ENGINEERING Artificial Intelligence
https://doi.org/10.2738/ENGAI.2026.0008

Original Research

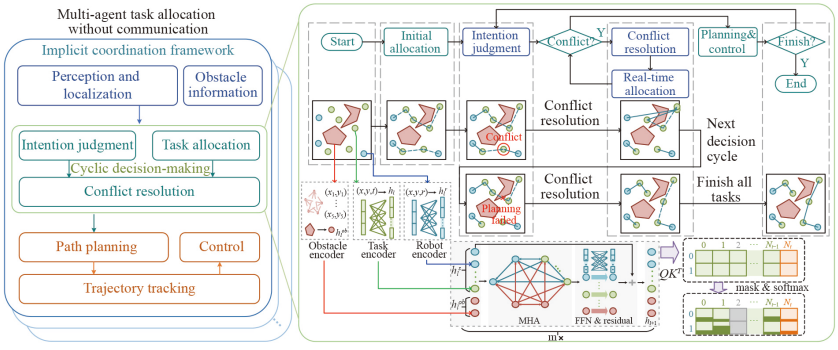
Subgraph Attention Framework for Implicit Coordination in Multi-Robot Task Allocation Without Communication

Zirui Wu[1*], Jianing Hu[1*], Weiran Yao (✉)[1], Jianxing Liu[1], Sergio Vazquez[2], Jose I. Leon[3], Leopoldo G. Franquelo[3]

1 School of Astronautics, Harbin Institute of Technology, Harbin 150001, China
2 Department of Electronic Engineering, Universidad de Sevilla, Sevilla 41004, Spain
3 ENGREEN Laboratory of Engineering for Energy and Environmental Sustainability, Universidad de Sevilla, Sevilla 41092, Spain

© The Author(s) 2026. This article is published by Higher Education Press.

GRAPHICAL ABSTRACT



HIGHLIGHTS

- A decentralized framework coordinates routing-based MRTA using local observations without inter-robot messages.
- The end-to-end SubG-AM allocator generates task sequences directly from coordinates, avoiding full pairwise state-node cost-matrix computation during online inference.
- Subgraph attention module infers peer targets from trajectories observed within a sliding window.

ABSTRACT A Subgraph Attention Neural Network and Conflict Resolution (SANN-CR) framework is proposed for routing-based multi-robot task allocation without inter-robot communication. The framework addresses the coupled challenges of computationally demanding online reallocation and inconsistent decisions arising from independent robot operation. Given a shared initial map and task list, robots rely on self-localization and onboard peer observations without exchanging messages during execution. An end-to-end allocator, SubG-AM, embeds robot, task, and obstacle coordinates through subgraph attention to generate task sequences without constructing a full pairwise cost matrix during online inference. A trajectory-based intention module estimates peers' target tasks, and a distance-based conflict resolution strategy uses these estimates to reconcile competing assignments through selective reallocation. Comparative allocation experiments, simulation ablations, and physical robot experiments were conducted to evaluate allocation quality, task completion, and computational efficiency. In the simulation ablations, SANN-CR completed all tasks in all 100 test instances at each of four problem scales and reduced redundant execution relative to the variant without conflict resolution. The three decision modules achieved a combined mean computation time of 5.3 ms on a Jetson Orin NX, excluding path planning. These results demonstrate the feasibility of decentralized task coordination without inter-robot message exchange under the stated sensing and initialization assumptions.

KEYWORDS Multi-robot systems; No-communication task allocation; Subgraph attention networks; Conflict resolution

Received 31 Jul 2026; accepted 19 Sep 2026

E-mail: yaoweiran@hit.edu.cn (Yao W.).

* These authors contributed equally to this work.

1 Introduction

Multi-robot task allocation (MRTA) coordinates task selection and execution in multi-robot systems (MRS). Maintaining collaboration during communication silence motivates coordination mechanisms that operate without inter-robot message exchange [1]. Marker-based and optical methods support relative localization and cooperative flight without task-level messages [2–4]. Auction and consensus approaches rely on exchanged bids, assignments, or winner information [5–7]. Behavior-based cooperation, game-theoretic allocation, and communication-constrained swarms provide complementary coordination principles [8–11], but their objectives and information assumptions differ from routing-based MRTA without message exchange.

Two coupled challenges arise. Routing-based MRTA involves NP-hard combinatorial optimization [12,13], and obstacle-aware allocation can incur substantial preprocessing costs from pairwise path searches. Frequent onboard reallocation must therefore operate within a limited computational budget. Meanwhile, independent decisions may produce duplicated assignments, and unfinished tasks must be recovered without receiving peers' assignments or execution-status messages.

This study proposes a Subgraph Attention Neural Network and Conflict Resolution (SANN-CR) framework for MRTA under no-communication (MRTA-NC). Robots share initial maps and task lists and rely on self-localization, onboard peer sensing, and local completion records, as specified in Section 3. Motivated by advances in vectorized scene representations and trajectory and intention prediction [14–17], SANN-CR integrates allocation, intention inference, and conflict resolution. Its contributions are:

(1) A decentralized framework coordinates routing-based MRTA using local observations without inter-robot messages during execution.

(2) The end-to-end SubG-AM allocator generates task sequences directly from coordinates, avoiding full pairwise state-node cost-matrix computation during online inference; training and sequence evaluation retain A* transition costs.

(3) A subgraph attention module infers peer targets from trajectories observed within a sliding window.

(4) A distance-based strategy resolves target conflicts through bounded reallocation; its impact on travel cost is analyzed under explicit assumptions.

Sections 2–6 present related work, the problem formulation, the proposed framework, experiments, and conclusions, respectively.

2 Related Work

2.1 Task allocation

Based on the types of robots and the attributes of tasks, Gerkey et al. have categorized the task allocation problem into eight major classes [12]. Korsah et al. further considered the temporal dependencies of tasks and supplemented the classification [13]. Except for the simplest classification, which can quickly find the optimal solution through the Hungarian algorithm [18], other classifications are almost all NP-complete or even NP-hard. Currently, numerous methods have been proposed to solve the MRTA problem, such as mixed-integer programming [19], heuristic algorithms [20,21], and genetic algorithms [22,23]. These methods can achieve near-optimal solutions with satisfactory optimization objectives. Market-based methods form another important family of distributed MRTA approaches. The MURDOCH auction system demonstrated dynamic task allocation on physical robot teams [5], while the consensus-based bundle algorithm improved robustness in decentralized allocation [6]. A recent survey provides a systematic classification of these market and communication schemes [7]. Although decentralized, these approaches still require explicit exchanges of bids, assignments, or consensus information and therefore cannot be applied directly under complete communication denial. MRTA problems may also involve constraints arising from the computational and kinematic limitations of robots. For instance, Zhang et al. studied energy-efficient allocation under time-window and precedence constraints [24], and Tian et al. explored decentralized mission planning under kinematic constraints [25]. Nevertheless, these methods always rely on adjacency matrices when applied to time-series MRTA problems, such as the multiple traveling salesman problem (MTSP). In environments with restricted areas or obstacles, computing adjacency matrices becomes computationally intensive, leading to a long preprocessing time.

In recent years, graph neural networks (GNNs) [26] have gained popularity for solving combinatorial optimization and integer programming problems due to their natural representation of relational structures. The message-passing formulation provides a general framework for learning node and graph representations [27], and recent surveys have connected GNN architectures systematically to combinatorial optimization and reasoning [28]. Since many types of MRTA problems can be abstracted as combinatorial optimization problems, GNN-based methods can be adapted for MRTA applications, and many researchers have

employed Markov decision processes and deep learning to solve the classic Traveling Salesman Problem [29−31]. With the success of attention mechanism in other domains [32], some researchers have integrated them with GNN and reinforcement learning to solve a wider range of combinatorial optimization problems [33,34], such as vehicle routing, minimum vertex cover, and maximum cut. These methods often outperform traditional approaches in terms of computational efficiency. However, they primarily target abstract mathematical models, whereas MRTA problems cannot be completely deconstructed through mathematical modeling due to application-specific and real-world constraints that are difficult to be represented by formulas directly. The no-communication conditions considered in this paper is one such constraint, rendering existing methods unsuitable for direct application to the MRTA problem under discussion.

2.2 Communication constraints and implicit coordination

Communication constraints in MRS range from intermittent or bandwidth-limited message exchange to complete denial of explicit inter-robot communication. Balch and Arkin analyzed how communication affects reactive multi-agent behavior [35], while Parker's ALLIANCE architecture demonstrated distributed, behavior-based action selection using mission requirements, teammate activities, environmental conditions, and internal robot states [9]. Arkin showed that schema-based robots can cooperate in navigation without direct communication [8]. Stigmergic approaches provide another form of implicit coordination: for example, pheromone robotics uses persistent virtual environmental cues rather than direct task-negotiation messages [36]. These studies establish important behavior-based and environment-mediated coordination principles, although their assumptions and objectives differ from routing-based MRTA under complete communication denial.

A second research stream addresses perception without exchanging task-level information. Walter et al. used ultraviolet markers and onboard vision for relative localization [2]; subsequent work used this capability for compact formation flight and search-and-rescue swarms without explicit communication [3,4]. These methods provide observable robot states, but they do not by themselves recover other robots' task decisions.

At the planning level, Dai et al. used an incomplete-information game to allocate tasks among soccer robots without communication [10], whereas Kanakia et al. modeled limited-information MRTA as a global game [37]. Other studies considered path planning under communication and GPS denial [38], multi-UAV exploration with limited communication and energy [39], and decentralized exploration that minimizes communication using a Markov decision process [40]. Alitappeh and Jeddisaravi decomposed exploration into area partitioning, allocation, and routing [41], while Talamali et al. showed that constraining communication can improve swarm adaptability in changing environments [11].

Overall, existing work predominantly addresses relative perception, navigation and formation, domain-specific allocation, or reduced rather than absent communication. Few studies address general routing-based MRTA in which each robot must allocate tasks independently, infer peers' intentions from observation, and resolve duplicated assignments without exchanging messages. The SANN-CR framework is designed for this decision-level gap.

