IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received August 1, 2020, accepted August 14, 2020, date of publication August 20, 2020, date of current version September 1, 2020.

*Digital Object Identifier 10.1109/ACCESS.2020.3018145*

# Ship Pipe Route Design Using Improved A* Algorithm and Genetic Algorithm

**ZONGRAN DONG[1,2] AND XUANYI BIAN[3]**

[1]School of Software, Dalian University of Foreign Languages, Dalian 116044, China
[2]Research Center for Language Intelligence, Dalian University of Foreign Languages, Dalian 116044, China
[3]School of Naval Architecture and Ocean Engineering, Dalian University of Technology, Dalian 116024, China

Corresponding author: Zongran Dong (dongzongran@163.com)

This work was supported in part by the Doctoral Scientific Research Foundation of Liaoning Province, China, under Grant 2019-BS-061, and in part by the Basic Research Foundation of Education Department of Liaoning Province, China, under Grant 2019-JYT-07.

**ABSTRACT** The goal of ship pipe route design (SPRD) is to seek the near-optimal paths that meet various constraints and objectives. Due to the complex construction of routing space, diverse piping constraints, and the large number of pipes, SPRD is one of the most difficult and time-consuming tasks even to a skilled pipe designer. This article proposes automatic approaches for solving SPRD with A* algorithm and genetic algorithm (GA). Firstly, by simplifying the equipment and decomposing the routing space into grids, the mathematical model of SPRD is created. Then, the improved A* algorithm (A*-Router) for single pipe routing is introduced. The evaluation function, auxiliary tables and algorithm framework of A*-Router are presented. To obtain high-quality and diverse layouts, the improved GA (A*-GA-Router) is formulated by A*-Router and the connection-points strategy. Several new genetic operators of A*-GA-Router are designed to improve the routing performance. For multiple pipes routing, the novel algorithm (Multi-Pipes-Router) which calls A*-GA-Router internally is put forward. It arranges pipes according to the specified routing sequence and can produce parallel layout under the function of GA optimization and connection-points strategy. To cope with branch-pipe routing widely existing in engineering, a new pipe router (Branch-Pipe-Router) is put forward using a modified Steiner Tree framework in combination with the proposed single pipe routing algorithms. Compared with the traditional methods based on coevolution, it is more versatile and can effectively balance the layout quality and time efficiency. Finally, the feasibility and effectiveness of the proposed algorithms are demonstrated by the experiments on the designed and actual cases.

**INDEX TERMS** A* algorithm, genetic algorithm (GA), single pipe routing, multiple pipes routing, branch pipe routing, ship pipe route design (SPRD).

## I. INTRODUCTION

Pipe route design (PRD) is a significant subject in industrial fields, such as aero-engine, large-scale integrated circuit, mechanical product, ship engineering, *et al.* ISO standard divides ship pipe design into five successive phases, including primary design, functional design, detailed design, production engineering, and system-support information [1]. Ship pipe route design (SPRD) plays a prominent role in the detailed design phase, and many other successor activities depend on it [2]. The SPRD problem is essentially a kind of shortest path-finding problem under piping constraints. The main goal is to develop the collision-free and optimal routes of ship pipes, which connect the pipe nozzles in routing space scattered with obstacles. Although CAD systems are widely used in modern ship design, in general, only interactive-operation environments are provided in the piping modules. Due to the complexity of piping system and the diversity of routing constraints, SPRD still highly depends on manual work in engineering. Therefore, it is significant to investigate novel methods for automatic pipe routing.

Shortest path-finding problem has been carried out by researchers for several decades. It is the key technology of PRD. At the early stage, many deterministic algorithms such as Dijkstra [3], A* [4], maze algorithm [5], and escape algorithm [6] are proposed and improved [7]–[9], but these methods tend to provide fixed solutions. Recently, PRD study has been prompted by the development of modern intelligent algorithms such as genetic algorithm (GA) [10], [11], ant colony optimization (ACO) [12], and particle swarm

The associate editor coordinating the review of this manuscript and approving it for publication was Heng Wang.

optimization (PSO) [13]. However, the optimization ability and efficiency of these methods are not ideal enough when dealing with large-scale or complex problems. Many improved studies have been conducted [14]–[19], which lead to the complexity of algorithm design process.

As the extension of PRD in ship field, SPRD has its particularity. Lots of new or improved methods have been put forward. Kang *et al.* [20] and Shao *et al.* [21] adopted expert systems to route pipe path automatically. Park and Storch [2] proposed the cell-generation approach to optimize pipe routes in ship engine room. Asmara [22] developed a pipe routing framework for detailed ship pipe design. Kim *et al.* [23] developed a pipe routing system in CAD environment using network optimization. Moreover, some researchers focus on solving SPRD with swarm intelligent algorithms. Fan *et al.* [24]–[26] and Jiang *et al.* [27], [28] proposed several ship piping methods based on ACO, GA and the combination of both. Dong and Lin [29], [30] used co-evolutionary PSO and GA with fixed-length encoding to route ship pipes. Sui *et al.* [31], Niu *et al.* [32], and Sui and Niu [33] used maze algorithm to create high-quality chromosomes for GA, whereas Wang *et al.* [34], [35] generated artificial chromosomes with the mechanism of human-computer cooperation to improve GA and ACO.

In practice, more than 70% of pipes have a branch [22], but the branch pipe routing has not drawn much attention like the traditional two-terminal routing problem until the twenty-first century. Park and Storch [2] considered branch pipe as the compound of two basic forms, i.e., end-forked and middle-forked, in their cell-generation methodology; Fan *et al.* [36] tried to route branch pipes based on the maze algorithm and Steiner Tree theory, the principle of which is to connect the pipeline terminals sequentially. In addition, Asmara and Nienhuis [37] used discrete PSO to determine the connecting order, and then applied Dijkstra's algorithm to connect the terminals sequentially. By dividing the branch pipe into several single pipes, Wu *et al.* [38] and Jiang *et al.* [27] employed an optimization framework on the basis of co-evolutionary algorithms and multi-ACO to route branch pipes in ship, which can ignore the locations of branch points and the connecting order of branches. Liu and Wang [19] presented a novel branch pipe routing algorithm for planning multi-terminal pipe routes with Steiner Minimal Tree theory and PSO technique in conjunction. Sui and Niu [33] proposed a GA-based approach to route branch pipe in ship, where branch pipes are also regarded as a combination of several two-terminal pipelines, the main techniques in their method are the improved maze algorithm for routing single branch and the devised genetic operators for GA.

These mentioned methods can be categorized into three classes: deterministic routing algorithms, non-deterministic routing algorithms, and hybrid routing algorithms. In general, deterministic algorithms have better time efficiency; non-deterministic algorithms have better optimization ability; and hybrid algorithms can balance the advantages of different methods, while with more computation time. Due to the complexity of SPRD, it is necessary to develop various algorithms with different characteristics and abilities.

Asmara [22] reviewed several path-finding algorithms, and pointed out that A* algorithm is the most suitable one for his router module. Our previous work [39] also use A* algorithm for pipe routing. However, the early studies considered little on time efficiency and routing constraints. In addition, A* algorithm only provides fixed solution and cannot find the global optimum sometimes. Therefore, this article proposes new piping methods based on improved A* algorithm and GA.

The rest of this article is set out as follows. Section 2 describes the SPRD problem in more detail. Section 3 presents the proposed ship pipe routing algorithms. Section 4 provides the simulation experiments and discusses the results. Finally, Section 5 concludes the paper.

## II. PROBLEM FORMULATION

### A. REPRESENTATION OF ROUTING SPACE

The routing space is represented by a cuboid, and the ship structure, internal equipment, and prohibited piping areas are represented by cuboids or their combinations. According to the diameter values of the pipelines to be routed and the minimum space between pipes, the routing space is decomposed into grids. The grids covered or partially covered by the obstacles, such as equipment, structures, and forbidden areas, are marked as obstacle-grids, whereas the other grids are marked as free-grids. It is assumed that $S_{grids}$ denotes the set of grids generated by space decomposition, and $S_{obs}$ denotes the set of cuboids representing the space of obstacles. Since the size of grid is much smaller than that of cuboids in $S_{obs}$, the process for judging obstacle-grid or free-grid can be simplified as follows: traverse each grid in $S_{grids}$, for the selected grid $c$, check whether any vertex of $c$ is located inside the space of $S_{obs}$ by comparing their coordinates. If it exists, mark $c$ as an obstacle-grid, otherwise mark $c$ as a free-grid. After traversing all the grids in $S_{grids}$, the process ends. The energy value is set according to the location of the grid. The grids near walls, floors, and equipment surface are set with low energy values, whereas the grids covered by areas around boiler, above electrical equipment or inside operation space are set with high energy values to reduce the probability of pipe passing. There are obvious advantages to solve SPRD by adopting such a gridding model. Firstly, many mature shortest path-finding algorithms are ready for use. Secondly, the major constraints are suitable to be represented and processed in the weighted gridding environment. Fig. 1 shows an example of the routing space model.

