SIN-CHN SCIENTIFIC PRESS PTE.LTD.

Molecular & Cellular Biomechanics 2025, 22(2), 515.
https://doi.org/10.62617/mcb515

Article

Thoughts on the rational layout of automatic pipelines in aircraft engines brought by biomechanical co-evolutionary algorithm and improved A* algorithm

Yingxue Li*, Guolong Gan, Xiaolin Zhang, Feng Chen, Yong Zhang, Mingliang Li

Avic Chengfei Commercial Aircraft Co., Ltd, Chengdu 610065, Sichuan, China
*** Corresponding author:** Yingxue Li, Liyx030@avic.com

CITATION

Li Y, Gan G, Zhang X, et al. Thoughts on the rational layout of automatic pipelines in aircraft engines brought by biomechanical co-evolutionary algorithm and improved A* algorithm. Molecular & Cellular Biomechanics.2025; 22(2): 515. https://doi.org/10.62617/mcb515

ARTICLE INFO

Received: 12 October 2024
Accepted: 1 November 2024
Available online: 7 February 2025

Abstract: In the field of biomechanics, the delicate structural layout of living organisms often brings many inspirations for engineering design. As in the case of aero-engine piping layout, the current single- and multi-tube layouts are ineffective and need to be optimised. Inspired by the efficient material transfer and space utilisation mechanisms of biological systems, we propose an automatic pipe layout method for aero-engine based on co-evolutionary algorithm and improved A* algorithm. Taking inspiration from how biological networks adapt and optimize their connections, we first construct an improved A* algorithm. Through optimizing node coordinate expression, enhancing the evaluation function, introducing a directional strategy, and improving the OPEN_LIST, it becomes a potent tool. When applied to single-pipe layout in aero-engines and compared with the original A* algorithm, its advantages are evident. Subsequently, mimicking the collaborative evolution seen in ecological systems, we combine the co-evolutionary algorithm with a new evaluation function to develop a further improved A* algorithm for multi-pipe layouts. Finally, simulations confirm the feasibility and effectiveness of our proposed method. For single pipes, similar to nature's way of streamlining structures, our method significantly reduces pipe length and the number of elbows while effectively avoiding key equipment. The improved A* algorithm cuts pipe lengths by 12.8275% and 19.4843% respectively and boosts the computation speed by remarkable percentages. For multi-pipes, it enhances space utilization and reduces time cost. The improved algorithm reduces the number of traversing nodes from 3067 to 1968 and shortens the planning time from 20.34 s to 7.26 s, demonstrating its great efficacy.

Keywords: aero-engine; biomechanics; automatic layout; co-evolutionary algorithm; A*algorithm

1. Introduction

The pipe layout of aero-engine is an important part of aero-engine design, and its quality directly affects the performance, reliability, lifecycle, and economic cost of the aero-engine [1]. Automatic optimization of pipeline layout design is an important way to improve the quality of pipeline layout, shorten pipeline layout time, and reduce layout cost [2]. There are many challenges in the traditional aero-engine pipeline layout design. For example, the structure of aero-engine is complex, and the pipeline layout needs to consider the characteristics of limited internal space, numerous components and compact layout. How to realize the rational arrangement of the pipeline in the limited space, not only to ensure the performance of the engine, but also to meet the safety and reliability requirements, is the primary problem faced by the designer.

In order to achieve automation and intelligent design of pipes of aero-engine, many scholars have conducted extensive research in recent years. Some methods have been proposed, most of them can be classified into heuristic routing algorithms, cell-based routing algorithms, and graph-based routing algorithms.

For the heuristic routing algorithms, Jiao et al. [3] proposed a multi-objective pipeline routing method based on improved MOEA/D, which is used to automatically generate non dominated solutions for pipeline layout in three-dimensional space and aero-engine rotation space. In addition, this method can be extended to surface to meet the needs of paving pipes on rotating surfaces of aero-engine. Yuan et al. [4] proposed an automatic multi pipes layout method for aero-engine that emphasizes parallel arrangements. This method constructs an algorithm by compressing the visibility map to quickly determine the rough path and interference relationship of the pipeline to be layout. Based on the paths and co-evolutionary algorithm, the optimization problem of multi pipes layout was solved. The simulation results on aero-engine have verified the feasibility and effectiveness of the method. Wu et al. [5]and Liu et al. [6] both proposed an improved particle swarm algorithm to find the optimal route for pipelines and the algorithm showed good computational convergence. Liu and Tong [7] proposed a multi-objective pipe routing algorithm based on NSGA-II to consider the vibration performance of aircraft engines, while taking into account pipeline length, smoothness, and natural frequency. For cell-based routing algorithms, Wittmann et al. [8] used a genetic algorithm (GA) method to optimize interactive planning of pipeline routing paths. Park [9] developed an element generation method that satisfies geometric constraints to replace traditional methods such as element decomposition or network optimization. This method has been validated through multiple simulations. Ren et al. [10] carefully considered the spatial characteristics and major engineering constraints of aero-engine to improve the genetic algorithm, including improvements initiation and direction guideline. This method is used for pipeline routing in subspaces, which can effectively avoid local optimal solutions during the routing process. For graph-based routing algorithms, Kim et al. [11] designed an automatic pipeline routing system that takes into account factors such as the complexity of physical and operational limitations. Liu and Wang [12] proposed a graph-based routing algorithm to attempt to find the shortest collision free pipeline path in the circumferential space of the aero-engine between the hub and the shell. And by combining two adaptive strategies, the algorithm is extended from the visibility map of robot path planning in 2D space to the 3D circumferential space of aero-engine. Qu et al. [13] developed a new method for routing straight branch pipes, which is used to automatically generate the optimal straight branch pipe routing in confined spaces. This method utilizes a novel concurrent Max-Min Ant System optimization algorithm, incorporating a concurrent search strategy and a dynamic update mechanism, to address the optimization problem of pipe layout. With the development of artificial intelligence, the pipeline layout methods based on A^* algorithm and co-evolutionary algorithm have also been applied. Yao [14] improved the A^* algorithm using the co-evolutionary algorithm and achieved automatic layout of a single pipeline. Wu et al. [15] also proposed an improved A^* algorithm, which introduces exclusion terms and dynamically adjusts weights for heuristic terms to design a new evaluation function. However, to a large extent, the above research cannot simultaneously consider the

utilization of time and space for layout. The cost-effective pipeline routing algorithm that can automatically generate multi-objective solutions is still an open research field. Wu and Hao [16] applied Lee algorithm and Tabu search algorithm to optimize the layout order to complete the multi-tube layout design. Wu et al. [17] used the improved firefly algorithm to optimize the sequence of pipeline layout. Liu et al. [18] introduced the Kriging agent model when planning the pipeline laying sequence, which improved the calculation efficiency. In addition, another feasible idea for planning multi-pipe systems is swarm intelligent optimization algorithm based on the overall coding mechanism. Ando et al. [19] use the overall coding idea for pipeline layout. The existing pipe layout optimization methods have made a beneficial exploration to solve the layout design problem of multi-pipe pipe system. Due to the complexity of the multi-pipe layout design problem, a complete solution has not yet been formed.

In this paper, an automatic pipe layout method for aero-engine based on co-evolutionary algorithm and improved A*algorithm is proposed. To address the shortcomings of the above research, this method considers the optimization of evaluation functions and state tables. The coordinate representation and directional guidance of the pipeline are also considered. The remainder of this paper is organized as follows: Section 2 gives the problem description of pipe layout for aero-engine; Section 3 gives the improved description of the A*algorithm and conducts simulation verification; Section 4 concludes this paper.

2. Problem description of pipe layout for aero-engine

2.1. Spatial representation

Figure 1 shows a simplified model of the layout space of the aircraft engine piping system. This space is a revolving space between the inner casing and the nacelle, and there are obstacles such as fuel system, hydraulic system, air system, etc. [20,21]. The outer surface of the inner casing and the inner surface of the nacelle can be considered as the rotational surface or approximate rotational surface generated by a certain curve rotating around a certain straight line. Therefore, the study of automatic pipe layout for aero-engine is of great significance.