3 Problem Statement

Table 1 lists the notation. In a 2D obstacle environment, N_r robots sequentially visit N_t task points. Robot i locally marks task j complete when $\|p_i - p_j\|_2 < \epsilon_{\text{task}}$, the prescribed completion threshold. The objective minimizes total travel distance:

$$\min J = \sum_{i=1}^{N_r} \text{dist}(S_i), \tag{1}$$

subject to

$$\sum_{i=1}^{N_r} \sigma_{ij} = 1, j = 1,2,\cdots,N_t. \tag{2}$$

$$\sigma_{ij} \in \{0,1\}, i = 1,2,\cdots,N_r, j = 1,2,\cdots,N_t. \tag{3}$$

where σ_{ij} indicates whether robot i executes task j;

Table 1 List of notations and their descriptions in problem statement.

Notations	Descriptions
state	Including robot initial position and task point
N_r	The number of robots
N_t	The number of tasks
N_s	The number of state nodes in the graph, which is equal to $N_r + N_t$
N_{ob}	The number of obstacles
J	The total cost
S_i	Task execution sequence of robot i
dist()	The distance traveled in accordance with a sequence S
σ_{ij}	The boolean variable that indicates whether robot i executes task j
costmat()	Cost matrix or adjacency matrix. Input two state nodes and return the distance between them
R_t	The initial reward of task
α	The decay coefficient

constraints (2)–(3) assign each task exactly once. $S_i = [t_{i,1}, \cdots, t_{i,l_i}]$ is robot i's ordered list of task identifiers or target nodes, not motor inputs, velocity commands, or a time-parameterized trajectory. Its travel distance is

$$\mathrm{dist}(S_i) = \sum_{k=0}^{l_i-1} \mathrm{costmat}(state_{i,k}, state_{i,k+1}), \tag{4}$$

$$state_{i,k} \in S_i, i = 1,2,\cdots,N_r, k = 1,2,\cdots,l_i, \tag{5}$$

where l_i is the number of assigned tasks, $state_{i,0}$ is robot i's initial state, and $state_{i,k}$ corresponds to target $t_{i,k}$. Routes are open, with no return-to-depot leg. Each costmat() entry is an A* grid shortest-path distance in the obstacle map. An alternative objective maximizes time-decaying reward [25]:

$$\max J_{\mathrm{reward}} = \sum_{i=1}^{N_t} R_t \exp(-\alpha t_i), \tag{6}$$

where R_t, α, and t_i denote the initial task reward, decay coefficient, and task i's completion time. The method can also use this objective; here we use the distance cost in Eq. (1).

This SR-ST-TA-ID problem [12,13] resembles the Multiple Traveling Salesman Problem (MTSP), but includes obstacle-dependent path costs and execution constraints such as planning failures and robot malfunctions. Without communication, each robot independently computes its sequence onboard. Although the system objective remains Eq. (1), independent assignments need not satisfy Eqs. (2)–(3), motivating conflict resolution.

Operational information assumptions. Robot i has access to:

(1) Shared priors: the static map, obstacle geometry, coordinate frame, task identifiers, and complete target coordinates supplied to all robots before deployment.

(2) Own state: its pose estimate, task sequence, and planner and execution status. OptiTrack supplies only its own pose in the physical experiments.

(3) Peer observations: relative poses and recent trajectories estimated with onboard LiDAR and cameras, from which intentions and stopped-motion signals are inferred locally.

(4) Local task records: its task list, completion flags, and observation history, which determine remaining-task candidates. A task is removed only upon local completion or inference from these observations.

No inter-robot messages are exchanged during execution: peers' motion-capture streams, assignments, private completion flags, and stopping causes are not directly available. There is no globally synchronized completion log; shared priors, self-localization, and passive peer sensing preclude interpreting this setting as complete information isolation.

4 Solution Methodology

4.1 Motivation and framework

Conventional allocation frameworks exchange assignments and execution status to maintain consistency and recover from failures. SANN-CR instead runs onboard each robot using the shared priors and local observations defined in Section 3. As shown in Fig. 1, periodic allocation supports recovery, intention judgement estimates peers' targets, and a common conflict-resolution rule handles duplicated assignments. Repeated reallocation requires a fast allocator. The output

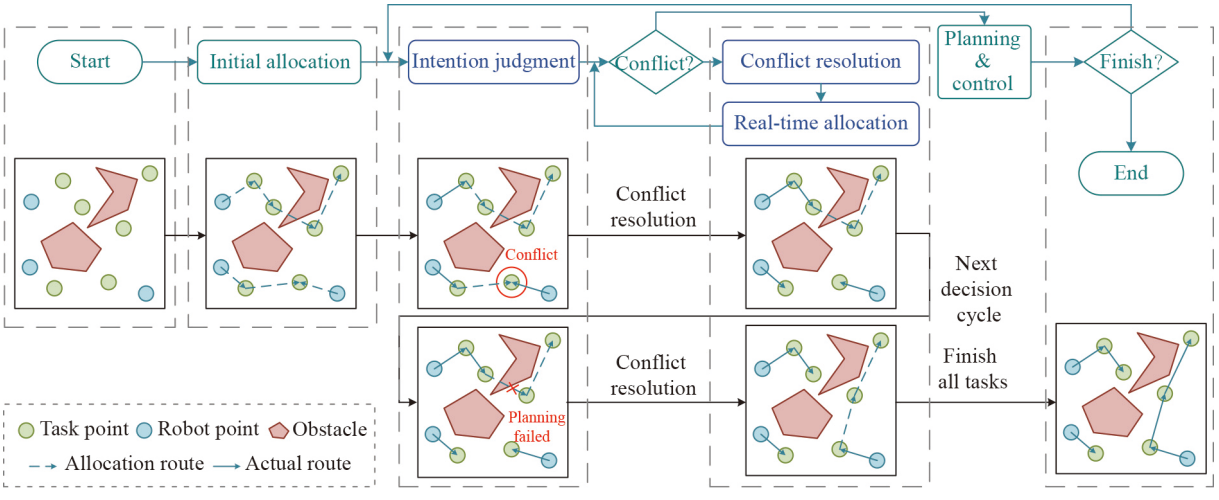


Fig. 1 SANN-CR framework. Intention judgement infers peer targets from locally observed motion. Conflict resolution invokes real-time allocation to resolve duplicated assignments.

is a task-target sequence. Hybrid A* generates paths using a point-state model with obstacles inflated by robot dimensions; infeasible targets trigger the local recovery procedure in Section 5.2.

4.2 Real-time task allocation

SubG-AM extends the TSP attention model in Ref. [29] to multi-start routing, using subgraphs to embed polygonal obstacles and restricted areas. It maps robot, task, and obstacle coordinates to an allocation sequence. Obstacle-aware behavior is learned during training; without a footprint input or explicit collision checker, the network provides no collision-free guarantee.

4.2.1 Subgraph and encoder

Fig.2 shows the encoder. Each of the N_{ob} obstacles forms a p-node subgraph, with p fixed for batch training; complex boundaries can be divided into p-edge polygons. Each node contains consecutive normalized vertex coordinates, $x_i^{\text{ob}} = (x_i, y_i, x_{i+1}, y_{i+1})$, with vertex $p+1$ equal to vertex 1. A feedforward network (FFN) embeds this input into e dimensions:

$$h_{i,0}^{\text{ob}} = \text{FFN}(x_i^{\text{ob}}), \tag{7}$$

for $i = 1, \cdots, p$. A GNN applies m multi-head attention (MHA) layers:

$$h_{i,l+1}^{\text{ob}} = \text{FFN}(\text{BN}(\text{MHA}(h_{1,l}^{\text{ob}}, \cdots, h_{p,l}^{\text{ob}}))) + h_{i,l}^{\text{ob}}, \tag{8}$$

where $l = 0, \cdots, m-1$ and $i = 1, \cdots, p$. Mean pooling gives the obstacle embedding:

$$h^{\text{ob}} = \frac{1}{p}\sum_{i=1}^{p} h_{i,m}^{\text{ob}}. \tag{9}$$

The endpoint pairs encode cyclic adjacency and closure, without a canonical starting vertex or clockwise/counterclockwise label. Fully connected attention and mean pooling aggregate location and boundary cues, but do not guarantee topological reachability. Downstream A* and Hybrid A* retain the polygon map for grid-distance and inflated-obstacle feasibility checks. Robot and task nodes are collectively called state nodes. Their 3D features contain normalized coordinates and a robot/task indicator, embedded into e dimensions:

$$h_{i,0}^{\text{s}} = \text{FFN}(x_i^{\text{s}}), \tag{10}$$

for $i = 1, \cdots, N_s$. In the global cross-attention graph (CAG), obstacle queries attend only to obstacles, whereas state queries attend to all nodes:

$$h_{l+1}^{\text{s}}, h_{l+1}^{\text{ob}} = \text{CAG}(h_l^{\text{s}}, h_l^{\text{ob}}), \tag{11}$$

which is equivalent to the following equations:

$$h_{j,l+1}^{\text{ob}} = \text{FFN}(\text{BN}(\text{MHA}(h_{1,l}^{\text{ob}}, \cdots, h_{N_{\text{ob}},l}^{\text{ob}}))) + h_{j,l}^{\text{ob}}, \tag{12}$$

$$h_{i,l+1}^{\text{s}} = \text{FFN}(\text{BN}(\text{MHA}(h_{1,l}^{\text{ob}}, \cdots, h_{N_{\text{ob}},l}^{\text{ob}}, h_{1,l}^{\text{s}}, \cdots, h_{N_s,l}^{\text{s}}))) + h_{i,l}^{\text{s}}, \tag{13}$$

where $l = 0, \cdots, m-1$, $i = 1, \cdots, N_s$, and $j = 1, \cdots, N_{\text{ob}}$. For head a, let $Q_a = X_q W_q^a$, $K_a = X_k W_k^a$, and $V_a = X_k W_v^a$. Then $A_a = \text{softmax}(Q_a K_a^T / \sqrt{d_k} + M) V_a$ and $\text{MHA} = \text{Concat}(A_1, \cdots, A_H) W_o$, where H is the number of heads, $d_k = e/H$, and all W are learned projections. The mask M is zero for allowed query–key pairs and $-\infty$ otherwise, implementing the CAG restrictions above; local obstacle attention allows all pairs. State queries encode robot/task coordinates, while obstacle keys and values encode obstacle positions and shapes. Query-dependent scores can therefore distinguish nearby and distant obstacles. Unidirectional updates keep obstacle embeddings independent of state features while retaining spatial information for state decisions. After m blocks with batch normalization (BN), FFNs, and residual connections, state-node embeddings are averaged to obtain the graph embedding:

$$h_g = \frac{1}{N_{\text{s}}}\sum_{i=1}^{N_{\text{s}}} h_{i,m}^{\text{s}}. \tag{14}$$

