), traffic signal control (Liao et al., 2025), and air traffic management (Guo et al., 2025c).

In MATA problems, there exist two fundamental yet interdependent decision phases: the task prioritizing heuristics (TPH), which determines the service order of task points, and the aircraft selection heuristics (ASH), which assigns specific aircraft to tasks. Decisions in each phase significantly influence the other, resulting in coupling effects that conventional GP algorithms, typically employing a single-tree representation per individual, fail to effectively capture (Zhang et al., 2018; Fan et al., 2024; Sun et al., 2024). Dual-/multi-tree GP extends expressiveness by assigning different subtrees to different decisions. However, most implementations rely on random subtree-level crossover/mutation and ignore role semantics and compatibility, frequently producing semantically misaligned offspring and weakened building blocks. Moreover, fitness is often attributed at the individual-tree level rather than to the TPH–ASH pair, weakening selection pressure toward jointly effective policies. Reinforcement learning (RL) has been applied to dynamic task assignment with promising results, yet most studies optimize a single decision (e.g., "which task to execute now?") while relying on heuristic or fixed policies for the complementary decision (e.g., "which aircraft executes the chosen task?"). This partial modeling underuses cross-decision feedback: The value of a TPH action depends on ASH, and conversely, the value of an ASH action is also influenced by TPH, making one-sided optimization suboptimal in coupled MATA settings.

To explicitly address this coupling issue, we propose a novel coevolutionary genetic programming (CoGP) framework specifically designed for large-scale and dynamic multi-aircraft task allocation problems. The key innovation of CoGP is to jointly evolve two interacting populations, i.e., TPH and ASH, and departs from conventional single-tree GP or dual-tree GP by pairwise (joint) fitness attribution that evaluates TPH–ASH synergies rather than individual trees. First, we construct comprehensive terminal sets that capture the dynamic states of both aircraft and tasks. These terminals are used within tree-based individual representations, allowing the aggregation and calculation of relevant features at each decision point. At every allocation step, the priorities of aircraft and tasks are computed and then integrated into a specially designed low-level heuristic template, which translates these priorities into executable task allocation decisions. Furthermore, rather than employing a single-tree representation to learn heuristics, our approach co-evolves two separate yet interacting populations: one dedicated to evolving effective TPH and the other to evolving ASH. The evolutionary process incorporates mutual consideration between the two populations, whereby individuals in one population are evaluated in conjunction with representative heuristics from the other. This coevolutionary mechanism guides the evolutionary search towards identifying heuristics that simultaneously perform well in both task prioritization and aircraft selection, ultimately providing robust and adaptive solutions suited for complex and dynamic emergency response environments.

Finally, we conduct experiments on public benchmark instances. The results indicate that the proposed CoGP method can learn more effective reactive heuristics compared with the state-of-the-art methods.

2 Problem formulation

In the static multi-depot drone-based emergency rescue model (Guo et al., 2025a), all customer $j \in C$ and their demands q_j are assumed known in advance. The objective in the static case is to plan drone routes from multiple depots $d \in D$ to serve all customers with minimum total arrival time (e.g. minimizing $\sum_{j\in C} T_j$, where T_j is the arrival/service completion time at customer j). The decision variable x_{ij}^k is defined as equal to 1 if the drone $k \in K$ travels along the arc from customer i to customer j, and 0 otherwise. First, the basic constraints include

$$\sum_{k\in K}\sum_{i\in N} x_{ij}^k = 1, \qquad \forall j \in C, \tag{1}$$

$$\sum_{j\in C} x_{ij}^k = \sum_{j\in N} x_{ji}^k, \qquad \forall i \in C,\ \forall k \in K, \tag{2}$$

$$\sum_{j\in C} x_{dj}^k \le 1, \qquad \forall k \in K,\ \forall d \in D, \tag{3}$$

$$\sum_{i\in C} x_{id}^k \le 1, \qquad \forall k \in K,\ \forall d \in D, \tag{4}$$

$$\sum_{j\in C} q_j \sum_{i\in N} x_{ij}^k \le Q_k, \qquad \forall k \in K. \tag{5}$$

Constraints (1)–(5) are identical to those in the static model: each customer j is served by exactly one drone (constraint (1)); flow conservation (constraint (2)) ensures that if a drone k visits a customer i (enters i), it also departs from i, thereby forming a continuous route; each drone departs from at most one depot and returns to at most one depot (constraints (3) and (4)); the total demand served by any drone k cannot exceed its payload capacity Q_k (constraint (5)).

Second, the drones have energy constraints. The distance between customers i and j is L_{ij}, with the drones flying at a constant speed v, resulting in a travel time $t_{ij} = L_{ij}/v$. The energy level of drone k when leaving customer i for customer j is represented by e_{ij}^k, while w_{ij}^k denotes the weight of supplies carried by drone k during this trip. The energy consumption of drone k for moving from customer i for customer j, considering its payload is expressed as

$$R_{ij}^k = \gamma_0 + \gamma w_{ij}^k + L_{ij}(\rho_0 + \rho w_{ij}^k), \tag{6}$$

where γ_0 represents the energy required for takeoff and landing for an empty drone, and γ denotes the additional energy needed for takeoff and landing with additional payload. Similarly, ρ_0 indicates the energy consumption per unit distance for an empty drone, while ρ accounts for the additional energy required per unit distance with an increased payload. Then, the energy-related constraints include

$$\sum_j w_{ji}^k - \sum_j w_{ij}^k = q_i, \forall i \in C,\ \forall k \in K, \tag{7}$$

$$w_{ij}^k \le W \cdot x_{ij}^k, \forall i, j \in C, \tag{8}$$

$$e_{ij}^k = E \cdot x_{ij}^k, \forall i \in D,\ \forall j \in C, \tag{9}$$

$$0 \le e_{ij}^k \le E \cdot x_{ij}^k, \forall i \in C,\ \forall j \in D, \tag{10}$$

$$\sum_{j\in C\setminus\{i\}} e_{ji}^k - \sum_{j\in C\setminus\{i\}} e_{ij}^k = \sum_{j\in C\setminus\{i\}} R_{ji}^k, \forall i \in C,\ \forall k \in K, \tag{11}$$

$$e_{ij}^k \ge R_{ij}^k, \forall i \in C,\ \forall j \in C \setminus \{i\},\ \forall k \in K. \tag{12}$$

Eq. (7) describes the change in the payload of the drone after serving a customer, which equals the demand of that customer. Constraint (8) ensures that the weight of supplies carried in any arc does not exceed the maximum payload W of the drone. Eq. (9) indicates that the battery of a drone is fully charged upon departure from the depot, with its maximum energy level represented by E. Constraint (10) enforces that the drone energy level remains within its maximum capacity at all times. Eq. (11) maintains the energy balance during drone operations. Constraint (12) ensures that the drone's remaining energy at each customer location is sufficient to reach the next customer. In the proposed formulation, aircraft energy consumption is modeled as a hard constraint to ensure mission feasibility. Each aircraft must retain sufficient residual energy to complete its assigned tasks and return to its depot safely. Task urgency, on the other hand, is represented in the objective function, which aims to minimize the overall rescue delay. Consequently, the optimization process seeks to allocate tasks in a manner that maximizes mission responsiveness under strict energy feasibility constraints, thereby achieving a balanced trade-off between timeliness and safety.

Based on the above definitions and constraints, we now extend this model to a dynamic scenario in

which not all customers are known a priori; instead, a proportion of customer requests are revealed over time. A simplified scenario is illustrated in Fig. 1. Given a specific instance $\xi \in \Xi$, the time is discretized into steps $t = 1, 2, \ldots, T^{\xi}$. Each customer $j \in C$ is associated with a random discovery time τ_j^{ξ}, which is the time step at which customer j becomes known (i.e., is revealed to the planner). At any time t, the set of known customers (available for dispatch) is $C_{\leq t} = \{j | j \in C, \tau_j^{\xi} \leq t\}$. Initially $(t = 1)$, only customers with $\tau_j^{\xi} \leq 1$ are known; additional customers appear stochastically as t increases. Once a drone has been dispatched on a route, its route is fixed and cannot be altered to accommodate later-discovered customers. Any customers revealed after a drone's departure must be served by a new route (either by a different drone or by the same drone after it completes its current route). The dynamic problem's objective remains to minimize the total customer arrival time, but now this objective must account for the uncertainty in discovery times by optimizing the expected total arrival time. In other words, we seek a routing policy that minimizes the expectation of the sum of all customers' arrival times, $\mathbb{E}_{\xi \in \Xi}\left[\sum_{j \in C} T_j(\xi)\right]$, where the expectation is taken over the stochastic revelation process of customer locations for instance ξ, i.e., $\mathcal{P}_{\xi}$.