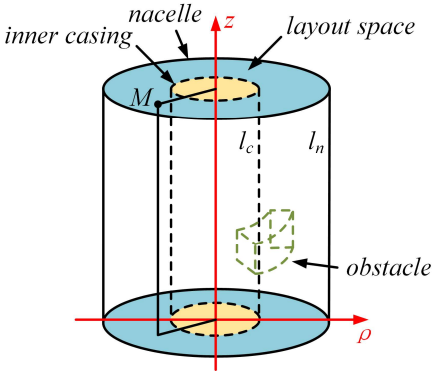


Figure 1. Simplified model of the layout space of the aircraft engine piping system.

Assuming l_c and l_n are the generatrixes of the outer surface of the inner casing and the inner surface of the nacelle, they can be expressed as:

$$l_c: \begin{cases} F_c(\rho, z) = 0 \ (0 \le z \le h_0) \\ \theta = 0 \end{cases} \tag{1}$$

$$l_n: \begin{cases} F_n(\rho, z) = 0 \ (0 \le z \le h_0) \\ \theta = 0 \end{cases} \tag{2}$$

where h_0 is the axial length of engine casing.

For ease of representation, the l_c and l_n are denoted as the $\rho_c = f_c(z)$ and $\rho_n = f_n(z)$, respectively. Therefore, the layout space can be expressed as:

$$\begin{cases} f_c(z) < \rho < f_n(z), \\ 0 \le \theta < 2\pi, \\ 0 < z < h_0 \end{cases} \tag{3}$$

Additionally, the obstacles in the pipe space based on the principle of minimum fan ring containment box can be expressed as:

$$\begin{cases} \rho_{\min} \le \rho \le \rho_{\max}, \\ \theta_{\min} \le \theta \le \theta_{\max}, \\ \theta_{\min} \le \theta \le \theta_{\max} \end{cases} \tag{4}$$

where $(\rho_{\min}, \rho_{\max}) \times (\theta_{\min}, \theta_{\max}) \times (z_{\min}, z_{\max})$ is the range of value for obstacle containment box in cylindrical coordinate system.

2.2. Mathematical model of layout

When arranging pipelines in three-dimensional space, the path can be viewed as a series of connected polylines, with the intersection points of these polylines representing the corresponding nodes, as shown in **Figure 2**.

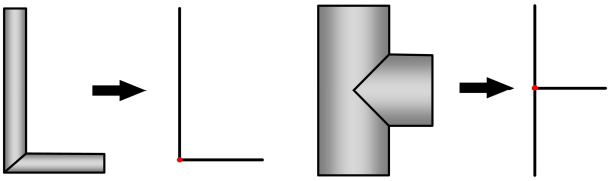


Figure 2. Pipeline simplified model.

Therefore, the path is represented using the variable length node position encoding:

$$path = [(x_s, y_s, z_s), \ldots, (x_i, y_i, z_i), \ldots, (x_T, y_T, z_T)] \tag{5}$$

where (x_i, y_i, z_i) is the spatial coordinates of the i-th node.

The mathematical model of pipeline layout which aims to optimize the shortest pipeline path can be expressed as:

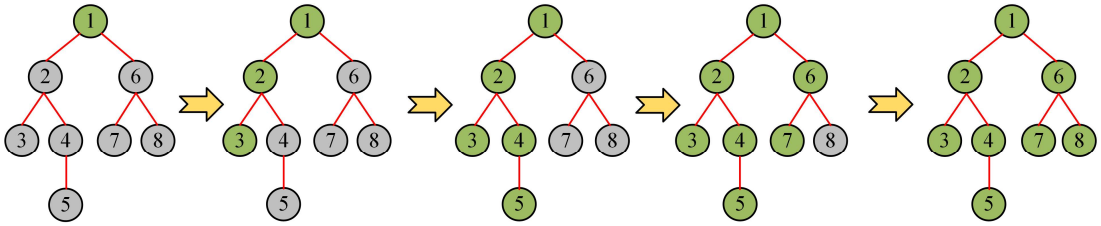
$$min\, f\,(l_1, \ldots, l_i, \ldots, l_n) = min \sum_{i=1}^{n} l_i \tag{6}$$

where l_i is the length of the i-th section of the pipeline, $l_i = \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2 + (z_i - z_{i+1})^2}$; n is the number of main pipe segments in a pipeline.

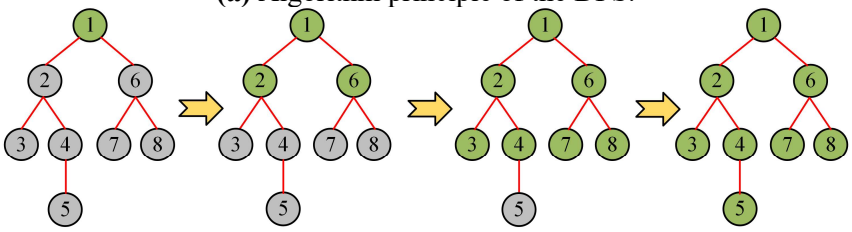
3. Theoretical backgrounds

3.1. A* algorithm

The A* algorithm is a typical heuristic search algorithm [22,23]. It can incorporate heuristic information and guidance paths related to the problem during the algorithm search process to reduce problem complexity and improve algorithm solving speed. At the same time, it combines the advantages of DFS algorithm (Depth First Search) and BFS algorithm (Breadth First Search), which improves the efficiency of algorithm while determining the optimal path that can be searched [24,25]. The algorithm principles of the DFS and BFS are shown in **Figure 3**.



(a) Algorithm principle of the DFS.



(b) Algorithm principle of the BFS.

Figure 3. Algorithm principles.

The pathfinding idea of A* algorithm consists of the following steps:

(1) Initialization. Add the start node to the open List and set its heuristic function estimate (usually denoted as *F* value) to 0 and the actual path length (usually denoted as *G* value) of the start node to 0;

(2) Circular search. When the open list is not empty, repeat the following steps:

1) Select the node with the smallest *F*-value in the open list as the current node;

2) Remove the current node from the open list and add it to the closed list (close List).

(3) Traverse the neighbor node. For each neighbor of the current node:

1) If the neighbor node is already in the close list, it is skipped.

2) Calculate the actual path length (*G* value) from the start node through the current node to the neighbor node:

G value = *G* value of the current node + the actual distance between the current node and the neighbor node;

3) If the neighbor node is not in the open list, add it to the open list and set its parent node as the current node;

4) If the neighbor node is already in the open list and the actual path length of the new path is smaller than the known path length, update the parent node to the current node and recalculate its actual path length.

(4) Update node information. For each node in the open list, update its heuristic function estimate (*F* Value):

1) *F* value = *G* value + *H* value, where *G* value is the actual path length from the node to the start node, and *H* value is the estimated heuristic function from the node to the target node;

2) The estimated value of the heuristic function can be Manhattan distance, Euclidean distance, etc., but it should be an optimistic estimate, that is, the actual path length should not be underestimated.

(5) Termination conditions. When the target node is added to the closed list or the open list is empty, the algorithm terminates.

(6) Backtracking path. If the destination node is found, the shortest path from the start node to the destination node can be reconstructed by retracing the nodes in the shutdown list.

The A* algorithm selects the node with the smallest evaluation function value among the adjacent nodes of the current node for expansion at each iteration, and its evaluation function can be expressed as:

$$f(n) = g(n) + h(n) \tag{7}$$

where $f(n)$ is the sum of actual cost value and estimated cost value; $g(n)$ is the actual cost value from starting node to node *n*; $h(n)$ is the inspiration factor, which is used to estimate the cost value from node *n* to the target node.

Assuming the $h'(n)$ is the actual cost value from node *n* to the target node:

a) If $h(n) = 0$, the A* algorithm will degrade to the Dijkstra algorithm due to the loss of heuristic information. At this point, the algorithm will perform a blind search. Although it can ensure finding the optimal path, it will greatly reduce the execution efficiency of the algorithm.

b) If $h(n) < h'(n)$, smaller value of $h(n)$ increases the search range and expands the number of nodes, which in turn leads to lower execution efficiency.

c) If $h(n) = h'(n)$, the algorithm will perform strict path optimization and maintain a certain level of efficiency.

d) If $h(n) > h'(n)$, although the algorithm can maintain a small search range and fewer expansion nodes, it cannot guarantee the search for the optimal path.

