Engineered Science
DOI: https://dx.doi.org/10.30919/es1689

A Multi-robot Farmland Operation Path Scheduling Method Integrating Jump Point Search and Multi-stage Collaborative Optimized Salp Swarm Algorithm for Balanced Task Allocation

Liwei Yang,[1,#] Ping Li,[2,#] Yun Ge,[1,3,4,5,*] He Zhang,[1,3,4] Fuyang Zhang,[1,3,4] Yijiang Zheng[1,5] and Zhongshuo Ding[1,3,4]

Abstract

To address high path costs, imbalanced task loads, and unstable convergence in multi-robot path planning and scheduling within unstructured farmland environments, this paper presents a balanced scheduling method combining Enhanced Jump Point Search (EJPS) with a Multi-Stage Collaborative Optimization Salp Swarm Algorithm (MCOSSA). Firstly, a farmland grid environment model is constructed to represent spatial accessibility and task locations. EJPS is used for global path planning, incorporating bidirectional jump expansion and adaptive heuristic adjustments. To further refine the path, a path-local segment node reconstruction (PLSNR) is proposed to compress redundant nodes and enhance path smoothness. Secondly, MCOSSA is employed for multi-robot task scheduling. A dual-factor encoding–decoding mechanism is designed, guided by a balanced fitness function minimizing the maximum path cost. The algorithm integrates dynamic subpopulation clustering, adaptive cooperative evolution, non-uniform Gaussian perturbations, and multi-neighborhood local search to improve convergence. Finally, experiments on benchmark functions, farmland maps, and large-scale task allocation scenarios show that the proposed method significantly outperforms particle swarm optimization, salp swarm algorithm, artificial bee colony, and others in terms of path length, load balancing, robustness, and stability, exhibiting strong potential for practical applications.

Keywords: Multi-robot; Farmland environments; Path planning; Task scheduling; Jump point search; Salp swarm algorithm.
Received: 13 June 2025; Revised: 19 July 2025; Accepted: 02 August 2025
Article type: Research article.

1. Introduction

With the rapid evolution of smart agriculture, multi-robot collaborative operations have emerged as a key enabler for enhancing the efficiency of large-scale fieldwork and advancing intelligent agricultural management.[1] In expansive and unstructured farmland environments, such as that characteristic of safflower cultivation, complex terrains and densely distributed tasks introduce substantial challenges. Conventional path planning and task scheduling approaches often fall short in achieving an effective trade-off among path optimality, workload balancing, and operational robustness under real-world constraints.[2] Consequently, there is a pressing need to develop efficient and balanced path scheduling optimization strategies specifically designed for multi-robot systems, in order to drive the intelligent transformation of agricultural equipment and elevate collaborative operational performance.

Path planning,[3] as a core technology for autonomous agricultural robot operations, widely employs algorithms such as A,* Dijkstra for global route construction and obstacle avoidance in agricultural tasks.[4] However, as operational environments expand in scale and complexity, these conventional methods encounter limitations, including low computational efficiency and poor scalability in high-dimensional, large-scale scenarios. To address these issues, an *et al.*[5] proposed an improved Jump Point Search (JPS) algorithm that significantly accelerates grid-based path searching through a jump expansion mechanism. Ni *et al.*[6] applied bidirectional rapidly-exploring random tree for pollination path planning of a pepper-pollination robotic arm, achieving a 95% success rate in narrow and complex environments. Guo *et al.*[7] improved the navigation performance of an electric tracked tractor in greenhouse settings by integrating an enhanced A* algorithm with dynamic window approach.

Meanwhile, bio-inspired intelligent algorithms have gained increasing traction in complex farmland path planning.

[1]*College of Mechanical and Electrical Engineering, Shihezi University, Shihezi, 832000, China*
[2]*College of Mechatronics and Automation Engineering, Xinjiang University of Technology, Hetian, 848023, China*
[3]*Xinjiang Production and Construction Corps Key Laboratory of Modern Agricultural Machinery, Shihezi, 832000, China*

Algorithms such as Ant Colony Optimization (ACO), Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Artificial Bee Colony (ABC), recognized for their strong global search capabilities, have been widely adopted for path optimization in robotic operations.[8] Xin *et al.*[9] introduced a greedy strategy-enhanced PSO for multi-objective task scheduling of unmanned surface vessels. Miao *et al.*[10] applied an improved ACO for indoor robot path optimization. The wheat harvesting problem has been formulated as multi-traveling salesman problem,[11] where He *et al.*[12] used GA to generate optimal coordination paths for harvesters and transport vehicles. Tian *et al.*[13] optimized unmanned aerial vehicle path planning for plant protection tasks based on ACO. In recent years, deep learning and reinforcement learning techniques[14-15] have also been explored, enabling adaptive path optimization under dynamic environments and varying obstacle distributions.

Multi-robot scheduling and collaborative optimization aim to ensure the efficient allocation of tasks and the coordinated planning of paths among multiple operational agents, with key objectives including minimizing total operation time, reducing path costs, and achieving workload balance.[16-17] Classical models such as the multi-traveling salesman problem (mTSP) and the vehicle routing problem provide essential theoretical underpinnings and have been widely adopted in agricultural scenarios to facilitate task scheduling and path coordination among multiple robots.[18] Zhang *et al.*[19] introduced an improved ACO for dynamic task allocation in multi-machine cooperation, enabling real-time determination of optimal scheduling schemes for harvesters and transport vehicles under failure conditions. Cao *et al.*[20] investigated ACO-based collaborative task planning to achieve efficient navigation and scheduling management in dynamic farmland environments. To address large-scale, heterogeneous multi-robot scheduling problems involving priority constraints and multi-objective conflicts, Liu *et al.*[21] proposed a reinforcement learning-enhanced PSO framework, which incorporates joint encoding of task sequences and robot subsets, thereby improving both solution diversity and convergence quality. Guo *et al.*[22] developed a cooperative discrete ABC algorithm for task assignment and scheduling in multi-robot farming systems, in which global exploration and local refinement were enhanced via dynamic neighborhood strategies and collaborative tracking mechanisms. In the context of load balancing, Wang *et al.*[23] proposed a hybrid optimization strategy that integrates the minimum spanning tree approach with ACO to address the mTSP under capacity and time window constraints. By modifying the pheromone update mechanism, their method achieved notable improvements in both solution efficiency and equilibrium. In recent years, coupled optimization of path planning and task allocation has garnered increasing attention, with a growing emphasis on joint modeling as well as modular decoupling strategies. Li *et al.*[24] applied an improved ACO algorithm to jointly optimize path connectivity and task sequencing across multiple farmland plots. Seyyed Hasani *et al.*[25] integrated an enhanced Clark – Wright algorithm with tabu search to optimize path planning for multiple agricultural machines, resulting in a 32% reduction in total operation time. Yang *et al.*[26] decomposed the multi-region routing problem into regional path generation and visitation sequence optimization, which were collaboratively solved using A* and GA, yielding strong generalizability across diverse agricultural environments. Furthermore, Yang *et al.*[27] addressed obstacle-induced uncertainty by incorporating task prioritization and conflict-avoidance strategies, enabling distributed obstacle avoidance optimization under dynamic conditions. Gao *et al.*[28] proposed an automatic scheduling and path optimization approach for multi-region operations based on task equivalent center modeling and the Lin–Kernighan–Helsgaun heuristic algorithm, significantly enhancing operational safety and transition efficiency across complex field environments.

To address the challenges of high path cost, unbalanced workload distribution, and limited convergence performance in multi-robot path planning and task scheduling within unstructured farmland environments, this study proposes a balanced multi-robot path scheduling framework that integrates EJPS and MCOSSA, as illustrated in Fig. 1. The main contributions are as follows:

(1) An EJPS algorithm is proposed, featuring bidirectional expansion and adaptive heuristic adjustment for efficient global path planning in large-scale farmland grid environments. A PLSNR is further introduced to optimize path structure, thereby improving both optimality and smoothness.

(2) A MCOSSA algorithm is developed, incorporating dual-factor encoding–decoding, fitness-cluster-driven dynamic subpopulation grouping, adaptive cooperative evolution, and multi-strategy Gaussian perturbations. This design significantly enhances path balance, convergence speed, and global solution quality.

(3) Comparative experiments conducted on standard benchmark functions, multiple farmland map scenarios, and large-scale task allocation cases demonstrate that the proposed method consistently outperforms PSO, Sine Cosine Algorithm (SCA), ABC, Salp Swarm Algorithm (SSA), and Multi-subpopulation based Symbiosis and Non-uniform Gaussian mutation SSA (MSNSSA)[29] in terms of path length, workload balance, robustness, and solution stability.

2 Problem description and modeling

2.1 Problem description

Fig. 2 presents the spatial distribution of task points across the farm, where each point corresponds to a distinct type of

[4]*Key Laboratory of Northwest Agricultural Equipment, Ministry of Agriculture and Rural Affairs, Shihezi, 832000, China*

[5]*Saffron Industry Research Institute, Shihezi University, Shihezi, 832000, China*

[#]*These authors contributed equally to this work.*

[*]*Email:* gy_mac@shzu.edu.cn (Y. Ge)

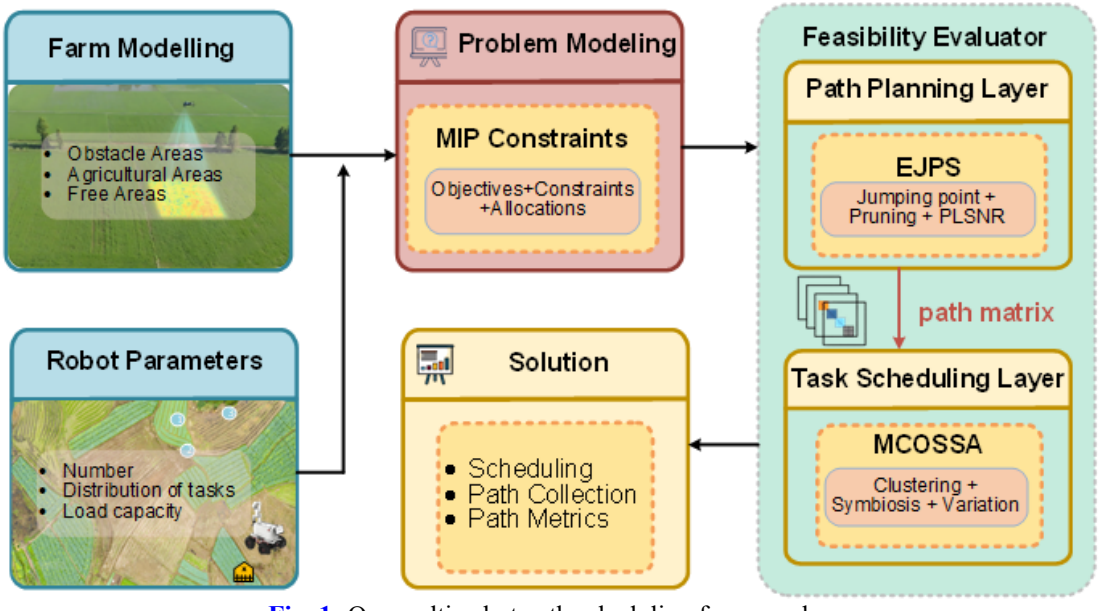


Fig. 1: Our multi-robot path scheduling framework.

agricultural operation. In this scenario, a centralized depot functions as the unified departure and return hub for all agricultural robots. A fleet of homogeneous robots is deployed from the depot to execute assigned tasks at various locations and subsequently return upon completion.

The overarching objective is to devise a scheduling strategy that minimizes the maximum completion time (*i.e.*, the make span across all robots), while ensuring a balanced distribution of task loads under the constraint that all tasks must be completed exactly once. To facilitate model formulation and algorithmic implementation, the following assumptions are adopted:

(1) All robots operate under idealized conditions, with no failures, collisions, or unforeseen interruptions;

(2) The robot fleet is homogeneous, possessing identical locomotion speeds and task execution capabilities;

(3) Robots traverse at constant velocities, and task durations are deterministic and dependent on task types;

(4) Each robot is constrained to operate within its designated region, with task point coordinates known a priori;

(5) All tasks must be completed exactly once; duplication and omission are strictly prohibited.

Conventional scheduling methods based solely on task point coordinates fail to account for the underlying spatial and traversal complexities inherent in real-world field operations. To address this limitation and enhance the fidelity of environmental modeling, a grid-based representation of the farmland is introduced (Fig. 3). The grid discretizes the operational environment into structured cells encoding traversable areas, static obstacles, cultivated zones, the depot, and task point locations. This spatial abstraction enables the incorporation of critical factors such as obstacle avoidance, path feasibility, and environmental constraints into the planning and optimization framework.

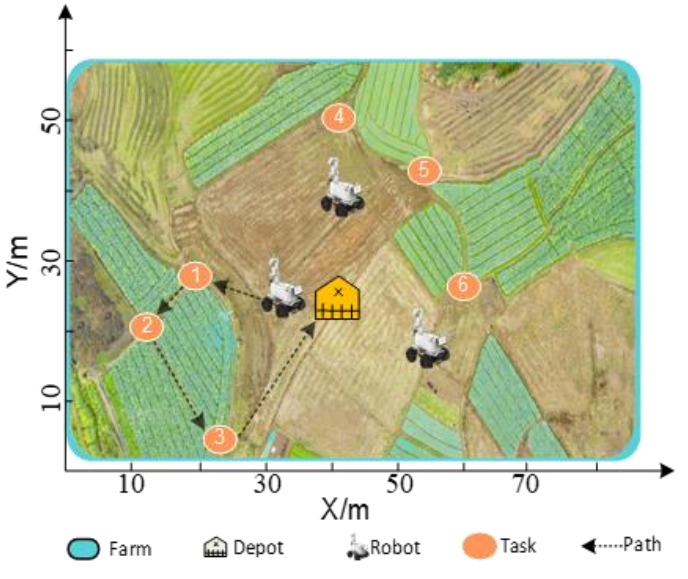


Fig. 2: Schematic of the farm.

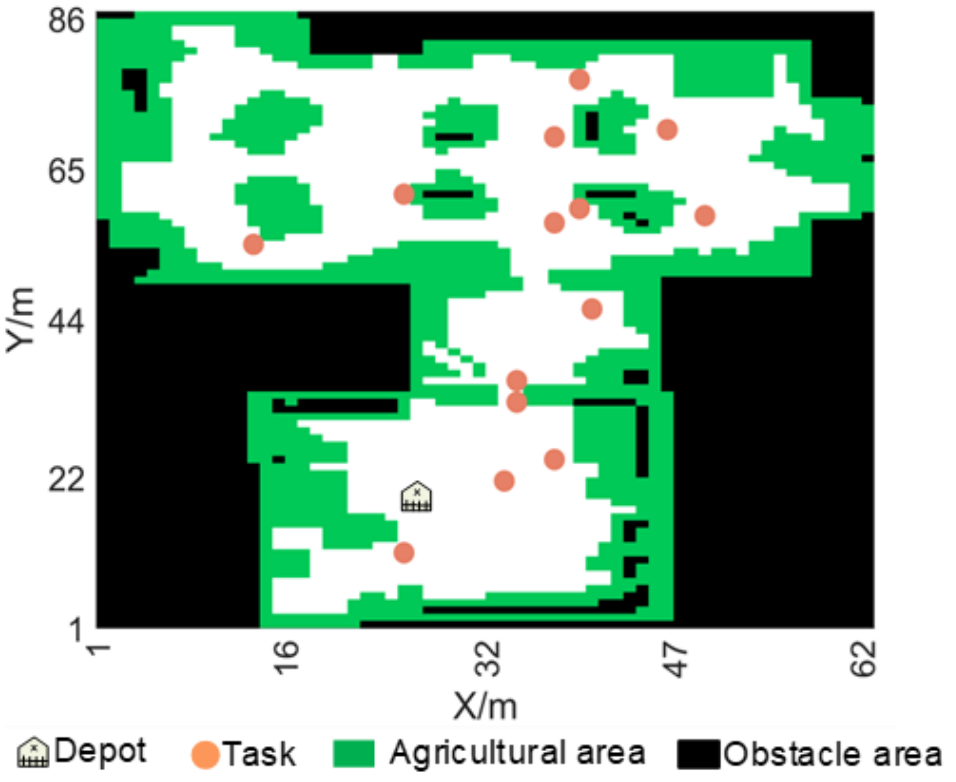


Fig. 3: Grid map.