When routing pipes with different diameter values, the layout space will be decomposed into grids based on the minimum pipe diameter which is equal to the side length of a grid. When routing a pipe $p$ with a larger diameter, the obstacles in the space should be extended outward by a certain distance in

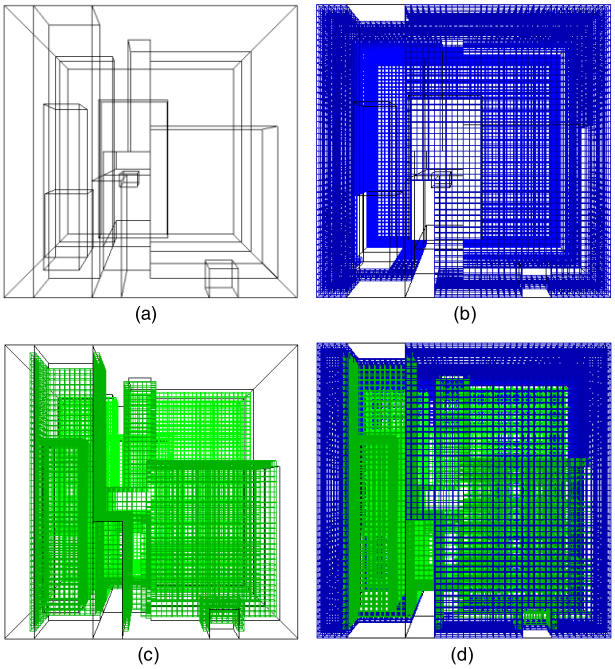



**FIGURE 1.** Routing space model. (a) Routing space. (b) Grids along walls and floors. (c) Grids along equipment. (d) Grids in (b) and (c).

advance, which can be calculated by formula (1) [32].

$$N_p = \begin{cases} \mathbf{int}\left(\dfrac{D_p}{2L}\right), & \dfrac{D_p}{2L} - \mathbf{int}\left(\dfrac{D_p}{2L}\right) \leq \mathbf{0.5} \\ \mathbf{int}\left(\dfrac{D_p}{2L}\right) + \mathbf{1}, & \dfrac{D_p}{2L} - \mathbf{int}\left(\dfrac{D_p}{2L}\right) > \mathbf{0.5} \end{cases} \tag{1}$$

where $N_p$ denotes the grid number that needs to be extended by obstacles, $D_p$ represents the largest diameter value of pipe $p$, and $L$ represents the side length of a grid. Note that the diameter value here includes the minimum allowable distance between pipelines.

The obstacle extension process is described as follows: Take the extension process of pipe $p$ as an example. As mentioned earlier, denote the grids generated by space decomposition as $S_{grids}$. In the pretreatment, all the grids located inside or at the boundaries of the obstacles have been marked as obstacle-grids. Based on this, the extension process only needs to traverse $S_{grids}$ once. If the currently traversed grid $c$ is an obstacle-grid, skip it, otherwise, check whether there is any obstacle-grid within $N_p$ distance from grid $c$. If any, mark $c$ as a temporary-obstacle-grid (a special kind of obstacle-grid that does not allow pipes to pass and will be restored to free-grid after routing), if not, $c$ is still a free-grid. The extension process ends after traversing all the grids in $S_{grids}$.

Note that the initial pipe nozzles may be covered by obstacle extension, it is necessary to extend the initial location of each pipe nozzle outward along the original direction to form a new connection point. Moreover, the obstacle extension should be performed prior to each pipe routing, and the routed pipes need to be put into the set of obstacles. Further description will be addressed in the case study section.

When searching a path in such gridding space, the pipe $p$ will not interfere with any obstacles even after extending it back to the actual size. This method decomposes the layout space only once for pipes of all sizes, and also improves the layout accuracy due to using small-size grid to represent obstacles and pipe paths.

### B. REPRESENTATION OF PIPE

Pipeline is a medium conveying channel that connects the starting and target nozzles without interference with the routing environment. To route pipe with algorithm, a pipe is represented by the outer envelope grids along its path. The pipe path can be represented by the sequence of center point coordinates of each grid, which is defined as follows:

$$p_1\,(x_1, y_1, z_1) \rightarrow p_2\,(x_2, y_2, z_2) \rightarrow \ldots \rightarrow p_n\,(x_n, y_n, z_n) \tag{2}$$

where $p_1$ is the starting grid, and $p_n$ is the target grid. In grid decomposition space, the grid-coordinate sequence can be converted to grid-position sequence as follows:

$$\begin{aligned} &p_1\,(r_1, c_1, l_1) \rightarrow p_2\,(r_2, c_2, l_2) \rightarrow \ldots \rightarrow p_n\,(r_n, c_n, l_n) \\ &\mathbf{s.t.}\ |r_i - r_{i-1}| + |c_i - c_{i-1}| + |l_i - l_{i-1}| = 1, \\ &\quad (1 < i \leq n) \end{aligned} \tag{3}$$

The final shape of the pipe can be obtained by expanding the path outward with the radius value of the pipe.

Parallel piping means the paths of two or more pipes are arranged as close as possible to form a bundle effect, so as to meet the requirements of aesthetics, space saving, brackets sharing, and easy for maintenance. Obviously, parallel piping commonly exists in SPRD.

Suppose $p_i$ and $p_j$ $(i \neq j)$ are two non-interference pipes, the grids sequences of $p_i$ and $p_j$ are defined as follows:

$$\begin{aligned} &p_1^i\left(r_1^i, c_1^i, l_1^i\right) \rightarrow p_2^i\left(r_2^i, c_2^i, l_2^i\right) \rightarrow \ldots \rightarrow p_n^i\left(r_n^i, c_n^i, l_n^i\right) \\ &p_1^j\left(r_1^j, c_1^j, l_1^j\right) \rightarrow p_2^j\left(r_2^j, c_2^j, l_2^j\right) \rightarrow \ldots \rightarrow p_m^j\left(r_m^j, c_m^j, l_m^j\right) \end{aligned} \tag{4}$$

Select two grids $p_k^i$ and $p_t^j$ from $p^i$ and $p^j$, respectively, then $p_k^i \neq p_t^j$, $(1 \leq k \leq n, 1 \leq t \leq m)$. If $p_k^i$ is adjacent to $p_t^j$, the Manhattan distance of the two grids must be 1, i.e., $\left|r_k^i - r_t^j\right| + \left|c_k^i - c_t^j\right| + \left|l_k^i - l_t^j\right| = 1$. Thus, the parallel piping effect of the two pipes can be measured by calculating the number of adjacent grid-pairs in $p_i$ and $p_j$.

If there are $N$ pipes $p_1, p_2, p_3, \ldots, p_N$ $(N \geq 2)$ to be routed in parallel, we have two options to arrange them.

1) Fixed-selection method. The current routing pipe should be arranged next to the specified routed pipe.
2) Free-selection method. The current routing pipe can be arranged next to any of the routed pipes.

Our approach will adopt the first option since it is more reasonable in practical piping.

### C. CONSTRAINTS AND OBJECTIVES

The constraints and objectives are very complicated in practical SPRD problem. To verify the effectiveness of the proposed algorithms, some major constraints and objectives are selected and considered.

#### 1) CONSTRAINTS

1) Connect the pipe nozzles and avoid interference with any obstacles.
2) Arrange pipes and rigid structures orthogonally.
3) Satisfy the minimum length requirement between two adjacent bends.

#### 2) OBJECTIVES

4) Minimize the path length as much as possible.
5) Minimize the number of bends as much as possible.
6) Route pipes along the surface of walls, floors or equipment for better support.
7) Arrange pipes in parallel if possible, for sharing the same series of racks.

### D. MATHEMATICAL MODELING

The constraints (1-3) of the SPRD problem are guaranteed by the grid decomposition model and A* routing algorithm, and the optimization objectives (4-7) are gradually improved with the evolution of genetic algorithm. The objective function of SPRD is defined as follows:

$$\mathbf{Min}.f_{Objective}\,(path) = \alpha \times f_L\,(path) + \beta \times f_B\,(path) + \gamma \times f_E\,(path) - \delta \times f_N(path) \quad (5)$$

where $f_L\,(path)$ denotes the number of grids on the path, corresponding to objective 4; $f_B\,(path)$ denotes the number of bends on the path, corresponding to objective 5; $f_E\,(path)$ denotes the sum of grid-energy values on the path, corresponding to objective 6. The first three sub-objectives are the smaller the better. $f_N(path)$ measures the parallel routing effect of the current pipe and the routed pipe, corresponding to objective 7. The larger this value is, the more the adjacent grid-pairs exist. $\alpha$, $\beta$, $\gamma$, and $\delta$ are all positive weight factors set according to user experience. $\delta$ should be selected to ensure that the value of the objective function is always positive. $\alpha$, $\beta$, and $\gamma$ are used to balance the relative importance among the first three sub-objectives. In this approach, the pipe path is created by the improved A* algorithm and will not collide with any obstacles. Therefore, it is no need to introduce penalty function like [10], [29].

## III. PROPOSED PIPE ROUTING METHODS

### A. OVERVIEW OF THE PROPOSED METHODS

This work tries to combine the optimization ability of A* algorithm and genetic algorithm to improve the efficiency and quality of pipe routing. Firstly, the ship pipe routing algorithm (A*-Router) based on the improved A* algorithm is presented. Besides avoiding interference with the obstacles, the other objectives and constraints such as path length, number of bends, minimum length of pipe segments, as well as routing along supports can be considered comprehensively in the optimization. Although A* algorithm can be used to find a fixed optimized solution, it is not equal to find the actual optimal solution in engineering. In addition, designers want to find more feasible reference solutions. Based on the improved A* algorithm and the connection-points strategy, a new genetic algorithm (A*-GA-Router) is proposed, which can guarantee the validity and quality of the evolutionary solutions. Compared with the improved A* algorithm, the new genetic algorithm can provide different optimal or sub-optimal solutions in different executions. In practical pipe engineering, multiple pipes and branch pipes are very common and important. Many studies for routing multiple pipes and branch pipes are based on evolutionary algorithms [19], [26], [27], [30], [32], [33], [38]. However, the layout quality, successful routing rate or time efficiency of the previous studies can be improved in some extent. Therefore, the method of multiple pipes routing (Multi-Pipes-Router) is developed based on A*-GA-Router; and to cope with branch-pipe routing, the modified Steiner Tree framework combined with LEE-Router, A*-Router or GA-A*-Router (Branch-Pipe-Router) is put forward.