And there are three common heuristic factors: Chebyshev distance, Manhattan distance, and Euclidean distance, which are expressed as follows [26]:

(1) The Chebyshev distance represents the maximum difference in coordinate distance between the current and target nodes:

$$h(n) = max(|x_n - x_1|, |y_n - y_1|, |z_n - z_1|) \tag{8}$$

(2) The Manhattan distance represents the sum of the distances between the current node and the target node on the horizontal and vertical axes:

$$h(n) = |x_n - x_1| + |y_n - y_1| + |z_n - z_1| \tag{9}$$

(3) The Euclidean distance represents the straight-line distance between the current node and the target node:

$$h(n) = \sqrt{|x_n - x_1|^2 + |y_n - y_1|^2 + |z_n - z_1|^2} \quad (10)$$

The Chebyshev distance, Manhattan distance, and Euclidean distance are shown in **Figure 4**.

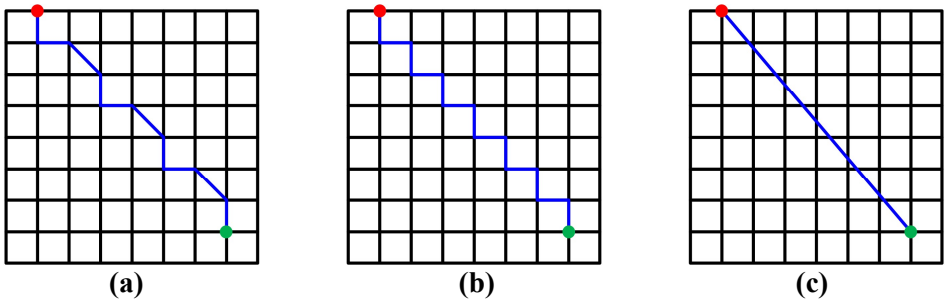


Figure 4. Types of distance calculation. **(a)** Chebyshev distance; **(b)** Manhattan distance; **(c)** Euclidean distance.

Additionally, the execution of the A* algorithm requires the introduction of two state tables (i.e., the OPEN_LIST and the CLOSE _LIST). During the search process, the starting node is added to the CLOSE_LIST table, and the 8 nodes around the starting node are added to the OPEN_LIST table. The cost value *f* of each node is calculated, and the node with the smallest cost value, which is the yellow node in the **Figure 5**. Then the algorithm will update the OPEN_LIST table to the position of the CLOSE_LIS table, and use the node with the lowest cost as the new starting node until it reaches the target node. And the flow chart of A*algorithm is shown in the **Figure 6**.

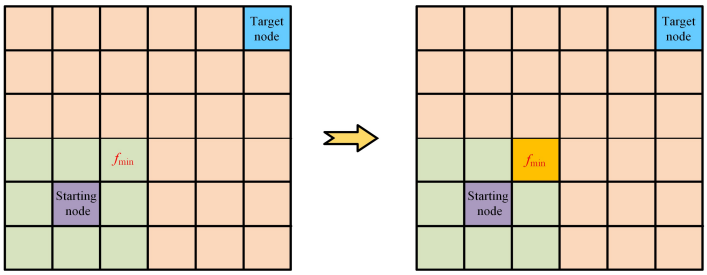


Figure 5. Schematic diagram of node selection for the A* algorithm.

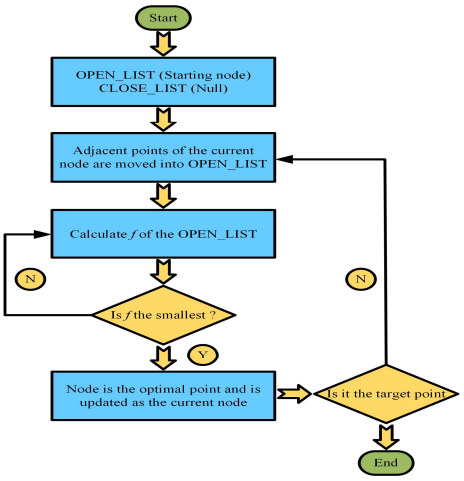


Figure 6. Flow chart of the A* algorithm.

3.2. Improved A* algorithm

Figure 7 shows a simplified structural layout of the aero-engine. From **Figure 6**, it can be seen that the internal structure of the aero-engine is complex. In the design of aircraft engine piping layout, the length of the pipes should be as short as possible, the number of bends should be minimized, and the pipes should avoid interference with other obstacles. Sometimes, pipelines should also be distributed orthogonally as much as possible. However, the traditional A* algorithm has a simple principle and relies heavily on evaluation function, directional strategy, and state tables, which is not conducive to solving complex aviation engine pipeline layout problems. Therefore, it is necessary to improve the traditional A* algorithm.

Figure 7. Simplified structural layout of the aero-engine.

3.2.1. Improvement of node coordinate expression method

The use of Cartesian coordinate system in pipeline layout using the A* algorithm may lead to inaccurate evaluation functions, low efficiency in accessing state tables, and inability to effectively handle the complexity and directionality of path. Therefore, the spatial position of grid nodes has been improved to convert Cartesian coordinate system into three-dimensional array coordinate system. The three-dimensional array coordinate system represents the row, column, and layer positions where the grid nodes are located. The Cartesian coordinate system and three-dimensional array coordinate system are shown in **Figure 8**.

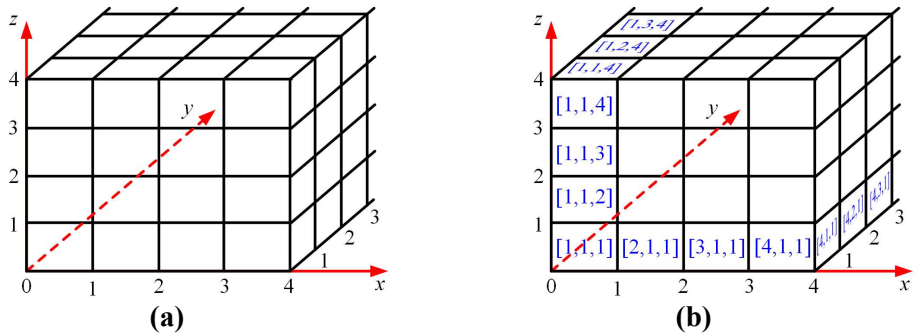


Figure 8. Cartesian coordinate system and three-dimensional array coordinate system. **(a)** Cartesian coordinate system; **(b)** Three-dimensional array coordinate system.

In the three-dimensional layout space, the range of values for the *x*-axis, *y*-axis, and *z*-axis of the Cartesian coordinate system are [0, *L*], [0, *B*], and [0, *H*], respectively. The space is divided into grid combinations using the grid method, with each grid

being a cube with a side length of a. So, the grid node [1, 1, 1] can be composed of point (0, 0, 0), (a, 0, 0), (a, a, 0), (0, a, 0), (a, 0, a), (a, a, a), and (0, a, a). Therefore, for every change of a unit on the x-axis, y-axis, and z-axis in the Cartesian coordinate system, the row, column, and layer coordinates in the three-dimensional array change by 1 unit accordingly. The three-dimensional array coordinate of the grid node is shown in **Figure 9**.

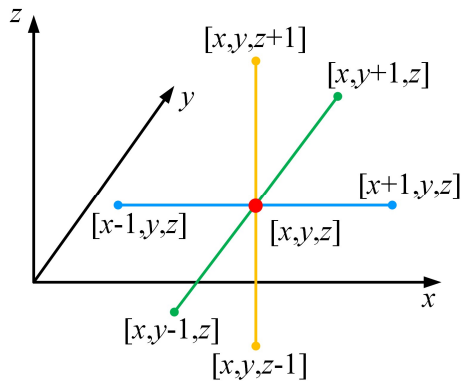


Figure 9. Three-dimensional array coordinate of the grid node.

3.2.2. Optimization of evaluation function

The evaluation function directly affects the efficiency and accuracy of path search in the A* algorithm [27]. The difference between estimated value and actual value is crucial for the performance of the algorithm. To balance the relationship between the two and implement the path planning algorithm more reasonably, an optimized inspiration factor $h(n)$ is introduced.