2.2 Problem modeling

Based on the description in Section 1.1, the mathematical model is established as follows:

Parameters:

$M = \{1,2,\ldots,m\}$	Robot set consisting of m robots.
$T = \{1,2,\ldots,n\}$	Task point set consisting of n tasks.
$V = T \cup \{0\}$	Position set including all points, where 0 denotes the depot and the rest represent task points.
p_{ij}^k	Path of robot k from point i to point j.
c_{ij}^k	Total cost of robot k traveling the path from i to j.
t_j	Fixed operation time required to execute task at point j.
$\mathcal{O} \subset Map$	Set of obstacle cells within the map.
$G = (N, E)$	Grid map modeled as a graph comprising a node set and an edge set.

Decision variables:

x_{ij}^k	Robot k executes the path from task point i to j: 1 if yes, 0 otherwise.0
y_j^k	Whether task point j is assigned to robot k: 1 if yes, 0 otherwise.
$C_k \geq 0$	Total task execution time of robot k.
$C_{max} \geq 0$	Maximum completion time among all robots.

Objective function:

$$\min C_{max} \quad (1)$$

Constraints:

$$\sum_{k\in M} y_j^k = 1, \forall j \in T \quad (2)$$

$$y_j^k = \sum_{i\in V, i\neq j} x_{ij}^k, \forall j \in T, \forall k \in M \quad (3)$$

$$\sum_{j\in T} x_{0j}^k = 1, \sum_{i\in T} x_{i0}^k = 1, \forall k \in M \quad (4)$$

$$\sum_{i\in V, i\neq j} x_{ij}^k = \sum_{l\in V, l\neq j} x_{jl}^k, \forall j \in T, \forall k \in M \quad (5)$$

$$C_k = \sum_{i\in V}\sum_{j\in V} x_{ij}^k \cdot c_{ij}^k + \sum_{j\in T} y_j^k \cdot t_j, \forall k \in M \quad (6)$$

$$C_k \leq C_{max}, \forall k \in M \quad (7)$$

$$x_{ij}^k = 0, \text{if any cell in } p_{ij}^k \in \mathcal{O} \quad (8)$$

$$x_{ij}^k \in \{0,1\}, y_j^k \in \{0,1\}, C_k \geq 0, C_{max} \geq 0 \quad (9)$$

Eq. (1) represents the objective function, aiming to minimize the maximum completion time. Eq. (2) enforces the unique task assignment constraint, ensuring that each task point j is assigned to exactly one robot. Eq. (3) specifies the path consistency constraint, guaranteeing coherence between task allocation and path planning. Eq (4) imposes the start and end point constraints, requiring all robots to depart from and ultimately return to the depot. Eq. (5) ensures path connectivity, stipulating that if a robot visits task point j, it must arrive from a preceding point and proceed to the subsequent point. Eq. (6) calculates the completion time for each robot, accurately reflecting the actual workload and providing a basis for the optimization objective. Eq. (7) synchronizes the maximum completion time variable, ensuring that C_{max} represents the maximum completion time across all robots. Eq. (8) enforces path feasibility constraints, where feasible paths are planned using our EJPS algorithm. Eq. (9) defines variable constraints, ensuring the model conforms to a mixed-integer programming structure.

2.3 Jump point search algorithm

The A* algorithm is a classical heuristic-based search method widely employed for optimal path planning. Its evaluation function is defined as:

$$f(n) = g(n) + h(n) \tag{10}$$

where $f(n)$ denotes the overall evaluation of the current node n; $g(n)$ represents the actual cost from the start node to n; $h(n)$ is the heuristic estimate from n to the goal.

JPS[30] is an efficient variant of A* that substantially improves search performance by limiting node evaluations to critical jump points, thereby eliminating redundant expansions. The core procedure of JPS consists of two main components:

(1) Pruning rules, which discard non-essential adjacent nodes based on directional constraints, effectively compressing the search space;

(2) Jumping rules, which recursively identify and evaluate jump points that satisfy predefined conditions within the grid, enabling the algorithm to guide the search process more efficiently toward the goal.

2.3.1 Pruning rules

In the JPS algorithm, the neighbor set of a node is defined as all its reachable adjacent nodes $neighbours(x)$, where movements along horizontal or vertical directions incur a cost of 1, while diagonal movements incur a cost of 2. The core concept of the pruning rules lies in recursively excluding redundant adjacent nodes that can be optimally reached without passing through the current node, thus effectively reducing unnecessary expansions, compressing the search space, and enhancing search efficiency. This pruning procedure is executed entirely online, requires no preprocessing steps, and introduces no additional memory overhead. The detailed pruning operations are as follows:

Case 1：neighbours (x) without obstacles

when the algorithm expands nodes along straight or diagonal directions, nodes n satisfying Eq. (11) and (12) are pruned, thereby minimizing redundant searches and significantly enhancing the overall search efficiency.

$$len(\langle p(x), \ldots, \mathrm{n}\rangle \backslash x) \leq len(\langle p(x), x, n\rangle) \tag{11}$$

$$len(\langle p(x), \ldots, n\rangle \backslash x) < len(\langle p(x), x, n\rangle) \tag{12}$$

where $len(.)$ denotes the path length cost; $\langle p(x), x, n\rangle$ denotes the path from parent node $p(x)$ traversing through node x to node n, and the direct path from parent node $\langle p(x), \ldots, n\rangle \backslash x$ to node $p(x)$ without passing through node n.

Case 2：neighbours(x) with obstacles

When obstacles are present, Case 1 cannot prune all non-natural adjacent nodes due to obstruction. Therefore, JPS introduces the concept of forced neighbors. Specifically, node n is defined as a forced neighbor if it is not a natural neighbor of node x and simultaneously satisfies the condition described in Eq. (13):

$$len(\langle p(x), x, n\rangle) < len(\langle p(x), \ldots, n\rangle \backslash x) \tag{13}$$

2.3.2 Jumping rules

In the JPS algorithm, jump point search is performed via recursive pruning, retaining only natural and forced neighbors while filtering valid successor jump points by skipping intermediate redundant nodes. A node y is identified as a jump point of node x if there exists the smallest positive integer k such that the jump condition is satisfied $y = x + k\vec{d}$ and at least one of the following criteria holds:

(1) Node y is the goal node.

(2) Node y has at least one forced neighbor.

(3) If the movement direction $\vec{d}$ is diagonal, and there exists a node z such that $z = y + k_i\vec{d}_i$ after moving $k_i (k_i \in N)$ steps in direction $\vec{d_i} (\vec{d_i} \in \{\vec{d_1}, \vec{d_2}\})$ from node y, and node z satisfies either condition (1) or (2), then z is a valid successor jump point of node y.

2.4 MSNSSA algorithm

The SSA algorithm, proposed by Seyedali[31] in 2017, performs global optimization by simulating the cooperative dynamics between leader and follower agents within the population. While effective in exploration, SSA exhibits a tendency to become trapped in local optima when addressing multimodal, high-dimensional, or nonlinear optimization problems—primarily due to its limited adaptive capability in adjusting search behavior. To overcome these shortcomings, Chen *et al.*[29] proposed the MSNSSA algorithm, which introduces a multi-subpopulation framework by dividing the population into three distinct roles: leaders, followers, and tailers.

The leader subgroup, the top one-third of the population, is primarily responsible for global exploration and serves to guide the swarm toward the global optimum. The position of each leader is updated with reference to the food source according to the following expression:[31]

$$x_j^i(t+1) = \begin{cases} F_j + c_1(c_2(ub_j - lb_j) + lb_j), c_3 \geq 0.5 \\ F_j - c_1(c_2(ub_j - lb_j) + lb_j), c_3 < 0.5 \end{cases} \tag{14}$$

where x_j^i denotes the position of the i-th leader salp in the j-th dimension; t is the current iteration number; F_j represents the position of the food source; ub_j and lb_j denote the upper and lower bounds, respectively; c_2 and c_3 are random numbers uniformly distributed in the range [0, 1]. The control parameter c_1 is defined as follows:[31]

$$c_1 = 2e^{-(\frac{4t}{T_{max}})^m} \tag{15}$$

where T_{max} denotes the maximum number of iterations, and the power exponential function is employed to balance exploration and exploitation capabilities.

The follower subgroup, the middle one-third of the population, is tasked with local exploitation. By leveraging a

symbiotic mechanism, it improves the precision of local search while remaining aligned with the global guidance provided by the leaders. The position of each follower is updated according to the following formulation:[31]

$$x_j^i(t+1) = x_j^i(t) + r \cdot (F_j - C \cdot R) \quad (16)$$

$$C = \frac{x_j^i(t) + x_j^{i-1}(t)}{2} \quad (17)$$

where, r is a random number in the range [0, 1]; C denotes the symbiosis coefficient, characterizing the relationship between the salps of i and $i-1$; $R \in \{1,2\}$ represents the mutualism factor, indicating the intensity of interaction among individuals.

The tailer subgroup, the last one-third of the population, preserves population diversity by applying non-uniform Gaussian mutation, thereby mitigating the risk of premature convergence to local optima. Its position update process is defined as follows:[29]

$$x_j^i(t+1) = x_j^i(t) + \Delta(t, GD_t^i) \quad (18)$$

$$\Delta(t, GD_t^i) = GD_t^i\left(1 - r^{(\frac{1-t}{T_{max}})^b}\right) \quad (19)$$

$$GD_t^i = N((F_j - x_j^i(t)), \sigma) \quad (20)$$

where, $\Delta(t, GD_t^i)$ denotes the non-uniform mutation step size, serving as a mutation operator that adaptively adjusts the step length based on a Gaussian distribution GD_t^i; b is a system constant that determines the degree of non-uniformity in the mutation process and is set to 2; σ represents the standard deviation of the Gaussian distribution.

3. Path planning phase: EJPS

3.1 Bidirectional alternating

Bidirectional search involves a forward search from the start point to the goal and a backward search from the goal to the start. However, this approach presents two primary challenges:

(1) As illustrated in Fig. 4, initiating search simultaneously from both the start and goal points may lead to the generation of two distinct paths.

(2) Theoretically, the forward and backward searches are expected to meet at their geometric midpoint, which would yield the highest search efficiency. In practice, however, due to varying obstacle densities and distances between jump points, the two paths may fail to converge at the center.

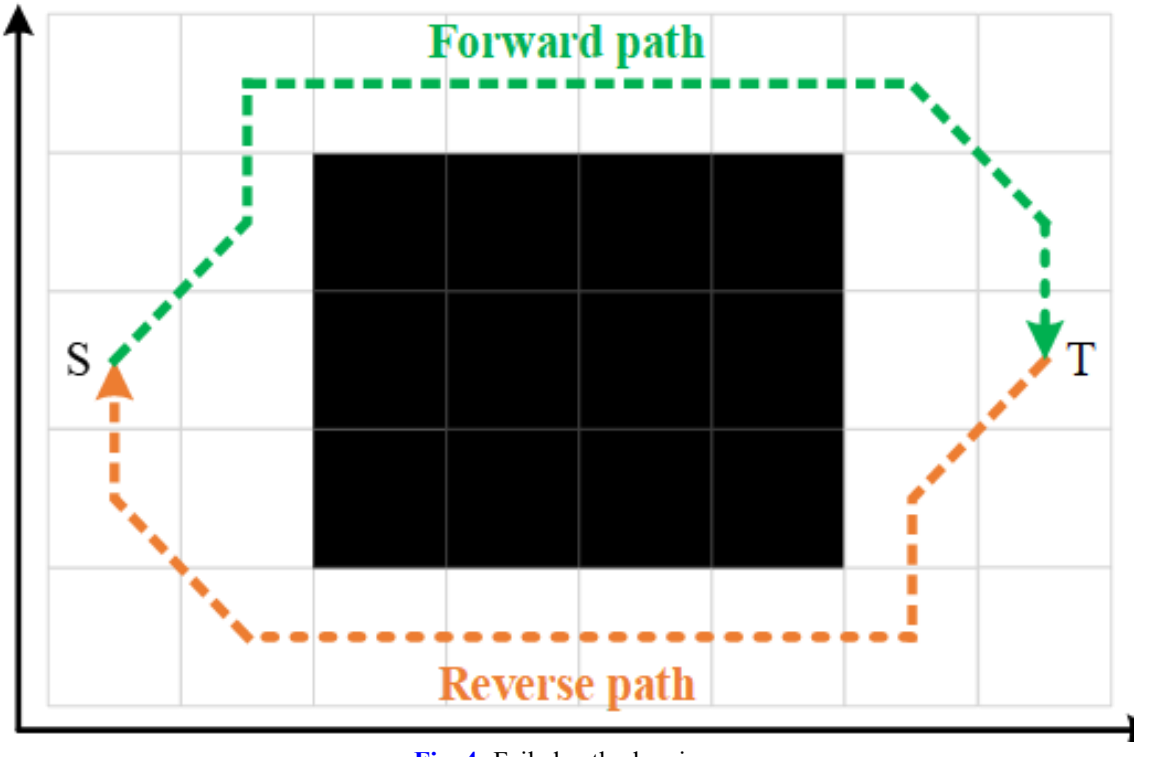


Fig. 4: Failed path planning.

To overcome the limitations of traditional bidirectional parallel search, particularly its inefficiency in merging paths near the geometric midpoint, we propose an enhanced algorithm based on a bidirectional alternating expansion mechanism. By interleaving forward and backward jump point searches, the proposed method effectively guides the convergence of paths toward the geometric center, thereby enhancing both path continuity and overall search efficiency. Specifically, the forward search originates from the start point, while the backward search is initiated in reverse from the goal. These two searches proceed alternately, dynamically narrowing the search gap and accelerating convergence. This strategy achieves a favorable balance between computational efficiency and path optimality, exhibiting strong potential for practical deployment and scalability. The pseudo-code of the algorithm is shown in Algorithm 1, and the implementation

procedure is described as follows:

Algorithm 1: Bidirectional Alternating Search Strategy

Input: Start node s, goal node g, grid map M
Output: Optimized path $\mathcal{P}$ from s to g, or failure

Initialize $\text{OPEN}_1 \leftarrow \{s\}$, $\text{OPEN}_2 \leftarrow \{g\}$;
Initialize $\text{CLOSE}_1 \leftarrow \emptyset$, $\text{CLOSE}_2 \leftarrow \emptyset$;
Set $g_1(s) \leftarrow 0$, $g_2(g) \leftarrow 0$;
Set $f_1(s) \leftarrow g_1(s) + h(s,g)$, $f_2(g) \leftarrow g_2(g) + h(g,s)$;
while $OPEN_1 \neq \emptyset$ **and** $OPEN_2 \neq \emptyset$ **do**
  $n_1 \leftarrow \arg\min_{n \in \text{OPEN}_1} f_1(n)$;
  ExpandJumpPoints(n_1, direction = FORWARD, using $h_1(n,g)$);
  Add n_1 to CLOSE_1;
  if $n_1 \in CLOSE_2$ **then**
    return *ReconstructPath*(n_1)
  $n_2 \leftarrow \arg\min_{n \in \text{OPEN}_2} f_2(n)$;
  ExpandJumpPoints(n_2, direction = BACKWARD, using $h_2(n,s)$);
  Add n_2 to CLOSE_2;
  if $n_2 \in CLOSE_1$ **then**
    return *ReconstructPath*(n_2)
return *Failure (no feasible path found)*

Step 1: Initialization
The algorithm begins by initializing two sets of OPEN and CLOSED lists for bidirectional search. Specifically, OPEN1 and CLOSED1 correspond to the forward search, recording jump points pending expansion and those already explored, respectively, while OPEN2 and CLOSED2 serve the same roles for the backward search. The start node S is inserted into OPEN1, and the goal node T into OPEN2. Both CLOSED lists are initially empty, establishing the structural basis for the forthcoming bidirectional alternating expansion.

Step 2: Alternating Bidirectional Jump Point Expansion
The algorithm proceeds with an alternating forward and backward jump point search. It begins with the forward expansion phase to iteratively identify successor jump points.

Step 2.1: If OPEN1 is non-empty, select the node n with the lowest estimated total cost $f(n)$. If $n$corresponds to the goal node, the search process terminates and the optimal path is reconstructed. Otherwise, remove n from OPEN1 and add it to Closed1.

Step 2.2: From the current jump point n, initiate directional expansion based on the orientation of natural successors: Initially search in horizontal and vertical directions. If an obstacle or map boundary is encountered, switch to diagonal expansion until a valid successor jump point t is located.

Step 2.2.1: If no valid jump point is found, or if the identified point t is already present in CLOSED1, discard it and continue the search.

Step 2.2.2: If t is not in OPEN1, add it to OPEN1, compute its cost values $g(t)$, $h(t)$, and $f(t)$, and set its parent to n.