### B. IMPROVED A* ALGORITHM FOR SINGLE PIPE ROUTING (A*-ROUTER)

A* algorithm was developed by Hart, Nilsson, and Raphael in 1968 [4], and it was improved to use in many applications, such as, robot path planning in 2D/3D space [40]–[42], pipe design in aeroengine [43] and electromechanical products [44]. It is not simply depth-first or breadth-first search, but optimal-first search. The algorithm introduces a heuristic function in its cost function. In each iteration, A* algorithm selects a node $n$ from the neighbor nodes of the current node to expand, and $n$ has the minimum cost value among these neighbors. The cost function is defined as $f(n) = g(n)+h(n)$, where $g(n)$ is the part to calculate actual cost from the start node to node $n$, and $h(n)$ is the part to calculate estimated cost from node $n$ to the target node. During the search, A* algorithm updates $g(n)$ and $h(n)$ continuously, and obtains heuristic information from $h(n)$. Therefore, the selection of each path node is based on the cost of the formed path and the estimated cost of the remaining path. A* algorithm is an improvement of Dijkstra algorithm. If $h(n)$ is set to 0, A* algorithm degenerates into Dijkstra algorithm. The time complexity of A* algorithm mainly depends on its Open table. Fibonacci heap [45] provides the best worst-case time as $O(|E|+|V|\log|V|)$, where $|E|$ and $|V|$ denote the number of edge and node, respectively. The time complexities of A* and Dijkstra are the same, but due to the role of $h(n)$, A* algorithm usually explores fewer nodes than Dijkstra algorithm. Several methods for solving the SPRD problem by improved A* algorithms were developed and compared in our previous work [46]. To construct a hybrid algorithm, an improved A* algorithm is given as follows.

#### 1) DESIGN OF THE EVALUATION FUNCTION

##### a: ACTUAL COST FUNCTION g (n)

The actual cost function of the formed pipe path from starting point to $n$ is defined as follows:

$$g(n) = a \times L_{cost}(n) + b \times E_{cost}(n) + c \times B_{cost}(n) \quad (6)$$

where $a$, $b$, $c$ are the weight factors. $L_{cost}(n)$, $E_{cost}(n)$, and $B_{cost}(n)$ are the length cost, energy cost, and bending number cost, respectively. Each time a new grid is explored in the search, it needs to add the length of the grid to $L_{cost}(n)$, add the energy of the grid to $E_{cost}(n)$, and add 1 to $B_{cost}(n)$ when the grid forms a new bend with its previous neighbors.

##### b: HEURISTIC FUNCTION h(n)

The heuristic function will be called repeatedly to compute the estimation cost of the exploring grids. Heuristic function not only affects the search scope, but also affects the computational complexity. In grid decomposition space, Manhattan distance is naturally adopted as the heuristic function to achieve better results. Assuming that $(x_n, y_n, z_n)$ denotes the location of grid $n$, $(x_t, y_t, z_t)$ denotes the location of target grid $t$, then the Manhattan distance between $n$ and $t$ is defined as follows:

$$h(n) = \text{abs}(x_n - x_t) + \text{abs}(y_n - y_t) + \text{abs}(z_n - z_t) \quad (7)$$

where abs() is a function to compute absolute value. Pipes need to be arranged orthogonally in SPRD problem, so Manhattan distance is the shortest distance between the two grids. However, due to the existence of obstacles in routing space, the actual distance is usually longer than Manhattan distance. To improve the computing efficiency, Manhattan distance can be properly magnified before using as the heuristic function.

##### c: EVALUATION FUNCTION f (n)

Based on the definitions of $g(n)$ and $h(n)$, the evaluation function $f(n)$ is defined as follows:

$$\begin{aligned} f(n) &= g(n) + w \times h(n) \\ &= a \times L_{cost}(n) + b \times E_{cost}(n) + c \times B_{cost}(n) \\ &\quad + w \times (\text{abs}(x_n - x_t) + \text{abs}(y_n - y_t) + \text{abs}(z_n - z_t)), \\ w &\geq 1 \end{aligned} \quad (8)$$

If b and c are 0, the search is only guided by the path length; if b and c are positive values, the constraints of bending number and routing along supports are taken into account.

#### 2) DESIGN OF THE AUXILIARY TABLES

Beside heuristic function, the computation of A * algorithm mainly relies on the Open table. A proper Open table with low average time complexity needs to be selected first. From the comparison in our previous work [46], we observed that binary heap is a better choice. In this method, priority queue as the implementation of binary heap is employed as the type of Open table ($pq$). It allocates contiguous area of memory to represent a tree structure. The top item of the queue has the smallest cost, so the complexity of getting the smallest element is $O(1)$. Inserting or removing elements in priority queue will cause heap adjustment, and its time complexity is $O(\log N)$. Updating element by key is not supported in priority queue; it has to traverse the queue to find the element before updating. The complexity of updating operation in priority queue is $O(N \log N)$, which is time-expensive and will be improved in the A*-Router.

The Close table of A* algorithm is used to judge whether a grid in routing space has been explored or not. Therefore, the auxiliary table $close\_map[k][j][i]$ is introduced to keep the status of the grid indexed by $(k, j, i)$. The initial value of each element in $close\_map$ is 0, and will be set to 1 after exploration. In addition, some high frequency operations, such as searching grid and getting grid cost, are implemented in another auxiliary table $open\_map[k][j][i]$ to improve the efficiency. The elements in $open\_map$ record the latest cost of grids. The last auxiliary table $dir\_map[k][j][i]$ keeps the sequential relationships for the grids on path. $dir\_map[k][j][i]$ stores the previous neighbor of grid $(k, j, i)$. The grid path of pipe can be tracked using $dir\_map$ and the pipe nozzles.

#### 3) FRAMEWORK OF THE A*-ROUTER

The high-level flowchart of A*-Router is shown in Fig. 2.

Subsequently, in order to explain the implementation and improvement methods of A*-Router, the detailed steps are described as follows.

In *Step 16*, locating and updating $(f_{\text{old}}, v)$ in $pq$ requires a large amount of queue operations, which causes continuous adjustment on the internal heap and leads to bad performance, so we proposed an improvement: In *Step 15*, for grid $v$ in Open table, if the result of current exploration is better, then put $(f(v), v)$ to the top of $pq$ directly instead of updating $(f_{\text{old}}, v)$ with $(f(v), v)$, (i.e., do the same operations as *Step 14*). The improvement can reduce high-cost queue operations, whereas the size of $pq$ will grow. But this side effect is trivial since the subsequent search only explores the grids that are not in the Close table and have the smallest cost.

### C. IMPROVED GENETIC ALGORITHM FOR SINGLE PIPE ROUTING (GA-A*-ROUTER)

Genetic algorithm (GA), as a famous non-deterministic algorithm, has excellent optimization capability. To solve the SPRD problem, we propose an improved GA combined with the A*-Router to generate chromosomes and design genetic operators. The aim of this GA is not only to ensure the quality of the solutions, but also to produce more feasible solutions for reference. The key steps of GA-A*-Router are described as follows.

#### 1) CHROMOSOME GENERATION

To create chromosomes based on A*-Router, the connection-points strategy is introduced:

1) Set the number of intermediate connection points $N$ excluding the starting point and the target point. $N$ depends on the requirement of the pipe and the characteristics of the routing space. When routing a single pipe, the role of

**Algorithm :** A*-Router

**Input:** the starting grid $s$, the target grid $e$, and $Sp[M][N][L]$ which denotes the grid-decomposition routing space. $M$, $N$, and $L$ are the layer number, column number, and row number of the space, respectively.

**Output:** grid path of the pipe.

***Step 1:***

Create an empty priority queue $pq$;
For each *grid* $c \in Sp[M][N][L]$, let
$open\_map[c.layer][c.column][c.row] \leftarrow 0$;
$close\_map[c.layer][c.column][c.row] \leftarrow 0$;
$dir\_map[c.layer][c.column][c.row] \leftarrow$ -1.

***Step 2:***

Calculate the evaluation function value $f(s)$ of the starting grid $s$;
Push the pair item $(f(s), s)$ into $pq$;
Save the actual cost of $s$:
$open\_map[s.layer][s.column][s.row] \leftarrow g(s)$.

***Step 3:***

If $pq$ is not empty, perform *Step 4* to *Step 17* in loops.

***Step 4:***

Assign the second part of the top element in $pq$ to $v$;
Pop up the top element from $pq$.

***Step 5:***

If $v$ is the target grid $e$, stop searching and build the path of current pipe according to $dir\_map[M][N][L]$, $s$, and $e$;
If $v$ is not the target grid $e$, perform the following steps.

***Step 6:***

Set the status of $v$ to "Explored", i.e.,
$open\_map[v.layer][v.column][v.row] \leftarrow 0$;
$close\_map[v.layer][v.column][v.row\ ] \leftarrow 1$.

***Step 7:***

Backup $v$ to $v0$: $v0 \leftarrow v$.

***Step 8:***

Explore the neighbor grids of $v$ in six orthogonal directions: $Sp[v.layer+k][v.column+j][v.row+i]$ denotes a neighbor grid, where $k$, $j$, and $i$ are integers between $[-1, 1]$; if one of them is $-1$ or 1, the other two must be 0; select one neighbor grid at a time, and then perform *Step 9* to *Step 17*.

***Step 9:***

If the conditions (a), (b), and (c) are met at the same time, perform *Step 10* to *Step 16*, otherwise perform *Step 17*.
(a) $(v.layer+k, v.column+j, v.row+i)$ is a valid grid coordinate in the routing space;
(b) $Sp[v.layer+k][v.column+j][v.row+i]$ is a free-grid, which is not in obstacles or other routed pipes;
(c)The value of $close\_map[v.layer+k][v.column+j][v.row+i]$ is 0, which means grid $v$ has not been explored.

***Step 10:***

Assign $v$ to a new location:
$v(layer, column, row) \leftarrow v(layer+k, column+j, row+i)$.

***Step 11:***

Judge whether *Step 10* forms a bend, if a new bend is formed, calculate the distance $d$ from new bend to the last bend;
If no bend has been formed or $d$ is not less than the minimum bending length $L_{min}$, then perform *Step 12*, otherwise go to *Step 17*.

***Step 12:***

Try to calculate the actual cost $g(v)$ of grid $v$, and assign it to $dist$: $dist \leftarrow g(v)$.

***Step 13:***

If $open\_map[v.layer][v.column][v.row]$ is 0, which means grid $v$ has not been explored, then perform *Step 14*, otherwise perform *Step 15*.