This study optimizes the distance to obtain heuristic factors through dynamic weighting, and selects weighting values based on the actual situation of the path. The improved inspiration factor$h(n)$ can be expressed as:

$$h(n)=\begin{cases}\omega_1\times(|x_n-x_1|+|y_n-y_1|+|z_n-z_1|),|x_n-x_1|+|y_n-y_1|+|z_n-z_1|>\lambda,\omega_1\geq 1\\ \omega_2\times(|x_n-x_1|+|y_n-y_1|+|z_n-z_1|),|x_n-x_1|+|y_n-y_1|+|z_n-z_1|\leq\lambda,0<\omega_2<1\end{cases} \quad (11)$$

where x_1, y_1, z_1 are the coordinates of the current node; x_n, y_n, z_n are the coordinates of the target node; λ is the threshold; ω_1 and $\omega_1$are the weight, respectively.

Specifically, the weight increases and the algorithm search speed become faster when the original heuristic factor $h(n)$ exceeds the threshold λ. On the contrary, the weight decreases and the optimal path is given priority consideration when it is below the threshold. The threshold is dynamically selected based on the factors such as computer computing performance and map size, and the weight values are adjusted according to the size and complexity of the set map.

To highlight the effectiveness of the improved inspiration factor, it is compared with Chebyshev distance, Manhattan distance, and Euclidean distance on the same map. Set the optimized distance parameters λ = 16, ω_1 = 4, and ω_2 = 0.7 during the comparison process. The comparison results are shown in **Figure 10**.

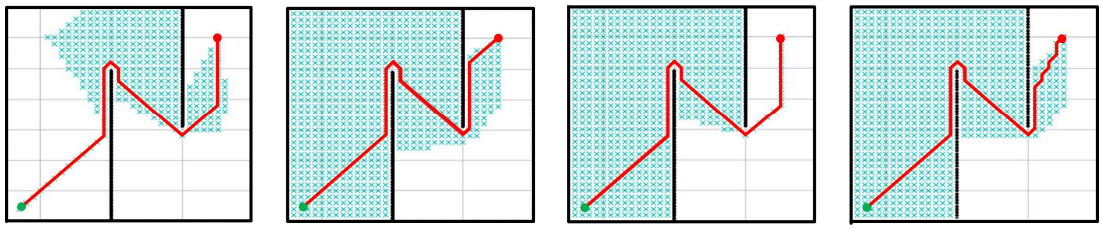


Figure 10. Three-dimensional array coordinate of the grid node. **(a)** Optimized distance; **(b)** Chebyshev distance;**(c)** Manhattan distance; **(d)** Euclidean distance.

From **Figure 10**, it can be seen that the paths using four heuristic factors can all avoid obstacles and reach the target node. Compared with other distance calculation methods, the nodes of optimized distance in this study are significantly reduced. Compared to Chebyshev distance and Euclidean distance, path of the optimized distance has fewer corners, which is more in line with the actual situation. The specific comparison information is shown in **Table 1**.

Table 1. Specific comparison information.

$h(n)$	Number of nodes	Computing time/ms	Number of corners
Optimized distance	325	293.2	7
Chebyshev distance	746	311.8	8
Manhattan distance	652	398.5	7
Euclidean distance	719	354.2	14

From **Table 1**, it can be concluded that the number of nodes of optimized distance nodes is only 325, with a calculation time of 293.2 ms. The simulation results demonstrate that the optimized heuristic factor significantly reduces the time required for path search and greatly enhances the efficiency of path planning while maintaining the accuracy of the path length.

3.2.3. Adjustment of directional strategy

When designing pipeline layout, it is also necessary to consider the demand relationship between pipelines and equipment. The pipelines should leave a certain safe space between equipment as much as possible. Meanwhile, the pipeline should avoid bypassing the equipment. Therefore, a directional strategy is introduced into A* algorithm to make the pipeline layout more in line with actual needs.

The key node method is an important method for achieving directional guidance in pipeline layout. In the evaluation function, the actual distance $g(n)$ represents the actual path length from the starting node to the current node. Direction guidance can be achieved by setting key nodes to adjust the value of $g(n)$.

For equipment that requires tight fitting of pipelines, two key nodes are set near the equipment to appropriately reduce the value of $g(n)$. At this point, the evaluation function for two key nodes is also relatively optimal. At this point, the evaluation function for key nodes is also relatively optimal. If the constraint rule of pipeline layout is converted into a length index, the algorithm will prioritize selecting these two nodes to attract pipeline layout, as shown in **Figure 11**. The adjustment formula for $g(n)$ is as follows:

$$g(n) = g(n-1) + (1-\gamma) \times d_{(n-1,n)} \quad \gamma \in (0,1) \tag{12}$$

where $d_{(n-1,n)}$ is the actual distance from node n−1 to node n; γ is the length index, which can adjust the numerical value of $g(n)$.

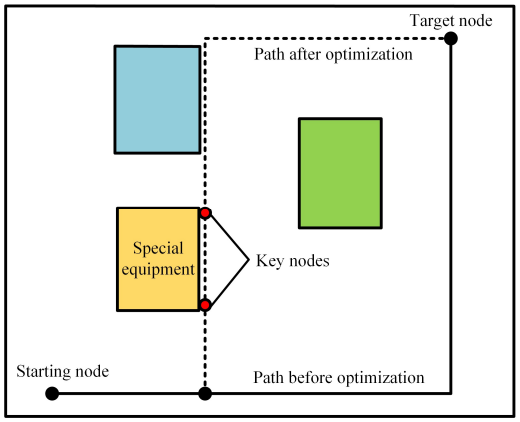


Figure 11. Key node attraction strategy.

For the equipment to be avoided, two key nodes are also set to increase $g(n)$ and evaluation function $f(n)$. In the process of pipeline layout, the A*algorithm will discard these two nodes to avoid equipment, as shown in **Figure 12**. The adjustment formula for $g(n)$ is as follows:

$$g(n) = g(n-1) + (1+\gamma) \times d_{(n-1,n)} \; \gamma \in (0,1) \tag{13}$$

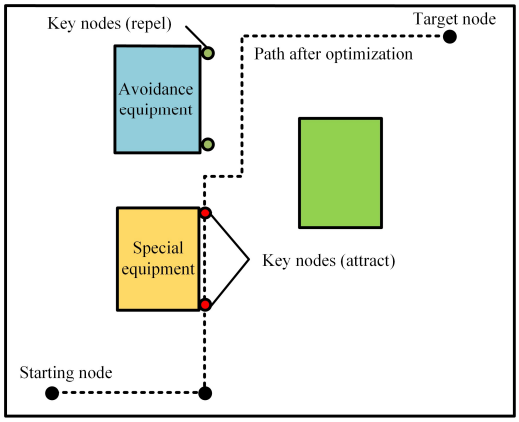


Figure 12. Key node exclusion strategy.

3.2.4. Optimization of OPEN_LIST

The two state lists (i.e., the OPEN_LIST and the CLOSE_LIST) are introduced to the A*algorithm to store nodes in path. Its running process can be simply summarized as: move in (move extended nodes into the OPEN_LIST)-sort-move out (move out of the OPEN_LIST). The most time-consuming operation is sorting, which aims to find the node with the smallest evaluation function value in the OPEN_LIST. The time is largely influenced by the structure of OPEN_LIST. In order to further improve the efficiency of the A* algorithm, the structure of the OPEN_LIST is optimized to accelerate the comparison process of the evaluation function and removal process of the nodes.

The common data structures include array, linked list, and binary tree [28], as shown in **Figure 13**.