To accommodate the dynamic aspects, we introduce new elements to the model while preserving the core structure of the static formulation. The decision variables are extended to instance-related version $x_{ij}^{k,\xi}$. In addition, the following modifications are considered: (1) a random parameter τ_j^{ξ} for each customer j, representing its discovery time; (2) a decision variable S_k^{ξ} (a nonnegative real) for each drone $k \in K$, denoting the departure time from its depot (i.e., the time step at which drone k launches its route); (3) an objective function that includes an expectation over the random discovery times. All drones and depots from the static model remain available in the dynamic model, and all original routing constraints (such as one-to-one customer assignment, route continuity, and drone capacity limits) are retained. The key new constraints ensure that no drone can service a customer before that customer has been discovered, and that a drone's route (once launched at time S_k^{ξ}) cannot include any customer revealed after S_k^{ξ}. The extended model is given below:

$$\min \quad \mathbb{E}_{\xi \in \Xi}\left[\sum_{j \in C} T_j(\xi)\right], \tag{13}$$

$$T_j(\xi) \geq T_i(\xi) + t_{ij}^{\xi} - M(1 - x_{ij}^{k,\xi}), \quad \forall \xi \in \Xi, \ \forall i, j \in C, \ \forall k \in K, \tag{14}$$

$$T_j(\xi) \geq S_k^{\xi} + t_{dj}^{\xi} - M(1 - x_{dj}^{k,\xi}), \quad \forall \xi \in \Xi, \ \forall d \in D, \ \forall j \in C, \ \forall k \in K, \tag{15}$$

$$T_j(\xi) \geq \tau_j^{\xi} \quad \forall \xi \in \Xi, \ \forall j \in C, \tag{16}$$

$$S_k^{\xi} \geq \tau_j^{\xi} \sum_{i \in N} x_{ij}^{k,\xi}, \quad \forall \xi \in \Xi, \ \forall j \in C, \ \forall k \in K, \tag{17}$$

$$x_{ij}^{k,\xi} \in \{0, 1\}, \forall \xi \in \Xi, \ \forall i, j \in N, \ \forall k \in K, \tag{18}$$

$$S_k^{\xi}, \ T_j(\xi) \geq 0, \forall \xi \in \Xi, \ \forall k \in K, \ \forall j \in C, \tag{19}$$

$$\text{Constraints (1)–(5), constraints (7)–(12),} \tag{20}$$

where M denotes a large positive number.

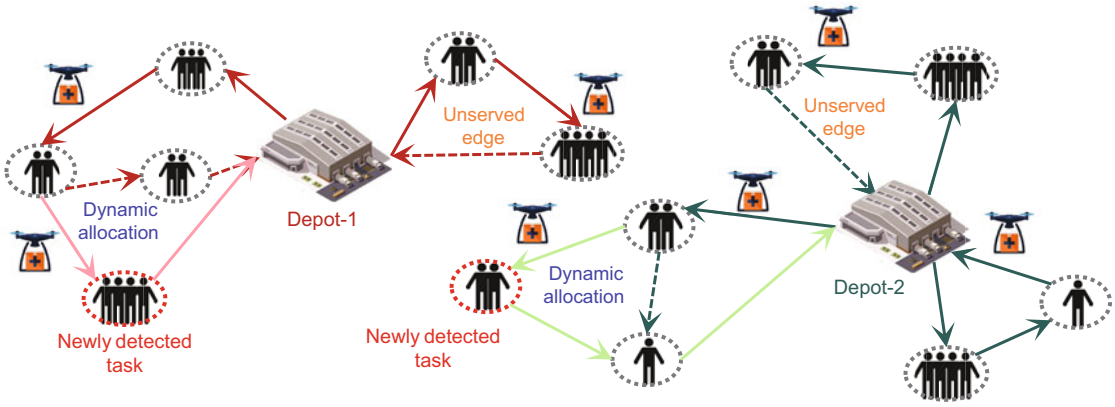

Fig. 1 Illustration of the considered scenario. Multiple aircraft depart from different depots to execute the detected tasks. During mission execution, new tasks may dynamically emerge, requiring the aircraft to reallocate their assignments to accommodate the newly detected tasks

In this model, the objective function (13) minimizes the expected total arrival time of all customers. Constraints (14) and (15) are the standard time propagation constraints, which define customer arrival times based on route sequences: If drone k travels directly from customer i to j, then $T_j(\xi)$ must be at least $T_i(\xi)$ plus the travel time t_{ij}^{ξ} (constraint (14)); if drone k's route starts at depot d and goes first to customer j, then $T_j(\xi)$ must be at least the drone's departure time S_k^{ξ} plus the travel time from depot d to j (constraint (15)). The new constraints (16) and (17) enforce the dynamic availability of customers and the no-reoptimization condition. Constraint (16) guarantees that no customer can be served before it is discovered: For each customer j, its service completion time $T_j(\xi)$ cannot occur earlier than its random discovery time τ_j^{ξ}. In other words, a drone cannot arrive at j's location before j becomes known. Constraint (17) links each drone's departure time to the discovery times of the customers it serves: For any customer j assigned to drone k (i.e., if $x_{ij}^{k,\xi} = 1$ for some predecessor i on k's route), we impose $S_k^{\xi} \geq \tau_j^{\xi}$. This means that drone k cannot launch its route until all of its assigned customers have been discovered. Equivalently, each route is planned only when the full information about all customers on that route is available, ensuring that a drone's route (once dispatched at time S_k^{ξ}) is "fixed" and will not be modified to pick up later-revealed customers. Together, constraints (16) and (17) formalize the requirement that drones cannot serve or plan for customers ahead of their revelation. Finally, constraints (18) and (19) specify the decision variable domains: $x_{ij}^{k,\xi}$ are binary variables indicating the route arcs taken (with $N = D \cup C$ denoting the set of all customers), and S_k^{ξ} and $T_j(\xi)$ are nonnegative continuous variables for departure and arrival times, respectively.

3 Solution methods

3.1 Method overview

The overall framework of the proposed CoGP is illustrated in Fig. 2. In this framework, two separate populations are co-evolved to discover high-quality reactive heuristics: the population $\mathcal{P}^{\mathrm{T}}$ is responsible for evolving TPH, while the population $\mathcal{P}^{\mathrm{A}}$ evolves ASH. At each evolutionary generation, individuals from both populations undergo genetic operations, namely parent selection, crossover, mutation, and reproduction, to produce offspring. Each offspring represents a specific tree structure, which defines how to aggregate and process terminal sets. These offspring are then instantiated as executable task allocation solutions using a low-level heuristic template.

The fitness of each offspring is evaluated based on the objective value achieved by its corresponding solution. It is important to note that each individual in the population can determine only either how to prioritize tasks or how to select aircraft. Consequently, the fitness evaluation of an individual $\mathcal{S}^{\mathrm{T}}$ in $\mathcal{P}^{\mathrm{T}}$ must account for its interaction with an individual $\mathcal{S}^{\mathrm{A}}$ from $\mathcal{P}^{\mathrm{A}}$. This interaction is facilitated by the coevolution process, which is described in detail in Section 3.4.

3.2 Tree-based representation

To enable flexible decision-making for MATA, the first population $\mathcal{P}^{\mathrm{T}}$ evolves task tree for prioritizing tasks, while the second population $\mathcal{P}^{\mathrm{A}}$ determines the aircraft selection (Fig. 3). The combined outputs of the two trees are used to construct a unified service priority score for each feasible aircraft-task pair, which in turn drives the dynamic assignment process.

3.2.1 Task tree

The task tree takes as input the real-time attributes of each unserved task and generates a task priority score. The terminal set for the task tree consists of the customer x-coordinate (CX) and y-coordinate (CY), customer demand (CD), and customer waiting time (CWT), i.e., the elapsed time since the customer request appeared.

3.2.2 Aircraft tree

The aircraft tree receives the real-time state of each aircraft as input and outputs a priority score reflecting its suitability to undertake the next task. The terminal set for the aircraft tree includes: the current aircraft x-coordinate (AX) and y-coordinate (AY), remaining aircraft capacity (CAP), remaining aircraft energy (EAP), and originating depot x- and y-coordinates (DX, DY). All the designed tree terminals are summarized in Table 1.