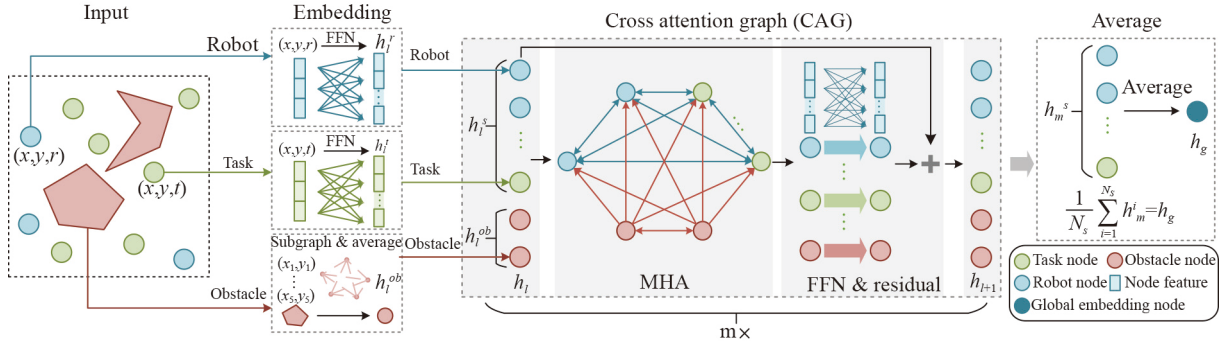


Fig. 2 SubG-AM encoder. FFNs embed robot/task features; attention aggregates normalized obstacle edge-endpoint features into subgraphs. Global attention with normalization and residual connections produces node embeddings.

4.2.2 Decoder

Algorithm 1 starts at a robot node and concatenates its embedding $h^{s}_{\text{idx},m}$ with h_g into the $2e$-dimensional h_{cat}. Cross-attention produces h_{state}, masking selected nodes. The first attention splits projected queries, keys, and values into H heads of width $d_k = e/H$, hence the divisor $\sqrt{d_k}$. Concatenation and projection restore e dimensions; the final selection logits use unsplit e-dimensional queries and keys, hence $\sqrt{e}$. Softmax yields the next-node probabilities P.

Training samples from P; testing selects its maximum. Decoding ends after every state node is selected. The sequence is divided into ordered task-target lists S_i, whose targets pass sequentially to Hybrid A* and then NMPC/PID tracking. These lists contain neither control commands nor time-parameterized trajectories.

$$S_1, S_2, \cdots, S_i = \text{divide}(Seq). \tag{15}$$

Robot nodes delimit task segments; incoming edges to robot nodes are excluded from the loss.

For two robots and four tasks, $[r_1, t_2, t_1, r_2, t_4, t_3]$ gives $S_1 = [t_2, t_1]$ and $S_2 = [t_4, t_3]$. Masking and complete decoding include every state node exactly once, so no task is omitted or duplicated. Consecutive or final robot nodes, including cases with fewer tasks than robots, yield empty segments: these robots are idle for this round, with no task discarded.

Algorithm 1 Decoder for task allocation.

Require: h_g, h^{s}_{m}, $Node_r$, $Node_s$, $training$

Ensure: Seq, pro

$Seq(0) \leftarrow node_0$, $node_0 \in Node_r$

$pro(0) \leftarrow 1$

$idx \leftarrow \text{GetIndex}(node_0)$

$masks \leftarrow zeros(N_s)$, $masks(idx) \leftarrow$ TRUE

for $i \in 1, 2, \cdots, N_s - 1$ **do**

  $h_{\text{cat}} \leftarrow \text{Concatenate}(h_g, h^{s}_{\text{idx},m})$

  $Q_{\text{cat}} \leftarrow W_{q1}(h_{cat})$

  $K_1 \leftarrow W_{k1}(h^{s}_{m}), V_1 \leftarrow W_{v1}(h^{s}_{m})$

  $h_{state} \leftarrow \text{FFN}(\text{Softmax}(\text{Mask}(\frac{Q_{\text{cat}} K_1^T}{\sqrt{d_k}}))\, V_1)$

  $Q_{state} \leftarrow W_q(h_{state}), K_2 \leftarrow W_{k2}(h^{s}_{m})$

  $P \leftarrow \text{Softmax}(\text{Mask}(\frac{Q_{state} K_2^T}{\sqrt{e}}))$

  if $training$ = TRUE **then**

    $idx \leftarrow \text{Sample}(P)$

  else

    $idx \leftarrow \text{Argmax}(P)$

  end if

  $pro(i) \leftarrow P(idx)$

  $Seq(i) \leftarrow node_{idx}, node_{idx} \in Node_s$

  $masks(idx) \leftarrow$ TRUE

end for

4.2.3 Training

Following the reinforcement learning method in Ref. [29], the loss is

$$L = (J_{\text{train}} - J_{\text{baseline}}) * \left(\sum_{i=0}^{N_s - 1} \log(pro(i)) \right) \tag{16}$$

where J_{train} and J_{baseline} are the respective model costs from (1), and $pro(i)$ is the decoder's selection probability. Their cost difference weights the policy-gradient update. The baseline is replaced when a t-test indicates that the trained model performs significantly better.

Robot starts, tasks, and obstacles are sampled within a normalized 1×1 plane, with random counts fixed within each batch. Training uses $p = 4$ vertices per obstacle, matching the physical experiments. A* supplies pairwise state-node costs for Eq. (4) during training and sequence evaluation. Online SubG-AM does not receive this matrix, avoiding its quadratic A* preprocessing; assigned targets still require downstream path planning.

4.3 Intention judgement

Observed trajectories and context support motion and intent inference [15–17]. Here, we classify the unfinished task pursued by each robot without receiving its assignment. The network in Fig.3 follows SubG-AM's local/global attention structure, with different inputs and a task-intention classifier.

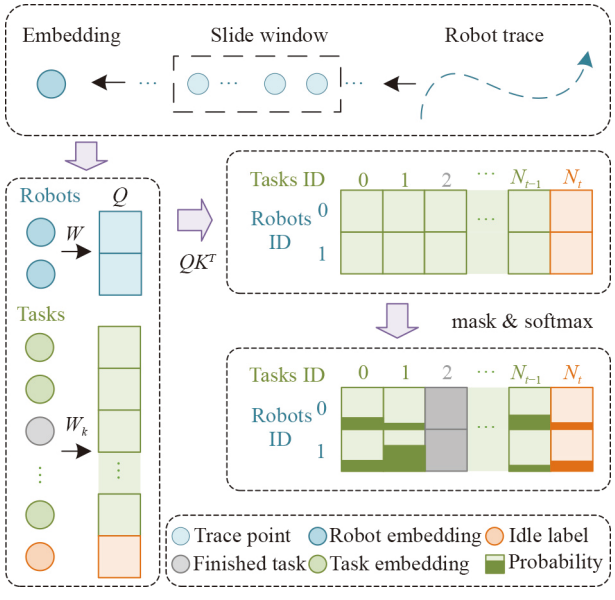


Fig. 3 Intention module. Subgraphs embed sliding-window trajectories; masked attention and softmax give task-intention probabilities, excluding tasks locally marked as finished.

Inputs comprise recent robot trajectories, tasks unfinished according to local completion flags, and obstacles. Task features contain coordinates and a completion flag; alternatively, a mask excludes completed tasks. Trajectory features contain coordinates and a time index, aggregated by local graph embedding. Obstacle and global encoding follow SubG-AM; query–key scores and softmax yield intention probabilities. Peer trajectories come from onboard LiDAR and cameras; self trajectories use own pose estimates. No peer task-state or completion messages are received. A visual-motion detector with a velocity threshold sets $I[m]=-1$ for a stopped robot. This denotes stopped motion without distinguishing task completion, planning failure, or hardware malfunction.

Supervised training pairs simulated trajectories with their assigned targets. The simulation generates random robot starts, tasks, and obstacles, using SubG-AM allocation, A* planning, and NMPC tracking with the experiment's kinematic parameters. Communication is available during data generation to obtain target labels.

4.4 Conflict resolution

Different local observations can produce conflicting allocations. Each robot locally predicts sequences for all robots but executes only its own; peer sequences are not received assignments. Algorithm 2 resolves only peer intentions that conflict with the executing robot's current target.

Here, R, T, I, id, and Seq denote robots, tasks, inferred intentions, the executing robot's identifier, and its task sequence. All flags are local: R_v contains observed non-stopped robots and T_v locally unfinished tasks. A stopped peer ($I[m]=-1$) triggers candidate reconstruction from the task list and observation history. This signal does not distinguish completion from planning or hardware failure; no temporary-wait action or global completion log is used. If a peer intends the same target at lower distance cost, the robot yields it, removes it from the valid task set, and virtually places the peer at that target. This simulates completion for the distance-based objective. SubG-AM then reallocates the modified sets until no conflict remains. At most one conflict is resolved per peer, giving at most N_r-1 reallocations.

Algorithm 2 Conflict Resolution

Require: R, T, Ob, Seq_i, I, id

Ensure: Seq_o

$R_v \leftarrow \emptyset, T_v \leftarrow \emptyset, Seq_o \leftarrow Seq_i$
for $m \in 1,2,\cdots,N_r$ **do**
  if $I[m] \neq -1$ **then**
    $R_v \leftarrow R_v \cup \{m\}$
for $n \in 1,2,\cdots,N_t$ **do**
  if $T[n].finished$ = FALSE **then**
    $T_v \leftarrow T_v \cup \{n\}$
do
  $conflict \leftarrow$ FALSE
  for $m \in 1,2,\cdots,N_r$ **do**
    if $I[m] = Seq_o[0]$ **and**
    dist($R[m], I[m]$)<dist($R[id], I[m]$) **then**
      $R[m].pos \leftarrow T[I[m]].pos$
      $T[I[m]].finished \leftarrow$ TRUE
      $Seq_o \leftarrow$ SubG-AM(R_v, T_v, Ob)
      $conflict \leftarrow$ TRUE
      break
while ($conflict$)

Next, the impact of the number of conflicts on the optimization metrics is analyzed. Similarly, the optimization metric in this case is the total travel distance of the robots. To facilitate the analysis, we make four necessary assumptions.