Step 2.2.3: If t is already in OPEN1, recompute its tentative cost $g(t)$. If the new path through n yields a lower cost, update the parent of t to n and refresh its evaluation value $f(t)$.

Step 3: Backward jump point expansion
Upon identifying a candidate jump point t in the forward search, control alternates to the backward search phase. The node m with the minimum evaluation value $f(m)$ is selected from OPEN_2 for expansion. This phase mirrors the forward search process, leveraging the same directional jump point expansion rules to iteratively explore valid successors. During this procedure, both OPEN2 and CLOSED2 are dynamically updated. Through continuous backward propagation from the goal, the backward search front steadily approaches the start node, facilitating progressive convergence between the two search waves.

Step 4: Search termination via path intersection
The forward and backward jump point searches proceed in an interleaved, alternating fashion. The search process terminates as soon as a jump point expanded in one direction is detected in the CLOSED list of the opposite direction. This condition signifies that the forward and backward search frontiers have intersected at a common node, indicating a successful connection. At this point, the search is halted and no additional nodes are expanded.

Step 5: Final path reconstruction
Following path intersection, the final solution is reconstructed by tracing parent links in both search directions. Starting from the meeting node, the algorithm backtracks along the forward search to retrieve the path from the start node, and similarly backtracks along the backward search toward the goal. The complete path is obtained by concatenating the forward and backward segments, forming a continuous, optimal trajectory from start to goal.

3.2 Heuristic enhancement

In bidirectional JPS, the accuracy of the heuristic function critically affects both search efficiency and path optimality. An underestimated heuristic promotes optimal path discovery but often incurs redundant expansions, while an overestimated one accelerates search at the expense of accuracy. The commonly used Euclidean distance heuristic tends to underestimate true costs, especially in long-distance scenarios, leading to reduced efficiency.

To address this, an adaptive heuristic weighting strategy is introduced. By dynamically scaling the heuristic term—amplifying it in early stages to speed up exploration and gradually reducing it near the goal—this approach balances expansion efficiency with path accuracy, achieving a more effective trade-off between speed and optimality.[32]

$$f(n) = g(n) + \left(1 + \frac{r}{R}\right) h(n) \tag{21}$$

where, R denotes the Euclidean distance from the start point to the goal, r represents the Euclidean distance from the current node n to the goal.

3.3 PLSNR strategy

Traditional grid-based paths consist of numerous discrete nodes that, while preserving connectivity, often result in redundant waypoints, elongated trajectories, and suboptimal execution efficiency. While existing post-processing pruning techniques[33-34] offer certain improvements, their effectiveness

is inherently limited by alignment with grid centers, making it difficult to achieve higher structural precision.

To overcome these limitations, we introduce PLSNR strategy, which combines dynamic segment discretization with goal-directed backtracking. This method effectively extracts critical control points, restructures the path with reduced complexity, and significantly improves execution quality. The pseudo-code is shown in Algorithm 2 and the steps are as follows:

Algorithm 2: Path-Local Segment Node Reconstruction (PLSNR)

Input: Original path $\mathcal{P}_{\text{orig}} = \{p_1, p_2, ..., p_n\}$
Output: Refined path $\mathcal{P}_{\text{ref}}$

$\mathcal{P}_{\text{ref}} \leftarrow \{p_1\}$; // Initialize with start node
Set $i \leftarrow 2$;
while $i < n$ **do**
  $prev \leftarrow \mathcal{P}_{\text{ref}}[-1]$;
  $curr \leftarrow p_i$, $next \leftarrow p_{i+1}$;
  Compute direction vectors $\vec{v}_1 = curr - prev$, $\vec{v}_2 = next - curr$;
  Compute turning angle $\theta = \angle(\vec{v}_1, \vec{v}_2)$;
  if $\theta > \theta_{thresh}$ **then**
    // Detected turning point
    if IsLineCollisionFree($prev$, $next$) **then**
      $i \leftarrow i + 1$; // Skip $curr$, direct connection is safe
    else
      Append $curr$ to $\mathcal{P}_{\text{ref}}$; // Obstacle detected, keep
  else
    $i \leftarrow i + 1$; // No turn, skip $curr$
Append p_n to $\mathcal{P}_{\text{ref}}$; // Ensure goal node included
return $\mathcal{P}_{ref}$

Step 1: Uniform discretization and initialization

Let the original path be defined as $P = \{p_0, p_1, \ldots p_n\}$, where each point lies at the center of a grid cell. To enhance point continuity and enable finer redundancy elimination, the path is uniformly discretized such that the Euclidean distance between any two adjacent nodes does not exceed a predefined threshold D_{min}. This is achieved by inserting intermediate points between adjacent nodes, yielding a refined path set $\tilde{P} = \{\widetilde{p_0}, \widetilde{p_1}, \ldots \widetilde{p_m}\}$ with $\|\widetilde{p_i} - \widetilde{p_{i+1}}\| \leq D_{min}$. The optimized path set $p^* = \{\widetilde{p_0}\}$ is then initialized, with the current reference index set to $i = 0$ and the terminal index set to m.

Step 2: Farthest feasible target backtracking

Starting from the current reference point $\widetilde{p_i}$, initialize the temporary target index $j = m$ and define a safety threshold $SaveD$. Construct a candidate segment $\overline{\widetilde{p_i}\widetilde{p_j}}$ and discretize it into the point set L_{ij}. If all points in $L_{ij} \subset$ FreeSpace and $\|L_{ij} - \text{ObstacleSpace}\| \leq SaveD$, lie within the traversable region and meet safety constraints, record the target as $T = \widetilde{p_j}$. Otherwise, decrement $j = j - 1$ and repeat the feasibility check until the farthest reachable point is identified or $T = \widetilde{p_j}$ reaches the immediate neighbor.

Step 3: Path set update and reference index advancement

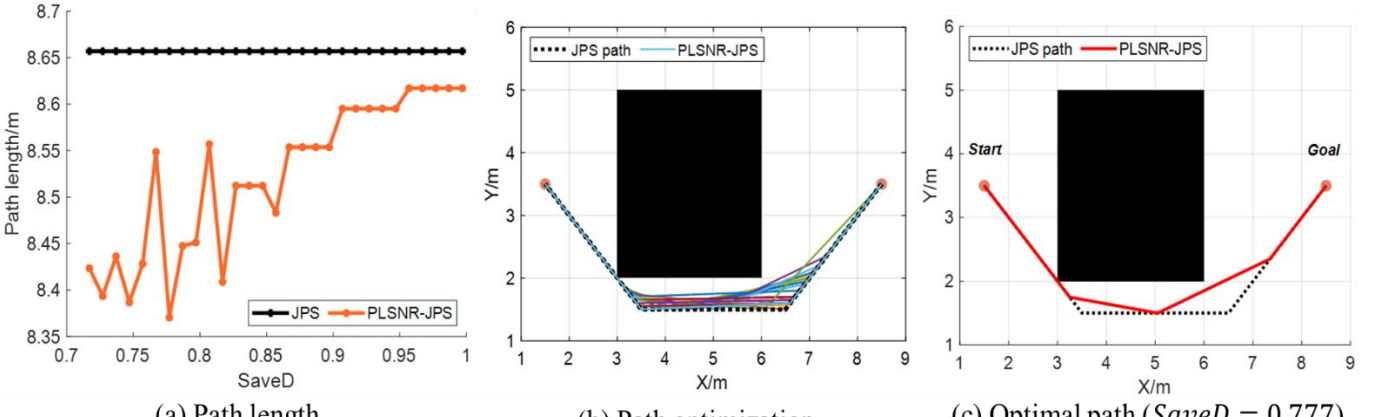
The farthest feasible point T is appended to the optimized path set p^*, and the reference index $i \leftarrow j$ is updated accordingly. The algorithm then proceeds from this new reference point: $p^* \leftarrow p^* \cup \{\widetilde{p_j}\}$

Step 4: Iterative completion

Steps 2 and 3 are repeated until the final destination $\widetilde{p_m}$ is included in p^*. At this point, all redundant intermediate nodes lying on linear segments have been effectively removed, leaving only key turning points to ensure path minimality and structural clarity.

The safety distance parameter $SaveD$, as a displacement control factor in the PLSNR strategy, exerts a substantial influence on path smoothness. To determine its optimal value, we varied $SaveD$ within the range of $[0.5 \times \sqrt{2}, 1.0]$ while fixing the discretization interval at $D_{min} = 0.01$. The initial paths were generated using the JPS algorithm. Fig. 5(a) illustrates how varying safety distance parameters affect the total path length. Fig. 5(b) compares the original JPS paths with the compressed paths generated by the PLSNR strategy, highlighting the improvements in path compactness and overall planning performance. Fig. 5(c) presents the optimized paths and the corresponding structural changes, which significantly influence the final path layout and length. When the safety distance is set to $SaveD = 0.777$, the refined path length decreases from 8.6569 m to 8.3705 m.

It is important to note that conventional smoothing strategies,[33-34] which rely solely on the original JPS nodes, are ineffective in this case. As the initial JPS path already



(a) Path length (b) Path optimization (c) Optimal path ($SaveD = 0.777$)

Fig. 5: Schematic of PLSNR optimization.

represents the shortest feasible solution under such discrete constraints, these methods offer no further improvement. In contrast, PLSNR achieves continuous-level refinement, enabling more precise and practical path optimization.

4. Task allocation phase: MCOSSA

While MSNSSA has improved the convergence and diversity of SSA, its performance remains constrained in complex nonlinear and large-scale scheduling scenarios due to limited evolutionary strategies and insufficient local exploitation. To overcome these shortcomings and enhance multi-robot task scheduling, we propose an improved multi-stage framework, termed MCOSSA, and the pseudo-code is shown in Algorithm3. MCOSSA incorporates four synergistic mechanisms: dynamic subpopulation division, fitness feedback-based symbiosis, hybrid Gaussian perturbations, and multi-neighborhood local refinements. These enhancements jointly strengthen the algorithm's adaptability and search capability at both the population and individual levels, enabling robust and scalable optimization in high-load, tightly coupled scheduling environments.

Algorithm 3: MCOSSA: Multi-stage Collaborative Optimization Salp Swarm Algorithm

Task point set $T = \{t_1, ..., t_n\}$, robot number K, cost matrix C Optimized task allocation and execution path set P Generate population Pop of size N using dual-factor encoding (A, R) // 1.
Apply K-means clustering to form: Pop^1 (exploration), Pop^2 (exploitation), Pop^3 (elite) // 2.
for $iter = 1$ *to* $MaxIter$ **do**
  Exploration Phase (Population Pop^1**)**
  foreach X_i
  in Pop^1 **do**
    Perform exploration updates:
    quad Use large adaptive mutation to enhance diversity
  Exploitation Phase (Population Pop^2**)**
  foreach X_i
  in Pop^2 **do**
    Apply random neighborhood:
    quad 1) Local swap, 2) Reversal, 3) Task exchange, 4) Task migration, 5) Elite guidance
  Guidance Phase (Population Pop^3**)**
  foreach X_i
  in Pop^3 **do**
    Perform Lévy flight-based fine-tuning
  Step 3: Merge and Elitism
  Merge Pop^1, Pop^2, and Pop^3
  Apply elitism: preserve top E individuals for the next iteration
Step 4: Final Solution
return the best solution X^*, corresponding to optimal task allocation and path schedule.

4.1 Multi-robot problem formulation

4.1.1 Dual-factor encoding and decoding

In multi-robot collaborative routing problems, the task execution sequence and path segmentation are intrinsically coupled. Traditional discrete modeling methods often fail to simultaneously ensure solution space continuity and path feasibility, thereby constraining the convergence efficiency and global search capability of optimization algorithms.[35]

To address this, we introduce a dual-factor random-key encoding scheme that integrates task ordering and robot assignment within a unified vector representation. Each individual $X_i$in the salp population is encoded as follows:[36]

$$X_i = [x_1, x_2, \ldots x_n, b_1, b_2, \ldots b_{m-1}] \tag{22}$$

where Each individual $X_i \in R^{n+m-1}$ encodes a solution comprising n task factors $\{x_1, x_2, \ldots x_n\}$ and $m-1$ segmentation factors $\{b_1, b_2, \ldots b_{m-1}\}$, where elements $x_n \in [0,1)$ and $b_m \in [1, m-1]$ denote the respective value ranges for the task and segmentation components.

The task execution sequence π is determined by sorting the task factors $\{x_1, x_2, \ldots x_n\}$ in ascending order. The differences between adjacent elements in this sequence are computed, and the $m-1$ largest differences are selected as segmentation indices S_1 through S_{m-1}, subject to the condition:

$$0 < S_1 < S_2 < \ldots < S_{m-1} < n \tag{23}$$

Accordingly, π is partitioned into m continuous subsequences, each corresponding to the task allocation for a specific robot. The decoding process for each individual X_i defines the robot task paths as:

$$T_k = [depot, \pi(S_{k-1}+1), \pi(S_{k-1}+2), \ldots, \pi(S_k), depot] \tag{24}$$

where T_k denotes the complete route of the k-th robot, and the depot ensures path closure. The segmentation boundaries are defined by:

$$\begin{cases} S_0 = 0 \\ S_m = n \\ k = 1,2,\ldots,m \end{cases} \tag{25}$$

Fig. 6 presents the complete procedure of the dual-factor random key-based encoding and decoding mechanism, demonstrated with a case involving three robots collaboratively executing ten tasks. Each individual consists of ten task-ordering keys (*e.g.*, 0.28, 0.15, …, 0.65) and two segmentation keys (*e.g.*, 1, 2). The task-ordering keys are first sorted in ascending order to derive the task execution sequence. Then, adjacent key differences are computed, and the two largest are selected as segmentation points—specifically between tasks 8 and 1, and tasks 4 and 5. Based on these, the sequence is partitioned into three segments. Finally, the decoding yields three closed-loop paths: [0–2–8–0], [0–1–3–4–5–0], and [0–9–6–7–10–0], each corresponding to one robot's assigned task route.

4.1.2 Fitness function design

In multi-robot path planning problems, conventional fitness functions typically pursue the minimization of the total travel cost across all robots, as follows:[35]

$$F(X) = \sum_{k=1}^{m} \sum_{i=1}^{p-1} D\left(P_{k,i}, P_{k,i+1}\right) \tag{26}$$

where $P_{k,i}$ denotes the i-th node in the k-th robot's path, $D\left(P_{k,i}, P_{k,i+1}\right)$ is the distance function, and p is the number of nodes assigned to robot k.

Although this objective effectively reduces overall operational cost, it often leads to load imbalance, where some robots undertake disproportionately longer routes, resulting in execution delays and reduced system efficiency. To address this, a balanced fitness function is introduced, aiming to minimize the longest individual path among all robots:[37]

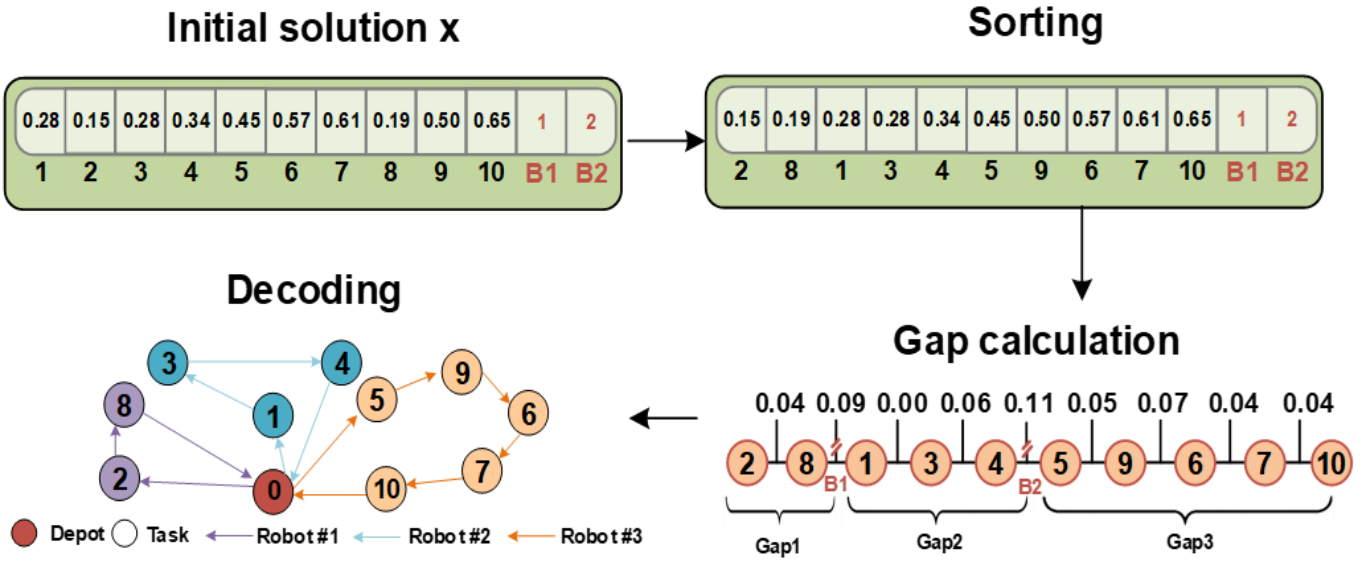


Fig. 6: Encoding-decoding example.

$$F(X) = max\{L_1, L_2, \ldots, L_m\} \tag{27}$$

$$L_k = \sum_{i=1}^{p-1} D\left(P_{k,i}, P_{k,i+1}\right) \tag{28}$$

where L_k denotes the total length of the k-th robot's route; $max\{\cdot\}$ denotes the cost of the longest path among m paths.