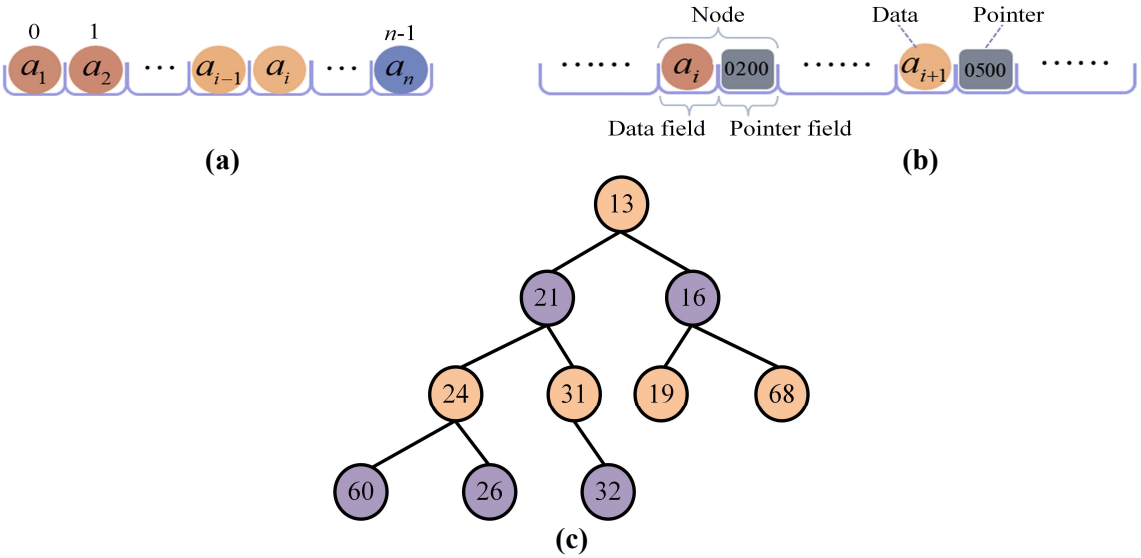


Figure 13. Common data structures. **(a)** Array; **(b)** Linked list; **(c)** Binary tree.

From **Figure 13**, it can be concluded that the array is a data structure of a linear table, where the data is unsorted. Therefore, the time complexity during operation is $O(n)$. For a linked list, its data structure is non continuous and nonsequential. The time complexity for moving out and in nodes is $O(1)$, while the time complexity for finding the most valuable node is $O(n)$. However, the binary tree is a tree structure in which the evaluation function of each node can be less than or equal to the evaluation function of nodes in other subtrees. The time complexity for performing operations such as moving in, sorting, and moving out is $O(\log n)$. Therefore, if the binary tree is used to store data, the A* algorithm only needs to read the node at the top of the heap each time, which greatly improves its running efficiency. The time complexity is shown in **Figure 14**.

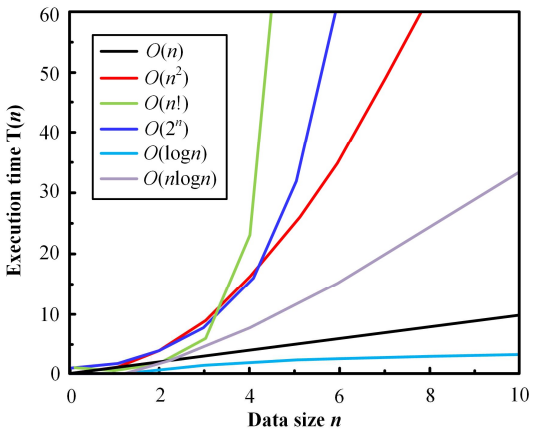


Figure 14. Time complexity.

3.2.5. Single pipeline simulation analysis

In order to validate the effectiveness of the improved A* algorithm, three paths are set up for simulation comparison, as shown in **Figure 15**.

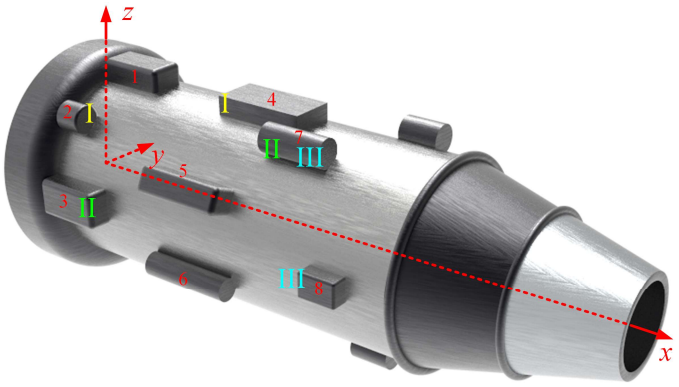


Figure 15. Path description 1.

From **Figure 15**, it can be seen that there are 3 paths to be arranged and 8 obstacles set up in the space. The starting and ending points of these three paths are different, and the coordinate information of the obstacles and starting and ending points are shown in **Tables 2** and **3**, respectively.

Table 2. Coordinate information of the obstacles.

Number	Diagonal coordinates
1	(200, −70, 350), (440, −430, 430)
2	(200, −280, 275), (300, −280, 275)
3	(200, −340, −70), (430, −430, 65)
4	(775, 345, −85, 345), (1120, 415, 85, 415)
5	(680, −335, 1200), (1000, −390, 220)
6	(675, −360, −160), (1020, −360, −160)
7	(1070, −180, 360), (1330, −180, 360)
8	(1380, −350, −470), (1520, −40, 40)

Table 3. Coordinate information of the starting and ending points.

Number	starting point	ending point
I	(300, −280, 275)	(820, −85, 390)
II	(430, −395, 0)	(1120, −220, 330)
III	(1260, −220, 330)	(1380, −395, 0)

The three pipelines are arranged separately using the improved A* algorithm. The results are compared with the unimproved A*algorithm, as shown in **Figure 16**.

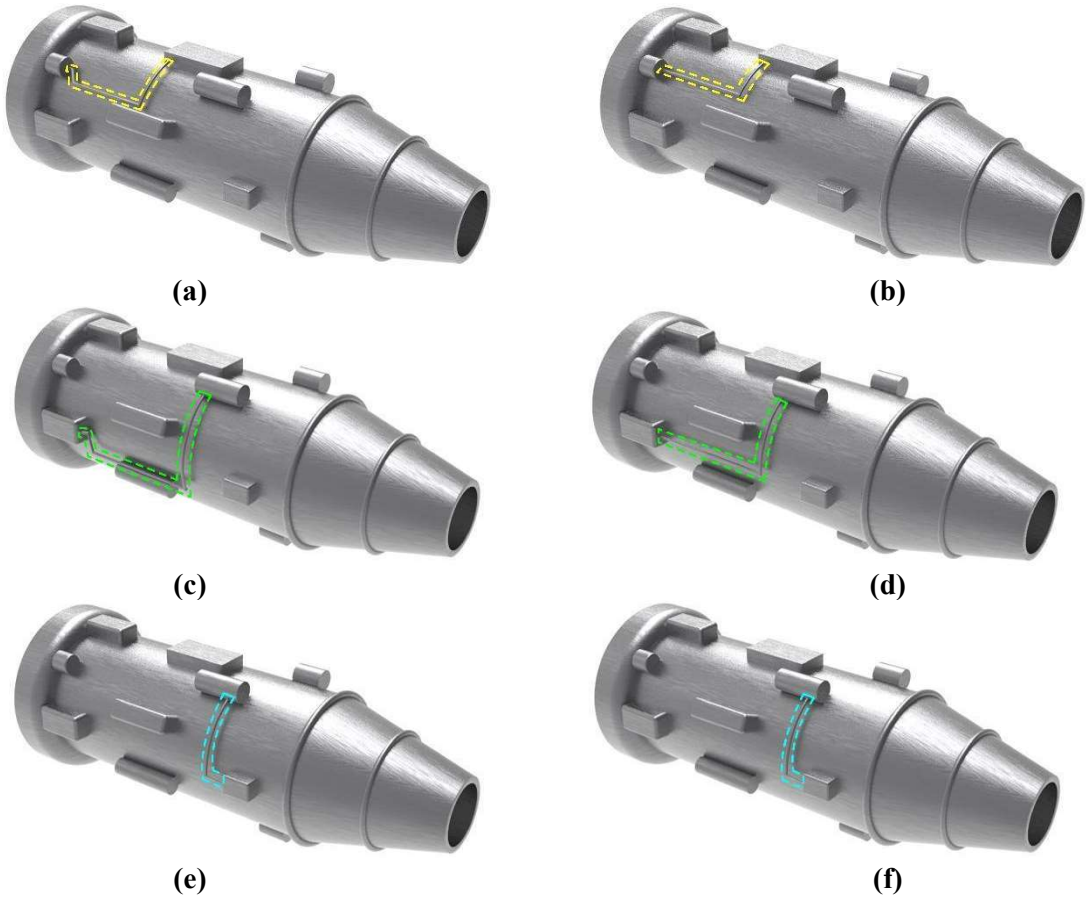


Figure 16. Comparison of results. **(a)** Path I with unimproved A* algorithm; **(b)** Path I with improved A* algorithm; **(c)** Path II with unimproved A* algorithm;**(d)** Path II with improved A* algorithm; **(e)** Path III with unimproved A* algorithm;**(f)** Path III with improved A* algorithm.