(1) The task allocation algorithm f used in conflict resolution is relatively stable. That is, with small perturbations in the input, executing the allocation algorithm multiple times on one robot or on different robots will result in the same output task sequence S for the task set T and robot set R under algorithm f.

$$f(T,R) = S = \{Seq_1, Seq_2, \cdots, Seq_{N_r}\}, \tag{17}$$

where Seq_i is the task sequence of robot i. The relative deviation of the output optimization objective does not exceed Δ:

$$\begin{aligned} \text{dist}(\left|f_x(T,R) - f_y(T,R)\right|) &\leqslant \text{dist}(\Delta f_x(T,R)), \\ \text{dist}(\left|f_x(T,R) - f_y(T,R)\right|) &\leqslant \text{dist}(\Delta f_y(T,R)), \end{aligned} \tag{18}$$

where $x, y \in R$, dist() is the distance function.

(2) Definition of the decomposition of the task set and robot set. Assume that the task set T can be divided into two subsets, T_1 and T_2, and that no task from T_1 and T_2 will appear in the same robot's task sequence. We say that the task set T can be fully decomposed into T_1 and T_2 under the allocation algorithm f.

$$T \overset{f}{=} T_1 + T_2. \tag{19}$$

if and only if:

$$\begin{aligned} &T_1 \cap T_2 = \emptyset, T_1 \cup T_2 = T, \\ &\forall t_1 \in T_1, t_2 \in T_2, Seq_i \in S, \{t_1, t_2\} \cap Seq_i = \emptyset. \end{aligned} \tag{20}$$

Eq. (20) can also be expressed as:

$$\begin{aligned} &T_1 \cap T_2 = \emptyset, T_1 \cup T_2 = T, \\ &T_1 = \{Seq_i | Seq_i \in S\}, T_2 = \{Seq_j | Seq_j \in S\}. \end{aligned} \tag{21}$$

Then, the robot set R under the allocation algorithm f and the complete decomposition of the task set T_1 and T_2 can also be correspondingly decomposed into R_1 and R_2:

$$R \overset{f,T_1,T_2}{=\!=\!=} R_1 + R_2. \tag{22}$$

Since the decomposition of the task set and the robot set correspond to each other, we abbreviate $f(T,R)$ as $f(T)$.

(3) The task allocation algorithm used in conflict resolution possesses internal consistency. If the task set T can be fully decomposed into T_1 and T_2 under the allocation algorithm f, and the corresponding robot set R can also be decomposed into R_1 and R_2 under f and the complete decomposition of T_1 and T_2:

$$\begin{aligned} T &\overset{f}{=} T_1 + T_2, \\ R &\overset{f,T_1,T_2}{=\!=} R_1 + R_2, \end{aligned} \tag{23}$$

then:

$$f(T,R) = f(T_1) + f(T_2). \tag{24}$$

(4) The distance traveled by the robot during the decision cycle is negligible compared to the total distance. This can be considered a minor perturbation in the input, denoted as ξ.

Based on the four axioms outlined above, we will proceed with the derivation. We will derive the changes in the total travel distance of the robots compared to the original allocation in the cases of one conflict and two conflicts. By analogy, this can be extended to the case of k conflicts.

To simplify the notation, we define the total travel distance of the robots under the allocation algorithm f as:

$$F(T) = \mathrm{dist}(f(T)). \tag{25}$$

Assume that a single conflict occurs. Regardless of the number of robots and tasks, they can be divided into two categories: The set of tasks for which the decision sequence changes after the conflict, denoted as T_C. The set of tasks for which the decision sequence remains unchanged, denoted as T_N.

Compared to the conflict-free case, the change in total travel distance can be expressed as:

$$\begin{aligned} |J_1 - J_0| &= |F_\mathrm{C}(T_\mathrm{C}) + F_\mathrm{N}(T_\mathrm{N}) - F_\mathrm{N}(T)| \\ &= |F_\mathrm{C}(T_\mathrm{C}) + F_\mathrm{N}(T_\mathrm{N}) - F_\mathrm{N}(T_\mathrm{C}) - F_\mathrm{N}(T_\mathrm{N})| \\ &= |F_\mathrm{C}(T_\mathrm{C}) - F_\mathrm{N}(T_\mathrm{C})| \\ &\leqslant \Delta F_\mathrm{N}(T_\mathrm{C}) + \xi. \end{aligned} \tag{26}$$

where F_C represents the result of reallocation after conflict resolution, and F_N represents the original allocation result without conflict. The transition from the first to the second step in the derivation uses (19), which states that T under f_N can be fully decomposed into T_C and T_N.

If two conflicts occur, the tasks can be divided into three categories: Tasks whose decisions are never affected by any conflict, denoted as T_N. Tasks whose decisions change after the first conflict but are not affected by the second, denoted as $T_{\mathrm{C}1}$. Tasks whose decisions change after the second conflict, denoted as $T_{\mathrm{C}2}$.

Compared to the conflict-free case, the change in total travel distance can be expressed as:

$$\begin{aligned} |J_2 - J_0| &= |F_{\mathrm{C}1}(T_{\mathrm{C}1}) + F_{\mathrm{C}2}(T_{\mathrm{C}2}) + F_\mathrm{N}(T_\mathrm{N}) - F_\mathrm{N}(T)| \\ &= |F_{\mathrm{C}1}(T_{\mathrm{C}1}) + F_{\mathrm{C}2}(T_{\mathrm{C}2}) - F_\mathrm{N}(T_{\mathrm{C}1} + T_{\mathrm{C}2})| \\ &\leqslant |F_{\mathrm{C}1}(T_{\mathrm{C}1} + T_{\mathrm{C}2}) - F_\mathrm{N}(T_{\mathrm{C}1} + T_{\mathrm{C}2})| \\ &\quad + |F_{\mathrm{C}2}(T_{\mathrm{C}2}) - F_{\mathrm{C}1}(T_{\mathrm{C}2})| \\ &\leqslant \Delta F_\mathrm{N}(T_{\mathrm{C}1} + T_{\mathrm{C}2}) + \Delta F_{\mathrm{C}1}(T_{\mathrm{C}2}) + 2\xi \\ &\leqslant \Delta F_\mathrm{N}(T_{\mathrm{C}1} + T_{\mathrm{C}2}) + \Delta(1+\Delta) F_\mathrm{N}(T_{\mathrm{C}2}) + 2\xi, \end{aligned} \tag{27}$$

where $F_{\mathrm{C}1}$ represents the result of reallocation after the first conflict, and $F_{\mathrm{C}2}$ represents the result after the second conflict. In the transition from the second to the third step of the derivation, the triangle inequality and Eq. (19) are applied. Since $\{T_{\mathrm{C}1} + T_{\mathrm{C}2}\}$ under f_N may not be fully decomposable into $T_{\mathrm{C}1}$ and $T_{\mathrm{C}2}$, Eq. (19) is applied to $f_{\mathrm{C}1}$. In the transition from the fifth to the sixth step, a variant of Eq. (18) is used, ensuring that the final result is expressed entirely in terms of F_N under f_N.

For k conflicts, the tasks can be divided into $k+1$ categories. T_N: tasks unaffected by any conflict. $T_{\mathrm{C}i}$: tasks affected by the i-th conflict but not by subsequent conflicts. The change in total travel distance can be expressed as:

$$|J_k - J_0| \leqslant \Delta \sum_{i=0}^{k-1} (1+\Delta)^i F_\mathrm{N}\Big(\sum_{j=i+1}^{k} T_{\mathrm{C}j}\Big) + k\xi. \tag{28}$$

By analyzing the formula, to minimize the changes in total travel distance under conflict scenarios, both the allocation algorithm f and the conflict resolution method need to meet certain requirements: The allocation algorithm f should possess stability and consistency to ensure that Δ is as small as possible under minor perturbations. Additionally, the number of conflicts and the size of the conflict-affected task sets should be minimized. The conflict resolution method should also aim to minimize the number of conflicts and exhibit good real-time performance to ensure ξ remains as small as possible.

5 Experiments

This section is divided into two subsections. The first subsection conducts experiments on the task allocation method SubG-AM proposed in this paper and compares it with other commonly used methods for similar task allocation problems to demonstrate the superiority of the proposed approach. The second subsection validates the effectiveness of the proposed SANN-CR framework under no-communication conditions through both

simulation and real-world experiments.

5.1 Real-time task allocation

The type of MRTA problem is described in Section 3, which is similar to the MTSP [42]. However, unlike the standard problem, it requires accounting for the impact of obstacles. We set the environment as a normalized 1×1 plane, where the starting points of the robots, task points, and obstacles are randomly generated. For the same reasons, we set the number of robots to $2 \leqslant N_r \leqslant 5$, the number of tasks to $5 \leqslant N_s \leqslant 12$, and the number of obstacles to $3 \leqslant N_{\text{ob}} \leqslant 20$. Users can adjust the task scale of the dataset according to their specific needs. The hyperparameter settings are shown in the Table 2.

Table 2 Hyperparameter settings.

Hyperparameter	Value
Learning rate	10^{-4}
Learning rate decay	0.98 (every 50 epochs)
Batch size	256
Batches	1024
Epochs	500
Encoder layers	3
Subgraph layers	3
Attention heads	8
Embedding size	128
Optimizer	Adam

A dataset of 1024 batches is generated under the aforementioned problem scales, with each batch containing 256 map data samples. These data include normalized robot, task, and obstacle coordinates. A* transition costs evaluate training/test sequences; online SubG-AM uses coordinates only. We trained the model on an RTX 4090 GPU with 24 GB of memory.

To demonstrate that the proposed method performs better in MRTA problem, we compare it with the following methods using computation time and total travel distance as evaluation metrics.

(1) Greedy Algorithm. This is an online algorithm that, at each step, calculates the shortest distance from all robots to all task points (the distance is computed using the A* algorithm). The robot-task pair with the shortest distance is then selected for assignment, and the robot's position is updated, continuing until all task points are assigned.

(2) Genetic Algorithm. This is an evolutionary algorithm that simulates the process of biological evolution to gradually optimize the task allocation results. We set the population size of the genetic algorithm to 50, with 500 iterations, and used bubble sort for population selection.