4.2 Search mechanism optimization

4.2.1 Fitness-Based dynamic subpopulation partitioning

To enhance the convergence capability and population diversity of SSA in multimodal and complex search spaces, a fitness-based dynamic clustering strategy is proposed to replace the traditional fixed-ratio division scheme used in MSNSSA, which statically segments the population into leaders, followers, and tailers. This strategy fully exploits fitness disparities among individuals to infer relative dominance, thereby enabling a more adaptive and problem-aware subpopulation structure. By treating fitness values as indicators of individual search performance, the population is dynamically partitioned using the K-means algorithm. This allows individuals to be grouped based on evolutionary potential rather than fixed roles, ensuring that subgroup evolution strategies are more appropriately aligned with the population's current state. The resulting adaptive stratification improves the algorithm's balance between exploration and exploitation across different stages of evolution.

Let the population consist of N individuals, with corresponding fitness values denoted as:[38]

$$F = \{F_1, F_2, \ldots, F_N\}, F_i = F\left(x_j^i\right) \tag{29}$$

where $x_j^i \in \Re^d$ denotes the position of the i-th individual in a d-dimensional search space. The fitness function values F are used as input features for clustering via the K-means algorithm, which aims to minimize the within-cluster sum of squared errors, as follows:[38]

$$\min_{\{C_1,\ldots,C_k\}} \sum_{k=1}^{K} \sum_{F_i \in C_k} \|F_i - \mu_k\|^2 \tag{30}$$

where C_k denotes the k-th cluster, μ_k is the centroid of that cluster, and k is the number of clusters. In this study, k is set to 3, corresponding to the leader, follower, and tailer subgroups, respectively.

After clustering, clusters are ranked by their average fitness $\overline{F_k}$ in ascending order (for minimization problems).

The cluster with the lowest average fitness is labeled as the leader group, followed by the follower and tailer groups accordingly:

$$\begin{cases} G_1 \rightarrow Leader, & if\ \overline{F_1} = \min_k \overline{F_k} \\ G_2 \rightarrow Folloewer, & if\ \overline{F_2} = \mathrm{mid}_k \overline{F_k} \\ G_3 \rightarrow Chainer, & if\ \overline{F_3} = \max_k \overline{F_k} \end{cases} \tag{31}$$

where $\overline{F_k} = \frac{1}{|C_k|} \sum_{F_i \in C_k} F_i$.

As illustrated in Fig. 7, the traditional fixed-ratio partitioning assigns roles purely based on index without accounting for individual fitness, leading to blurred subgroup boundaries and reduced diversity. In contrast, the proposed K-means-based fitness clustering adaptively captures population heterogeneity and yields a more accurate role assignment, thus enhancing search coordination and algorithmic robustness.

4.2.2 Fitness-Driven symbiotic strategy

In the original MSNSSA, the evolution of follower individuals is guided by constructing a symbiotic center based on the average of current and previous positions. However, this approach neglects the role of fitness information, resulting in limited responsiveness to environmental variability and a higher risk of premature convergence due to inefficient information propagation.

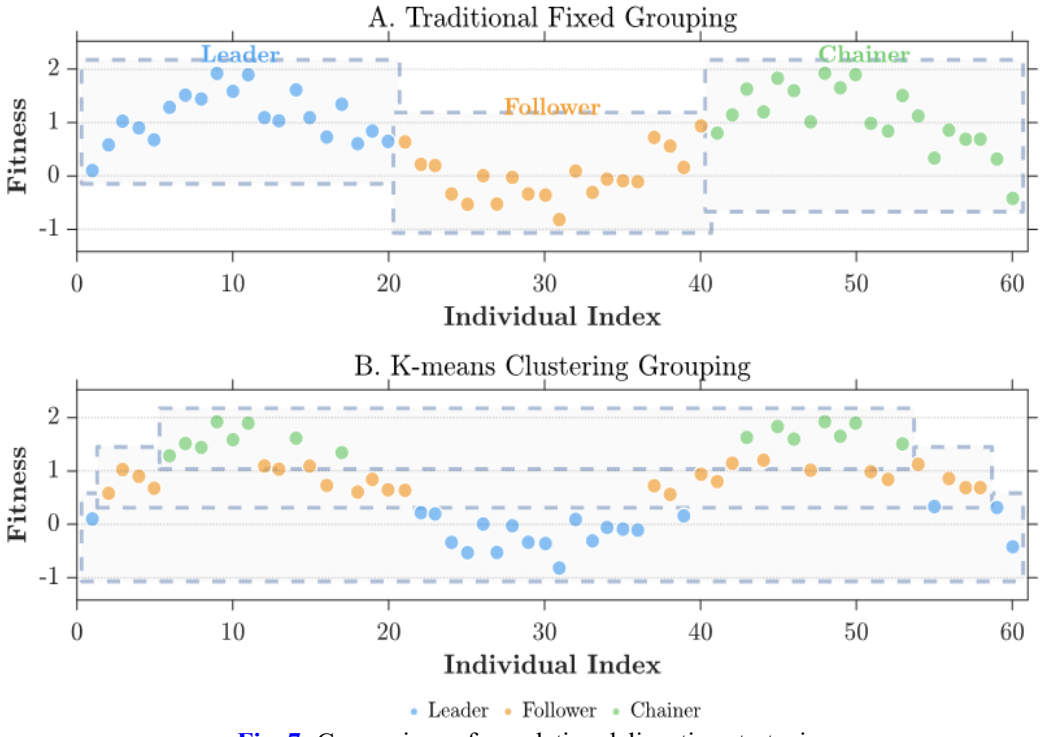


Fig. 7: Comparison of population delineation strategies.

To address this limitation, a fitness-driven symbiotic update mechanism is introduced in this study. By incorporating a fitness feedback coefficient, the proposed strategy adaptively adjusts the search direction toward superior solutions, thereby enhancing collaborative evolution and overall algorithmic stability.[39] The updated position of each follower is computed as:

$$x_j^i(t+1) = x_j^i(t) + r(F_j - C \cdot R) + \mu \cdot \omega_i \cdot (F_j - x_j^i(t)) \quad (32)$$

where $\mu = 1 - \kappa$ denotes the feedback guidance coefficient, which controls the influence of the fitness-driven component; ω_i is the fitness feedback weight, designed to enhance the guidance effect of higher-quality individuals, and is defined as:

$$\omega_i = \frac{f_{i-1}}{f_i + f_{i-1} + \epsilon} \quad (33)$$

where f_i represents the fitness value of the i-th individual, and $\epsilon = 1 \times 10^{-8}$ is a small positive constant introduced to prevent division by zero.

4.2.3 Multi-strategy mutation fusion

In the MSNSSA framework, tailing individuals are responsible for maintaining population diversity through non-uniform Gaussian perturbation. However, relying solely on a single mutation strategy often leads to insufficient exploratory capability, making it difficult to escape local optima in complex, high-dimensional search spaces.

To address this, a multi-strategy mutation fusion mechanism is introduced, which combines non-uniform Gaussian mutation with Lévy flight to enhance both local refinement and global exploration. The position update rule is defined as:

$$x_j^i(t+1) = \begin{cases} x_j^i(t) + \Delta G(t), r < \vartheta \\ x_j^i(t) + Lévy(\lambda) \end{cases} \quad (34)$$

where $\Delta G(t)$ represents the non-uniform Gaussian perturbation defined as:

$$\Delta G(t) = GD_t^i \left(1 - r\left(1 - \frac{t}{T_{max}}\right)^b\right) \quad (35)$$

where $Lévy(\lambda)$ denotes a Lévy flight-based long-distance mutation; $r \in [0,1]$ is a uniformly distributed random variable; ϑ is a predefined fusion probability controlling the balance between the two strategies.

4.2.4 Local refinement search strategy

As swarm intelligence algorithms evolve, population diversity gradually diminishes, especially in the late stages of optimization. This convergence often leads to premature stagnation around local optima, hindering further improvements in solution accuracy and overall algorithmic progress.

To address this, a local refinement strategy is introduced. A subset of top-performing individuals is perturbed using small-scale Gaussian noise to generate candidate solutions. These candidates are selectively retained based on fitness evaluation, thus enhancing local exploitation and promoting the discovery of superior solutions. The update formula is given by:

$$x_j^i(t+1) = x_j^i(t) + \varphi \mathbb{N}(0,1) \quad (36)$$

where φ denotes the perturbation strength; $\mathbb{N}(0,1)$ represents a standard normal distribution.

4.3 Multi-neighborhood local search strategy

To further refine path quality and enhance local convergence, this study introduces a multi-neighborhood local search mechanism within each iteration. While maintaining fixed start and end points, five classical neighborhood structures are applied to iteratively perturb and reconstruct the path, enabling fine-grained local optimization.

Let the current path be denoted as a vector $Route$, with its cost defined as $D(Route)$. The local search process explores the following five neighborhood operations:

(1) 2-opt: Reverses a subsegment to eliminate loops or overlaps.

(2) Swap: Exchanges two non-terminal nodes to adjust local ordering.

(3) Insert: Moves a node to a new position to reshape the visiting sequence.

(4) Or-opt: Relocates a contiguous subsequence of length two to another position.

(5) Simplified 3-opt: Disconnects and reconnects three path segments to explore better combinations.

In each search round, these operators are executed in sequence. If any operation yields an improved cost, the modification is accepted immediately and the process restarts. The search terminates when no further improvements can be made. Fig. 8 visually illustrates the effect of each operator on path restructuring.

5. Integrated algorithmic framework

We propose a balanced multi-robot task allocation method for large-scale field operations. Fig. 9 illustrates the overall workflow of the proposed approach, which includes generating a task cost matrix using the EJPS algorithm, followed by encoding, scheduling, and path optimization through the MCOSSA, ultimately achieving balanced task loads among multiple robots.

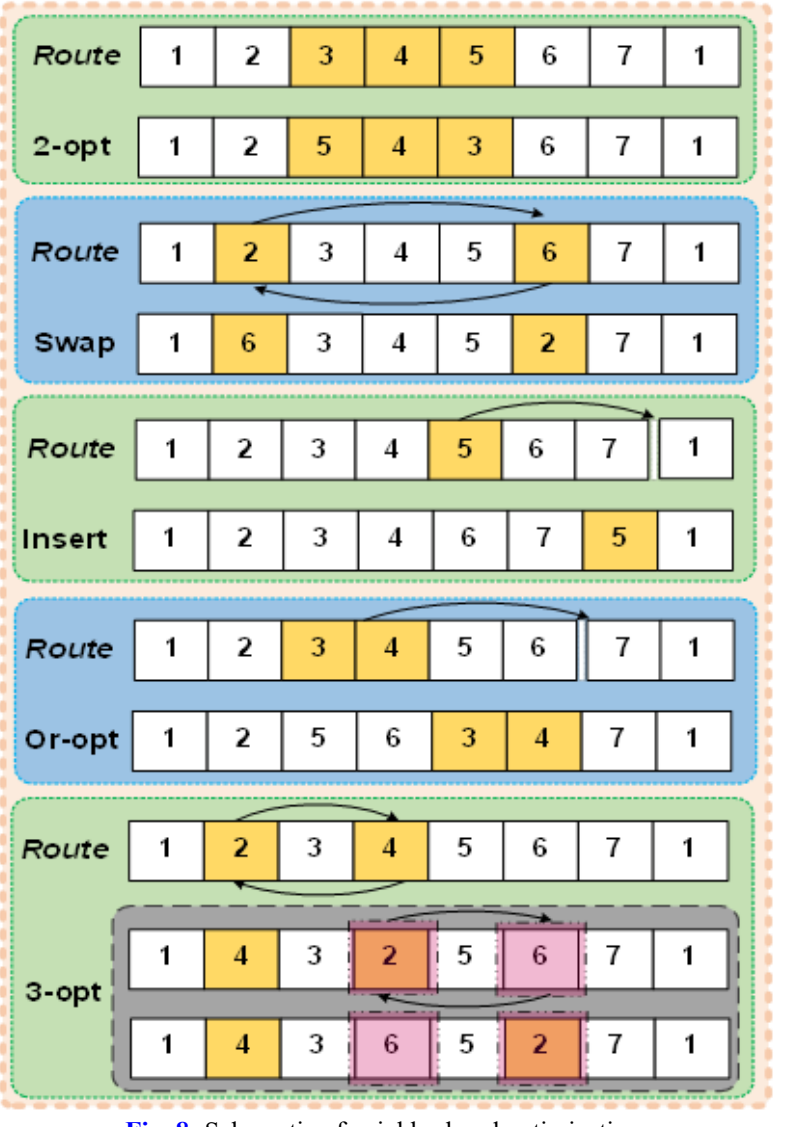


Fig. 8: Schematic of neighborhood optimization.

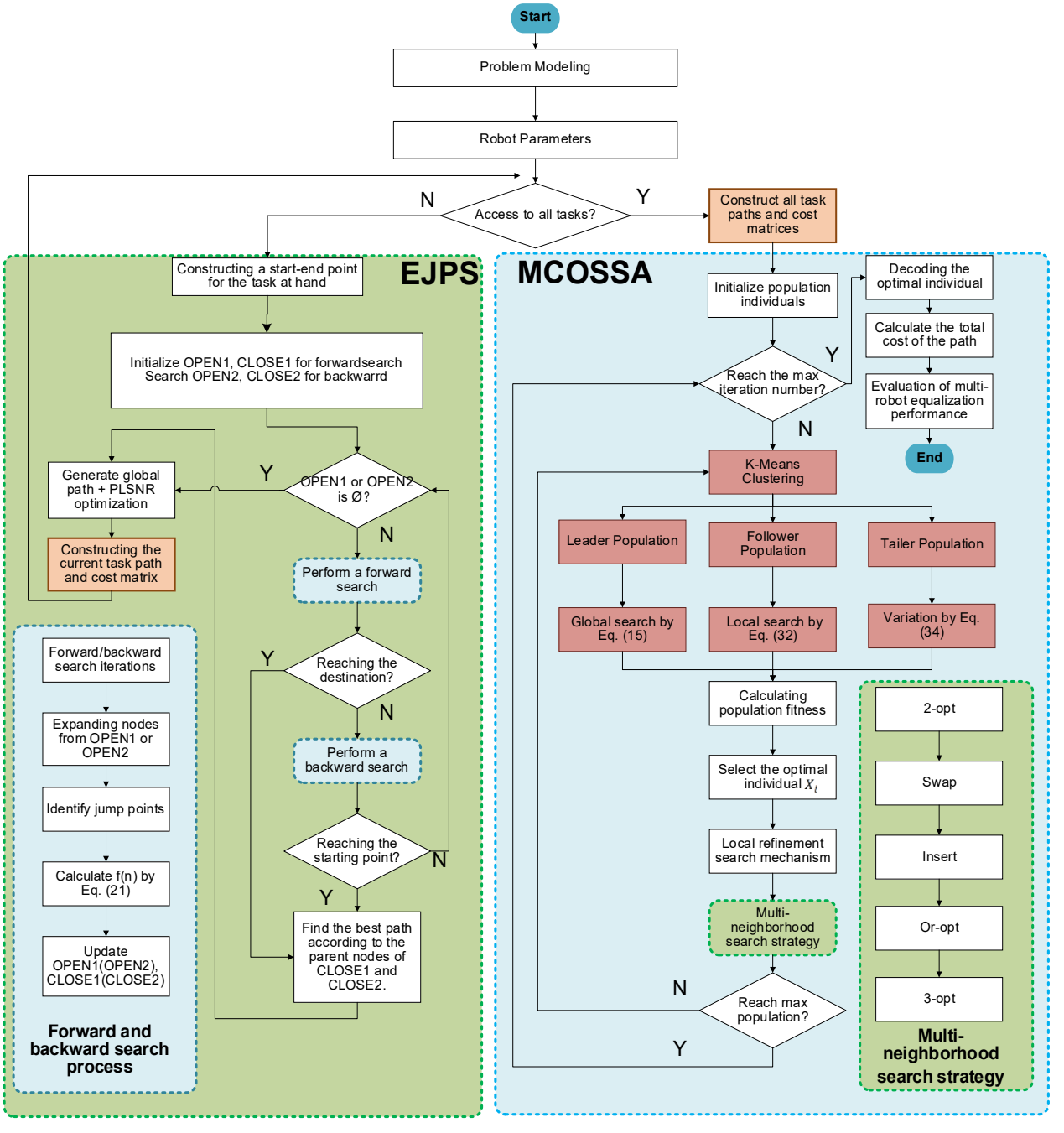


Fig. 9: Algorithm flow.