***Step 14:***

Compute $g(v)$ and $h(v)$ of grid $v$, update $f(v)$:
$f(v) \leftarrow g(v) + w \times h(v)$;
Update the cost of $v$:
$open\_map[v.layer][v.column][v.row] \leftarrow g(v)$;
Save the id of the previous grid of $v$ into $dir\_map$:
$dir\_map[v.layer][v.column][v.row] \leftarrow$
$Sp[v.layer\text{-}k][v.column\text{-}j][v.row\text{-}i].id$;
Push $(f(v), v)$ into $pq$.

***Step 15:***

If $open\_map[v.layer][v.column][v.row]$ is larger than $dist$, which means grid $v$ has been explored, but the new path is better, then perform *Step 16*, otherwise perform *Step 17*.

***Step 16:***

Compute $g(v)$ and $h(v)$ of grid $v$, update $f(v)$:
$f(v) \leftarrow g(v) + w \times h(v)$;
Update the cost of $v$:
$open\_map[v.layer][v.column][v.row] \leftarrow g(v)$;
Save the id of the previous grid of $v$ into $dir\_map$:
$dir\_map[v.layer][v.column][v.row] \leftarrow$
$Sp[v.layer\text{-}k][v.column\text{-}j][v.row\text{-}i].id$;
Find $(f_{\text{old}}, v)$ in $pq$ by $v$, and then update $(f_{\text{old}}, v)$ with $(f(v), v)$.

***Step 17:***

Restore $v0$ to $v$: $v \leftarrow v0$;
Go to *Step 8* to explore the next neighbor grid.

***Step 18:***

The path from $s$ to $e$ has not been found, return some failure information.

intermediate connection points is to improve the diversity and quality of the initial paths. The value of $N$ can be slightly increased when the pipe passes through a complex routing space. In general, the range from 1 to 5 is sufficient. When routing multiple pipes in parallel, the intermediate connection points can improve the parallel effect between pipes. If the length of parallel path will be long, $N$ can take a slightly larger value, and the recommended range of $N$ is from 2 to 7. However, the randomness of routing path and the number of pipe segments will increase along with the value of $N$, consequently, $N$ should not take big values.

2) Generate the intermediate connection points at random between the starting point and target point inside the specified

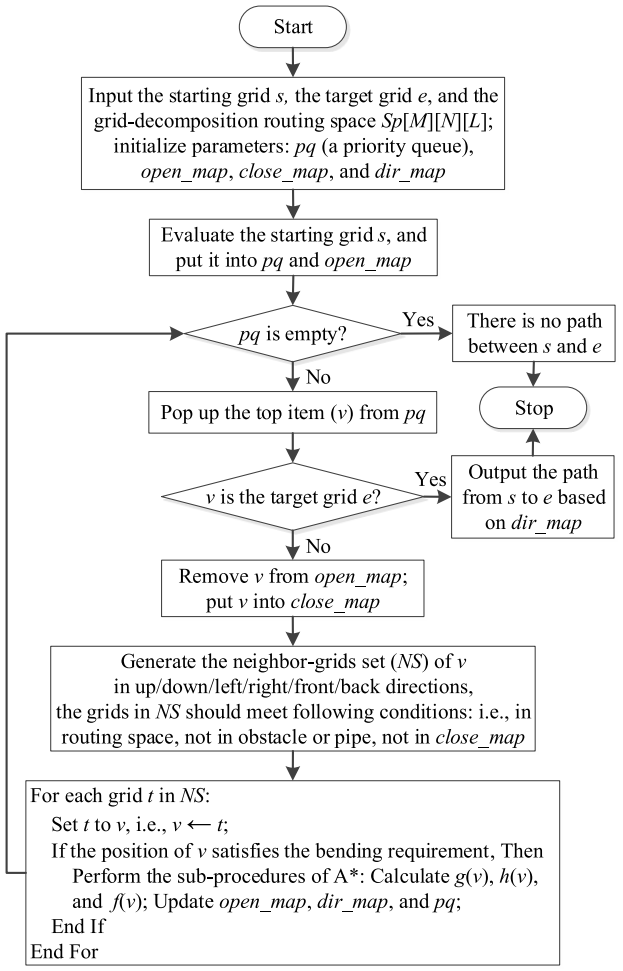



**FIGURE 2.** The flowchart of A*-router.

space. Note that, the connection points cannot locate in any obstacles to ensure the validation of the pipe path, and the specified area for generating connection points is usually the cuboid with the starting point and target point as the diagonal. Of course, the size, location, and valid area of this cuboid space can be adjusted according to the user's requirements. It should be mentioned that, if there are few obstacles in the cuboid space or the space is too large, the distribution of connection points will be relatively scattered. It is not conducive to generating high-quality chromosomes within short time. To deal with such situation in practice, the space for generating connection points is preferred to be narrowed further. For instance, restrict the connection points to be generated near the sides or bottom of the cuboid. The layouts obtained under this strategy may not be optimal, but the solutions can be obtained in less time, which can provide more feasible references for the designer.

3) Sort the intermediate connection points by a specified direction of their coordinates.

4) Create the sub-paths between each adjacent connection points using A*-Router. Consequently, the whole routing path can be formed.

In this procedure, the sorting operation on connection points can effectively improve the pipe-path quality. See illustration in Fig. 3. The sorting direction is usually the same as the coordinate direction in which the starting point and the target point has the largest difference.

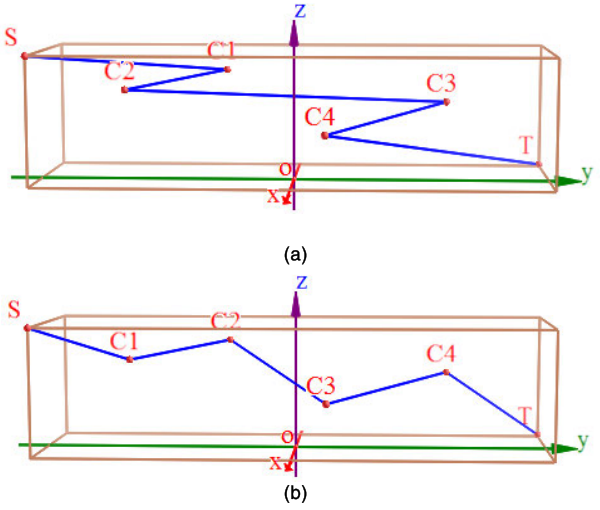



**FIGURE 3.** A schematic diagram of connection points sorting. (a) Path formed with the initial connection points. (b) Path formed with the connection points sorted in y coordinate.

### 2) SELECTION OPERATOR AND FITNESS FUNCTION

Selection operator selects the offspring chromosomes from parent chromosomes according to the specified strategy and the fitness value of each chromosome. It embodies the theory of fittest survival in GA. In A*-GA Router, roulette method is used as the selection strategy, and it is suitable for solving the maximum optimization problem. Therefore, the objective function (5) is converted to formula (9).

$$\textbf{Max}.\, f_{\mathrm{GA-Objective}}\,(path) = \mathrm{K} - f_{Objective}\,(path) \qquad (9)$$

where K is a large constant so that the difference value between K and $f_{\mathrm{Objective}}\,(path)$ is positive.

In order to improve the convergence of this GA, elitism is adopted in the selection operator to avoid losing the global optimal solution.

### 3) CROSSOVER OPERATOR

Crossover operator is the main evolution power of GA. It generates offspring chromosomes from the genes of parent chromosomes. Fig. 4 shows an example of the crossover operator. The crossover points are selected from the parent chromosomes (P1 and P2) at random, and then A*-Router algorithm is used to generate a sub-path connecting the two crossover points. Finally, the sub-path and the path segments of the parent chromosomes are connected appropriately to form the offspring chromosomes.

The chromosomes of this method use variable-length encoding, and a chromosome can be mapped to pipe path

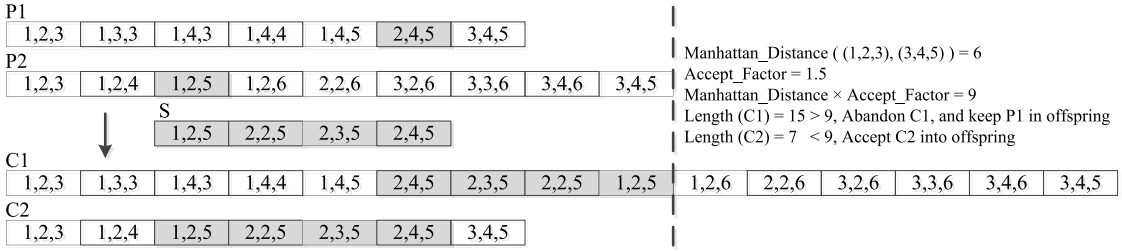



**FIGURE 4.** An example of crossover operator.

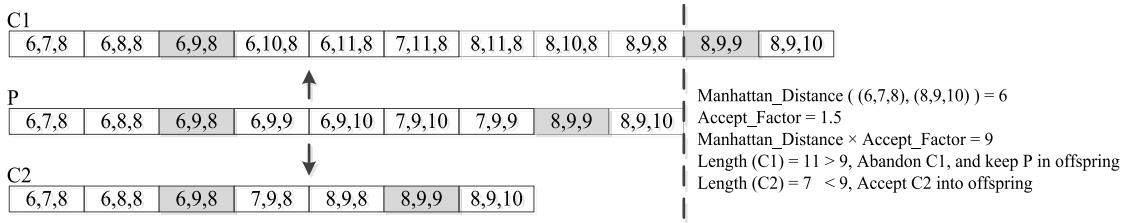



**FIGURE 5.** An example of mutation operator.

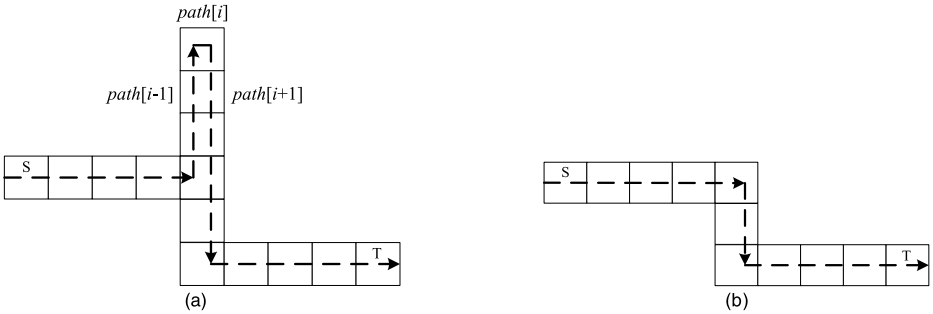



**FIGURE 6.** Removal of overlapping path. (a) Path with overlap. (b) Path after removal of overlap.

directly without additional conversion. However, due to the uncertainty of the sub-path generated by the crossover operator, the length of the offspring chromosome may be much longer than that of the parent chromosome. If there is no restriction to deal with it, the quality of the GA population will deteriorate step by step. So, the acceptable factor method was proposed in this article, i.e., if the length of the new chromosome is not greater than the product of the acceptable factor and the Manhattan distance of pipe nozzles, the new chromosome will be put into the offspring generation, otherwise the original parent chromosome will be kept. The acceptable factor is set by user, and the recommended value is in the range of [1.2, 1.8]. This method can accept inferior solutions to some extent, so it could be used to balance the optimization and convergence abilities of the algorithm.