(3) CO-AM [29]. This is a method that utilizes attention mechanism to solve routing-type combinatorial optimization problems. Since the MRTA problem can be considered a routing-type combinatorial optimization problem, we use it as one of the comparison methods. However, the original method does not account for the potential presence of obstacles in real-world environments. To adapt it to our problem type, we made adjustments to the original implementation by incorporating the obstacle vertices into the attention mechanism. This is done by fully connecting the vertices and averaging their values, in order to account for the influence of obstacles. The training scale, hyperparameters, and hardware setup for this method are consistent with those used in our proposed approach.

The test datasets are randomly generated under different scales, with the number of robots N_{r} set to 2, 4, 6, and 8, N_{t} set to 5, 10, 15, and 20, and N_{ob} set to 5, 10, 15, and 20. For each scale, 100 test instances were generated. The generation method was consistent with that of the training set but used different random seeds. Specifically, robot initial points, tasks, and obstacle center points were randomly generated using a uniform distribution, ensuring that robot initial points and task points were not located within obstacles. We intentionally set scales of some test to exceed the range of the training set to evaluate the model's generalization capability. All methods were tested on a server equipped with an Intel(R) Xeon(R) Platinum 8357B CPU. Table 3 selects $N_{\text{r}} = 2, 8$, $N_{\text{t}} = 10, 20$, and $N_{\text{ob}} = 5, 20$; Fig. 4A includes all four robot counts at $N_{\text{t}} = N_{\text{ob}} = 20$. Table entries average 100 instances; comparisons are descriptive, not significance tests.

To provide a more intuitive understanding of the task allocation results for the four methods, we randomly selected a test instance under the scale of $N_{\text{r}} = 4$, $N_{\text{t}} = 20$, and $N_{\text{ob}} = 20$ to demonstrate the task allocation paths of the four methods. A* generates the comparison paths from each method's task sequences in Fig. 5. Physical execution uses Hybrid A* instead (Section 4.1).

In Table 3, the time represents the average computation time of each method, measured in seconds. For deep learning-based methods, this corresponds to the inference time. For adjacency matrix-based methods, the time is the sum of the method's own execution time and the time required to compute the adjacency matrix, represented as the total time by using the "+" symbol to connect them. All timings cover allocation only, excluding downstream A* or Hybrid A* path generation. Under these settings, SubG-AM and CO-AM allocate faster than the matrix-based methods. As the problem size increases, the time required to compute the adjacency matrix grows rapidly, making these methods unsuitable for real-time applications. In terms of optimization quality, for smaller problem scales, our method is slightly inferior to the genetic algorithm but outperforms the greedy algorithm

Table 3 Performance comparison of four methods at selected task scales.

N_r	N_t	N_{ob}	Method	Cost(m)	Time(s)
2	10	5	SubG-AM	2.436	0.0283
			CO-AM	2.626	0.0264
			Greedy	2.702	6.2E-5 + 5.5*
			GA	2.359	62.9 + 5.5
		20	SubG-AM	2.522	0.0293
			CO-AM	2.766	0.0266
			Greedy	2.776	6.1E-5 + 5.2
			GA	2.442	62.3 + 5.2
	20	5	SubG-AM	3.820	0.0911
			CO-AM	4.895	0.0874
			Greedy	4.158	1.79E-4 + 18.2
			GA	3.560	99.2 + 18.2
		20	SubG-AM	3.836	0.0915
			CO-AM	4.489	0.0885
			Greedy	4.260	1.78E-4 + 15.5
			GA	3.624	101.3 + 15.5
8	10	5	SubG-AM	1.733	0.0701
			CO-AM	2.078	0.0676
			Greedy	1.974	6.21E-5 + 11.5
			GA	2.063	71.3 + 11.5
		20	SubG-AM	1.761	0.0725
			CO-AM	2.043	0.0678
			Greedy	1.911	6.22E-5 + 10.7
			GA	2.054	68.9 + 10.7
	20	5	SubG-AM	3.079	0.1127
			CO-AM	3.724	0.1097
			Greedy	3.364	1.82E-4 + 27.7
			GA	3.490	101.6 + 27.7
		20	SubG-AM	3.162	0.1144
			CO-AM	3.620	0.1086
			Greedy	3.370	1.82E-4 + 25.5
			GA	3.560	91.1 + 25.4

* For adjacency matrix-based methods, the time is the sum of the method's own execution time and the time required to compute the adjacency matrix, represented as the total time by using the "+" symbol to connect them.

and CO-AM. However, as the problem scale increases, the optimization quality of our method surpasses that of the genetic algorithm, as well as the other two methods. This is because, with larger problem sizes, the genetic algorithm requires significantly larger population sizes and more iterations to ensure optimal results. From the perspective of computation time, the current hyperparameter settings for the genetic algorithm already far exceed the requirements for real-time performance.

Fig.4 illustrates the variation in total task cost of the four methods as the number of robots, tasks, and obstacles change. As the number of robots increases, the total task cost of all four methods decrease. However, due to the exponential growth in the number of potential task sequences, the advantage of the genetic algorithm gradually diminishes. In contrast, as the number of tasks increases, the total task cost of all four methods rise correspondingly, and the superiority of the proposed method becomes increasingly evident. The number of obstacles, however, does not significantly affect the

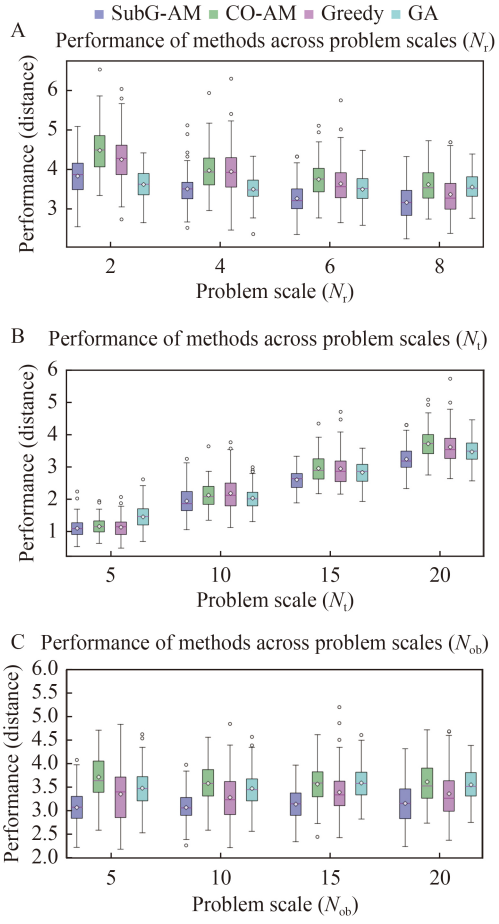


Fig. 4 Comparison of total task cost across four methods under varying problem scales. (A) Robot counts $N_r = 2,4,6,8$ with $N_t = N_{ob} = 20$, including the intermediate counts omitted from Table 3. (B) Setting $N_r = 6$, $N_{ob} = 20$, the total task cost are evaluated as the number of tasks N_t varies. (C) Keeping $N_r = 8$, $N_t = 20$, the total task cost are examined as the number of obstacles N_{ob} changes.

performance of the four methods.

Under the tested settings, SubG-AM is faster and has lower cost than CO-AM.

Regarding task cost, when the task scale is small, SubG-AM performs slightly worse than the GA algorithm, which has an extremely long execution time. However, as the task scale increases, SubG-AM surpasses the GA algorithm in performance. Furthermore, on the tested problem scales beyond the training set's range, our method achieves the best performance among the four methods; this is empirical generalization within the tested distributions.

5.2 Framework for MRTA-NC problem

We evaluate SANN-CR through simulation and physical ablations.

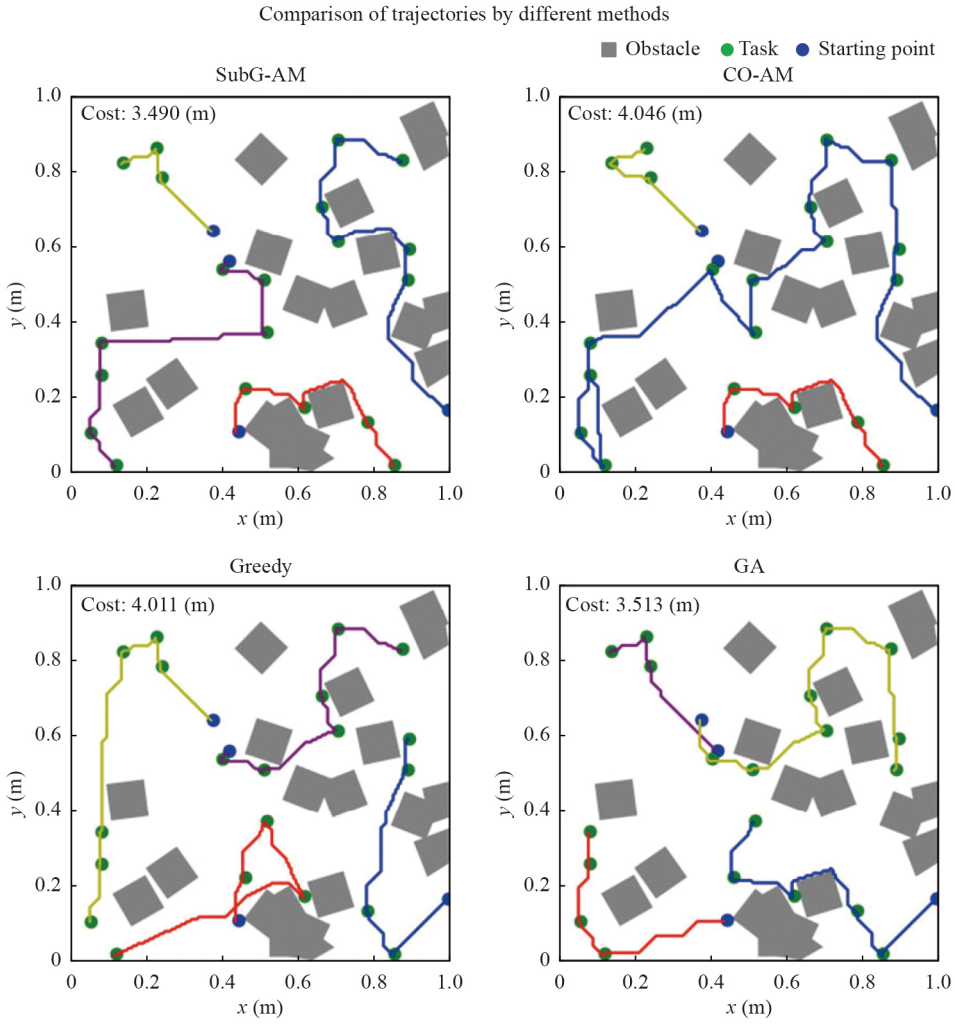


Fig. 5 Comparison of task allocation trajectories for different methods.