6. Simulation experiments Supporting Information

To evaluate the effectiveness and robustness of the proposed EJPS-MCOSSA framework in solving path planning and task allocation problems, a series of experiments were conducted using MATLAB 2024a on a system equipped with an R7-5800H processor and 32GB RAM:

(1) Optimization Benchmarking: MCOSSA is compared with PSO, SCA, ABC, SSA, and MSNSSA across eight standard test functions (Table 1), which span unimodal, multimodal, separable, and non-separable characteristics. Each algorithm is run 50 times under consistent parameters (Table 2) to assess convergence accuracy and stability.

(2) Path Planning Evaluation: EJPS is tested against classic algorithms including A*, 16A*, 24A*, and Bidirectional Jump Point Search (BJPS) on public agricultural-style maps (den204d, den403d, den998d) from.[40] Fig. 10 presents the scene characteristics of three types of maps: (a) an open area, (b) an obstacle-dense area, and (c) a curved-path scenario, each representing different levels of complexity in farmland environments.

(3)Multi-Robot Scheduling: Using den998d, 60 task points are assigned via EJPS-derived paths. MCOSSA is then evaluated against PSO, SCA, ABC, SSA, and MSNSSA under varying robot numbers. Each setup is repeated 50 times to assess total cost, task balance, and runtime efficiency across different deployment scales.

Table 1: Benchmark function.

Title	Function	Dimension	Variable Range	Optimal Solution
F1	$f_1(x)=\sum_{i=1}^{n}x_i^2$	30	[-100,100]	0
F2	$f_2(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	30	[-10,10]	0
F3	$f_3(x)=\sum_{i=1}^{n}\left(\sum_{j=1}^{i}x_j\right)^2$	30	[-100,100]	0
F4	$f_4(x)=\max_i\{\lvert x_i\rvert,1\le i\le n\}$	30	[-10,10]	0
F9	$f_4(x)=\sum_{i=1}^{n}\left[x_i^2-10\cos(2\pi x_i)+10\right]$	30	[-5.12,5.12]	0
F10	$f_{10}(x)=-20exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-exp\left(\frac{1}{n}\sqrt{\sum_{i=1}^{n}cos(2\pi x_i)}\right)+20+e$	30	[-32,32]	0
F18	$f_{18}(x)=[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)][30+(2x_1-3x_2)^2(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$	2	[-2,2]	3
F19	$f_{19}(x)=-\sum_{i=1}^{4}c_i exp\left(-\sum_{j=1}^{3}a_{ij}(x_j-p_{ij})\right)$	3	[0,1]	-3.86

Table 2: Algorithm parameters.

Algorithm	Parameter	Experiment 1 Common Parameters	Experiment 3 Common Parameters
PSO	Inertia weight 0.9, Individual Learning Factor 2, Social Learning Factor 2.	The maximum number of iterations is 100, the population size is 60, and the upper and lower search bounds and dimensions are shown in Table 1.	The maximum number of iterations is 1000, the population size is 60, the upper and lower search bounds is ±10, and the dimensionality is 60.
SCA	Control factor 2.		
ABC	Upper limit of acceleration factor 1.		
SSA	Dynamic parameter m is 2.		
MSNSSA	Dynamic parameter m is 2.5.		
Our MCOSSA	Dynamic parameter m is 5, Lévy flight step 5, $\vartheta=0.5$，$\varphi=0.01$，$\eta=10$.		

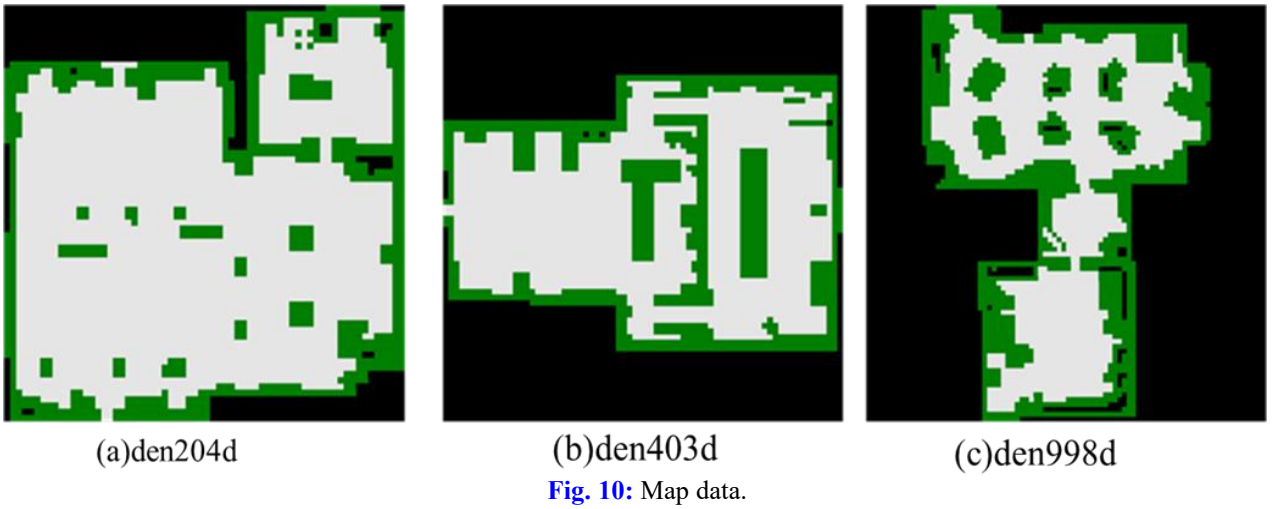


Fig. 10: Map data.

6.1 Test function experiment

To rigorously evaluate the global search performance, convergence accuracy, and robustness of the proposed algorithm, a suite of eight widely recognized benchmark functions (F1, F2, F3, F4, F9, F10, F18, and F19) was selected. These functions encompass diverse landscape characteristics, including unimodal, multimodal, separable, and non-separable structures, thereby enabling a comprehensive performance assessment across varied optimization scenarios. our MCOSSA is benchmarked against five mainstream algorithms: PSO, SCA, ABC, SSA, and MSNSSA. All algorithms were executed independently for 50 runs under identical parameter configurations to ensure fairness and statistical significance. The performance metrics recorded include average value (Ave), standard deviation (Std), best result (Best), worst result (Worst), and average runtime (Time/s). The comparative results are summarized in Table 3 and Appendix A (see *Supporting Information file*), with the best results highlighted in bold.

Experimental findings reveal that Our MCOSSA consistently achieves superior performance across all benchmark functions, reflecting its strong global search capability and convergence stability. In terms of Ave, MCOSSA attained the best performance on all eight test functions. For instance, on F1, it achieved an exceptionally low average value of 8.1661e−15, substantially outperforming PSO (1.2037e+02), SCA (1.7868e+03), ABC (3.5972e+03), and SSA (1.6749e+00). Similarly, on F2, F3, and F4, MCOSSA reached values of 5.3535e−10, 1.9759e−13, and 2.0605e−09, respectively, clearly dominating all competing algorithms. Regarding Std, MCOSSA exhibited remarkable robustness and convergence consistency. For example, on F4, it reported a standard deviation of 5.5369e−09, significantly lower than that of PSO (3.1391e−01) and ABC (1.0528e+00), highlighting its ability to maintain search stability across runs. In terms of Best, MCOSSA either reached or closely approximated the theoretical optima on several functions, including F1 (4.3330e−18), F2 (1.9831e−11), F3 (2.0519e−18), F18 (3.0000e+00), and F19 (−3.8607e+00), thereby demonstrating its excellent global optima-hunting capability. Even under Worst-case scenarios, MCOSSA maintained a high degree of resilience. On F3, for instance, the worst performance was 3.1780e−12, substantially outperforming SSA (5.4377e+01) and SCA (3.3552e+02), indicating superior avoidance of local optima. While the computational cost of MCOSSA is marginally higher than that of lightweight metaheuristics such as SSA, its runtime remains within acceptable limits—e.g., 1.3169e−01 s on F3 and 1.2881e−01 s on F4—demonstrating an effective balance between solution precision and algorithmic efficiency.

In conclusion, MCOSSA achieves consistently superior performance across all tested benchmark functions, excelling in accuracy, stability, and robustness. Its ability to address a wide range of optimization landscapes underscores its potential as a robust and generalizable optimization framework for complex real-world applications.

6.2 Path planning experiment

To evaluate the performance advantages of our EJPS algorithm in path planning tasks, a series of comparative experiments were conducted on publicly available grid map datasets: den204d, den403d, and den998d. The baseline algorithms selected for comparison include A*, 16A*, 24A*, and BJPS. Evaluation metrics encompass path length, number of traversed nodes, number of turning points, and computational time. Detailed results are reported in Table 4 and illustrated in Figs. 11–16.

In Map#1 (open area), the traditional A* algorithm generated a path of 75.6985 m, with 919 explored nodes and a runtime of 3.7625 s. The improved 16A* and 24A* algorithms slightly reduced the path lengths to 72.1356 m and 72.7214 m, and the number of nodes to 831 and 845, respectively, although with a slight increase in computation time. BJPS* exhibited excellent exploration efficiency, requiring only 73 nodes and0.4798 s, but failed to improve the path quality, with a longer path (76.5269 m) and unchanged number of turning points (10). In contrast, the proposed EJPS algorithm achieved the shortest path of 70.5393 m, with only 58 explored nodes and 4 turning points, at a runtime of 2.6575 s, demonstrating superior overall performance in both path quality and efficiency.

In Map#2 (obstacle-dense environment), higher demands are placed on path smoothness and obstacle avoidance. The A* algorithm produced the longest path (78.8406 m) and the highest number of explored nodes (1368). Although BJPS* achieved the fastest runtime (0.2786 s), its path was unstable and lengthy (79.6690 m) with 12 turning points. 16A* and 24A* provided moderate improvements in path length (75.0241 m and 75.6099 m) and node count (1247 and 1260), but did not significantly reduce path complexity. The proposed EJPS algorithm outperformed all others with a shorter path (74.6996 m), the fewest explored nodes (51), only 5 turning points, and a runtime of 1.2349 s, indicating a significant improvement in path quality and planning efficiency.

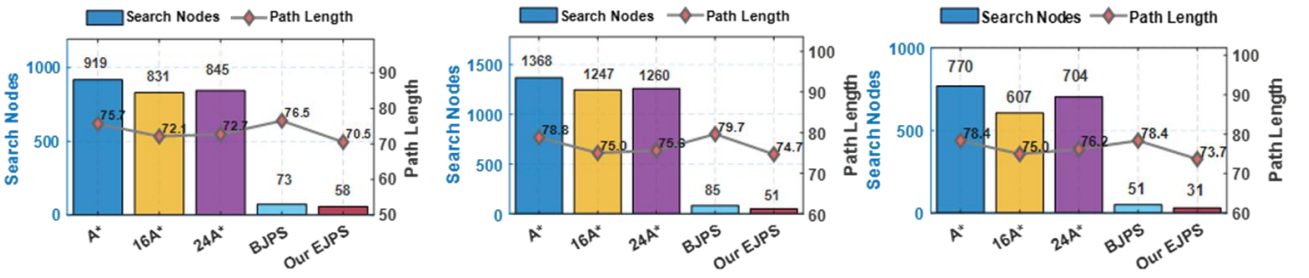
In Map#3 (complex curved terrain), the A* algorithm resulted in the longest path (78.3553 m), with the highest number of turning points (15) and the longest runtime (5.9291 s). BJPS* generated the same path length but with unstable structure (12 turning points), despite exploring only 51 nodes. 16A* and 24A* offered slightly shorter paths (74.9706 m and 76.1935 m) with moderate runtime improvements. In contrast, the proposed EJPS algorithm achieved a significantly shorter path (73.6788 m) by exploring only 31 nodes, with 10 turning points and a runtime of just 1.6248 s, demonstrating its robustness and high efficiency in challenging environments.

Table 3: Test function results.

Title	PSO					SCA				
	Ave	Std	Best	Worst	Time/s	Ave	Std	Best	Worst	Time/s
F1	1.2037e+02	3.3068e+01	4.3618e+01	2.0406e+02	5.1396e-03	1.7868e+03	1.2179e+03	7.6888e+01	5.7412e+03	2.7539e-0
F2	3.1387e+00	2.8128e+00	1.0259e+00	1.1618e+01	4.0235e-03	3.2998e-02	3.0243e-02	3.4253e-03	1.3007e-01	1.2766e-02
F3	1.0491e+01	5.5579e+00	2.1428e+00	2.5915e+01	1.6512e-02	3.3433e+01	5.7529e+01	7.7456e-01	3.3552e+02	2.4786e-02
F4	9.7575e-01	3.1391e-01	3.8860e-01	1.8385e+00	4.2368e-03	1.4499e+00	1.4729e+00	1.0549e-01	8.2457e+00	1.2718e-02
F9	4.4540e+01	1.2273e+01	2.4554e+01	7.7123e+01	6.4107e-03	1.1694e+01	1.2382e+01	1.0344e-02	4.5717e+01	1.5797e-02
F10	3.2450e+00	5.1891e-01	1.9553e+00	4.3688e+00	6.6671e-03	6.6637e-01	2.7115e+00	1.1950e-02	1.9342e+01	1.4938e-02
F18	3.0020e+00	2.1889e-03	3.0000e+00	3.0108e+00	2.4666e-03	3.0005e+00	8.2491e-04	3.0000e+00	3.0048e+00	5.4578e-03
F19	-3.8596e+00	3.6143e-03	-3.8628e+00	-3.8548e+00	6.1588e-03	-3.8540e+00	3.8590e-03	-3.8621e+00	-3.8437e+00	9.8856e-03
Title	ABC					SSA				
	Ave	Std	Best	Worst	Time/s	Ave	Std	Best	Worst	Time/s
F1	3.5972e+03	6.6117e+02	1.8262e+03	4.5887e+03	1.8547e-01	1.6749e+02	9.4251e+01	3.4121e+01	4.8621e+02	4.0509e-02
F2	1.2850e-02	3.2882e-03	7.2962e-03	2.4693e-02	1.8224e-01	4.4730e-02	9.2908e-02	2.7225e-05	4.3394e-01	2.3704e-02
F3	3.5433e+02	1.2807e+02	1.2608e+02	6.3305e+02	2.0943e-01	8.4839e+00	1.0754e+01	5.5741e-03	5.4377e+01	3.6342e-02
F4	4.1621e+00	1.0528e+00	2.1443e+00	7.3338e+00	1.8435e-01	1.0598e-01	2.1003e-01	1.7363e-04	1.1327e+00	2.3825e-02
F9	3.1698e+01	5.5747e+00	1.9349e+01	4.3555e+01	2.0849e-01	1.1402e+01	6.0378e+00	3.9798e+00	2.9849e+01	2.6607e-02
F10	1.7828e+01	9.5707e-01	1.4228e+01	1.8929e+01	1.8964e-01	7.8980e-01	1.0272e+00	1.7779e-05	4.2985e+00	2.5844e-02
F18	3.0000e+00	1.3596e-10	3.0000e+00	3.0000e+00	1.7524e-01	3.0000e+00	6.8137e-13	3.0000e+00	3.0000e+00	1.7238e-02
F19	-3.8615e+00	2.0365e-03	-3.8627e+00	-3.8513e+00	2.0096e-01	-3.8627e+00	2.7687e-04	-3.8628e+00	-3.8612e+00	2.2260e-02
Title	MSNSSA					Ours				
	Ave	Std	Best	Worst	Time/s	Ave	Std	Best	Worst	Time/s
F1	6.1747e-09	2.4811e-09	3.3104e-09	1.4177e-08	4.2971e-02	8.1661e-15	1.5148e-14	4.3330e-18	7.9066e-14	1.4196e-01
F2	1.0089e-05	2.0317e-06	5.3949e-06	1.5305e-05	2.6574e-02	5.3535e-10	7.9468e-10	1.9831e-11	4.8182e-09	1.1931e-01
F3	3.7970e-09	2.6512e-09	7.5143e-10	1.1729e-08	3.9796e-02	1.9759e-13	5.9510e-13	2.0519e-18	3.1780e-12	1.3169e-01
F4	2.1296e-05	4.2282e-06	1.4115e-05	3.0206e-05	2.8685e-02	2.0605e-09	5.5369e-09	3.7215e-11	3.5822e-08	1.2881e-01
F9	2.2256e-03	6.9399e-04	9.5319e-04	3.8170e-03	3.0403e-02	1.0746e+00	3.0542e+00	0.0000e+00	1.5919e+01	1.5584e-01
F10	1.5042e-05	3.1175e-06	6.6499e-06	2.0103e-05	2.0103e-05	1.1122e-08	4.8701e-08	4.3010e-11	3.4068e-07	1.3346e-01
F18	3.0000e+00	1.4716e-12	3.0000e+00	3.0000e+00	2.0904e-02	3.0000e+00	5.0658e-15	3.0000e+00	3.0000e+00	9.7547e-02
F19	-3.8613e+00	6.9528e-03	-3.8628e+00	-3.8164e+00	2.6601e-02	-3.8608e+00	5.6986e-05	-3.8607e+00	-3.8608e+00	1.1952e-01