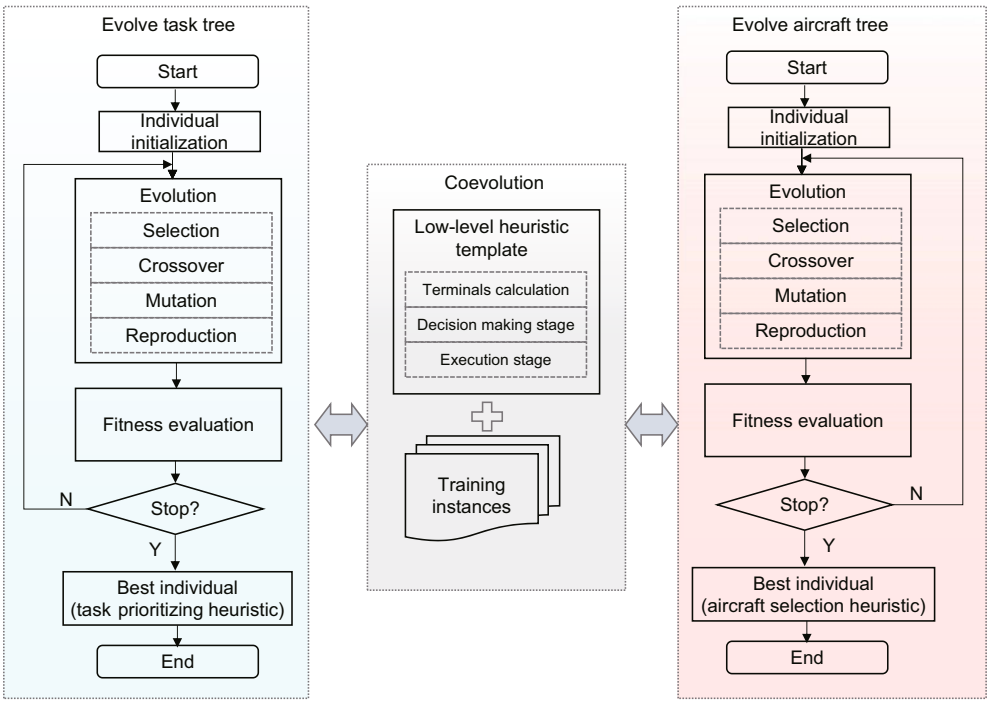

Fig. 2 Framework of the proposed CoGP algorithm

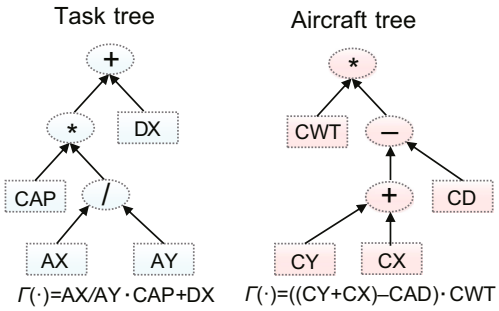

Fig. 3 Illustration of the two trees for deciding how to prioritize tasks and how to select aircraft. $*$ is a placeholder

Table 1 Terminal set for multitree GP representation

Type	Terminal	Description
Aircraft tree	AX	Aircraft's current x-coordinate
	AY	Aircraft's current y-coordinate
	CAP	Aircraft's remaining capacity
	EAP	Aircraft's remaining energy
	DX	Departure depot x-coordinate
	DY	Departure depot y-coordinate
Task tree	CX	Customer (task) x-coordinate
	CY	Customer (task) y-coordinate
	CD	Customer demand
	CWT	Customer waiting time

3.3 Low-level heuristic

Each individual in the proposed CoGP framework is composed of two trees: an aircraft tree and a task tree. These tree structures define how the corresponding terminal sets are aggregated and computed to produce priority values. We develop a two-stage low-level heuristic template, which decodes individuals into executable constructive heuristics and evaluates their effectiveness. The two stages are: (1) decision-making and (2) execution.

At the decision-making stage, the given individual is decoded into a pair of constructive heuristics that dynamically guide all aircraft in completing all tasks. At each decision timestep t, for an individual ind_i in the task selection population $\mathcal{P}^{\mathrm{T}}$, the task tree structure is denoted as $\Gamma_i(\cdot)$. For each available task j, we compute its priority by applying the task tree to the relevant terminal vector $\boldsymbol{\mathcal{X}}_{tj}^{\mathrm{TS}}$:

$$\pi_{tj}^{\mathrm{TS}} = \Gamma_i\left(\boldsymbol{\mathcal{X}}_{tj}^{\mathrm{TS}}\right), \tag{21}$$

where π_{tj}^{TS} denotes the priority score of task j at time t for individual ind_i. The task with the highest priority is then selected:

$$j^* = \arg\max_j \left(\pi_{tj}^{\mathrm{TS}}\right). \tag{22}$$

Similarly, for the same decision timestep t, the

priority of each available drone k is computed using the corresponding aircraft tree $\Psi_i(\cdot)$ from individual ind_i in the aircraft selection population $\mathcal{P}^{\text{A}}$. The aircraft-related terminal vector is denoted by $\boldsymbol{\mathcal{X}}_{tk}^{\text{AS}}$, yielding the aircraft priority:

$$\pi_{tk}^{\text{AS}} = \Psi_i\left(\boldsymbol{\mathcal{X}}_{tk}^{\text{AS}}\right). \tag{23}$$

The aircraft with the highest priority is selected:

$$k^* = \arg\max_k \left(\pi_{tk}^{AS}\right). \tag{24}$$

The selected aircraft-task assignment at decision timestep t is thus given by $\text{Plan}_t = (k^*, j^*)$. By repeating this process across all timesteps, we obtain the complete sequence of assignments: $\text{Plan} = \{\text{Plan}_1, \text{Plan}_2, \ldots, \text{Plan}_T\}$.

At the execution stage, the obtained flight plan is used to simulate and evaluate the actual performance of the individual. The fitness of each individual is computed based on the total cost incurred by executing the sequence Plan.

3.4 Coevolution process

As described above, the aircraft-task assignment sequence is jointly determined by the task tree (an individual ind_i from the population $\mathcal{P}^{\text{T}}$) and the aircraft tree (an individual ind_j from the population $\mathcal{P}^{\text{A}}$). Consequently, the two populations are intrinsically coupled through their shared objective value and cannot be evolved independently. In other words, the fitness of an individual ind_i in the task selection population $\mathcal{P}^{\text{T}}$ depends not only on its own structure but also on the paired individual ind_j from the aircraft selection population $\mathcal{P}^{\text{A}}$.

To address this interdependence, we employ a coevolutionary strategy. Specifically, two separate populations, $\mathcal{P}^{\text{T}}$ and $\mathcal{P}^{\text{A}}$, are evolved to optimize the TPH and ASH, respectively. As shown in Fig. 4, to evaluate an individual from one population, it must be paired with a representative individual from the other population to form a complete aircraft-task assignment solution. The fitness of the focal individual is then defined as the fitness of this paired solution.

During the initialization phase, representative individuals from each population are selected at random to form evaluation pairs. As evolution progresses, the best-performing individuals (according to fitness) from each subpopulation are chosen as representatives for the purpose of fitness evaluation. This coevolutionary mechanism allows both populations to adapt in response to the evolving strategies of the other, encouraging the emergence of more effective and robust aircraft-task assignment heuristics.

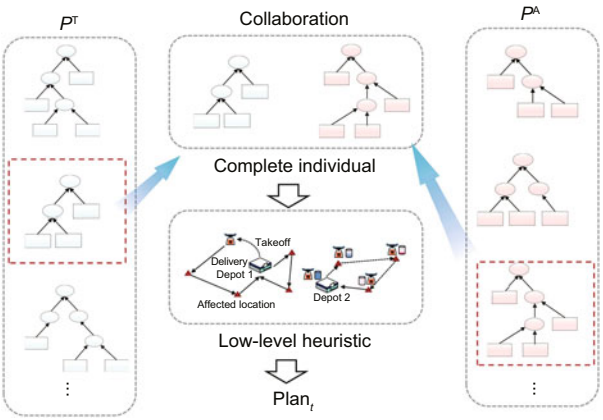

Fig. 4 Illustration of the coevolution process

3.5 Genetic operators

In the proposed CoGP, two primary genetic operators, crossover and mutation, are used to explore and exploit the heuristic space. The overall process of generating offspring using genetic operators is shown in Algorithm 1. Specifically, crossover is applied with a predefined probability p_{c} to a pair of selected parent individuals. During this operation, a crossover point is randomly chosen on each parent tree, corresponding to a particular node within the tree structure. The subtrees (tree blocks) rooted at these crossover points are then swapped between the two parent trees, resulting in two new offspring (Fig. 5). This subtree exchange allows for the effective recombination of structural and behavioral characteristics from both parents, thereby enhancing the quality of solutions within the population.

Mutation, on the other hand, is performed on an individual tree with a certain probability p_{m}. Specifically, a mutation point is randomly selected within the individual's tree structure. A newly generated random subtree is then created and replaces the original subtree rooted at the mutation point (Fig. 6). This operator introduces new genetic material into the population and helps prevent premature convergence by exploring previously unvisited regions of the heuristic space.