Two variants isolate the effects of periodic allocation and conflict resolution.

(1) Single-Step Allocation. Each robot allocates once from the common initial state and executes its own sequence without updates. Initial assignments agree, but tasks left unfinished by planning failures or malfunctions cannot be reassigned.

(2) Multi-Step Allocation. This variant retains periodic allocation and intention judgement but removes conflict resolution. Stopped peers and unfinished-task candidates are identified locally as in Section 4.4; stopping causes remain unknown. Reallocation can recover unfinished tasks, but asynchronous decisions may assign the same task to multiple robots, increasing travel distance or reducing the time-decayed reward in Eq. (6).

Considering both real-time performance and optimization, the task allocation algorithm for the three frameworks utilizes the SubG-AM method proposed in this paper. The Hybrid A* algorithm is used for path planning, and a nonlinear MPC controller is employed for trajectory tracking. In the simulation experiments, we generated four different scales of test sets, each consisting of 100 test instances. The number of obstacles is fixed at 10, with robot counts of 2, 3, 4, and 5, and task counts of 5, 10, 15, and 20. The results of the ablation experiments are shown in Fig.7 and Table 4. Table 4 presents the number of test instances where tasks were successfully completed for each of the three frameworks across four problem scales (with 100 different test instances for each task scale). The results indicate that in the single-step allocation framework, where the entire reallocation module is removed, task allocation consistency is ensured, but task completion cannot be guaranteed. In the multi-step allocation framework, where the conflict resolution process is removed, task completion is ensured, but task allocation consistency is compromised, leading to an increase in the total travel distance of the robots.

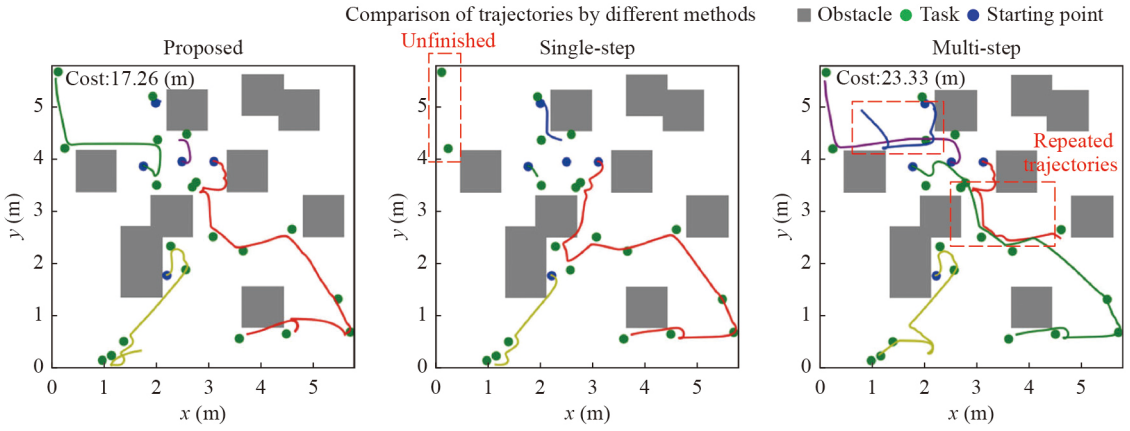


Fig. 6 Comparison of robot motion trajectories under different frameworks. In Scale 4 ($N_r = 5$, $N_t = 20$, $N_{ob} = 10$), a randomly selected test instance was used to plot the robot motion trajectories under the three frameworks. In the single-step allocation framework, the robot trajectories are relatively short, but some tasks remain uncompleted. In the multi-step allocation framework, all tasks are completed; however, some tasks are redundantly executed.

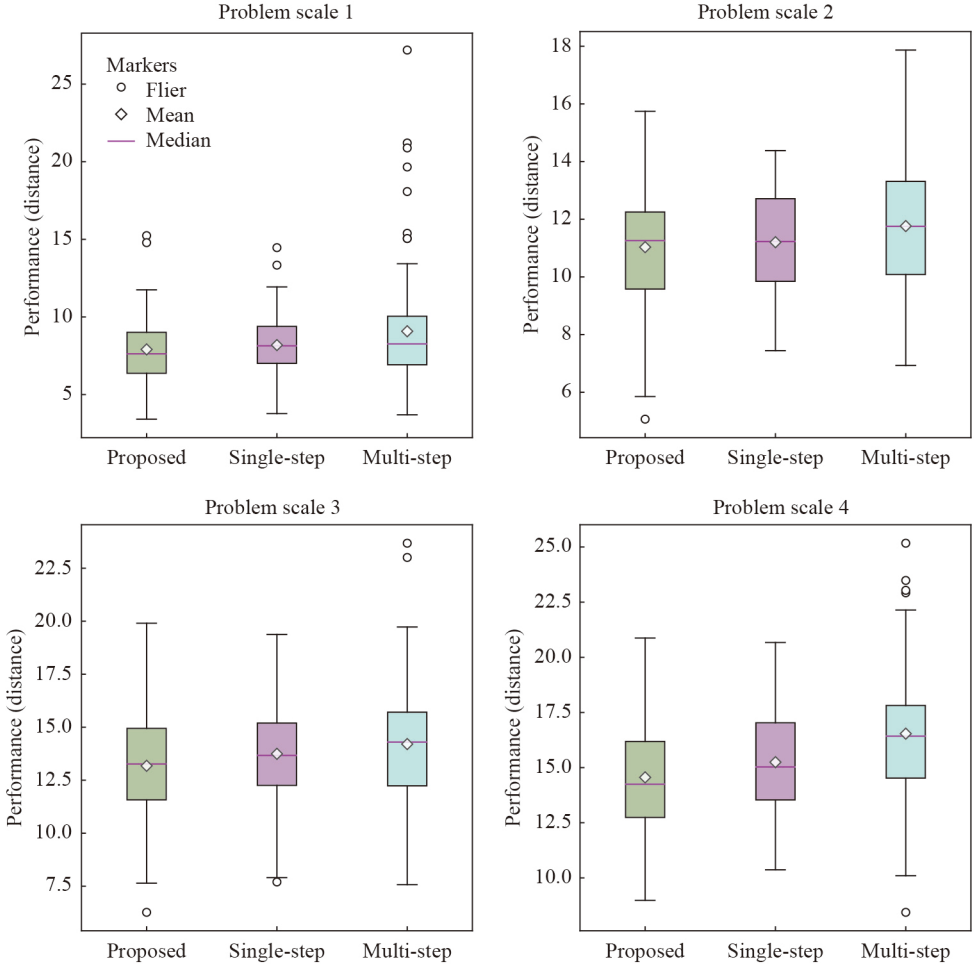


Fig. 7 Ablation experiment results comparison across different problem scales. In Scale 1, $N_r = 2$, $N_t = 5$, $N_{ob} = 10$; In Scale 2, $N_r = 3$, $N_t = 10$, $N_{ob} = 10$; In Scale 3, $N_r = 4$, $N_t = 15$, $N_{ob} = 10$; In Scale 4, $N_r = 5$, $N_t = 20$, $N_{ob} = 10$.

Table 4 Number of test instances successfully completing all tasks for different methods and problem scales.

Method	Scale 1	Scale 2	Scale 3	Scale 4
Proposed	100	100	100	100
Single-step	92	89	78	76
Multi-step	100	100	100	100

To visually illustrate the differences among the three frameworks, Fig.6 presents the robot motion trajectories for a test instance in Scale 4.

The results of the ablation experiments demonstrate that the proposed framework effectively addresses the MRTA problem under no-communication conditions. It ensures both task allocation consistency and task completion while achieving relatively optimal performance in terms of optimization.

Fig. 8 illustrates the experimental setup, equipment, and results of the physical experiments. In the physical experiments, with 2 robots, 5 tasks, and 10 obstacles, we compared the performance of the three frameworks. The 5.8 m × 5.8 m area contains 0.45 m square obstacles represented by $p = 4$ vertices, as in training. The robots used are tracked robots, and the kinematic aspects of motion are modeled using the differential drive kinematics model. The robots have a maximum wheel speed of 0.5 m/s and a wheelbase of 0.238 m. The onboard computers are equipped with Jetson Orin NX running Ubuntu 20.04, and the robots use the ROS2 Foxy version for their operating system. Allocation, intention judgement, and conflict resolution averaged 5.3 ms combined on Orin NX, excluding path planning. The Hybrid A* algorithm is used for path planning, a nonlinear MPC controller is employed for trajectory tracking, and a PID controller is used for wheel speed control. Hybrid A* uses the point-state model and robot-size obstacle inflation defined in Section 4.1. Update periods, distinct from computation time, are 1.5 s for reallocation, 0.1 s for planning and tracking, and 0.02 s for wheel control. OptiTrack supplies each robot only its own pose; onboard LiDAR and cameras estimate peer motion. All decisions, planning, and control run onboard. Experimenters supply only start/stop commands; robots exchange no messages. Evaluation determines task completion from executed trajectories without providing shared runtime completion flags.

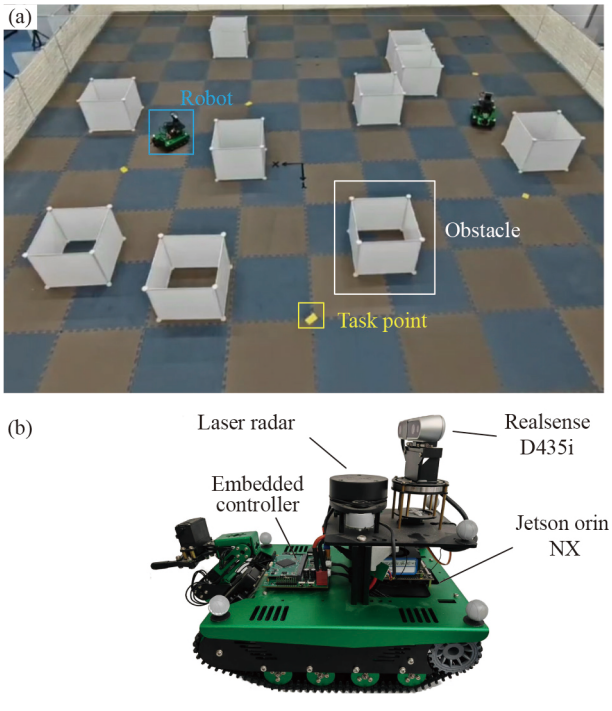


Fig. 8 Physical Experiments. (A) One of the physical experimental scenarios; (B) Structural diagram of the tracked robots used in the experiment.