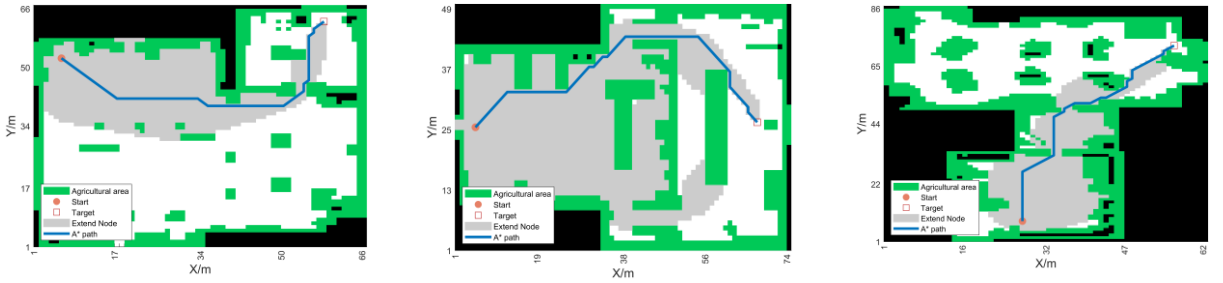
Table 4: Path planning results.

Map	Algorithm	Path length/m	Number of nodes traversed/pc	Inflection points/pc	Time consumed/s
#1	A*	75.6985	919	10	3.7625
	16A*	72.1356	831	8	4.0832
	24A*	72.7214	845	7	3.5851
	BJPS	76.5269	73	10	0.4798
	Our EJPS	70.5393	58	4	2.6575
#2	A*	78.8406	1368	12	3.8057
	16A*	75.0241	1247	6	3.8537
	24A*	75.6099	1260	6	4.4795
	BJPS	79.6690	85	12	0.2786
	Our EJPS	74.6996	51	5	1.2349
#3	A*	78.3553	770	15	5.9291
	16A*	74.9706	607	10	5.9337
	24A*	76.1935	704	9	5.9582
	BJPS	78.3553	51	12	0.3908
	Our EJPS	73.6788	31	10	1.6248



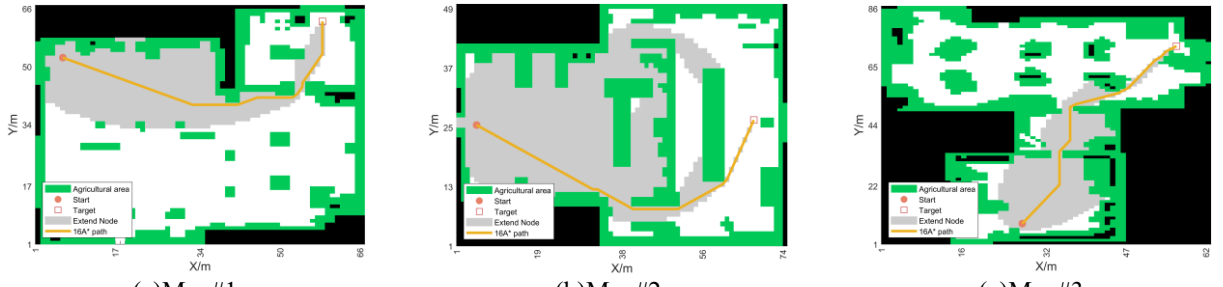
(a) Map#1 (b) Map#2 (c) Map#3

Fig. 11: Comparison of path length and number of nodes traversed.



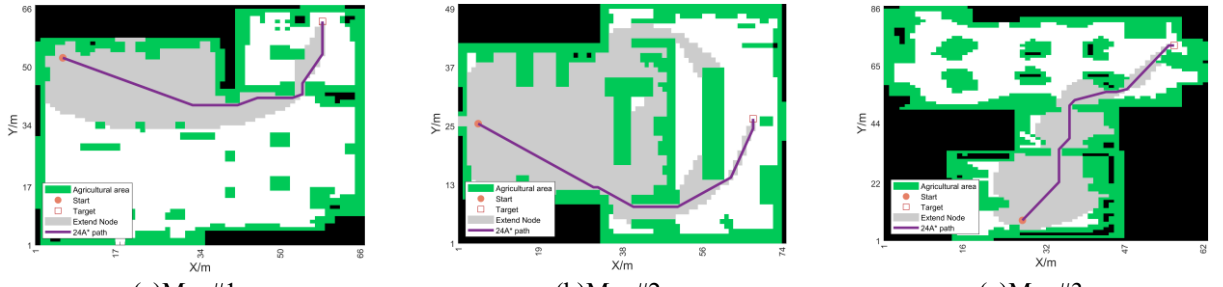
(a)Map#1 (b)Map#2 (c)Map#3

Fig. 12: A* algorithm.



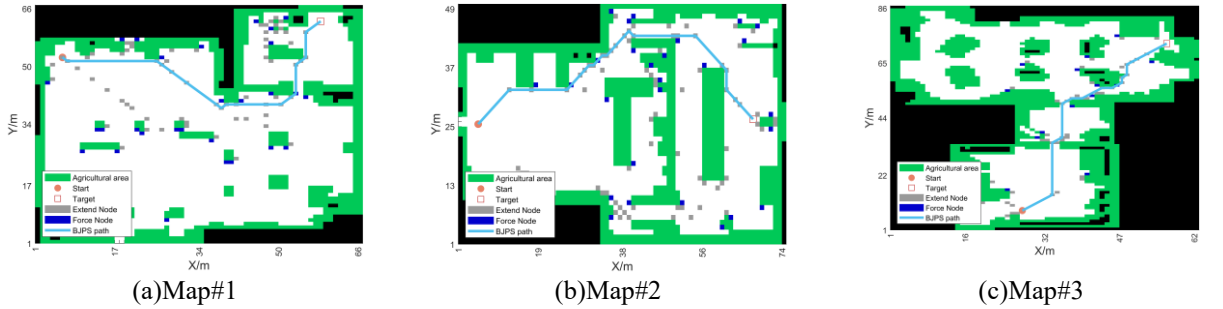
(a)Map#1 (b)Map#2 (c)Map#3

Fig. 13: 16A* algorithm.



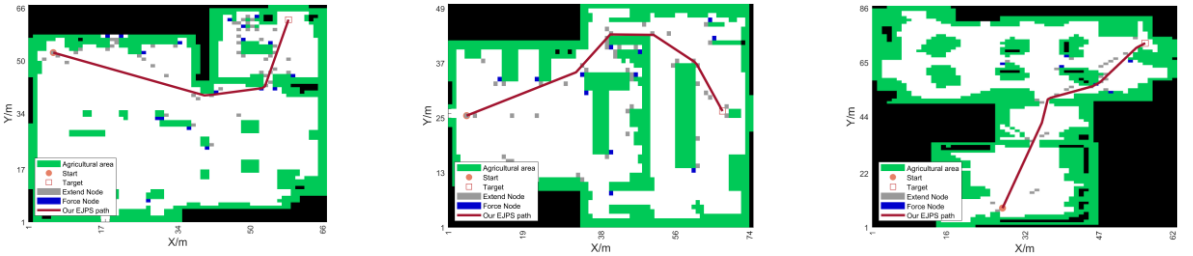
(a)Map#1 (b)Map#2 (c)Map#3

Fig. 14: 24A* algorithm.



(a)Map#1 (b)Map#2 (c)Map#3

Fig. 15: BJPS algorithm.



(a)Map#1 (b)Map#2 (c)Map#3

Fig. 16: Our EJPS algorithm.

Collectively, these results underscore the superior adaptability and comprehensive performance of the proposed EJPS algorithm across diverse environmental settings. Compared to A* variants and BJPS, EJPS consistently yields more compact and smoother paths, reduces redundant node expansions, and maintains competitive runtime performance. These empirical findings substantiate its effectiveness and scalability for global path planning in both structured and obstacle-rich agricultural domains.

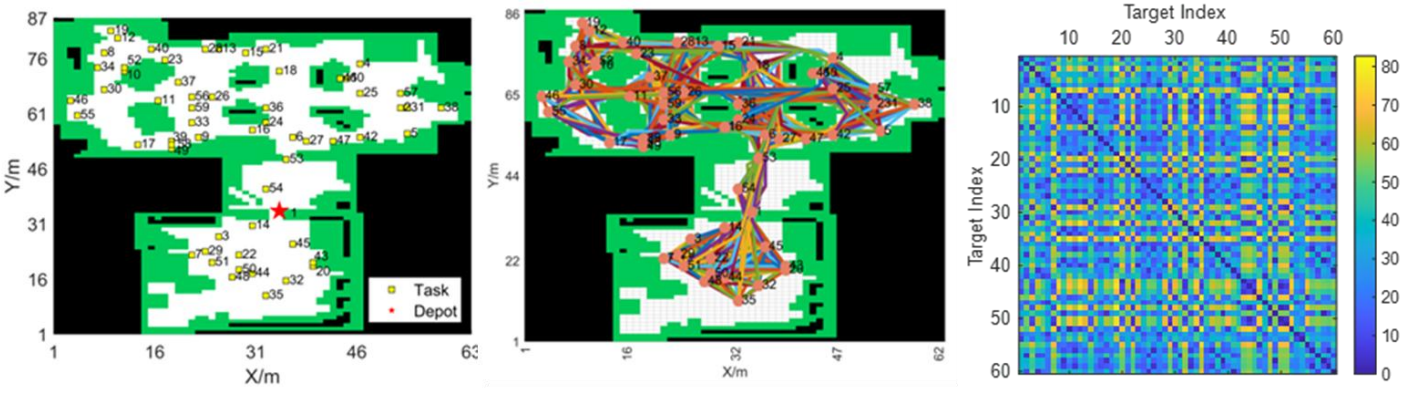
6.3 Path scheduling experiment

Building upon the preceding experiments—which respectively validated the optimization capability of MCOSSA on benchmark functions and demonstrated the path planning effectiveness of EJPS across three distinct environmental scenarios—a comprehensive evaluation of the proposed multi-robot path scheduling framework was conducted on the den998d map. In this experiment, 60 unique task points were randomly generated, with task point #1 at coordinates (35, 35) designated as the central depot. All robots are initialized at the depot, tasked with completing assigned operations by visiting selected task points, and returning to the depot upon completion. To accurately capture spatial traversal costs, the EJPS algorithm was employed to compute pairwise shortest paths between all task points. The resulting fully connected graph and associated cost matrix, depicted in Fig. 17, provide the structural foundation for subsequent multi-robot task allocation and scheduling via MCOSSA. Specifically, Fig. 17(a) shows the spatial distribution of task points and the depot in the den998d map; Fig. 17(b) illustrates the fully connected paths generated by EJPS; and Fig. 17(c) presents the corresponding cost matrix used for task scheduling.

6.3.1 Statistical quantification

To comprehensively assess the scalability, robustness, and task coordination performance of the proposed EJPS-MCOSSA framework under varying team sizes, a set of multi-robot scheduling experiments was conducted with robot fleets ranging from 1 to 6 agents. On a unified cost matrix constructed via EJPS-based pairwise traversal, the proposed method was benchmarked against five representative metaheuristics: PSO, SCA, SSA, ABC, and MSNSSA. Each experimental setting was independently repeated 50 times to ensure statistical consistency. Detailed experimental outcomes are summarized in Table 5 and visually analyzed in Figs. 18–19. The optimal results in the table are highlighted in bold. Fig. 18 compares the average total path cost across various robot team sizes, providing insights into the scalability and coordination efficiency of each method. Fig. 19 illustrates the convergence process, where the number of iterations (x-axis) and corresponding path cost (y-axis) reveal the convergence speed and stability of the proposed algorithm.

A multi-dimensional performance evaluation protocol was adopted, encompassing: Multi-robot total cost: including average (Ave), standard deviation (Std), best (Best), and worst (Worst) values; Task distribution equity: measured via standard deviation, coefficient of variation (CV), and Gini coefficient;



(a) Task point setting (b) Path planning (c) Cost matrix

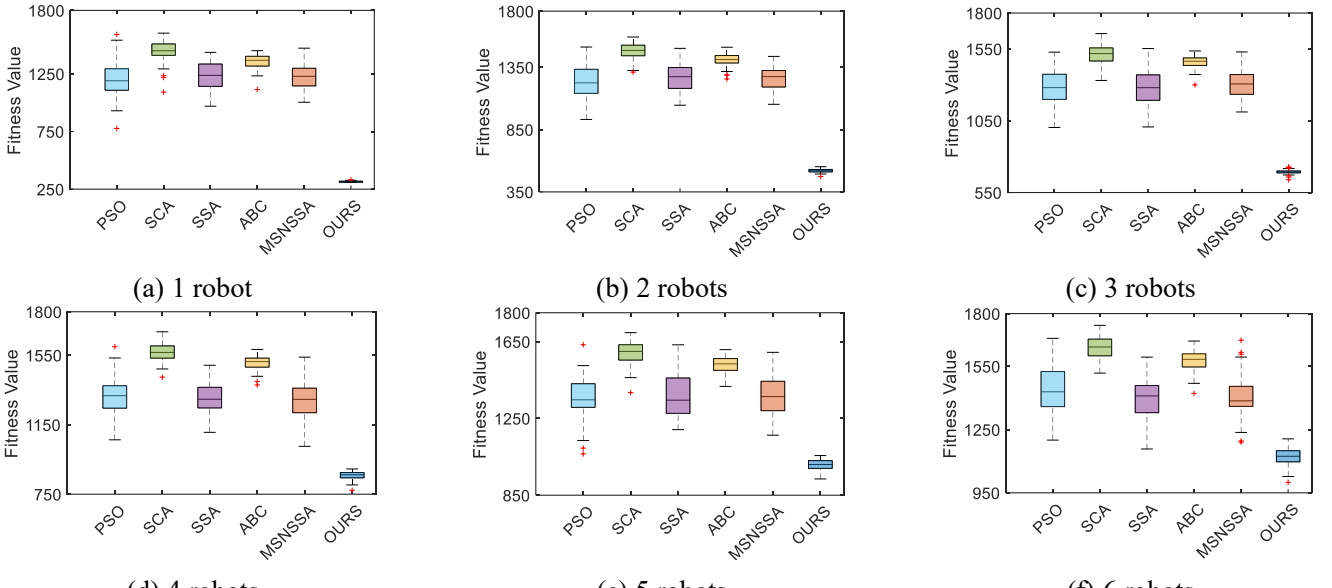
Fig. 17: Task point traversal.

Table 5: Task scheduling results.