Note that to select parent individuals for

Algorithm 1 Breeding($\mathcal{P}, \Xi$)

Require: Population $\mathcal{P}$; training instances Ξ
Ensure: One individual of offspring S'
1: rm ← random(0,1)
2: **if** rm $< p_c$ **then**
3: // Crossover
4: $S_1, S_2 \leftarrow$ LexicaseSelection($\mathcal{P}, \Xi$)
5: $S' \leftarrow$ crossover(S_1, S_2)
6: **else if** $p_c <$rm $< p_c + p_m$ **then**
7: // Mutation
8: $S' \leftarrow$ mutation(LexicaseSelection($\mathcal{P}, \Xi$))
9: **else**
10: // Reproduction
11: $S' \leftarrow$ copy(LexicaseSelection($\mathcal{P}, \Xi$))
12: **end if**
13: **return** S'

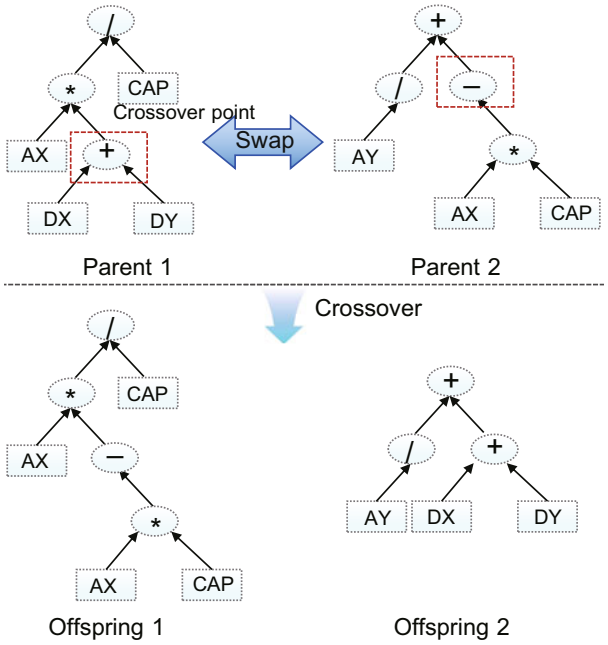

Fig. 5 Illustration of how to generate a new tree structure based on a crossover operator

crossover and individuals for mutation, we employ the Lexicase Selection operator, which promotes diversity by selecting solutions based on their performance across randomly ordered test cases.

3.6 Computational complexity analysis

After training, CoGP yields a best-performing individual that is used as a reactive heuristic for aircraft task allocation during testing. The testing-time cost is dominated by evaluating terminal features, each of which can be computed in (near) constant time. Consequently, the overall time complexity scales primarily with the number of aircraft ($|K|$) and the average number of tasks ($|C|$), i.e., $O(|K| \cdot |C|)$. In practice, the testing-phase complexity of CoGP is effectively linear in the number of aircraft and tasks, making it well-suited for real-time allocation in large-scale settings.

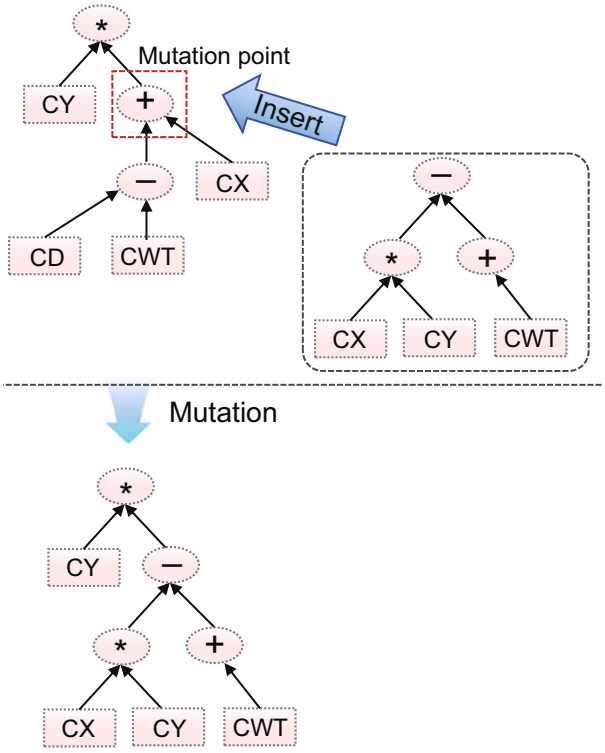

Fig. 6 Illustration of how to generate new subtree blocks using a mutation operator

4 Experiments

4.1 Dataset

The experimental evaluation is conducted on 23 publicly available benchmark instances of the multi-depot capacitated vehicle routing problem (MDVRP), which can be accessed at http://www.bernabe.dorronsoro.es/vrp/. These instances cover problem sizes ranging from 50 to 360 customers and 2 to 9 depots. To better reflect real-world operational conditions and the practical constraints of drone-based rescue missions, the benchmark instances are adapted following the methodology in Guo et al. (2025a). Specifically, all coordinates are normalized within a 0–20 km range, and customer payload demands are rescaled to lie between 0 and 2.3 kg. Each drone is assumed to have a curb weight of 6 kg and a maximum payload capacity of 3 kg. Each drone is equipped with a battery capacity

of 504 Wh. The cruising speed of the drones varies with payload: 32 km/h when fully loaded and 48 km/h when empty.

Moreover, to more accurately capture the dynamic and uncertain nature of real emergency response scenarios, the static customer information in each instance is partially withheld at the start of each simulation. Specifically, only 80% of customer locations and demands are revealed initially, while the remaining 20% are introduced progressively over time to mimic the ongoing discovery of new rescue tasks. For each benchmark instance with $|C|$ tasks, we designate $K_{\mathrm{d}} = \max\{1, \lfloor 0.2C \rceil\}$ tasks as dynamically revealed, where $\lfloor \rceil$ indicates rounding up. We uniformly sample without replacement an index set $\mathcal{S} \subset \{1, 2, \ldots, N_{\mathrm{d}}\}$ with $|\mathcal{S}| = K_{\mathrm{d}}$. For every $i \in \mathcal{S}$, the location and demand are exactly those provided by the original benchmark (no perturbation); i.e., dynamic tasks are consistent with the initial static specification. Reveal times are assigned at equal intervals over the planning horizon $[t_0, t_{\mathrm{end}}]$: if $\mathcal{S} = \{i_1, i_2, \ldots, i_{K_{\mathrm{d}}}\}$ (any fixed order), then

$$r_{i_\lambda} = t_0 + \lambda \cdot \Delta, \quad \Delta = \frac{t_{\mathrm{end}} - t_0}{K_{\mathrm{d}} + 1}, \quad \lambda = 1, 2, \ldots, K_{\mathrm{d}},$$

where r_{i_λ} denotes the revealed timestep of customer i_λ.

Thus, dynamic arrivals are uniform in time. All non-selected tasks $\{1, 2, \ldots, N_{\mathrm{d}}\} \setminus \mathcal{S}$ are available at t_0. A summary of the detailed characteristics for all benchmark instances employed in this study is provided in Table 2.

Table 2 Instance characteristics

Instance index	Instance name	Number of tasks	Number of depots	Number of aircraft
1	P01_50_4	50	4	16
2	P02_50_4	50	4	8
3	P03_75_5	75	5	15
4	P04_100_2	100	2	16
5	P05_100_2	100	2	10
6	P06_100_3	100	3	18
7	P07_100_4	100	4	16
8	P08_249_2	249	2	28
9	P09_249_3	249	3	36
10	P10_249_4	249	4	32
11	P11_249_5	249	5	30
12	P12_80_2	80	2	10
13	P13_80_2	80	2	9
14	P14_80_2	80	2	8
15	P15_160_4	160	4	20
16	P16_160_4	160	4	19
17	P17_160_4	160	4	18
18	P18_240_6	240	6	30
19	P19_240_6	240	6	28
20	P20_240_6	240	6	26
21	P21_360_9	360	9	45
22	P22_360_9	360	9	40
23	P23_360_9	360	9	35

4.2 Experimental settings

The CoGP algorithm for learning reactive scheduling heuristics is implemented using the distributed evolutionary algorithms in Python (DEAP) framework (Fortin et al., 2012). The function set for both trees includes basic arithmetic operators $\{+, -, \times, \div\}$ (protected division) and min/max operators. Conditional functions such as if–then–else are also included to enable complex priority heuristics. This function set is commonly used in GP for allocation/scheduling (e.g., job-shop scheduling, arc routing, and robot task allocation). During initialization (ramped half-and-half), functions are sampled uniformly from the function set under type safety and arity constraints. During evolution, function operator usage emerges from selection. Parameter settings are chosen according to conventional practices in the literature (Nguyen et al., 2013; Liu YX et al., 2020). Specifically, the population size is set to 1000, and the algorithm is run for 50 generations as the stopping criterion. The crossover, mutation, and reproduction rates are set to 0.8, 0.15, and 0.05, respectively. The maximal depth of each GP tree is limited to 8. For each test scenario, the CoGP algorithm is executed independently 30 times to ensure robust statistical evaluation. All experiments are conducted on computers equipped with an Intel Core i7-6700 CPU at 3.40 GHz.