The detailed comparison of results is presented in **Table 4**.

Table 4. Specific comparison of results.

Number	Method	Pipeline length/m	Pipeline elbow	Avoidance distance/m	Time/ms
I	unimproved A* algorithm	4.1723	3	0.0527	546.8712
	improved A* algorithm	3.6371	1	0.0916	228.6983
II	unimproved A* algorithm	5.6117	3	0.0291	746.4349
	improved A* algorithm	4.5183	1	0.0962	248.8951
III	unimproved A* algorithm	2.8524	1	0.2076	352.4237
	improved A* algorithm	2.8524	1	0.2076	170.1286

From **Table 4**, it can be concluded that the improved A*algorithm performs excellently. Specifically, the length of the pipeline of improved A*algorithm is 3.6371 m, which is less than the pipeline length of the unimproved A*algorithm in path I. And the number of pipeline elbow is smaller than that of the unimproved A* algorithm. This greatly improves space utilization. The improved A*algorithm has a significant avoidance effect on obstacle 5, with an avoidance distance of 0.0916 m. In addition, the improved A* algorithm has a pipeline layout time of 228.6983 ms, which is

58.1806% less than the unimproved A*algorithm. In path II, the performance of the improved A*algorithm is similar to that in path I. Due to the increase in pipeline length, both have increased in time. But the improved A*algorithm still performs well, with a 66.6554% reduction in computation time compared to the unimproved A*algorithm. In path III, there are almost no obstacles, so the difference between the two algorithms is only reflected in the time. The improved A*algorithm reduces 51.7261% compared to the unimproved A*algorithm. The above analysis demonstrates that the improved A* algorithm is effective in the layout of aircraft engine pipelines.

3.3. Improved A* algorithm based on co-evolutionary algorithm

However, the layout of aircraft engine pipelines is not a single pipeline layout problem, but a process of simultaneously arranging multiple pipelines [29]. Therefore, the single optimization algorithm is difficult to meet all requirements. In this section, A* algorithm is further improved based on co- evolutionary algorithm to solve the problem of multiple pipeline layout.

In order to verify the performance of the improved A* algorithm, MATLAB software was used for simulation and comparison experiments, and raster map was used for simulation experiments. The raster map size was 20 × 20 and 50 × 50 respectively. Set the safety distance of quadratic line optimization to 1, the expansion node unit before path smoothing to 0.1, and the moving window size to 9. The performance of the algorithm is compared and analyzed in different scale raster maps under the same conditions.

3.3.1. Co-evolutionary algorithm

The co-evolutionary algorithm from the theory of biological coevolution and is anew evolutionary algorithm based on traditional evolutionary algorithm [30]. Its principle is to establish two or more populations and add appropriate population relationship conditions to achieve evolution between populations, thereby obtaining the optimal solution. The coevolutionary relationship between populations is shown in **Figure 17**.

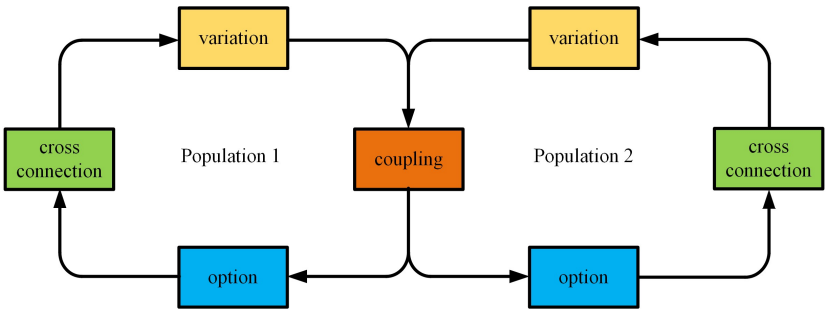


Figure 17. Relationship between population coevolution.

In addition, the co-evolutionary algorithm includes competitive co-evolutionary algorithm, lotka-volterraco-evolutionary algorithm, and cooperative co-evolution algorithm. Among them, competitive co-evolutionary algorithm and lotka-volterraco-evolutionary algorithm have the problems of large search space range and low efficiency, respectively. However, the cooperative co-evolution algorithm can ensure that the current population is in an evolutionary state while the population is in a silent

state, which not only reduces the complexity of the algorithm but also obtains the complete optimal solution of the final problem [31,32]. Therefore, for the problem of pipeline layout in the aircraft engine studied in this article, it is obvious that the cooperative co evolutionary algorithm is more suitable for the needs. The simple steps of the pipeline steps under this algorithm are as follows:

Step 1: The population model, which is based on the pipeline layout problem, is segmented into multiple subpopulation models. Additionally, the algorithm's parameters, evaluation function, and termination conditions are defined;

Step 2: Algorithm iteration is performed to find the optimal solution for each subpopulation according to the constraint rules;

Step 3: The solution of each population is determined whether it is the optimal solution. Otherwise, steps 2 and 3 will be executed.

Step 4: During the iteration process, the optimal solutions of each subpopulation adapt to each other, influence each other, and evolve together. The final optimal solution of the overall problem is obtained, which is the set of optimal solutions of all subpopulations.

3.3.2. Co-evolutionary improved A* algorithm

Based on the analysis of the pipeline layout problem of aircraft engines, the mathematical model of pipeline layout can be obtained as follows:

$$\begin{cases} f(N) = \min\left[f_1(N_1) + f_2(N_2) + \cdots + f_n(N_n)\right] \\ s.t.\ \ N_1 \cap N_2 \cap \cdots \cap N_n \neq \varnothing \\ s.t.\ \ N_1 \,||\, N_2 \,||\, \cdots \,||\, N_n \ \ or\ N_1 \perp N_2 \perp \cdots \perp N_n \end{cases} \tag{14}$$

where $f(N)$ is the evaluation function of multi pipeline layout algorithm; N is the multi pipeline system; $N_i$is the i-th sub path of the multi pipeline system; $||$ and $\perp$ are parallel pipelines and branch pipelines, respectively.

The biggest difference between multi pipeline layout and single pipeline layout is that multi pipeline systems have multiple different pipeline connection points, which results in the complexity of multi pipeline systems. The solution to this problem is to divide the pipeline into multiple single pipelines, which are interconnected yet relatively independent. Then each pipeline is expressed separately. The schematic diagram of branch pipeline decomposition is shown in **Figure 18**.

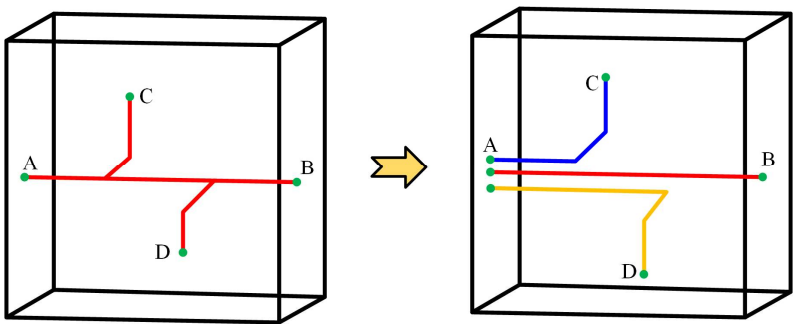


Figure 18. Schematic diagram of branch pipeline decomposition.

From **Figure 18**, it can be seen that the original branch pipeline has been decomposed into three single pipelines: AB, AC, and AD. The mathematical model of the branch pipelines based on Equation (12) is represented as follows:

$$N = \{N_1, N_2, \ldots, N_i\}, i = 1,2, \ldots, n \tag{15}$$

$$N_{Bend} = (N_{Bend}(N_1), N_{Bend}(N_2), \ldots, N_{Bend}(N_i)), i = 1,2, \ldots, n \tag{16}$$

where N_{Bend} is the number of bends in the entire branch pipeline.