#### 4) MUTATION OPERATOR

Mutation operator modifies the chromosome gene with a small probability to maintain the diversity of the population without destroying the convergence of the evolution. It is helpful to improve the searching performance of GA. Fig. 5 shows an example of the mutation operator. Two mutation points are selected in a chromosome (P) at random, A*-Router is used to regenerate the sub-path between the two points. Finally, the chromosome with the new sub-path goes into the offspring chromosomes. Similarly, the length of the offspring chromosome could be increased by the mutation operator, thus the acceptable factor method used in crossover operator is also applied here.

#### 5) REPAIR OPERATOR

In the process of chromosome generation, crossover, and mutation, there is a probability to obtain chromosomes with overlapping path or loop path. As shown in Fig. 6 and Fig. 7, although the pipe path with overlap(s) or loop(s) maintains the continuity between the starting point and the target point, it is in an illegal state, which needs to be repaired or discarded in time during the evolution. The repair operator consists of two parts: removal of overlapping path and removal of loop path.

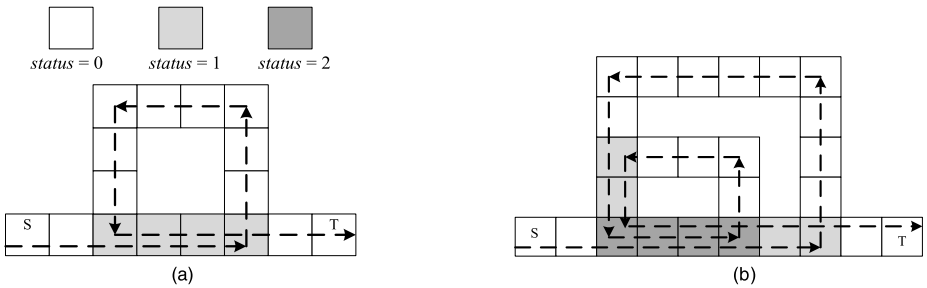



**FIGURE 7.** Path with loop(s). (a) Path with a single loop. (b) Path with loop nesting.

#### *a: REMOVAL OF OVERLAPPING PATH*

Suppose that the path grids information, e.g., coordinates (*layer*, *column*, *row*) and status, are stored in the array $path[0, \ldots, n-1]$, where $n$ is the number of grids on the path. The algorithm traverses the path grids from the starting grid successively. If the coordinates of $path[i-1]$ and $path[i+1]$ are found to be equal during the traversal, it indicates that some path segments on both sides of the symmetry point $i$ overlaps. Compare $path[i-k]$ and $path[i+k]$ $(k = 2, \ldots .t, \ldots)$ continuously, the iteration stops until $path[i-t] \neq path[i+t]$. Connect path segment $path[i+t, \ldots, n]$ to the grid at $path[i-t+1]$, then a time of overlap removal has been finished. Repeat the above steps until all overlaps in the path are removed.

#### *b: REMOVAL OF LOOP PATH*

Removal of overlaps should be executed before the removal of loops to ensure there is no overlap(s) in the path. Firstly, initialize $path[i].status$ to $-1$ $(0 \leq i < n)$, then the algorithm traverses the path grids successively from the starting grid to the target. When traversing grid $i$ $(0 \leq i < n)$, it performs $path[i].status \leftarrow path[i].status+1$, and check whether $path[i].status$ is greater than 0. If grid $i$ has been traversed only once, $path[i].status$ must be 0. Once $path[i].status$ is greater than 0, it means that the grid $i$ has been traversed at least twice, there are loop(s) or loop nesting(s) in the path, as shown in Fig. 7. If the algorithm is designed to remove all loops, it needs to handle many complex situations and the repairing effort is big. Therefore, our strategy is to discard the current path and rebuild a new one if the path contains loop(s).

The repair operator can be embedded into the process of chromosome generation, crossover, and mutation, which can effectively improve the quality of the chromosomes and the convergence of the algorithm.

#### 6) THE FLOWCHART OF GA-A*-ROUTER

The flowchart of GA-A*-Router for single pipe routing is shown in Fig. 8.

The parameters used in this GA include starting grid ($s$), target grid ($t$), identifier of the routing pipe (*pid*), identifier of the adjacent routed pipe (*adj_pid*), the number of intermediate connection points for chromosome (*pts_num*), population size (*pop_size*), the number of generation (*max_gen*), selection probability (*select_prob*), crossover probability (*crossover_prob*), and mutation probability (*mutate_prob*).

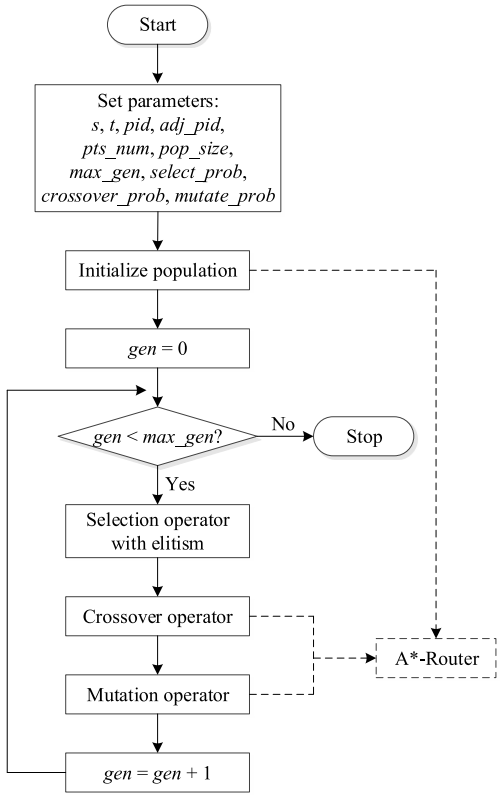



**FIGURE 8.** The flowchart of GA-A*-Router.

It can be seen from Fig. 8 that A*-router algorithm will be called repeatedly to generate sub-path in the implementations of population generation, crossover and mutation.

### *D. IMPROVED GENETIC ALGORITHM FOR MULTIPLE PIPES ROUTING (MULTI-PIPES-ROUTER)*

For multiple pipes routing, if the layout space of pipe is relatively isolated from each other, the routing sequence has little impact on the layout. Then, multiple pipes routing can be divided into several single pipe routing. For the space where pipes are centralized, the routed pipes could change the

layout environment. Thus, the layout depends on the routing sequence. The problem about routing sequence has been studied in [22], [46]. Their researches divide the pipes into different groups by the diameter values, the group with larger diameter value has high priority, and the routing sequence of pipes in a group is controlled by optimization algorithm. The routing algorithm evaluates the layout of each routing sequence, and adjusts the sequences according to the feedback. This process proceeds iteratively until the stopping criteria is met. The best practices in determining routing sequence are summarized as follows: (a) route the pipe with larger diameter first; (b) route the pipe with longer installation path first; (c) route the pipe with special requirement first; (d) route pipes in layers as much as possible, and give high priority to the pipes in lower layer; (e) route pipes according to the principle of piping system, e.g., pipes of the same type should be arranged at the same phase, water pipes should be arranged before oil pipes and steam pipes, oil pipes and steam pipes should be arranged separately; (f) for pipes having the same priority, compare the layouts based on different routing sequences. It can be seen that the critical task in multiple pipes routing is to arrange pipes with the same priority in a shared space.

To reduce the installation cost and improve the layout aesthetics, parallel routing of pipes with similar properties and adjacent nozzles are often required in engineering. The previous studies [26], [27], [30] used co-evolutionary algorithms to route multiple pipes in parallel, in which the multiple pipes corresponded by the populations evolve simultaneously. In the evolutionary process, the layout effect of a pipe is measured with the cooperation value computed by the chromosome of current pipe and the representatives from other populations. However, these methods are difficult to obtain ideal results and very time-consuming for solving complex problems.

Therefore, we generate the pipe routing sequence by using the mentioned strategies in advance, and then arrange pipes in parallel by the following algorithm (Multi-Pipes-Router), which is based on GA-A*-Router.

*Step 1:* Set the pipes to be routed in parallel by user input, and the routing sequence is denoted as $rs = [\mathrm{P1}, \mathrm{P2}, \mathrm{P3}, \ldots, \mathrm{P}n]$, which is a list of the pipe identifiers.