4.3 Experimental studies

To comprehensively evaluate the effectiveness of the proposed CoGP approach for MATA problems, we compare it against three groups of baselines.

4.3.1 Manually designed reactive heuristics

We design four intuitive and rule-based heuristics based on domain knowledge, where the first two heuristics are used for task prioritization and the

latter two for aircraft selection.

1. NNT: select the task with the nearest distance to the aircraft.

2. MinD: select the task with the minimum current demand.

3. MaxC: select the aircraft with the maximal remaining capacity.

4. MaxE: select the aircraft with the maximal remaining energy.

By combining each TPH (NNT or MinD) with each ASH (MaxC or MaxE), we construct four distinct heuristics for MATA, including MNT_MaxC, MNT_MaxE, MinD_MaxC and MinD_MaxE. These combinations provide interpretable and practical baseline solutions that represent standard decision-making strategies in dynamic task allocation. The comparison results can be found in Table 3. The experimental results include the mean value and standard deviation of evaluation metrics, and the t-tests results in which the significance level is set to 0.05.

As shown in Table 3, the proposed CoGP framework consistently outperforms all manually designed reactive heuristics—MNT_MaxC, MNT_MaxE, MinD_MaxC, and MinD_MaxE—across all 23 benchmark instances. For instance, in instance 17, the total arrival time is reduced from 10 625.8 s (MNT_MaxC) to 10 409.7 s (CoGP). The superior performance arises because the CoGP algorithm automatically discovers heuristics that exploit complex problem structures and dynamically evolving conditions, which human-designed rules may overlook. Unlike fixed heuristics that remain static throughout deployment, CoGP continuously refines both task prioritization and aircraft selection decisions simultaneously, effectively capturing interdependencies and optimizing system-wide performance in dynamic environments.

Fig. 7 illustrates the task allocation process for instance P06_100_3. At the early stage, only a few tasks are available. As time progresses, new tasks continuously appear and require assignment. The dispatching rules evolved by CoGP can promptly adapt to these changes and efficiently generate updated solutions to accommodate the newly emerging tasks.

Table 3 Performance comparison of CoGP and manually designed reactive heuristics

Instance index	Total arrival time of all customers (s)				
	CoGP	MNT_MaxC	MNT_MaxE	MinD_MaxC	MinD_MaxE
1	942.8 (16.7)	966.9 (17.7) (+)	999.0 (18.8) (+)	977.5 (17.4) (+)	992.6 (20.0) (+)
2	1465.3 (27.4)	1516.5 (31.3) (+)	1566.9 (26.9) (+)	1547.9 (27.6) (+)	1550.3 (28.9) (+)
3	1512.7 (29.7)	1564.7 (26.1) (+)	1586.1 (32.6) (+)	1635.7 (27.8) (+)	1617.7 (28.7) (+)
4	2752.9 (54.0)	2779.2 (49.8) (=)	2809.2 (52.5) (+)	2852.6 (47.8) (+)	2870.2 (47.2) (+)
5	3532.2 (74.5)	3556.2 (57.3) (=)	3675.6 (65.1) (+)	3676.2 (69.9) (+)	3696.3 (73.8) (+)
6	2214.0 (35.7)	2231.9 (45.9) (=)	2270.0 (36.1) (+)	2299.4 (40.1) (+)	2288.1 (38.5) (+)
7	2420.5 (46.4)	2461.1 (39.9) (+)	2496.2 (46.3) (+)	2505.9 (49.2) (+)	2541.5 (47.5) (+)
8	16 787.6 (306.1)	17 052.3 (296.6) (+)	17 267.2 (285.3) (+)	17 560.8 (308.2) (+)	17 570.3 (285.8) (+)
9	13 519.9 (263.0)	13 756.1 (278.0) (+)	13 813.3 (243.3) (+)	14 047.3 (283.7) (+)	14 150.8 (259.1) (+)
10	13 455.2 (223.2)	13 562.2 (254.2) (+)	13 737.2 (221.2) (+)	13 931.8 (243.1) (+)	14 024.2 (236.2) (+)
11	14 015.2 (264.1)	14 181.5 (227.8) (=)	14 342.6 (261.1) (+)	14 605.9 (269.6) (+)	14 616.1 (247.5) (+)
12	5017.9 (86.7)	5138.8 (88.1) (+)	5211.0 (103.8) (+)	5348.1 (106.3) (+)	5330.8 (93.5) (+)
13	5627.8 (106.1)	5736.8 (94.7) (+)	5811.1 (96.5) (+)	5994.4 (127.1) (+)	5995.7 (128.6) (+)
14	6425.7 (123.7)	6493.8 (135.3) (=)	6576.3 (130.7) (+)	6677.1 (133.2) (+)	6725.7 (133.8) (+)
15	9483.4 (181.3)	9586.7 (166.2) (+)	9688.8 (182.8) (+)	9812.8 (176.8) (+)	9904.2 (198.2) (+)
16	9820.9 (167.3)	9972.1 (186.5) (+)	10 071.2 (201.5) (+)	10 192.5 (163.2) (+)	10 249.3 (176.2) (+)
17	10 409.7 (196.7)	10 625.8 (182.7) (+)	10 812.8 (205.7) (+)	10 903.8 (178.1) (+)	11 027.5 (187.2) (+)
18	13 687.7 (234.6)	13 906.5 (257.7) (+)	13 993.1 (251.8) (+)	14 244.9 (252.1) (+)	14 319.9 (258.1) (+)
19	15 109.1 (271.1)	15 501.6 (274.9) (+)	15 610.7 (282.1) (+)	15 791.8 (265.2) (+)	15 934.5 (272.1) (+)
20	16 116.2 (309.9)	16 516.8 (273.3) (+)	16 721.7 (299.8) (+)	17 038.2 (288.9) (+)	17 162.6 (298.2) (+)
21	19 730.7 (349.2)	20 095.7 (379.1) (+)	20 374.7 (336.2) (+)	20 682.7 (368.9) (+)	20 751.9 (374.2) (+)
22	22 061.6 (404.1)	22 484.5 (391.2) (+)	22 741.7 (399.1) (+)	23 029.2 (414.2) (+)	23 234.9 (427.7) (+)
23	25 890.2 (412.4)	26 400.8 (429.1) (+)	26 690.1 (464.2) (+)	27 056.3 (489.7) (+)	27 291.5 (465.2) (+)

Each row represents a single test instance, and the columns correspond to different task allocation methods. The results are reported in terms of the mean, standard deviation, and significance test. The symbol (+) means that CoGP is significantly better than the compared algorithm. The symbol (=) means there is no significant difference between CoGP and the compared algorithm, which is met when the t-test results are above 0.05

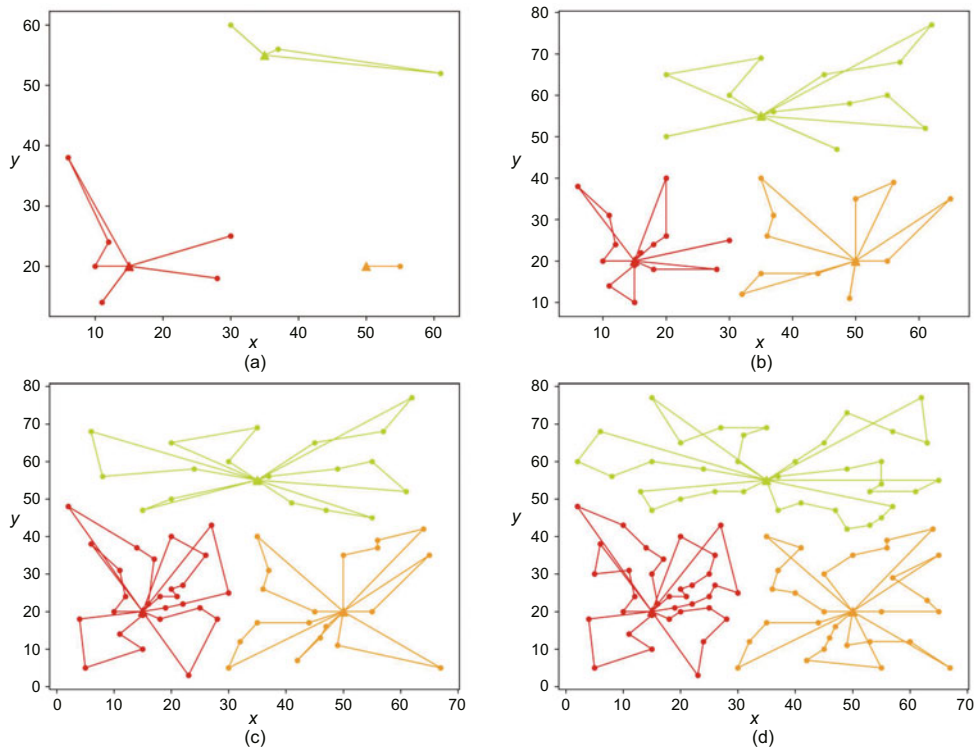

Fig. 7 Visualization of the task allocation process: (a) 5 s; (b) 25 s; (c) 55 s; (d) 75 s. Different colors represent rescue fleets departing from different depots. References to color refer to the online version of this figure

4.3.2 Existing state-of-the-art reactive heuristics

We benchmark CoGP against several representative state-of-the-art reactive algorithms widely adopted in the literature, including the recruitment method (RECRU) (Gao et al., 2022), dynamic deployment (DD) (Chen et al., 2022), and auction-based assignment (AUCT) (Chen et al., 2022). These methods are selected for their demonstrated effectiveness and relevance to dynamic task and resource allocation scenarios. To ensure comparison fairness, the hyperparameters of all the baselines are run with the originally reported settings from their papers. The comparison results are presented in Table 4.