As shown in **Figure 19**, if the starting and ending nodes and bending nodes of the branch pipeline are assigned node numbers, each connecting node in the pipeline can be represented as:

$$grid_k = (X_k, Y_k, Z_k), k = 1,2, \ldots, n \tag{17}$$

The mathematical expression of branch pipelines is as follows:

$$\begin{cases} N = \{N_1, N_2, N_3\} \\ N_1 = (grid_1^{start}, grid_4^{end}) \\ N_2 = (grid_1^{start}, grid_2, grid_5, grid_6^{end}) \\ N_3 = (grid_1^{start}, grid_3, grid_7, grid_8^{end}) \end{cases} \tag{18}$$

Additionally, the evaluation function for the problem of multi pipeline layout also needs to be redefined. The updated evaluation function is represented as:

$$\begin{aligned} f(N) &= g(N) + E \times h(N) \\ &= E_1 \sum_{i=1}^{n} L_{cost}(N_i) + E_2 \sum_{i=1}^{n} \omega N_{Bend}(N_i) + E \sum_{i=1}^{n} h(N_i) \end{aligned} \tag{19}$$

where $L_{cost}(N_i)$ is the length cost of the i-th single pipeline; $N_{Bend}(N_i)$ is the bending cost of the i-th single pipeline; ω is the cost of one bending; E, E_1, E_2 are the proportion of each evaluation factor in the evaluation function, respectively.

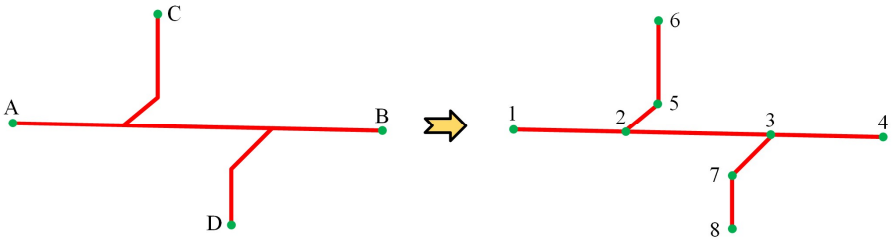


Figure 19. Branch pipeline node number.

3.3.3. Multi pipeline simulation analysis

In order to verify the effectiveness of the improved A* algorithm based on co-evolutionary algorithm, the path is further increased, as shown in **Figure 20**. And the partial paths in **Figure 20** have been combined, as shown in **Table 5**.

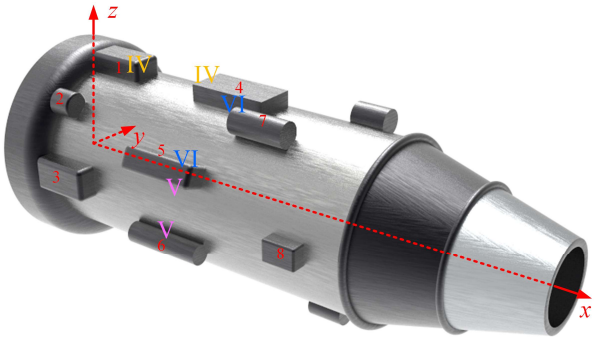
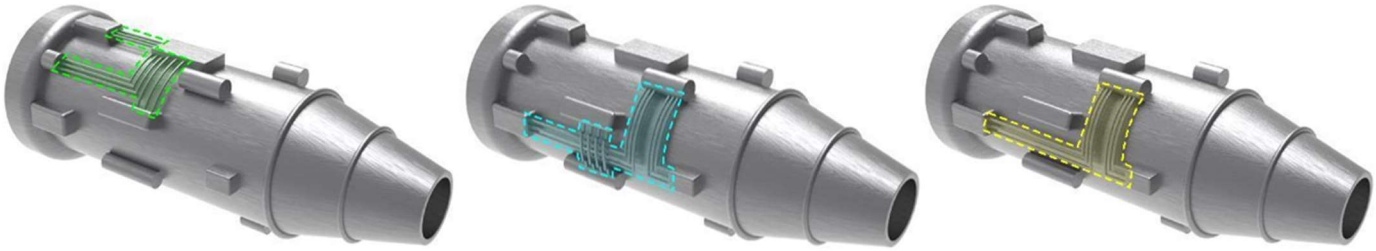


Figure 20. Path description 2.

Table 5. Relationship of combination.

Case	Path 1	Path 2	Path 3
A	I	IV	VI
B	II	III	V
C	II	III	

The simulation parameters are set to $\omega = 3$, $E_1 = 1$, $E_2 = 10$, $E = \gamma$. The simulation results are shown in **Figure 21**.



(a) **(b)** **(c)**

Figure 21. Simulation results. **(a)** Case A; **(b)** Case B; **(c)** Case C.

The specific results are shown in **Table 6**.

Table 6. Specific results.

Case	Total length of pipeline/m	Total elbow of pipeline	Total time/ms
A	17.6569	3	645.8941
B	26.8642	12	729.6457
C	15.5864	6	486.1251

From **Figure 21** and **Table 6**, it can be concluded that there is no overlapping arrangement of independent pipelines in the process of multi pipeline layout, and the overall layout is relatively beautiful the layout time is relatively short. This not only improves space utilization, but also reduces time cost. Therefore, A* algorithm can meet the basic requirements of aircraft engine pipeline layout.

The improved algorithm can greatly reduce the number of traversing nodes without affecting the optimal path, so that the number of nodes is reduced from the original 3067 to 1968, and the planning time is also shortened from the original 20.34 s to 7.26 s, which proves the effectiveness of the algorithm. (see **Table 7**):

Table 7. Data comparison of the path planning algorithm experiment.

Algorithm type	Average path length/mm	Number of traversed nodes/node	Average running time/s
Traditional algorithm	1406	3067	20.34
This study improves the algorithm	1360	1968	7.26

4. Conclusion

To address the current issues with single pipe and multi pipes layout in aero-engine, an automatic pipe layout method for aero-engine based on co-evolutionary algorithm and improved A* algorithm is proposed. The method has positive reference value for further improvement and implementation of automatic layout of aircraft engine piping systems. The main conclusions are as follows:

(1) The improved A*algorithm based on improved node coordinate expression, improved evaluation function, introduction of directional strategy, and improved OPEN_LIST is proposed. The algorithm is applied to the automatic layout of single pipe in aero-engine, and its simulation results are compared with the unimproved algorithm. Under different single pipeline paths, the improved A*algorithm has relatively shortened the length of pipes by 12.8275% and 19.4843%, respectively. According to different paths, the number of elbows also varies, but the number of elbows is much smaller than that of the unimproved algorithm. For time of pipes layout, the improved A* algorithm has relatively shortened the length of pipes by 58.1806%, 66.6554%, and 51.7261%, respectively. This lays the theoretical foundation for the single pipe layout in aero-engine.

(2) For automatic layout of multiple pipes, combining with co-evolutionary algorithm and new evaluation function, the new improved A* algorithm is proposed to solve the problem of automatic layout of multiple pipes in aero-engine. This algorithm is further optimized based on the former. Both space utilization and layout efficiency have been further improved. This provides support for obtaining various pipe layout schemes in aero-engine that better meet engineering conditions.

The method proposed in this paper can optimize the layout of multiple pipelines at the same time, and improve the efficiency and rationality of the overall layout. In the process of searching, it can converge to the optimal solution more quickly. This improved A* algorithm can effectively reduce the search space and improve the computational efficiency when dealing with pipeline layout problems. The method in this paper not only considers the layout efficiency of the pipeline, but also takes into account the length of the pipeline, bending degree, thermal expansion and other factors to achieve multi-objective optimization. This comprehensive consideration makes the pipeline layout more in line with the actual engineering needs.

However, in practical applications, especially when dealing with large-scale pipeline layout problems, the computational complexity is still high and the computation time is long. The method proposed in this paper is mainly aimed at the problem of pipeline layout in aero engines. For other types of pipeline layout problems, further research and improvement may be needed.

In the future, it is necessary to explore the combination of the proposed method with other optimization algorithms (such as genetic algorithm, particle swarm optimization algorithm, etc.) to further improve the performance of the algorithm.