*Step 2:* Set the adjacent relationship for pipes in *rs* by using Fixed-Selection method mentioned in Section II-B. The relationship is represented by $ns = \{\mathrm{R_{P1}}, \mathrm{R_{P2}}, \mathrm{R_{P3}}, \ldots, \mathrm{R_{P}}n\}$, where $\mathrm{R_{P}}i$ stands for the pipe identifier that pipe P$i$ ($1 \leq i \leq n$) needs to be routed in parallel. Set $\mathrm{R_{P1}}$ to $-1$, which means P1 is the first pipe to route;

*Step 3:* Pop the first element of *rs* and set it to *pid*. Set the starting grid and target grid of pipe *pid* to *s* and *t*, respectively;

*Step 4:* Pop the first element of *ns* and set it to *adj_pid*;

*Step 5:* Set the number of connection points to *pts_num*; calculate the space for generating intermediate connection points and denote the area by *pts_space* (Section III-C);

*Step 6:* Set specific parameters of GA-A*-Router for pipe *pid* or use the default parameters, including *pop_size*, *max_gen*, *select_prob*, *crossover_prob*, and *mutate_prob*;

*Step 7:* Prepare the layout space, i.e., extend the obstacles and the routed pipes outward by a certain grid number *ext_num* calculated by formula (1). The extension grids are marked as temporary-obstacle-grids;

*Step 8:* Use GA-A*-Router to route pipe *pid* in the prepared layout space with the parameters, including *pid*, *s*, *t*, *adj_pid*, *pts_num*, and *pts_space*;

*Step 9:* Mark the grids belonging to (on the path or in the scope of pipe extension) pipe *pid* as obstacle-grids, and set the pipe identifier of these grids to *pid*;

*Step 10:* Restore the layout space by setting the temporary-obstacle-grids to free-grids;

*Step 11:* If *rs* is not empty, go to *Step 3*, otherwise go to *Step 12*;

*Step 12:* Output the layout result.

And, as noted, there is no special treatment if pipe *pid* is the first routing pipe, but for the pipes arranged later, in the processes of population initialization and evolution, the intermediate connection points should be generated next to the specified routed pipe and then be sorted in the selected coordinate direction. This method could guide the parallel layout of pipes, but the randomly generated intermediate connection points may be close or far from each other, which weakens the guiding effect on parallel routing. Another method is to generate the intermediate connection points around the equidistant points of the pipe to be routed in parallel. The positions of these equidistant points are determined by the number of connection points. This method can ensure the uniform distribution and omit sorting operation on the connection points, but it reduces the randomness of the connection points, and sometimes affects the diversity of genetic population. The first method based on connection-points sorting is adopted in our simulations. The schematic diagram of parallel routing via connection points is shown in Fig. 9. Note that, the connection points cannot be located in any obstacles or routed pipes, and the number of adjacent grid-pairs for measuring the effect of parallel layout is computed as a bonus in the fitness function.

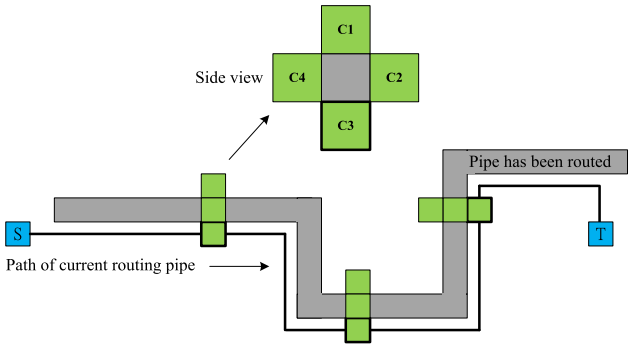



**FIGURE 9.** A schematic diagram of parallel routing based on connection points.

### E. IMPROVED ALGORITHM FOR BRANCH PIPE ROUTING (BRANCH-PIPE-ROUTER)

Branch pipe routing in ship is also a multi-terminal pipe routing problem, and it is formulated as a Euclidean Steiner Minimal Tree with Obstacles (ESMTO) problem. ESMTO is NP-hard even in 2D spaces [19], which makes the branch pipe routing in ship becomes much more difficult even for skilled pipe designers.

Through the analysis on the previous work about branch pipe routing in Section I, we can see that the methods based on connecting terminals sequentially [36], [37] or making some assumptions about branch shapes [2] can guarantee feasible solutions but may be not the optimums, however, they are polynomial-time algorithms and easy to implement; the methods based on intelligent optimization algorithms may provide more reasonable solutions, but they take more computation time.

Considering the characteristic of ship pipe routing, e.g., arrange pipes orthogonally with rigid structures, explore pipe routes in complex 3D space, route many pipes with different diameter values, *et al.*, the computation time and layout quality should be balanced together in the practical branch-pipe-routing algorithms. Therefore, we utilize the modified Steiner Tree framework (i.e., create a feasible solution of Steiner Tree by connecting the terminals sequentially [36], [37]) combined with the proposed single pipe routing algorithms to arrange branch pipe in this article.

In general, the branch pipe in ship can be grouped into two categories: the first one has the same diameter value for all the branches, namely, the equal-diameter branch pipe, and the other one has different diameter values for the branches, namely, the unequal-diameter branch pipe. For routing the unequal-diameter branch pipe, the connecting sequence of nozzles is determined by the user requirements, for instance, some require that the subbranches should be connected to the main branch, while some require that all branches are classified into different grades by the diameter values. The subbranch with a higher grade is arranged in priority, and the branch point should be formed on the subbranch with the adjacent higher grade. But for routing equal-diameter branch pipe, the constraints are not so much.

The routing methods are mainly implemented by the evolutionary algorithms or the modified Steiner Tree algorithms.

In most evolutionary algorithms [27]–[30], [33], [38], [39], the branches of a pipe are split into several single pipes that start at the same point but end at different points (with the same diameter value), then each single pipe is routed using population-based algorithm. In the process, the cooperation value and the fitness value of the subbranches are measured and compared.

The procedures of modified Steiner Tree algorithm are as follows: (a) generate an ordered list $S$ including all the terminals of the branches; (b) find the path between the first two terminals in $S$; (c) for $k = 3, \ldots, n$, do the following steps, take out the $k$-th terminal and find the path from terminal $k$ to the routed branches according to the specified connection strategy, e.g., shortest path first, overall of path length and bend numbers first, etc., repeat the step until all the terminals are connected.

In terms of applicability, the modified Steiner Tree algorithm is more versatile than the evolutionary algorithms.

The modified Steiner Tree framework combined with LEE-Router, A*-Router or GA-A*-Router is described as follows (Branch-Pipe-Router):

*Step 1:* Generate the routing sequence $S$ for the terminals of branch pipe, in which the terminals are sorted by descending order of the diameters. The order of the terminals with the same diameter value can be determined by the optimization algorithm;

*Step 2:* Judge the category of the branch pipe, if it is equal-diameter branch pipe, go to *Step 3*, else it is unequal-diameter branch pipe, go to *Step 4*;

*Step 3:* Route the equal-diameter branch pipe using the modified Steiner Tree algorithm mentioned above. When finished, go to *Step 7*;

*Step 4:* Set the connection strategy for branches: if the subbranches only connect to the main branch, go to *Step 5*, else if a subbranch can connect to another subbranch in higher grade, go to *Step 6*;

*Step 5:* Generate the main branch path between the first two terminals in $S$, and then for $k = 3, \ldots, n$, do the following: find the sub-path between the $k$-th terminal and the branch point on the main branch. When finished, go to *Step 7*;

*Step 6:* Generate the branch path between the first two terminals in $S$, and then for $k = 3, \ldots, n$, do the following: find the path between the $k$-th terminal and the branch point on a subbranch, e.g., the branch in the adjacent higher grade;

*Step 7:* Output the layout for the branch pipe.

Some notes of Branch-Pipe-Router are illustrated as follows:

1) Two decisions need to be made during the process. The first one is to give the sequence in which the terminals are connected. The second one involved the question if the branch point should appear in the main branch or that it can also be in a different branch. These problems can be manually defined by the user, or can be processed as a task by the optimization algorithm.
2) In step 3, 5 and 6, LEE-Router, A*-Router and GA-A*-Router can be used to route a single branch pipe.
3) The algorithms used to find the branch points on the routed branches in step 3, 5 and 6 can be LEE-Router or A*-Router. LEE-Router can find the shortest paths from a specified terminal grid to any grids on the routed branches in one execution, while A*-Router only finds the path between two grids at a time and needs to be called lots of times to find an appropriate branch point. However, during the exploration, A*-Router can take more constraints into account than LEE-Router does, which results in better layouts. In this router, both

of them can meet the routing quality with acceptable computation time.

4) To ensure the success for routing branch pipe, the grids in the routed subbranches which are not expected to be connected must be marked as obstacle-grids.

## IV. SIMULATIONS AND RESULTS

Two experiments are carried out to demonstrate the feasibility and effectiveness of the proposed algorithms. The first case is based on a well-designed pipe routing problem, it is used to verify the optimization performance of the algorithms, for instance, the path-searching ability, the time cost, the convergence rate and speed, as well as the robustness. The second case is extracted from a practical fuel piping system of ship engine room, in which more actual routing requirements are involved, e.g., the direction constraint on nozzles, the arrangement for branch pipes and parallel pipes, the arrangement for pipes with different diameter values, etc. The experiments are conducted based on the following environment.

Runtime environment: Intel (R) Core (TM) i5-7500 CPU 3.40 GHz; 4.00 GB RAM; Windows 7 64-bit OS.

Development environment: Microsoft Visual Studio 2013 VC++; Compiler optimization option is set to Maximum Optimization (Favor Speed) (/O2).

### A. CASE STUDY 1

#### 1) CASE DESCRIPTION

In this case, a piping model with several obstacles is designed. The routing space is a cube with side length of 100 mm, and the coordinate of its center is (0, 0, 0). The obstacles scattered in the routing space are represented by the cuboids whose diagonal coordinates are listed in Table 1.

The routing space is divided into cubic grids, and each grid can be indexed by a grid coordinate (*row*, *column*, *layer*). The four piping cases are shown in Table 2.