In comparison with RECRU, AUCT, and DD approaches, CoGP again exhibits superior overall efficacy across all 23 test scenarios (Table 4). Recruitment and auction mechanisms generally perform well in early-stage task assignment but degrade when system congestion increases, as their decisions depend on localized or immediate reward estimations. Similarly, DD relies on predefined heuristic templates, limiting its adaptivity. Conversely, CoGP evolves decision-making heuristics through a coevolutionary learning process that internalizes both short-term gains and long-term impacts of each allocation, achieving strategies that remain efficient under varying operational intensities and stochastic task arrivals.

4.3.3 Single-tree GP methods

To further evaluate the effectiveness of the dual-population coevolution, we compare CoGP with the following four variants with single-tree GP. Specifically, we conduct two types of single-tree GP experiments.

1. Ablation-TPH-only: GP is used to evolve only the TPH, combined with either MaxC or MaxE as the aircraft selection rule;

2. Ablation-ASH-only: GP is used to evolve only the ASH, combined with either NNT or MinD as the task prioritization rule.

This design enables us to isolate and evaluate the impact of evolving each decision phase independently, relative to the coupled coevolutionary approach.

Table 4 Performance comparison of CoGP and the existing state-of-the-art reactive heuristics

Instance index	Total arrival time of all customers (s)			
	CoGP	RECRU	DD	AUCT
1	936.0 (18.4)	969.1 (16.7) (+)	968.4 (19.8) (+)	972.1 (19.0) (+)
2	1468.0 (25.6)	1510.1 (33.2) (+)	1512.1 (28.1) (+)	1511.0 (24.9) (+)
3	1538.5 (32.5)	1585.0 (29.6) (+)	1586.5 (27.8) (+)	1582.7 (29.8) (+)
4	2758.6 (53.1)	2822.2 (52.1) (+)	2824.3 (55.6) (+)	2807.2 (46.1) (+)
5	3499.1 (71.4)	3619.5 (66.6) (+)	3607.8 (68.3) (+)	3616.2 (65.1) (+)
6	2217.7 (46.2)	2280.5 (46.1) (+)	2274.1 (40.6) (+)	2266.8 (37.6) (+)
7	2408.4 (39.4)	2479.6 (47.7) (+)	2484.1 (50.4) (+)	2497.4 (45.3) (+)
8	16 888.5 (292.6)	17 152.3 (294.9) (+)	17 289.2 (357.2) (+)	17 360.1 (280.5) (+)
9	13 609.6 (243.6)	14 015.9 (248.7) (+)	13 967.1 (299.3) (+)	13 850.0 (298.4) (+)
10	13 016.9 (274.1)	13 443.4 (239.3) (+)	13 445.0 (283.3) (+)	13 487.8 (296.3) (+)
11	13 958.8 (304.7)	14 169.1 (261.0) (+)	14 292.0 (299.8) (+)	14 253.4 (241.7) (+)
12	5024.2 (109.2)	5143.8 (89.8) (+)	5162.6 (101.1) (+)	5212.2 (88.6) (+)
13	5578.7 (111.7)	5726.9 (100.1) (+)	5728.8 (115.5) (+)	5715.9 (123.5) (+)
14	6322.4 (135.2)	6536.6 (132.8) (+)	6527.4 (109.5) (+)	6473.5 (121.2) (+)
15	9373.4 (164.7)	9655.4 (160.4) (+)	9656.5 (166.4) (+)	9723.4 (174.2) (+)
16	9656.5 (186.0)	10 019.3 (190.6) (+)	9994.6 (166.2) (+)	10 073.3 (221.5) (+)
17	10 448.2 (207.1)	10 763.6 (185.0) (+)	10 699.9 (192.7) (+)	10 725.7 (192.3) (+)
18	13 778.1 (295.6)	14 040.9 (306.0) (+)	14 085.3 (284.6) (+)	14 182.1 (311.9) (+)
19	15 251.3 (292.1)	15 636.6 (252.5) (+)	15 559.1 (260.2) (+)	15 549.4 (253.0) (+)
20	16 170.7 (276.3)	16 657.2 (295.6) (+)	16 651.5 (339.0) (+)	16 673.8 (337.8) (+)
21	20 022.5 (369.8)	20 274.0 (393.7) (=)	20 469.0 (329.5) (+)	20 475.3 (410.0) (+)
22	22 284.3 (446.4)	22 574.9 (444.0) (=)	22 769.4 (463.8) (+)	22 973.0 (458.8) (+)
23	26 095.4 (460.7)	26 816.0 (571.2) (+)	26 708.7 (511.1) (+)	26 521.2 (518.2) (+)

Each row represents a single test instance, and the columns correspond to different task allocation methods. The results are reported in terms of the mean, standard deviation, and significance test. The symbol (+) means that CoGP is significantly better than the compared algorithm. The symbol (=) means there is no significant difference between CoGP and the compared algorithm, which is met when the t-test results are above 0.05

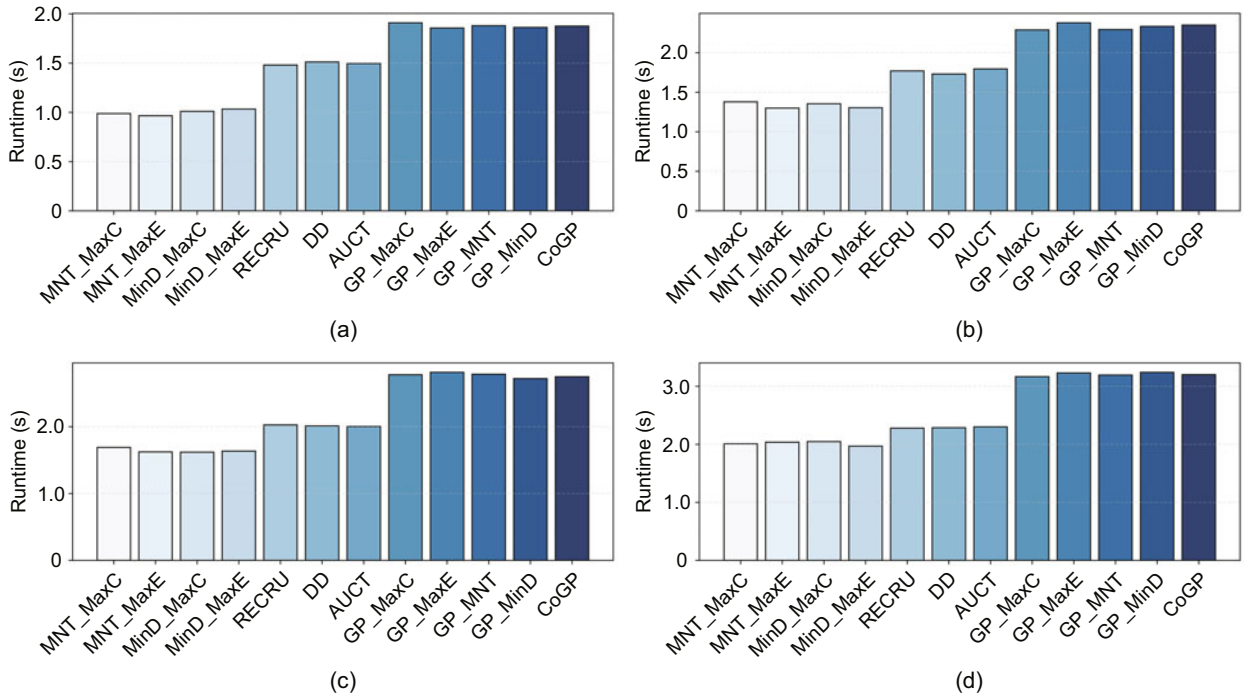

Fig. 8 Runtime comparison of different methods across four instances: (a) instance 2; (b) instance 6; (c) instance 11; (d) instance 22