In physical experiment scenario 1, the initial task allocation under the proposed framework SANN-CR assigned Robot 1 to tasks 1, 3, and 5, while Robot 2 was assigned tasks 2, 4, and 5. When both robots attempted to execute task 5, a conflict occurred. Robot 1 resolved the conflict using the conflict resolution module, yielding task 5. Since no other tasks were available, Robot 1 stopped. Robot 2, also leveraging the conflict resolution module, decided to continue executing task 5. In the single-step allocation framework, Robot 1 performs tasks 1, 3, and 5, while Robot 2 performs tasks 2 and 4. Due to the sparse obstacles in scenario 1, both robots are able to successfully complete path planning. Under the multi-step allocation framework, Robot 1 executed tasks 1, 3, and 5, while Robot 2 executed tasks 2, 4, and 5. When both robots attempted task 5, a conflict occurred. Since there was no conflict resolution module, both robots continued executing task 5, resulting in redundant execution of the task.

In physical experiment scenario 2, the distance between obstacles is relatively small. Robot-size obstacle inflation closes narrow passages, splitting the area into two connectivity domains. The initial allocation result for all three frameworks is as follows: Robot 1 performs tasks 1 and 3, and Robot 2 performs tasks 2, 4, and 5. Before Robot 2 moves from task 4 to task 5, path planning fails due to insufficient space between the obstacles, as the Hybrid A* algorithm cannot find a feasible path. Since both the SANN-CR and multi-step frameworks perform reallocation in each decision cycle, after Robot 2 stops, Robot 1 can reallocate task 5 to itself. However, the single-step framework is unable to complete the task. After Hybrid A* failure, the robot checks its observations and local task flags: it stops if no unfinished task remains; otherwise, SANN-CR reallocates a target compatible with inferred peer intentions.

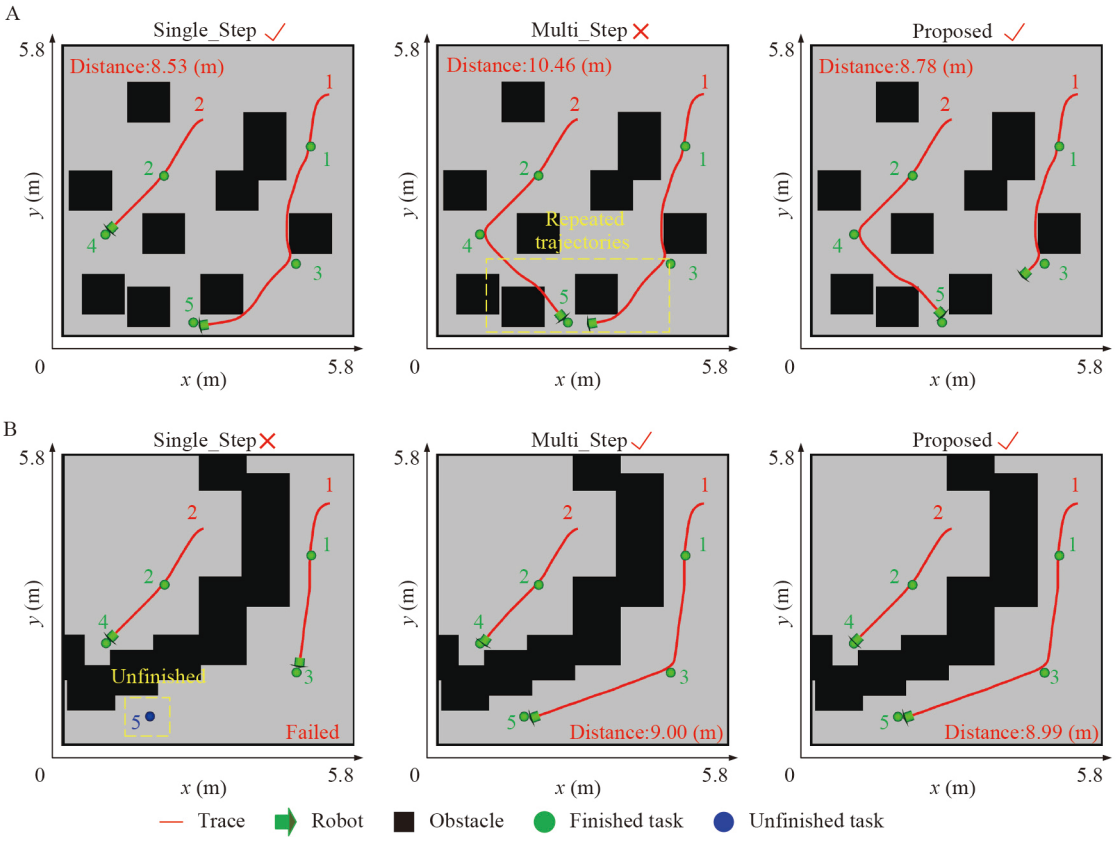


Fig. 9 Physical Experiments Trajectories including two scenarios. (A) Experimental trajectories of the three frameworks in a fully connected scenario; (B) Experimental trajectories of the three frameworks in a scenario with connectivity domain disruption (where path planning failures may occur).

These experiments show reallocation within the decision period and reduced redundant execution under the tested conditions. Recovery reassigns locally identified unfinished tasks after observed stops, without identifying their causes.

6 Conclusions

This study presented SANN-CR, a decentralized framework for routing-based MRTA without inter-robot message exchange. The framework integrates SubG-AM allocation, trajectory-based intention inference, and conflict resolution to reconcile locally generated task assignments. SubG-AM encodes robot, task, and obstacle coordinates directly, avoiding full pairwise state-node cost-matrix computation during online inference. Intention inference and selective reallocation enable robots to adapt their task sequences to observed peer behavior. Allocation comparisons showed lower travel costs than the greedy and CO-AM baselines under the tested settings. In simulation ablations, SANN-CR completed all tasks in all 100 instances at each of four problem scales and reduced redundant execution relative to the variant without conflict resolution. Physical experiments demonstrated the reassignment of unfinished tasks following path-planning failures. Allocation, intention inference, and conflict resolution together averaged 5.3 ms on a Jetson Orin NX, excluding path planning, supporting the feasibility of frequent onboard allocation updates.

These findings rely on shared initial maps and task lists, self-localization, onboard peer observations, and local completion records; physical self-localization was externally aided. Task execution still requires Hybrid A* feasibility checks and path generation with robot-size obstacle inflation. Observed stops do not identify their causes, and the framework includes neither an explicit temporary-wait state nor global completion or failure records. Its applicability therefore remains confined to the stated information assumptions and routing-based task formulation. Future work will extend the framework to task time windows, heterogeneous collaboration, and additional MRTA constraints.

Data availability No datasets were generated or analyzed for this work.

Acknowledgments This work was supported in part by the National Natural Science Foundation of China under Grant 62473109, in part by the Heilongjiang Provincial Science and Technology Talent Support Program under Grant CYQN24036, and in part by the Fundamental Research Funds for the Central Universities under Grant XNJKKGYDJ2026008. The authors used GPT-5.6 as a language-editing tool to improve grammar, clarity, and readability. The AI tool was not used for data analysis, data generation, image processing, or scientific content creation. All outputs were carefully reviewed and verified by the authors, who take full responsibility for the final content.

Competing interests The authors declare no conflicts of interest.

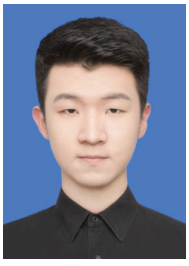
References

1. Yao W R, Tian H Y, Liu J X, et al. Time-extended consensus for multi-agent non-interactive collaboration. Automatica, 2025, 171: 111985
2. Walter V, Staub N, Franchi A, et al. UVDAR system for visual relative localization with application to leader–follower formations of multirotor UAVs. IEEE Robotics and Automation Letters, 2019, 4(3): 2637−2644
3. Petráček P, Walter V, Báča T, et al. Bio-inspired compact swarms of unmanned aerial vehicles without communication and external localization. Bioinspiration & Biomimetics, 2021, 16(2): 026009
4. Horyna J, Baca T, Walter V, et al. Decentralized swarms of unmanned aerial vehicles for search and rescue operations without explicit communication. Autonomous Robots, 2023, 47(1): 77−93
5. Gerkey B P, Mataric M J. Sold!: auction methods for multirobot coordination. IEEE Transactions on Robotics and Automation, 2002, 18(5): 758−768
6. Choi H L, Brunet L, How J P. Consensus-based decentralized auctions for robust task allocation. IEEE Transactions on Robotics, 2009, 25(4): 912−926
7. Quinton F, Grand C, Lesire C. Market approaches to the multi-robot task allocation problem: a survey. Journal of Intelligent & Robotic Systems, 2023, 107(2): 29
8. Arkin R C. Cooperation without communication: multiagent schema-based robot navigation. Journal of Robotic Systems, 1992, 9(3): 351−364
9. Parker L E. ALLIANCE: an architecture for fault tolerant multirobot cooperation. IEEE Transactions on Robotics and Automation, 1998, 14(2): 220−240
10. Dai W, Lu H M, Xiao J H, et al. Task allocation without communication based on incomplete information game theory for multi-robot systems. Journal of Intelligent & Robotic Systems, 2019, 94(3−4): 841−856
11. Talamali M S, Saha A, Marshall J A R, et al. When less is more: robot swarms adapt better to changes with constrained communication. Science Robotics, 2021, 6(56): eabf1416
12. Gerkey B P, Matarić M J. A formal analysis and taxonomy of task allocation in multi-robot systems. The International Journal of Robotics Research, 2004, 23(9): 939−954
13. Korsah G A, Stentz A, Dias M B. A comprehensive taxonomy for multi-robot task allocation. The International Journal of Robotics Research, 2013, 32(12): 1495−1512
14. Gao J Y, Sun C, Zhao H, et al. VectorNet: encoding HD maps and agent dynamics from vectorized representation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. Seattle, WA, USA, 2020, 11522–11530
15. Alahi A, Goel K, Ramanathan V, et al. Social LSTM: human trajectory prediction in crowded spaces. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Las Vegas, NV, USA, 2016, 961–971
16. Casas S, Luo W J, Urtasun R. IntentNet: learning to predict intention from raw sensor data. In: Proceedings of the 2nd Annual Conference on Robot Learning. Zürich, Switzerland, 2018, 947–956
17. Salzmann T, Ivanovic B, Chakravarty P, et al. Trajectron++: dynamically-feasible trajectory forecasting with heterogeneous data. In: Proceedings of the 16th European Conference on Computer Vision. Glasgow, UK, 2020, 683–700
18. Kuhn H W. The Hungarian method for the assignment problem. Naval Research Logistics Quarterly, 1955, 2(1–2): 83−97
19. Gombolay M C, Wilcox R J, Shah J A. Fast scheduling of robot teams performing tasks with temporospatial constraints. IEEE Transactions on Robotics, 2018, 34(1): 220−239
20. Gendreau M, Hertz A, Laporte G, et al. A generalized insertion heuristic for the traveling salesman problem with time windows. Operations Research, 1998, 46(3): 330−335
21. Zhu D Q, Cao X, Sun B, et al. Biologically inspired self-organizing map applied to task assignment and path planning of an AUV system. IEEE Transactions on Cognitive and Developmental Systems, 2018, 10(2): 304−313
22. Eun Y, Bang H. Cooperative task assignment/path planning of multiple unmanned aerial vehicles using genetic algorithm. Journal of Aircraft, 2009, 46(1): 338−343
23. Edison E, Shima T. Integrated task assignment and path optimization for cooperating uninhabited aerial vehicles using genetic algorithms. Computers & Operations Research, 2011, 38(1): 340−356
24. Zhang L X, Zhao J Z, Lamon E, et al. Energy efficient multi-robot task allocation constrained by time window and precedence. IEEE Transactions on Automation Science and Engineering, 2025, 22: 18162−18173
25. Tian H Y, Yao W R, Dong B, et al. Task-extended utility tensor method for decentralized multi-vehicle mission planning. IEEE Transactions on Automation Science and Engineering, 2024, 21(4): 6081−6093
26. Scarselli F, Gori M, Tsoi A C, et al. The graph neural network model. IEEE Transactions on Neural Networks, 2009, 20(1):