**TABLE 1. The diagonal coordinates of the obstacles.**

| No. | Diagonal coordinates $(x_1, y_1, z_1) \sim (x_2, y_2, z_2)$ |
|---|---|
| 1 | (-45, -50, 20) ~ (-30, 20, 0) |
| 2 | (0, -50, 30) ~ (50, 10, 10) |
| 3 | (0, -50, -10) ~ (50, 50, -30) |
| 4 | (-40, -50, 50) ~ (-20, 50, 30) |
| 5 | (-50, -50, -20) ~ (-20, 50, -30) |
| 6 | (-20, -10, -40) ~ (0, 0, 20) |
| 7 | (-50, -50, -20) ~ (-30, -20, 0) |
| 8 | (-30, -50, -50) ~ (10, 30, -40) |
| 9 | (-20, -50, -40) ~ (0, -40, 20) |
| 10 | (-10, 0, -10) ~ (0, 50, 10) |
| 11 | (30, -50, 50) ~ (20, -40, 40) |
| 12 | (-10, -50, 50) ~ (-20, -10, 20) |
| 13 | (-14, -10, -16) ~ (-6, -16, 8) |

#### 2) SIMULATION FOR SINGLE PIPE ROUTING

##### a: PARAMETER SETTINGS

The parameters for A*-Router: the factors in evaluation function $a = 1$, $b = 10$, $c = 10$, $w = 1$; the minimum bending length $L_{min}$ = side length of a grid.

The parameters for A*-GA-Router: the constant and factors in fitness function K = 1000000, $\alpha = 0.2$, $\beta = 0.4$, $\gamma = 0.4$, $\delta = 0$ ($\delta$ is zero for single pipe routing); the number of intermediate connection points $pts\_num = 3$; acceptable factor $acept\_factor = 1.4$; population size $pop\_size = 40$, the number of generation $max\_gen = 50$, selection probability $select\_prob = 1.0$, crossover probability $cross\_prob = 0.8$, mutation probability $mutate\_prob = 0.05$.

To compare the performances of different algorithms, the idea of [31]–[33] using maze algorithm to construct chromosomes has been implemented in our environment, which is named as LEE-GA-Router, and the maze algorithm in LEE-GA-Router is named as LEE-Router. For comparison purposes, LEE-GA-Router uses the parameter settings of A*-GA-Router as much as possible, including fitness function, population size, number of generations, selection probability, crossover probability and mutation probability. But the number of intermediate connection points (auxiliary point in [31]–[33]) is 1 in LEE-GA-Router.

The energy distribution of the routing space is configured as follows: the grids near the walls, floors, and equipment are set to low energy values. i.e., the energy of the grid adjacent to the support is set to 0; the grid energy is increased by 5 if the grid is far away from the support by a grid-side length; and the energy of the grid with more than 5 grid-side length away from the support is set to 25.

##### b: RESULTS AND CONCLUSIONS

In order to evaluate the average performance for single pipe routing, each algorithm is executed 10 times for Case 1 and Case 2, respectively. The statistical results are shown in Table 3.

The optimal layouts are shown in Fig. 10 ~ 13.

Analyze the results in Table 3, the following conclusions can be drawn:

1) A*-Router and LEE-Router are both deterministic algorithms, which produced the same results during

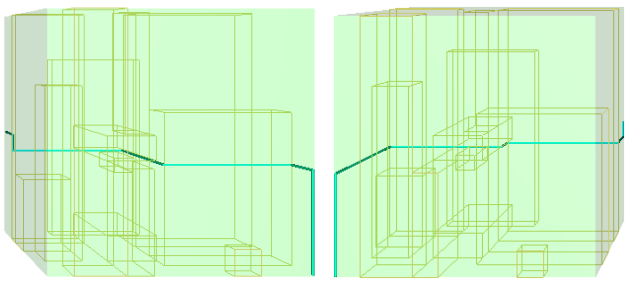



**FIGURE 10. The optimal layouts of A*-Router. (a) Case 1. (b) Case 2.**

**TABLE 2.** Information of the piping cases.

| Case No. | Routing type | Space decomposition | Grid coordinates of pipe nozzles |
|---|---|---|---|
| Case 1 | Single pipe routing | 100×100×100 | P1: s(99, 0, 99) ~ t(0, 46, 0) |
| Case 2 | Single pipe routing | 100×100×100 | P1: s(0, 0, 99) ~ t(99, 46, 0) |
| Case 3 | Multiple pipes routing | 50×50×50 | P1: s(49, 0, 49) ~ t(0, 23, 0)<br>P2: s(48, 0, 49) ~ t(1, 23, 0)<br>P3: s(47, 0, 49) ~ t(2, 23, 0) |
| Case 4 | Multiple pipes routing | 100×100×100 | P1: s(99, 0, 99) ~ t(0, 46, 0)<br>P2: s(98, 0, 99) ~ t(1, 46, 0)<br>P3: s(97, 0, 99) ~ t(2, 46, 0) |

**TABLE 3.** Result comparison of single pipe routing.

| Case No. | Algorithm name | Average length (mm) | Average number of bends | Average energy value | Average fitness value | Fitness value of the algorithm's best solution | Number of times to get global optimal solution | Average convergence generations | Average execution time (ms) |
|---|---|---|---|---|---|---|---|---|---|
| Case 1 | A*-Router | 245.0 | 6.0 | 0 | 999948.63 | 999948.63 | 0 | NA | 98.3 |
| | LEE-Router | 245.0 | 8.0 | 0 | 999947.81 | 999947.81 | 0 | NA | 421.2 |
| | A*-GA-Router | 245.0 | 5.3 | 0 | 999948.89 | 999949.00 | 7 | 21.6 | 74,601.1 |
| | LEE-GA-Router | 245.0 | 7.0 | 0 | 999948.19 | 999948.19 | 0 | 18.3 | 284,580.2 |
| Case 2 | A*-Router | 245.0 | 6.0 | 0 | 999948.63 | 999948.63 | 10 | NA | 104.3 |
| | LEE-Router | 245.0 | 9.0 | 0 | 999947.38 | 999947.38 | 0 | NA | 435.3 |
| | A*-GA-Router | 245.0 | 6.6 | 0 | 999948.37 | 999948.63 | 5 | 18.0 | 81,342.0 |
| | LEE-GA-Router | 245.0 | 9.0 | 0 | 999947.38 | 999947.38 | 0 | 4.2 | 299,624.7 |

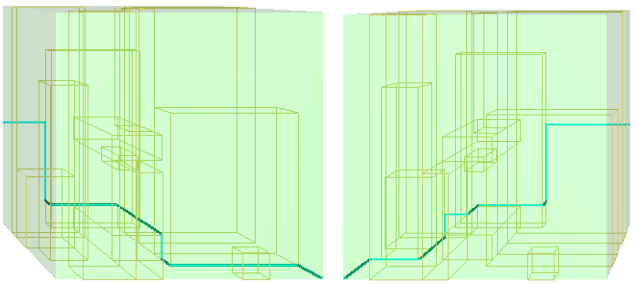



**FIGURE 11.** The optimal layouts of LEE-Router. (a) Case 1. (b) Case 2.

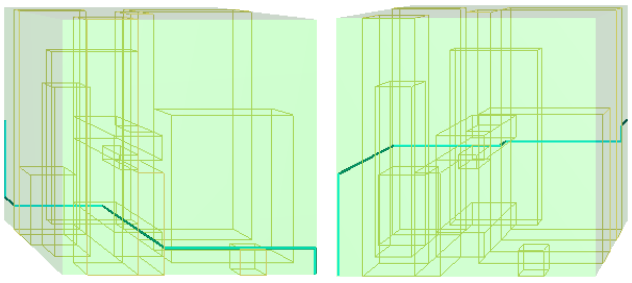



**FIGURE 12.** The optimal layouts of A*-GA-Router. (a) Case 1. (b) Case 2.

10 executions; A*-Router can get the global optimal solution in Case 2, while LEE-Router cannot get the global optimal solutions in the experiments of the two cases, therefore, neither algorithm is a global optimization algorithm; A*-Router can evaluate the bending cost when exploring the grids, it can get the global optimal path with less number of bends compared with LEE-Router; In terms of time performance, A*-Router is much better since it is optimal-first search, the time cost of A*-Router is about 1/4 of LEE-Router.

2) A*-GA-Router and LEE-GA-Router are non-deterministic algorithms, which may produce different results in different executions; In the experiments of the two cases, A*-GA-Router converges to the global optimal solutions with a probability of not less than 50%, whereas LEE-GA-Router cannot converge to any of the global optimal solutions. The main reason is that A*-GA-Router is based on multiple connection points, which can generate a better initial population. The fitness value curves of the simulations with A*-GA-Router are shown in Fig. 14; A*-GA-Router is also better than LEE-GA-Router in terms of optimization quality and time efficiency. It produces pipe routes with fewer bends and takes about 1/4 of the execution time of LEE-GA-Router. These

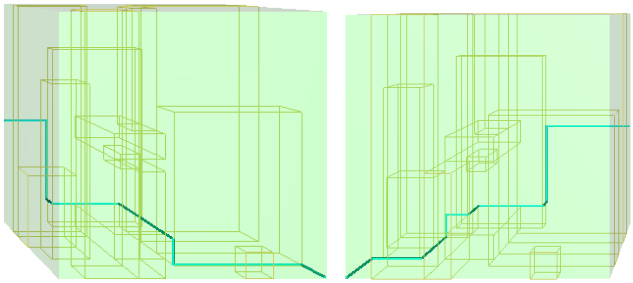



**FIGURE 13.** The optimal layouts of LEE-GA-Router. (a) Case 1. (b) Case 2.

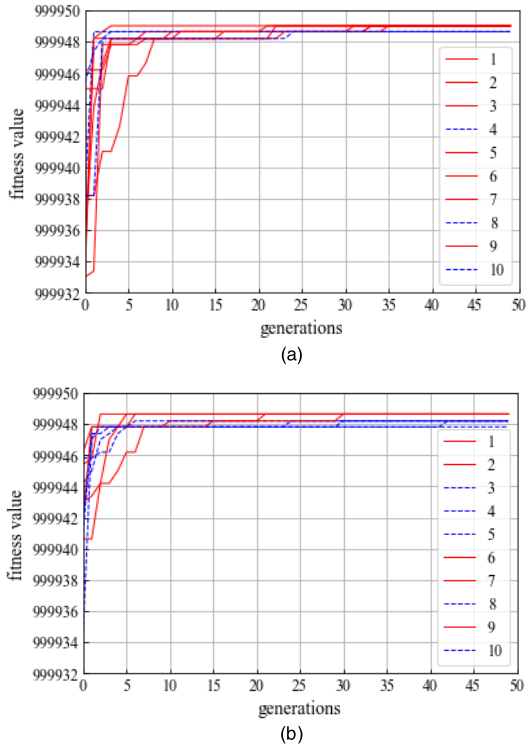



**FIGURE 14.** Fitness value curves of A*-GA-Router. (a) Case 1. (b) Case 2.

results are consistent with the fact that the two algorithms use A*-Router and LEE-Router internally.