Table 5 Performance comparison of CoGP and single-tree GP methods

Instance index	Total arrival time of all customers (s)				
	CoGP	GP_MaxC	GP_MaxE	GP_MNT	GP_MinD
1	945.5 (15.7)	958.1 (20.7) (+)	956.6 (20.7) (+)	947.9 (20.2) (=)	949.3 (19.6) (=)
2	1480.6 (26.9)	1504.8 (24.9) (+)	1498.9 (28.9) (=)	1487.4 (32.2) (=)	1502.1 (28.6) (+)
3	1539.2 (30.3)	1566.7 (29.7) (+)	1561.4 (30.6) (+)	1559.7 (27.9) (=)	1559.1 (27.8) (=)
4	2748.1 (50.7)	2787.0 (47.2) (+)	2808.9 (57.4) (+)	2774.9 (51.1) (=)	2779.8 (45.4) (+)
5	3522.7 (71.4)	3595.4 (63.4) (+)	3568.1 (64.5) (=)	3566.1 (67.1) (=)	3585.0 (58.9) (=)
6	2235.2 (41.9)	2265.2 (38.5) (+)	2268.1 (38.8) (+)	2236.5 (41.8) (=)	2250.1 (41.5) (+)
7	2414.2 (40.3)	2447.7 (43.8) (+)	2444.8 (43.0) (+)	2430.3 (47.2) (=)	2450.0 (48.6) (+)
8	16 736.4 (363.9)	17 045.9 (315.9) (+)	17 161.7 (317.7) (+)	16 973.8 (362.5) (+)	16 959.9 (320.9) (+)
9	13 711.9 (291.9)	13 828.0 (268.4) (=)	13 846.7 (239.2) (=)	13 774.6 (260.4) (=)	13 753.5 (243.3) (=)
10	13 121.0 (226.8)	13 258.0 (275.6) (+)	13 303.6 (242.4) (+)	13 183.2 (265.9) (+)	13 221.3 (279.0) (+)
11	13 917.7 (234.3)	14 087.3 (302.4) (=)	14 078.2 (282.5) (=)	14 016.6 (257.0) (=)	14 024.1 (292.4) (+)
12	5029.4 (106.8)	5132.9 (82.6) (+)	5126.5 (108.4) (+)	5072.7 (99.9) (+)	5103.8 (94.0) (=)
13	5564.1 (111.2)	5692.1 (114.1) (+)	5647.8 (116.8) (+)	5612.3 (109.6) (+)	5665.6 (91.4) (+)
14	6391.6 (108.0)	6435.6 (114.9) (=)	6503.6 (106.6) (+)	6415.4 (121.4) (=)	6433.1 (116.4) (=)
15	9408.1 (159.5)	9558.4 (187.0) (+)	9565.5 (172.9) (+)	9482.4 (207.4) (=)	9499.2 (162.7) (=)
16	9679.2 (160.9)	9898.2 (195.9) (+)	9950.8 (188.1) (+)	9778.2 (187.4) (=)	9868.2 (209.9) (+)
17	10 408.4 (195.0)	10 664.4 (217.8) (+)	10 610.9 (177.2) (+)	10 557.2 (203.0) (=)	10 620.8 (181.4) (+)
18	13 743.0 (301.6)	13 940.7 (232.2) (+)	13 963.5 (252.9) (+)	13 790.5 (283.5) (=)	13 868.3 (275.3) (=)
19	15 229.1 (278.2)	15 410.4 (326.7) (=)	15 449.4 (280.8) (=)	15 318.4 (263.7) (=)	15 426.7 (304.9) (=)
20	16 264.1 (266.0)	16 452.2 (281.3) (+)	16 671.9 (307.3) (+)	16 462.2 (341.1) (+)	16 544.1 (281.2) (+)
21	19 955.3 (328.0)	20 148.8 (425.5) (=)	20 350.9 (408.3) (+)	19 981.0 (386.7) (=)	20 166.9 (381.0) (+)
22	22 289.7 (453.7)	22 439.1 (426.3) (+)	22 512.0 (427.9) (+)	22 405.7 (391.3) (+)	22 439.3 (459.0) (+)
23	26 169.7 (530.2)	26 473.2 (545.0) (=)	26 498.5 (539.8) (=)	26 321.7 (425.7) (=)	26 348.1 (506.9) (=)

Each row represents a single test instance, and the columns correspond to different task allocation methods. The results are reported in terms of the mean, standard deviation, and significance test. The symbol (+) means that CoGP is significantly better than the compared algorithm. The symbol (=) means there is no significant difference between CoGP and the compared algorithm, which is met when the t-test results are above 0.05

When compared with conventional single-tree GP variants, CoGP consistently achieves the best performance across all 23 instances (Table 5). Single-tree GP methods that independently evolve TPH or ASH yield inferior outcomes because they neglect the mutual dependencies between these decisions. The explicit coevolutionary structure in CoGP—where TPH and ASH evolve jointly and are evaluated through pairwise fitness attribution—allows the algorithm to learn synergistic policies that optimize both phases simultaneously. This interdependent learning substantially enhances robustness, as observed in large-scale dynamic scenarios with frequent task insertions, where CoGP maintains consistent superiority while other approaches exhibit performance degradation.

4.4 Comparison of runtime

To assess the real-time decision-making capability of different algorithms, we conducted a runtime comparison experiment across four benchmark instances of varying scales, namely, instance 2, instance 6, instance 11, and instance 22. For each instance, we measured the average runtime required for a single decision step under identical hardware and thread configurations. The results are summarized in Fig. 8, where the mean runtime is reported over multiple independent runs.

As expected, the runtime of most algorithms increases with instance scale, reflecting the higher computational demand of larger problem sizes. Among the compared methods, the heuristic approaches constructed using greedy priority rules (i.e., MNT_MaxC, MNT_MaxE, MinD_MaxC, and MinD_MaxE) exhibit the fastest response, typically within 1.0–2.0 s per decision step. The intermediate group, including RECRU, DD, and AUCT, shows moderate decision latency (approximately 1.5–2.3 s), while the GP-based algorithms (GP_MaxC, GP_MaxE, GP_MNT, GP_MinD, and CoGP) require slightly longer processing times, up to about 3.0 s. This can be attributed to their more sophisticated decision logic, where additional intermediate variables are computed to evaluate the relative priorities of tasks and aircraft.

Nevertheless, across all four scales, the per-decision runtimes remain within a narrow range (below 3.0 s), demonstrating that all algorithms—including the proposed CoGP—are capable of making real-time decisions under dynamic conditions. These results confirm that the CoGP framework achieves competitive computational efficiency while maintaining its superior decision quality.

5 Conclusions

In this study, we develop a CoGP approach to automatically generate effective MATA strategies for emergency delivery operations in post-disaster scenarios. In such missions, multiple aircraft must rapidly deliver supplies to spatially dispersed and time-sensitive locations under stringent resource and operational constraints. The proposed CoGP framework achieves this objective through the design of informative terminal sets, a low-level heuristic template, and a two-population coevolutionary scheme that simultaneously develops effective heuristics for both task prioritization and aircraft selection. Comprehensive experiments on public benchmark instances demonstrate that CoGP consistently outperforms existing state-of-the-art GP and heuristic approaches. Compared with conventional single-tree GP methods, CoGP exploits more effectively the interactive information between the two decision processes, thereby enhancing adaptability and decision quality in dynamic mission environments.

While the current study assumes idealized conditions, such as perfect communication, uninterrupted task execution, and sufficient flight endurance, the proposed framework can be extended to more realistic operational settings. Future work will incorporate additional real-world considerations, including mid-mission return-to-base for refueling, safety redundancy, and online route adjustment mechanisms. In addition, a hierarchical two-layer optimization framework will be explored, where the upper layer focuses on robust strategy evolution and the lower layer addresses dynamic event responses in real time.

Contributors

Ce YU, Bo ZHANG, and Tong GUO designed the research. Ce YU and Bo ZHANG processed the data. Ce YU and Tong GUO drafted the paper. Bo ZHANG helped organize the paper. Wenbo DU and Xianbin CAO revised and finalized the paper.

Conflict of interest

Xianbin CAO is a guest editor of the Special Feature on Engineering and Technology for Low-Altitude Economy Infrastructure of *Frontiers of Information Technology & Electronic Engineering*; he was not involved with the peer review process of this paper. All the authors declare that they have no conflict of interest.

Data availability

The datasets are publicly available at http://www.bernabe.dorronsoro.es/vrp/. The other data that support the findings of this study are available from the corresponding author upon reasonable request.