61−80
27. Gilmer J, Schoenholz S S, Riley P F, et al. Neural message passing for quantum chemistry. In: Proceedings of the 34th International Conference on Machine Learning. Sydney, NSW, Australia, 2017, 1263–1272
28. Cappart Q, Chételat D, Khalil E B, et al. Combinatorial optimization and reasoning with graph neural networks. The Journal of Machine Learning Research, 2023, 24(1): 130
29. Kool W, van Hoof H, Welling M. Attention, learn to solve routing problems!. In: Proceedings of the 7th International Conference on Learning Representations (ICLR). New Orleans, LA, USA, 2019
30. Wang Z Y, Gombolay M. Learning scheduling policies for multi-robot coordination with graph attention networks. IEEE Robotics and Automation Letters, 2020, 5(3): 4509−4516
31. Vinyals O, Fortunato M, Jaitly N. Pointer networks. In: Proceedings of the 29th International Conference on Neural Information Processing Systems. Montreal, Canada, 2015, 2692–2700
32. Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. Long Beach, CA, USA, 2017, 6000–6010
33. Yu J J Q, Yu W, Gu J T. Online vehicle routing with neural combinatorial optimization and deep reinforcement learning. IEEE Transactions on Intelligent Transportation Systems, 2019, 20(10): 3806−3817
34. Dai H J, Khalil E B, Zhang Y Y, et al. Learning combinatorial optimization algorithms over graphs. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. Long Beach, CA, USA, 2017, 6351–6361
35. Balch T, Arkin R C. Communication in reactive multiagent robotic systems. Autonomous Robots, 1994, 1(1): 27−52
36. Payton D, Daily M, Estowski R, et al. Pheromone robotics. Autonomous Robots, 2001, 11(3): 319−324
37. Kanakia A, Touri B, Correll N. Modeling multi-robot task allocation with limited information as global game. Swarm Intelligence, 2016, 10(2): 147−160
38. Xu Y H, Wei Y R, Wang D, et al. Multi-UAV path planning in GPS and communication denial environment. Sensors, 2023, 23(6): 2997
39. Cesare K, Skeele R, Yoo S H, et al. Multi-UAV exploration with limited communication and battery. In: 2015 IEEE International Conference on Robotics and Automation (ICRA). Seattle, WA, USA, 2015, 2230–2235
40. Matignon L, Jeanpierre L, Mouaddib A I. Coordinated multi-robot exploration under communication constraints using decentralized Markov decision processes. In: Proceedings of the 26th AAAI Conference on Artificial Intelligence. Toronto, Ontario, Canada, 2012, 2017–2023
41. Alitappeh R J, Jeddisaravi K. Multi-robot exploration in task allocation problem. Applied Intelligence, 2022, 52(2): 2189−2211
42. Chakraa H, Guérin F, Leclercq E, et al. Optimization techniques for multi-robot task allocation problems: review on the state-of-the-art. Robotics and Autonomous Systems, 2023, 168: 104492



Zirui Wu received the B.S. and M.S. degrees in Automation from Harbin Institute of Technology, Harbin, China, in 2023 and 2026, respectively. His research interests include reinforcement learning, multi-robot mission planning, and multi-agent control systems.



Jianing Hu received the B.S. degree in Automation from Harbin Institute of Technology, Harbin, China, in 2023. He is currently pursuing the Ph.D. degree in Control Science and Engineering at Harbin Institute of Technology. His research interests include unmanned vehicles, interaction awareness of unmanned vehicles, and path planning.



Weiran Yao received the B.S. (with Hons.), M.S. and Ph.D. degrees in aeronautical and astronautical science and technology from the School of Astronautics, Harbin Institute of Technology (HIT), Harbin, China, in 2013, 2015, and 2020, respectively. At present, he is a Full Professor at the Department of Control Science and Engineering, HIT. His research interests include autonomous decision-making of unmanned systems, multi-robot task planning and control, etc.



Jianxing Liu received the B.S. degree in Mechanical Engineering and the M.E. degree in Control Science and engineering from the Harbin Institute of Technology, Harbin, China, in 2004 and 2010, respectively, and the Ph.D. degree in Automation from the Technical University of Belfort-Montbeliard, Belfort, France, in 2014. He is currently a Professor in the Department of Control Science and Engineering, Harbin Institute of Technology. His current research interests include nonlinear control and observation, computational and intelligent systems and and robot control. He is currently serving as an Associate Editor for several journals, including *IEEE Transactions on Industrial Electronics*, *IEEE Transactions Aerospace and Electronic Systems*, *IEEE Transactions Automation Science and Engineering*, *Control Engineering Practice*, *IEEE Systems Journal*, *Nonlinear Dynamics*. He is also an Associate Editor of the Conference Editorial Board, *IEEE Control Systems Society*.



Sergio Vazquez was born in Seville, Spain, in 1974. He received the M.S. and Ph.D. degrees in industrial engineering from the University of Seville (US), Seville, in 2006 and 2010, respectively. Since 2002, he has been with the Power Electronics Group working in R&D projects. He is currently an Associate Professor with the Department of Electronic Engineering, US. His research interests include power electronics systems and modeling, modulation, and control of power electronics converters applied to renewable energy technologies. Dr. Vazquez was the recipient as a coauthor of the 2012 Best Paper Award of the *IEEE Transactions on Industrial Electronics* and 2015, 2022, and 2023 Best Paper Awards of the *IEEE Industrial Electronics Magazine*. He is involved in the Energy Storage Technical Committee of the IEEE Industrial Electronics Society and is currently an Associate Editor for *IEEE Transactions on Industrial Electronics*.



Jose I. Leon was born in Cadiz, Spain. He received the B.S., M.S., and Ph.D. degrees in Telecommunications Engineering from Universidad de Sevilla, Seville, Spain, in 1999, 2001, and 2006, respectively. He is currently an Associate Professor with ENGREEN, Universidad de Sevilla. Since 2019, he has also been a Chair Professor with the Harbin Institute of Technology, Harbin, China. His research interests include modulation and control of power converters for high-power applications and renewable energy systems. Dr. Leon was a corecipient of the 2008, 2015, 2020, and 2022 Best Paper Award of *IEEE Industrial Electronics Magazine*, and the 2012 Best Paper Award of the *IEEE Transactions on Industrial Electronics*. He was the recipient of the 2014 IEEE J. David Irwin Industrial Electronics Society Early Career Award, the 2017 IEEE Bimal K. Bose Energy Systems Award, and the 2017 Manuel Losada Villasante Award.



Leopoldo G. Franquelo was born in Malaga, Spain. He received the M.Sc. and Ph.D. degrees in Electrical Engineering from the Universidad de Sevilla, Seville, Spain, in 1977 and 1980, respectively. He was an Associate Professor from 1982 to 1986 with the Electronics Engineering Department, Sevilla University, where since 1986, he has been a Professor. He has also been a Professor with the Department of Control Science and Engineering, Harbin Institute of Technology, Harbin, China, since 2016. He has participated in more than 100 Industrial and R&D projects and has authored and co-authored more than 300 papers, 108 of them in *IEEE* Journals and 25 of them in *IEEE*. His current research interests include modulation techniques for multilevel inverters and application to power electronic systems for renewable energy systems. Dr. Franquelo has been an *IEEE Industrial Electronics Society* (IES) Distinguished Lecturer since 2006. He became an Associate Editor in 2007, Co-Editor-in-Chief in 2014, and an Editor-in-Chief in 2016–2018 for the *IEEE Transactions on Industrial Electronics*. He is currently an Editor-in-Chief for *IEEE Open Journal of the Industrial Electronics Society* . He was a Member-at-Large of the IES Administrative Committee (AdCom) (2002–2003), the Vice President for Conferences (2004–2007), and the President Elect of the IES (2008–2009). He was the President of the IES (2010–2011) and is an IES AdCom Life member. In 2009 and 2013, he received the prestigious Andalusian Research Award and FAMA Award recognizing the excellence of his research career. He was the recipient of a number of Best Paper Awards from *IEEE* journals.