Robots	Algorithm	Total Multi-Robot Cost				Multi-Robot Balance Metrics			Algorithm Performance	
		Ave/m	Best/m	Worst/m	Std/m	Standard Deviation	Coefficient of Variation	Gini Coefficient	Time Consumption/s	Iteration Stability/Generation
1	PSO	1205.64	776.55	1590.98	161.46	N/A	N/A	N/A	1.06	698.34
	SCA	1435.11	1091.40	1602.78	96.09	N/A	N/A	N/A	1.76	622.06
	SSA	1238.03	969.43	1436.16	118.74	N/A	N/A	N/A	2.65	609.10
	ABC	1356.90	1114.00	1450.77	65.17	N/A	N/A	N/A	5.97	754.00
	MSNSSA	1227.86	1003.56	1472.57	108.08	N/A	N/A	N/A	2.51	864.00
	Ours	317.15	310.17	332.36	5.71	N/A	N/A	N/A	19.62	407.74
2	PSO	1239.35	931.12	1512.25	137.83	70.8935	0.1141	0.0403	1.22	718.56
	SCA	1478.40	1310.81	1592.75	65.82	79.5046	0.1073	0.0379	1.96	633.88
	SSA	1275.82	1045.31	1501.56	109.34	61.1753	0.0951	0.0336	2.26	589.96
	ABC	1410.24	1255.72	1510.52	54.67	73.1331	0.1040	0.0368	6.29	651.74
	MSNSSA	1265.65	1052.56	1436.95	92.44	61.1571	0.0954	0.0337	2.15	853.62
	Ours	520.28	473.60	552.28	15.06	13.6007	0.0528	0.0187	16.76	360.36
3	PSO	1274.45	1002.89	1527.68	126.48	112.9633	0.2644	0.1126	1.40	675.16
	SCA	1510.34	1330.53	1656.98	60.89	70.6649	0.1405	0.0592	2.13	608.50
	SSA	1287.24	1006.84	1552.71	122.78	53.8064	0.1259	0.0532	2.36	602.10
	ABC	1456.09	1299.49	1536.08	44.20	66.9393	0.1382	0.0589	6.38	676.22
	MSNSSA	1300.56	1111.09	1529.18	96.38	63.4704	0.1479	0.0626	2.36	860.42
	Ours	693.62	638.75	727.34	16.68	13.6748	0.0598	0.0254	13.44	367.04
4	PSO	1316.99	1063.06	1598.94	100.77	92.5848	0.2820	0.1299	1.51	762.38
	SCA	1569.24	1423.64	1684.85	50.14	67.6386	0.1725	0.0797	2.30	604.94
	SSA	1296.81	1106.13	1492.04	93.06	55.5297	0.1705	0.0781	2.47	598.08
	ABC	1505.25	1378.73	1583.18	42.92	72.9676	0.1944	0.0889	6.59	666.66
	MSNSSA	1296.33	1025.57	1538.67	115.21	57.0105	0.1756	0.0802	2.76	819.04
	Ours	856.10	772.47	895.30	25.05	15.9674	0.0756	0.0346	10.91	369.00
5	PSO	1354.99	1064.92	1633.16	114.97	82.8035	0.3044	0.1458	1.57	765.48
	SCA	1589.20	1384.20	1696.38	62.14	69.8421	0.2198	0.1051	2.42	571.06
	SSA	1369.73	1191.41	1633.05	115.36	54.6337	0.1991	0.0950	2.53	600.40
	ABC	1529.90	1416.43	1607.91	42.17	66.3956	0.2170	0.1032	6.84	623.92
	MSNSSA	1364.55	1162.48	1593.90	108.85	55.4891	0.2035	0.0968	3.01	852.28
	Ours	1006.07	934.71	1056.51	30.66	19.4597	0.0981	0.0470	8.79	336.20
6	PSO	1436.33	1199.96	1682.47	113.72	82.3498	0.3456	0.1634	1.62	684.32
	SCA	1637.47	1517.76	1744.14	61.20	65.0908	0.2383	0.1167	2.50	501.62
	SSA	1397.40	1157.81	1593.43	91.98	54.8145	0.2348	0.1131	2.60	608.66
	ABC	1575.75	1421.32	1669.77	49.89	62.2036	0.2373	0.1156	7.03	610.76
	MSNSSA	1404.40	1190.35	1673.51	94.70	49.1563	0.2089	0.1002	3.14	864.76
	Ours	1120.31	999.71	1205.83	41.30	25.2645	0.1377	0.0666	7.47	357.88

Algorithmic efficiency: evaluated through runtime and convergence iteration count. Lower total cost reflects enhanced system-wide operational efficiency. Lower CV and Gini values indicate improved workload balance and fairer task distribution. Reduced runtime and iteration count suggest superior algorithmic convergence characteristics and computational practicality.

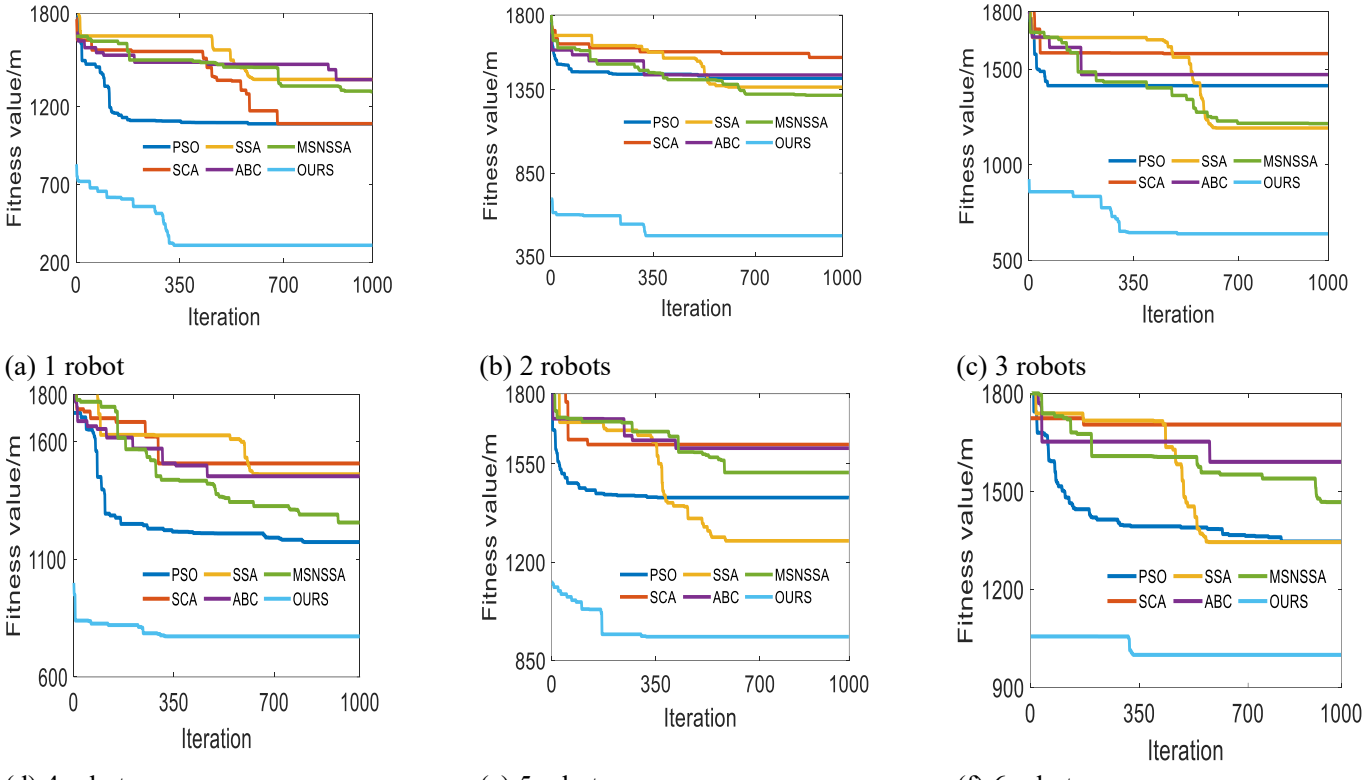
The experimental findings consistently demonstrate the superiority of MCOSSA across all evaluation metrics. In terms of total cost, the proposed method yielded the best performance across all robot scales. For instance, in the 3-robot scenario, MCOSSA achieved a total cost of 693.62 m, compared to 1274.45 m (PSO), 1510.34 m (SCA), and 1456.09 m (ABC)—representing a relative reduction exceeding 40%. Moreover, the rate of cost growth with increasing robot count remained significantly flatter for MCOSSA, attesting to its scalability and structural resilience. Regarding task load balancing, MCOSSA exhibited superior equity. In the 6-robot configuration, it achieved a Gini coefficient of 0.0666, markedly lower than PSO (0.1634) and ABC (0.1156).



(a) 1 robot (b) 2 robots (c) 3 robots

(d) 4 robots (e) 5 robots (f) 6 robots

Fig. 18: Path length comparison.



(a) 1 robot (b) 2 robots (c) 3 robots

(d) 4 robots (e) 5 robots (f) 6 robots

Fig. 19: Convergence diagram.

Likewise, its CV was only 0.1377, outperforming SSA (0.2348) and MSNSSA (0.2089), indicating a more harmonized allocation of workload across robots. In terms of solution stability, MCOSSA consistently delivered low variance in output quality. For instance, in the 5-robot setup, it reported a Std of only 30.66, significantly lower than PSO (114.97) and MSNSSA (108.85), affirming its robustness under stochastic conditions. Furthermore, the average number of iterations to convergence across 3 to 6 robots remained below 400, whereas exceeded 600, highlighting MCOSSA's superior convergence efficiency. Although the proposed method incurs a marginally higher computational cost compared to lightweight heuristics (*e.g.*, PSO typically completes within 2 s), the substantial improvements in total cost reduction, task fairness, and convergence stability justify this trade-off. These advantages render MCOSSA particularly well-suited for real-world applications in intelligent agriculture, warehouse logistics, and other domains that demand both solution robustness and equitable resource coordination.

6.3.2 Optimal path evaluation

To rigorously evaluate the path-level load balancing capability of the proposed multi-robot scheduling framework under optimal conditions, we further introduce three quantitative fairness metrics—Std, CV, and Gini coefficient (Gini)—to characterize the dispersion and equity of individual robot path costs after task allocation. These metrics were used to compare the optimal scheduling outputs of six algorithms—PSO, SCA, SSA, ABC, MSNSSA, and the proposed method—across varying fleet sizes from 1 to 6 robots. The results, summarized in Table 6 and illustrated in Figs. 20–21 and Appendix B, (see *Supporting Information file*), where Fig. 20 illustrates the path length distribution under the optimal scheduling results of each algorithm, serving to evaluate the load balancing performance. Each group of bars corresponds to a different number of robots, with individual bars representing the path length assigned to each robot, thereby reflecting the fairness and stability of task allocation. Fig. 21 and Appendix B, (see *Supporting Information file*) present the path allocation diagrams for 1 to 6 cooperating robots, where each curve denotes a complete task path of a single robot, enabling a visual comparison of task division, path overlap, and spatial distribution across different algorithms.

The experimental results indicate that, under increasing team size and task density (60 tasks), most baseline algorithms exhibit a significant degradation in path-level balance. This is especially evident in Fig. 21 and Appendix B, (see *Supporting Information file*) where overlapping and entangled trajectories reveal pronounced task clustering and inefficient spatial decoupling. For instance, in the 6-robot configuration, PSO yields a Std of 44.77, a CV of 0.2239, and a Gini coefficient of 0.1134, underscoring a marked imbalance in workload distribution. Similar patterns are observed with ABC and SSA, which frequently report elevated Std values (e.g., ABC reaches 103.69 in the 3-robot case), suggesting that even when global cost is reduced, intra-team task fairness remains poorly controlled. In contrast, the proposed EJPS-MCOSSA framework consistently demonstrates superior load balancing across all configurations. Notably, in the 5-robot scenario, it achieves a Std of 21.26 and a Gini coefficient of 0.0571, significantly outperforming MSNSSA (Std = 48.26, Gini = 0.0939) and ABC (Std = 56.55, Gini = 0.0922). In the most challenging 6-robot scenario, our method attains the lowest Gini coefficient of 0.0378 among all candidates, indicating outstanding fairness and minimal workload disparity under high-dimensional task coordination. Compared with conventional algorithms such as PSO, ABC, and MSNSSA, the proposed EJPS–MCOSSA method achieves superior results in terms of total path cost, workload balance, and convergence smoothness.

These improvements stem from three key innovations: (1) the use of bidirectional jumping and heuristic enhancement in EJPS, which streamlines path generation; (2) a dual-factor encoding and multi-stage optimization strategy in task scheduling, enhancing search precision; and (3) a fitness-driven subpopulation clustering mechanism combined with adaptive mutation, which preserves diversity and avoids local optima. Together, these mechanisms form an integrated framework that fundamentally outperforms existing methods in complex multi-robot planning scenarios.

Table 6: Comparison results of optimal programs.

Total number of robots	Robot number	Algorithm					
		PSO	SCA	SSA	ABC	MSNSSA	Ours
1	#1	776.4496	1091.0620	969.0871	1113.3521	1003.5716	294.3526
2	#1	475.7542	649.7015	580.4369	538.9893	514.2057	239.6811
	#2	455.4024	660.8230	464.4943	716.2468	538.2693	233.4616
3	#1	338.4935	335.3529	320.0696	347.1008	284.7827	222.9457
	#2	347.9788	483.7153	352.9068	403.7438	349.1055	210.9128
	#3	316.2054	511.0497	333.9327	548.1978	477.0514	210.4169
4	#1	276.0108	384.9313	276.5026	217.4655	230.4393	202.5515
	#2	225.8636	290.2287	237.8930	392.6113	238.6327	200.4667
	#3	170.0407	378.6683	314.0859	375.1629	263.8876	169.0102
	#4	390.8431	369.4747	277.5951	392.9402	292.4749	184.4906
5	#1	306.6036	267.5310	266.2249	221.7176	150.9267	162.0711
	#2	183.6078	247.1751	178.9448	282.5080	269.8160	167.1655
	#3	106.3933	215.1577	231.5416	272.4770	244.8423	200.5155
	#4	152.5230	291.8824	271.1166	263.4435	230.7331	176.9135
	#5	315.4235	361.8456	243.3218	375.4156	265.8706	210.8791
6	#1	214.3531	193.1754	176.7774	135.9995	204.4562	174.3552
	#2	203.5931	271.7855	95.7163	218.0896	181.9614	174.7815
	#3	161.3430	211.6386	206.6885	291.5612	229.2559	171.3610
	#4	137.2047	275.1695	202.9466	210.7295	163.2399	135.0643
	#5	220.6153	287.5032	165.4951	252.2505	189.3688	175.8812
	#6	262.7706	277.9849	310.0969	312.7192	222.0009	168.2670

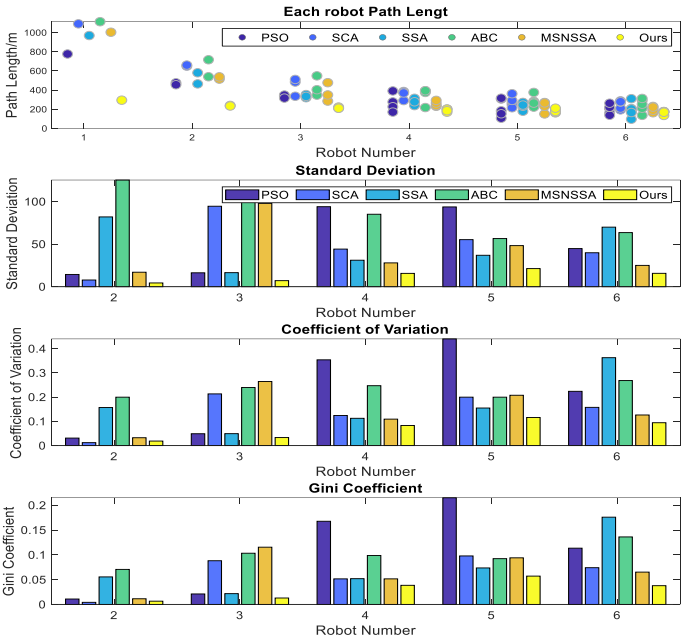
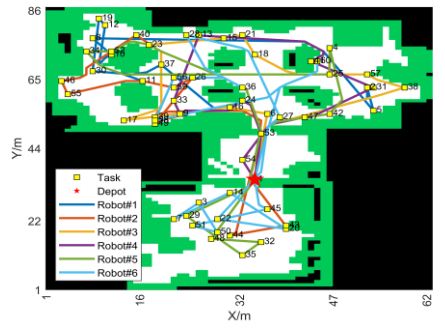
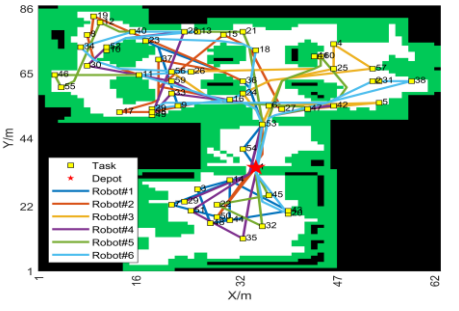


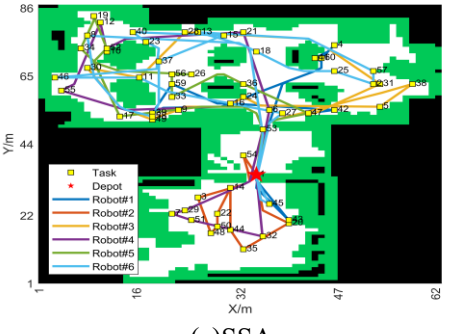
Fig. 20: Optimal path equalization comparison.