References

Ardeh MA, Mei Y, Zhang MJ, 2022. Genetic programming with knowledge transfer and guided search for uncertain capacitated arc routing problem. *IEEE Trans Evol Comput*, 26(4):765-779. https://doi.org/10.1109/TEVC.2021.3129278

Chen J, Guo YQ, Qiu ZF, et al., 2022. Multiagent dynamic task assignment based on forest fire point model. *IEEE Trans Autom Sci Eng*, 19(2):833-849. https://doi.org/10.1109/TASE.2021.3061757

Fan QL, Bi Y, Xue B, et al., 2024. A multi-tree genetic programming-based ensemble approach to image classification with limited training data [research frontier]. *IEEE Comput Intell Mag*, 19(4):47-62. https://doi.org/10.1109/MCI.2024.3446148

Fortin FA, De Rainville FM, Gardner MA, et al., 2012. DEAP: evolutionary algorithms made easy. *J Mach Learn Res*, 13:2171-2175.

Gao GQ, Xin B, 2019. A-STC: auction-based spanning tree coverage algorithm formotion planning of cooperative robots. *Front Inform Technol Electron Eng*, 20(1):18-31. https://doi.org/10.1631/FITEE.1800551

Gao GQ, Mei Y, Xin B, et al., 2022. Automated coordination strategy design using genetic programming for dynamic multipoint dynamic aggregation. *IEEE Trans Cybern*, 52(12):13521-13535. https://doi.org/10.1109/TCYB.2021.3080044

Guo T, Jiang N, Li BY, et al., 2021. UAV navigation in high dynamic environments: a deep reinforcement learning approach. *Chin J Aeronaut*, 34(2):479-489. https://doi.org/10.1016/j.cja.2020.05.011

Guo T, Mei Y, Du WB, et al., 2025a. Emergency scheduling of aerial vehicles via graph neural neighborhood search. *IEEE Trans Artif Intell*, 6(7):1808-1822. https://doi.org/10.1109/TAI.2025.3528381

Guo T, Mei Y, Zhang MJ, et al., 2025b. Enhanced evolution of parallel algorithm portfolio for vehicle routing problem via transfer optimization. *IEEE Trans Evol Comput*, early access. https://doi.org/10.1109/TEVC.2025.3616385

Guo T, Mei Y, Zhang MJ, et al., 2025c. Genetic programming with multi-fidelity surrogates for large-scale dynamic air traffic flow management. *IEEE Trans Evol Comput*, 29(6):2671-2685. https://doi.org/10.1109/TEVC.2024.3512552

Guo T, Mei Y, Zhang MJ, et al., 2025d. Learning-aided neighborhood search for vehicle routing problems. *IEEE Trans Patt Anal Mach Intell*, 47(7):5930-5944. https://doi.org/10.1109/TPAMI.2025.3554669

Hou ZY, You T, Wang W, 2025. Seismic resilience assessment-informed UAV task allocation framework for post-earthquake survey. *Int J Disaster Risk Reduct*, 116:105160. https://doi.org/10.1016/j.ijdrr.2024.105160

Hu LP, Zhang JQ, Liang XL, et al., 2025. A prescribed-time distributed constrained negotiation allocation algorithm for UAV swarms. *IEEE Trans Aerosp Electron Syst*, 61(5):14961-14980. https://doi.org/10.1109/TAES.2025.3588487

Jia QL, Xiao JP, Feroskhan M, 2024. Multitarget assignment under uncertain information through decision support systems. *IEEE Trans Ind Inform*, 20(8):10636-10646. https://doi.org/10.1109/TII.2024.3397392

Liao XC, Mei Y, Zhang MJ, 2025. GPLight+: a genetic programming method for learning symmetric traffic signal control policy. *IEEE Trans Evol Comput*, early access. https://doi.org/10.1109/TEVC.2025.3578575

Liu SY, Yi L, Xiong XR, et al., 2025. Explainable attention-based AAV target detection for search and rescue scenarios. *IEEE Int Things J*, 12(5):4922-4934. https://doi.org/10.1109/JIOT.2024.3519158

Liu YX, Mei Y, Zhang MJ, et al., 2020. A predictive-reactive approach with genetic programming and cooperative coevolution for the uncertain capacitated arc routing problem. *Evol Comput*, 28(2):289-316. https://doi.org/10.1162/evco_a_00256

Nguyen S, Zhang MJ, Johnston M, et al., 2013. A computational study of representations in genetic programming to evolve dispatching rules for the job shop scheduling problem. *IEEE Trans Evol Comput*, 17(5):621-639. https://doi.org/10.1109/TEVC.2012.2227326

Nguyen S, Mei Y, Zhang MJ, 2017. Genetic programming for production scheduling: a survey with a unified framework. *Complex Intell Syst*, 3(1):41-66. https://doi.org/10.1007/s40747-017-0036-x

Peng Q, Wu HS, Li N, et al., 2024. A dynamic task allocation method for unmanned aerial vehicle swarm based on wolf pack labor division model. *IEEE Trans Emerg Top Comput Intell*, 8(6):4075-4089. https://doi.org/10.1109/TETCI.2024.3386614

Ponda SS, Johnson LB, How JP, 2012. Distributed chance-constrained task allocation for autonomous multi-agent teams. American Control Conf, p.4528-4533. https://doi.org/10.1109/ACC.2012.6315626

Recchiuto CT, Sgorbissa A, 2018. Post-disaster assessment with unmanned aerial vehicles: a survey on practical implementations and research approaches. *J Field Robot*, 35(4):459-490. https://doi.org/10.1002/rob.21756

Sengupta R, Nagi R, Sreenivas RS, 2024. Robust task allocations by distributing the risk among agents: theory and algorithms. *IEEE Trans Autom Sci Eng*, 22:6475-6491. https://doi.org/10.1109/TASE.2024.3446456

Sun ZX, Mei Y, Zhang FF, et al., 2024. Multi-tree genetic programming hyper-heuristic for dynamic flexible workflow scheduling in multi-clouds. *IEEE Trans Serv Comput*, 17(5):2687-2703. https://doi.org/10.1109/TSC.2024.3394691

van Steenbergen RM, van Heeswijk WJA, Mes MRK, 2025. The stochastic dynamic postdisaster inventory allocation problem with trucks and UAVs. *Transp Sci*, 59(2):360-390. https://doi.org/10.1287/trsc.2023.0438

Wang SL, Mei Y, Zhang MJ, et al., 2022. Genetic programming with niching for uncertain capacitated arc routing problem. *IEEE Trans Evol Comput*, 26(1):73-87. https://doi.org/10.1109/TEVC.2021.3095261

Wang SL, Mei Y, Zhang MJ, 2023. A multi-objective genetic programming algorithm with α dominance and archive for uncertain capacitated arc routing problem. *IEEE Trans Evol Comput*, 27(6):1633-1647. https://doi.org/10.1109/TEVC.2022.3195165

Xing JH, Guo T, Tong L, 2024. Reliable truck-drone routing with dynamic synchronization: a high-dimensional network programming approach. *Transp Res Part C Emerg Technol*, 165:104698. https://doi.org/10.1016/j.trc.2024.104698

Xu M, Mei Y, Zhang FF, et al., 2024a. Genetic programming for dynamic flexible job shop scheduling: evolution with single individuals and ensembles. *IEEE Trans Evol Comput*, 28(6):1761-1775. https://doi.org/10.1109/TEVC.2023.3334626

Xu M, Mei Y, Zhang FF, et al., 2024b. Genetic programming and reinforcement learning on learning heuristics for dynamic scheduling: a preliminary comparison. *IEEE Comput Intell Mag*, 19(2):18-33. https://doi.org/10.1109/MCI.2024.3363970

Yang F, Chakraborty N, 2020. Chance constrained simultaneous path planning and task assignment for multiple robots with stochastic path costs. IEEE Int Conf on Robotics and Automation, p.6661-6667. https://doi.org/10.1109/ICRA40945.2020.9197354

Yu YY, Tang QR, Jiang QC, et al., 2025. A deep reinforcement learning-assisted multimodal multiobjective bilevel optimization method for multirobot task allocation. *IEEE Trans Evol Comput*, 29(3):574-588. https://doi.org/10.1109/TEVC.2025.3535954

Zhang FF, Mei Y, Zhang MJ, 2018. Genetic programming with multi-tree representation for dynamic flexible job shop scheduling. 31st Australasian Joint Conf on Artificial Intelligence, p.472-484. https://doi.org/10.1007/978-3-030-03991-2_43

Zhang FF, Mei Y, Nguyen S, et al., 2023. Multitask multiobjective genetic programming for automated scheduling heuristic learning in dynamic flexible job-shop scheduling. *IEEE Trans Cybern*, 53(7):4473-4486. https://doi.org/10.1109/TCYB.2022.3196887