Author contributions: Conceptualization, YL and GG; methodology, YL; software, XZ; validation, YL, GG and FC; formal analysis, FC; investigation, YZ; resources, ML; data curation, YL; writing—original draft preparation, YL; writing—review and editing, GG; visualization, XZ; supervision, ML and FC; project administration, YZ; funding acquisition, YL. All authors have read and agreed to the published version of the manuscript.

Ethical approval: Not applicable.

Data availability: The data used to support the findings of this study are available from the corresponding author upon request.

Conflict of interest: The authors declare no conflict of interest.

References

1. Zhang Y, Bai XL, Wu PR. Automatical optimization of aero-engine pipe-routing layout based on ABC algorithm. Journal of Northeastern University (Natural Science), 2013, 34(10):4. https://doi.org/10.3969/j.issn.1005-3026.2013.10.022 (2013).
2. Qu YF, Jiang D, Zhang XL.A new pipe routing approach for aero-engines by octree modeling and modified max-min ant system optimization algorithm. Journal of Mechanics, 2016:1-9. https://doi.org/10.1017/jmech.2016.86 (2016).
3. Jiao GS, Liu Q, Mao L, et al. Pipe routing for aero-engine using modified MOEA/D. 2017 10th International Symposium on Computational Intelligence and Design (ISCID), 2017.https://doi.org/10.1109/ISCID.2017.39 (2017).
4. Yuan HX, Yu JP, Jia D, et al. Group-based multiple pipe routing method for aero-engine focusing on parallel layout. Frontiers of mechanical engineering, 2021(16-4). https://doi.org/10.1007/s11465-021-0645-3 (2021).
5. Wu Y, Wang L, Zhuang X, et al. A cooperative coevolutionary algorithm with problem-specific knowledge for energy-efficient scheduling in serum system. Based Syst. 2023, 274:110663. https://doi.org/10.1016/j.knosys.2023.110663(2023).
6. Liu X, Sun W, Gao Y, et al. Optimization of pipeline system with multi-hoop supports for avoiding vibration, based on particle swarm algorithm. ARCHIVE Proceedings of the Institution of Mechanical Engineers Part C Journal of Mechanical Engineering Science 1989-1996 (vols 203-210), 2020. https://doi.org/10.1177/0954406220947115 (2020).
7. Liu Q, Tong B. Multi-pipe Routing in Bundles for Aero-engine using MOPSO. DEStech Transactions on Computer Science and Engineering, 2019.https://doi.org/10.12783/DTEEES/ICCIS2019/31770 (2019).
8. Wittmann J, Ochsenfarth F, Sonneville V, et al. Centralized vs. Decoupled Dual-arm Planning Taking into Account Path Quality. Journal of Intelligent & Robotic Systems, 2024, 110(4):1-19. https://doi.org/10.1007/s10846-024-02175-3(2024).
9. Park JH. Pipe-routing algorithm development for a ship engine room design. University of Washington, 2002.
10. Ren T, Zhu ZL, DimirovskiGM, et al. A new pipe routing method for aero-engines based on genetic algorithm. Proceedings of the Institution of Mechanical Engineers Part G Journal of Aerospace Engineering, 2014, 228(3):424-434.https://doi.org/10.1177/0954410012474134 (2014).
11. Kim SH, Ruy WS, Jang BS. The development of a practical pipe auto-routing system in a shipbuilding CAD environment using network optimization. International Journal of Naval Architecture and Ocean Engineering, 2013, 5(3):468-477. https://doi.org/10.2478/ijnaoe-2013-0146 (2013).
12. Liu Q, Wang C.A graph-based pipe routing algorithm in aero-engine rotational space. Journal of Intelligent Manufacturing, 2015, 26(6):1077-1083. https://doi.org/10.1007/s10845-013-0840-0 (2015).
13. Qu Y, Jiang D, Yang Q. Branch pipe routing based on 3D connection graph and concurrent ant colony optimization algorithm. Journal of Intelligent Manufacturing, 2016.https://doi.org/10.1007/s10845-016-1203-4 (2016).
14. Yao PX. Research on automatic layout of pipeline in coal preparation plant based on improved A*algorithm. Shandong University of Science and Technology, 2023.
15. Wu X, Zhang Q, Guo BG. A self-adaptive safe A* algorithm for AGV in large-scale storage environment. Intelligent Service Robotics, 2024, 17(2):221-235. https://doi.org/10.1007/s11370-023-00494-2 (2024).
16. Wu Q, Hao J K, Glover F. Multi-neighborhood tabu search for the maximum weight clique problem. Annals of Operations Research, 2012, 196(1):611-634. https://doi.org/10.1007/s10479-012-1124-3.1 (2012).

17. Wu J R, Wang Y G, Ding Z. An improved firefly algorithm for global continuous optimization problems. Expert Systems with Applications, 149. https://doi.org/10.1016/j.eswa.2020.113340 (2020).
18. Liu X, Zhao W, Wan D. Multi-fidelity Co-Kriging surrogate model for ship hull form optimization. Ocean engineering, 2022(243-Jan.1). https://doi.org/10.1016/j.oceaneng.2021.110239 (2021).
19. Ando N, Okuhara H, Kojima T, et al. Glitch-aware variable pipeline optimization for CGRAs. 2018. https://doi.org/10.1109/RECONFIG.2017.8279797 (2018).
20. Liu Q, Jiao G.A pipe routing method considering vibration for aero-engine using kriging model and NSGA-II.IEEE Access, 2018, PP(99):1-1. https://doi.org/10.1109/ACCESS.2018.2789361 (2018).
21. Yuan HX, Yu JP, Duo J, et al. Group-based multiple pipe routing method for aero-engine focusing on parallel layout. Frontiers of mechanical engineering, 2021, 16(4): 16. DOI:10.1007/s11465-021-0645-3(2021).
22. Ran DK, Peng FL, Li HG. A review of path planning research based on A*algorithm. Electronic Technology & Software Engineering, 2020.
23. Wang X, Lu JJ, et al. Research on AGV task path planning based on improved A*algorithm. Virtual Reality & Intelligent Hardware, 2023, 5(3):249-265.https://doi.org/10.1016/j.vrih.2022.11.002 (2023).
24. Gao ZZ, Wan LJ, Cai M, et al. Research on lazy theta* route planning algorithm based on grid point optimization. Applied Sciences, 2022, 12(20). https://doi.org/10.3390/app122010601 (2022).
25. Xiong YG, Li B, Yao T, et al. Global path planning of mobile robot with improved A*algorithm. Electronic measurement technology, 2024,47(05):31-36. https://doi.org/10.19651/j.cnki.emt.2415299 (2024).
26. Zhang YM, Wang J, Fu CX. Research on AGV path planning based on improved A*algorithm. Journal of Qingdao University (Engineering & Technology Edition), 2024:1-8. http://kns.cnki.net/kcms/detail/37.1268.TS.20240807.1518.002 (2024).
27. Xu Z, Liu X, Chen Q. Application of improved Astaralgorithm in global path planning of unmanned vehicles. 2019 Chinese Automation Congress (CAC), IEEE, 2019.https://doi.org/10.1109/CAC48633.2019.8996720 (2019).
28. Feng D, Deng L, Sun T, et al. Global Path Planning Based on an Improved A* Algorithm in ROS//The International Conference on Applied Nonlinear Dynamics, Vibration and Control. Springer, Singapore, 2022.DOI:10.1007/978-981-16-5912-6_84(2022).
29. Yuan HX. Research on automatic routing and optimization method for external pipeline of aero-engine. Northeastern University, 2021.
30. Cui FZ, Xu ZZ, Wang XK, et al. A dual-system cooperative co-evolutionary algorithm for satellite equipment layout optimization. Proceedings of the Institution of Mechanical Engineers Part G Journal of Aerospace Engineering, 2017. https://doi.org/10.1177/0954410017715280 (2017).
31. Liu W, Zhou Y, Li B, et al. Cooperative co-evolution with soft grouping for large scale global optimization. IEEE, 2019.https://doi.org/10.1109/CEC.2019.8790053 (2019).
32. Sun G, Fang X, Zhu L, et al. Path planning of plant protection UAV based on improved A* algorithm under wind conditions. 2021.https://doi.org/10.33440/j.ijpaa.20200304.125(2021).