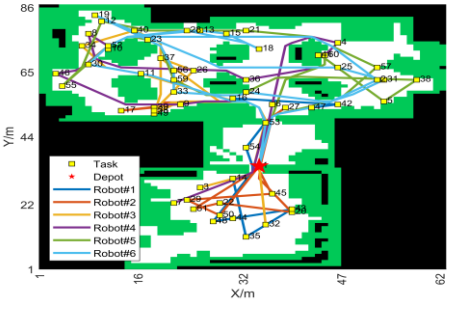
(a)PSO



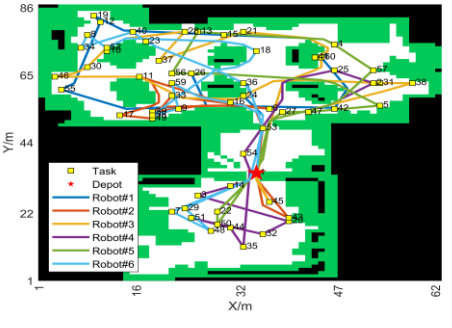
(b)SCA



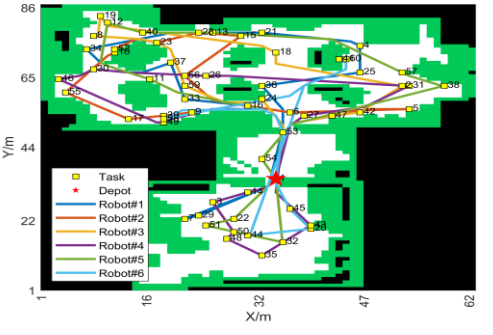
(c)SSA



(d)ABC



(e)MSNSSA



(f)Ours

Fig. 21: Six robots.

7. Conclusion

This paper investigates the problem of multi-robot path planning and task scheduling in unstructured and complex farmland environments and proposes an integrated optimization framework, EJPS-MCOSSA. Our EJPS algorithm leverages bidirectional jump expansion and an adaptive heuristic mechanism to significantly improve path search efficiency and connectivity. In conjunction with a path-level optimization strategy, it further enhances path optimality and smoothness. For task allocation, a dual-factor stochastic encoding mechanism is designed to support efficient multi-robot task decomposition and scheduling. By adopting the minimization of the maximum path cost as the objective, the approach explicitly promotes load-balanced execution across the robot fleet. Our MCOSSA algorithm enhances global exploration through subpopulation-based cooperative search and multi-strategy perturbation, while incorporating a rich set of local search operators—including 2-opt, swap, insertion, Or-opt, and 3-opt—to refine convergence accuracy and solution stability. Comprehensive simulation results demonstrate that the proposed method consistently outperforms state-of-the-art algorithms in terms of path cost reduction, workload equity, and convergence robustness. These findings validate the effectiveness and applicability of EJPS-MCOSSA as a scalable and reliable solution for coordinated multi-robot operations in dynamic, real-world agricultural settings. This study presents a unified and efficient optimization framework for multi-robot task planning in complex farmland environments, demonstrating strong practical value and potential for broader application. However, the current approach is developed and evaluated within idealized simulation scenarios, without considering robot heterogeneity or dynamic environmental changes. Future work will incorporate reinforcement learning strategies and online scheduling mechanisms to enhance the algorithm's adaptive capabilities, and validation experiments will be conducted on real-world agricultural platforms.

Acknowledgments

This work was supported by the Xinjiang University of Technology "New Youth Talent Cultivation Program Research Start-up Project" (2025XQYM060), Xinjiang safflower industry development fund and National Natural Science Foundation of China (52065057; 32260430)

Conflict of Interest

There is no conflict of interest.

Supporting Information

Applicable.

CRediT Statement

Liwei Yang and **Ping Li:** Writing, Original draft, Software, Methodology. **Yun Ge:** Validation, Resources, Investigation, Funding acquisition. **He Zhang:** Writing, Review and editing, Supervision, Methodology. **Fuyang Zhang** and **Yijiang Zheng:** Visualization, Supervision, Formal analysis. **Zhongshuo Ding:** Validation, Data curation, Resources, Investigation. It should be noted that **Liwei Yang** and **Ping Li:** contributed equally to the manuscript.

References

[1] J. Y. Chen, Q.-K. Pan, J. S. Neufeld, Z.-H. Miao, Optimization of task assignment for multi-farm multi-weeding robots based on discrete artificial bee colony algorithm, *Expert Systems with Applications*, 2025, **267**, 126182, doi: 10.1016/j.eswa.2024.126182.

[2] F. Zeng, C. Fan, S. Shirafuji, Y. Wang, M. Nishio, J. Ota, Task allocation and scheduling to enhance human–robot collaboration in production line by synergizing efficiency and fatigue, *Journal of Manufacturing Systems*, 2025, **80**, 309-323, doi: 10.1016/j.jmsy.2025.03.006.

[3] K. Chaudhary, V. Chand, A. Prasad, B. Sharma, Robot motion control using dual avoidance scheme, *Engineered Science*, 2024, **32**, 1261, doi: 10.30919/es1261

[4] G. Cao, B. Zhang, Y. Li, Z. Wang, Z. Diao, Q. Zhu, Z. Liang, Environmental mapping and path planning for robots in orchard based on traversability analysis, improved LeGO-LOAM and RRT* algorithms, *Computers and Electronics in Agriculture*, 2025, **230**, 109889, doi: 10.1016/j.compag.2024.109889.

[5] Z. An, C. Li, Y. Han, M. Niu, Improved bidirectional JPS algorithm for mobile robot path planning in complex environments, *Computers, Materials & Continua*, 2025, **83**, 1347-1366, doi: 10.32604/cmc.2025.059037.

[6] Z. Ni, Q. Li, M. Zhang, Efficient motion planning for chili flower pollination mechanism based on BI-RRT, *Computers and Electronics in Agriculture*, 2025, **232**, 110063, doi: 10.1016/j.compag.2025.110063.

[7] H. Guo, Y. Li, H. Wang, C. Wang, J. Zhang, T. Wang, L. Rong, H. Wang, Z. Wang, Y. Huo, S. Guo, F. Yang, Path planning of greenhouse electric crawler tractor based on the improved A* and DWA algorithms, *Computers and Electronics in Agriculture*, 2024, **227**, 109596, doi: 10.1016/j.compag.2024.109596.

[8] S. Li, M. Zhang, N. Wang, R. Cao, Z. Zhang, Y. Ji, H. Li, H. Wang, Intelligent scheduling method for multi-machine cooperative operation based on NSGA-III and improved ant colony algorithm, *Computers and Electronics in Agriculture*, 2023, **204**, 107532, doi: 10.1016/j.compag.2022.107532.

[9] J. Xin, J. Zhong, S. Li, J. Sheng, Y. Cui, Greedy mechanism based particle swarm optimization for path planning problem of an unmanned surface vehicle, *Sensors*, 2019, **19**, 4620, doi: 10.3390/s19214620.

[10] C. Miao, G. Chen, C. Yan, Y. Wu, Path planning optimization of indoor mobile robot based on adaptive ant colony

algorithm, *Computers & Industrial Engineering*, 2021, **156**, 107230, doi: 10.1016/j.cie.2021.107230.

[11] F. Ding, X. Luo, Z. Zhang, L. Hu, X. Wu, K. Bao, J. Zhang, B. Yuan, W. Zhang, Dual-unloading mode autonomous operation strategy and cotransporter system for rice harvester and transporter, *Engineering*, 2025, **48**, 220-233, doi: 10.1016/j.eng.2024.11.006.

[12] P. He, J. Li, A joint optimization framework for wheat harvesting and transportation considering fragmental farmlands, *Information Processing in Agriculture*, 2021, **8**, 1-14, doi: 10.1016/j.inpa.2020.04.006.

[13] H. Tian, Z. Mo, C. Ma, J. Xiao, R. Jia, Y. Lan, Y. Zhang, Design and validation of a multi-objective waypoint planning algorithm for UAV spraying in orchards based on improved ant colony algorithm, *Frontiers in Plant Science*, 2023, **14**, 1101828, doi: 10.3389/fpls.2023.1101828.

[14] J. Yang, J. Ni, Y. Li, J. Wen, D. Chen, The intelligent path planning system of agricultural robot via reinforcement learning, *Sensors*, 2022, **22**, 4316, doi: 10.3390/s22124316.

[15] H. Fu, Z. Li, W. Zhang, Y. Feng, L. Zhu, X. Fang, J. Li, Research on path planning of agricultural UAV based on improved deep reinforcement learning, *Agronomy*, 2024, **14**, 2669, doi: 10.3390/agronomy14112669.

[16] Y. Jiao, Y. Wang, X. Su, F. Wang, an ant colony hybrid simulated annealing algorithm for collaborative optimization of robotic mixed-model parallel two-sided assembly lines balancing, *Computers & Operations Research*, 2025, **182**, 107113, doi: 10.1016/j.cor.2025.107113.

[17] Y. Yao, Q. Wang, C. Wang, X. Li, L. Gao, K. Xia, Knowledge-based multi-objective evolutionary algorithm for energy-efficient flexible job shop scheduling with mobile robot transportation, *Advanced Engineering Informatics*, 2024, **62**, 102647, doi: 10.1016/j.aei.2024.102647.

[18] Z. Lu, Y. Wang, F. Dai, Y. Ma, L. Long, Z. Zhao, Y. Zhang, J. Li, A reinforcement learning-based optimization method for task allocation of agricultural multi-robots clusters, *Computers and Electrical Engineering*, 2024, **120**, 109752, doi: 10.1016/j.compeleceng.2024.109752.

[19] C. Zhang, L. Jia, S. Liu, G. Dou, Y. Liu, B. Kong, Dynamic job allocation method of multiple agricultural machinery cooperation based on improved ant colony algorithm, *Scientific Reports*, 2024, **14**, 22414, doi: 10.1038/s41598-024-73385-w.

[20] R. Cao, S. Li, Y. Ji, Z. Zhang, H. Xu, M. Zhang, M. Li, H. Li, Task assignment of multiple agricultural machinery cooperation based on improved ant colony algorithm, *Computers and Electronics in Agriculture*, 2021, **182**, 105993, doi: 10.1016/j.compag.2021.105993.

[21] X.-F. Liu, Y. Fang, Z.-H. Zhan, J. Zhang, Strength learning particle swarm optimization for multiobjective multirobot task scheduling, *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2023, **53**, 4052-4063, doi: 10.1109/TSMC.2023.3239953.

[22] H. Guo, Z. Miao, J. Ji, Q. Pan, An effective collaboration evolutionary algorithm for multi-robot task allocation and scheduling in a smart farm, *Knowledge-Based Systems*, 2024, **289**, 111474, doi: 10.1016/j.knosys.2024.111474.

[23] M. Wang, T. Ma, G. Li, X. Zhai, S. Qiao, Ant colony optimization with an improved pheromone model for solving MTSP with capacity and time window constraint, *IEEE Access*, 2020, **8**, 106872-106879.

[24] W. F. Li, L. X. Xu, L. L. Yang, W. R. Liu, K. P. Pan, & C. L. Li. Multi-field path planning method and test for agricultural robots based on improved ant colony algorithm, *Journal of Nanjing Agricultural University*, 2024, **47**(4), 823-834, doi: 10.16383/j.aas.c190684.

[25] H. Seyyedhasani, J. S. Dvorak, Using the Vehicle Routing Problem to reduce field completion times with multiple machines, *Computers and Electronics in Agriculture*, 2017, **134**, 142-150, doi: 10.1016/j.compag.2016.11.010.

[26] L. Yang, P. Li, T. Wang, J. Miao, J. Tian, C. Chen, J. Tan, Z. Wang, Multi-area collision-free path planning and efficient task scheduling optimization for autonomous agricultural robots, *Scientific Reports*, 2024, **14**, 18347, doi: 10.1038/s41598-024-69265-y.

[27] L. Yang, Y. Ge, Y. Zheng, H. Zeng, Cross-area scheduling and conflict-free path planning for multiple robots in non-flat environments, *Expert Systems with Applications*, 2025, **272**, 126767, doi: 10.1016/j.eswa.2025.126767.

[28] W. Gao, C. Liu, Y. Zhan, Y. Luo, Y. Lan, S. Li, M. Tang, Automatic task scheduling optimization and collision-free path planning for multi-areas problem, *Intelligent Service Robotics*, 2021, **14**, 583-596, doi: 10.1007/s11370-021-00381-8.

[29] Z. Z. Chen, D. M. Zhang, & Z. Y. Xin. Multi-subpopulation-based symbiosis and non-uniform Gaussian mutation salp swarm algorithm, *Acta Automatica Sinica*, 2022, **5**, 1307-1317, doi: 10.16383/j.aas.c190684.

[30] B. Fan, L. Guo, An improved JPS algorithm for global path planning of the seabed mining vehicle, *Arabian Journal for Science and Engineering*, 2024, **49**, 3963-3977, doi: 10.1007/s13369-023-08232-7.

[31] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, S. M. Mirjalili, Salp Swarm Algorithm: a bio-inspired optimizer for engineering design problems, *Advances in Engineering Software*, 2017, **114**, 163-191, doi: 10.1016/j.advengsoft.2017.07.002.

[32] X. X. Li, T. H. Fei, J. Y. Yang, L. H. Yang, & X. J. Li. Research on random obstacle avoidance method for robots based on the fusion of improved A* algorithm and dynamic window

method, *Journal of Instrumentation*, 2024, **42**(3), 132-140, doi:10.19650/j.cnki.cjsi.J2007064.
[33] S. Guo, J. Gong, H. Shen, L. Yuan, W. Wei, Y. Long, DBVSB-P-RRT*: a path planning algorithm for mobile robot with high environmental adaptability and ultra-high speed planning, *Expert Systems with Applications*, 2025, **266**, 126123, doi: 10.1016/j.eswa.2024.126123.
[34] Z. Feng, L. Zhou, J. Qi, S. Hong, DBVS-APF-RRT*: a global path planning algorithm with ultra-high speed generation of initial paths and high optimal path quality, *Expert Systems with Applications*, 2024, **249**, 123571, doi: 10.1016/j.eswa.2024.123571.
[35] K. Li, X. Yan, Y. Han, Multi-mechanism swarm optimization for multi-UAV task assignment and path planning in transmission line inspection under multi-wind field, *Applied Soft Computing*, 2024, **150**, 111033, doi: 10.1016/j.asoc.2023.111033.
[36] K. Panwar, K. Deep, Discrete salp swarm algorithm for euclidean travelling salesman problem, *Applied Intelligence*, 2023, **53**, 11420-11438, doi: 10.1007/s10489-022-03976-5.
[37] S. Mahmoudinazlou, C. Kwon, A hybrid genetic algorithm for the Min–max Multiple Traveling Salesman Problem, *Computers & Operations Research*, 2024, **162**, 106455, doi: 10.1016/j.cor.2023.106455.
[38] Y. Li, H. Wu, A clustering method based on K-means algorithm, *Physics Procedia*, 2012, **25**, 1104-1109, doi: 10.1016/j.phpro.2012.03.206.
[39] R. Zhu, T. Hou, Y. Guo, X. Pei, J. Zhu, Improved multi-strategy salp swarm algorithm for UAV path planning, *IEEE International Conference on Unmanned Systems (ICUS)*, Hefei, China, IEEE, October 13-15, 2023, 1601-1606, doi: 10.1109/ICUS58632.2023.10318445.
[40] N. R. Sturtevant, Benchmarks for grid-based pathfinding, *IEEE Transactions on Computational Intelligence and AI in Games*, 2012, **4**, 144-148, doi: 10.1109/TCIAIG.2012.2197681.