3) Compared with A*-Router, A*-GA-Router can provide a variety of layouts and has stronger optimization ability, e.g., it can find the path with fewer bends; A*-GA-Router is based on the mechanism of evolution, and calls A*-Router to create sub-path segments in each iteration, so it is much slower than A*-Router. In general, A*-GA-Router is suitable for searching high-quality solutions, whereas A*-Router is better for creating initial solutions.

### 3) SIMULATION FOR MULTIPLE PIPES ROUTING

Some early studies tried to use algorithms based on multi-populations with co-evolution mechanism to route pipes in parallel, and have observed the effect of parallel piping [26]–[30], [39]. However, such algorithms mainly rely on fitness and penalty functions to avoid interference between pipes, which may produce invalid layouts or the quality is not ideal. Moreover, the shortcomings will be more serious when solving large-scale or complex problems.

The proposed algorithm (Multi-Pipes-Router) uses A*-Router and the connection-point strategy to generate chromosomes, and the previous routed pipes are regarded as obstacles. Consequently, the subsequent pipe can completely avoid collision with the routed pipes. The following experiments will verify the algorithm in terms of computation time and optimization quality.

#### *a: PARAMETER SETTINGS*

The parameters of Multi-Pipes-Router and A*-GA-Router are the same except for the parameter $\delta$. $\delta$ related component in the fitness function plays a role in guiding the parallel layout of pipes. If $\delta$ is too small, it leads to insufficient parallel layout; otherwise, if $\delta$ is too big, it leads to excessive parallel layout. Users should set $\delta$ to an appropriate value based on the testing feedback. In this experiment, set $\delta = 0.3$ in Case 3 and set $\delta = 0.1$ in Case 4.

#### *b: RESULTS AND CONCLUSIONS*

Since the routed pipes are considered as obstacles in multiple pipes routing, the overall layouts are related to the routing sequence. In the experiments of Case 3 and Case 4, the routing sequence is set to P1→P2→P3.

In order to evaluate the average performance for multiple pipes routing, the algorithm is executed 10 times for Case 3 and Case 4, respectively. The statistical results are shown in Table 4.

The optimal layouts are shown in Fig. 15.

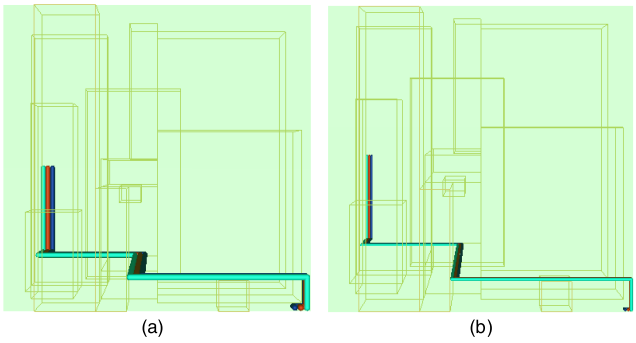



**FIGURE 15.** The optimal layouts of Multi-Pipes-Router. (a) Case 3. (b) Case 4.

By analyzing the results in Table 4, the following conclusions can be drawn:

1) The grids number of Case 4 is 8 times that of Case 3, and the average execution time of Case 4 is also about 8 times that of Case 3, which indicates the execution time increases linearly with the problem size. In addition, to solve Case 4 with $10^6$ grids and $max\_gen = 50$, the time for routing three pipes is about 150s. It shows

**TABLE 4.** Results comparison of multiple pipes routing.

| Case No. | Algorithm name and grids decomposition | Average length (mm) | Average number of bends | Average energy value | Average fitness value | Fitness value of the algorithm's best solution | Number of times to get the global optimal solution | Average execution time (ms) |
|---|---|---|---|---|---|---|---|---|
| Case 3 | Multi-Pipes-Router (50×50×50) | 736.8 | 20.1 | 0 | 2999917.28 | 2999917.95 | 6 | 20,501.7 |
| Case 4 | Multi-Pipes-Router (100×100×100) | 736.8 | 20.4 | 0.5 | 2999893.05 | 2999894.00 | 7 | 153,080.6 |

that the time performance is acceptable in practical routing.

2) In the two cases, the difference between the average fitness value and the fitness value of the best solution is very small. It means that the algorithm can converge to the sub-optimal solution when the best solution cannot be found.
3) The average results such as path length, bending number and energy value in the two cases are compared, and the values of each pair are very close. Moreover, both cases can converge to the global optimal solution for multiple times. It indicates that the optimization capability of the algorithm is not sensitive to the problem size.

The fitness value curves of one simulation (i.e., the 9$^{th}$) for Case 3 and Case 4 are shown in Fig. 16. P1 is the first routed pipe, since the searching space is big and flexible, it requires more generations to converge to the global optimal solution; P2 and P3 are then routed one by one. To pursue parallel pipe layout, the connection points are generated around the routed pipe. Since the searching space is small, the algorithm could converge to the optimal solution within fewer generations.

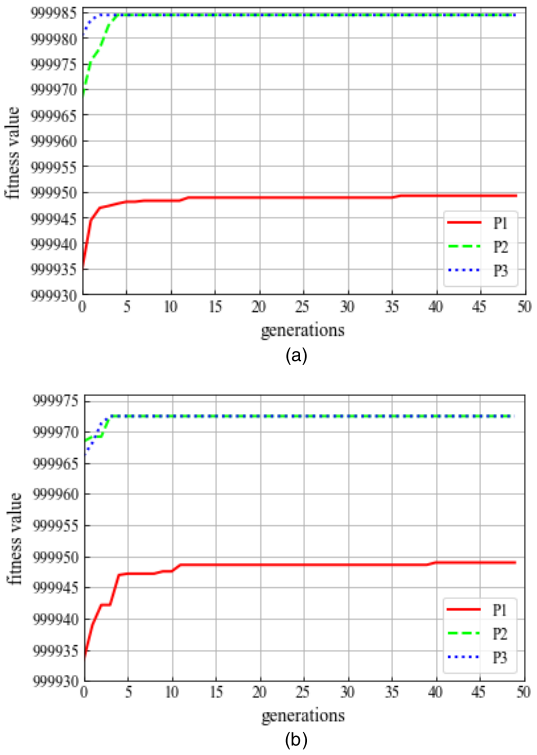



**FIGURE 16.** Fitness value curves of Multi-Pipes-Router. (a) Case 3. (b) Case 4.

Set $\delta$ to 0, A*-GA-Router will not consider the reward of parallel layout between pipes, e.g., Fig. 17 is the layout of routing pipes independently for Case 3.

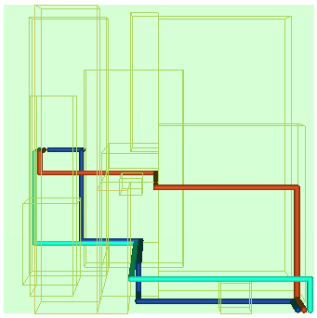


**FIGURE 17.** A layout of Case 3 with $\delta = 0$.

Set $\delta$ to a big value, A*-GA-Router will reward more for the parallel layout of pipes, even ignore the increase in path length or bending number, e.g., Fig. 18 shows a layout of Case 3 with $\delta = 2$.

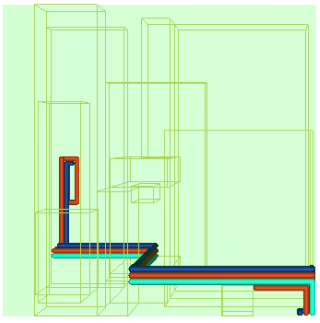


**FIGURE 18.** A layout of Case 3 with $\delta = 2$.

Set the routing sequence to P3→P2→P1, then new optimal results of Case 3 can be produced. Two layouts are shown

**TABLE 5.** Diagonal coordinates of the simplified equipment model.

| Equipment name | Diagonal coordinates $(x_1, y_1, z_1) \sim (x_2, y_2, z_2)$ |
|---|---|
| Fuel oil tank | (-5000, 200, -1000) ~ (-2000, 2000, -3000) |
| Fuel transfer pump 1 | (-1000, -350, -2300) ~ (-500, -250, -3000); (-900, -100, -2500) ~ (-600, -250, -2800); (-1000, 250, -2500) ~ (-500, -100, -2950); (-875, -100, -2250) ~ (-625, 150, -2500) |
| Fuel transfer pump 2 | (-1000, -1350, -2300) ~ (-500, -1250, -3000); (-900, -1100, -2500) ~ (-600, -1250, -2800); (-1000, -750, -2500) ~ (-500, -1100, -2950); (-875, -1100, -2250) ~ (-625, -850, -2500) |
| Fuel oil storage tank 1 | (400, -400, -1800) ~ (1800, 1400, -3000) |
| Fuel oil storage tank 2 | (2600, -400, -1800) ~ (4000, 1400, -3000) |
| Diesel generator 1 | (-2800, -1800, 2700) ~ (-1700, -2000, 900); (-2550, -1300, 1350) ~ (-1950, -1800, 900); (-2410, -1550, 1590) ~ (-2090, -1800, 1350); (-2650, -1150, 2230) ~ (-1885, -1800, 1590); (-2465, -1600, 2530) ~ (-1615, -1800, 2230); (-1885, -1600, 2230) ~ (-1615, -1800, 1930) |
| Diesel generator 2 | (1200, -1800, 2700) ~ (2300, -2000, 900); (1450, -1300, 1350) ~ (2050, -1800, 900); (1590, -1550, 1590) ~ (1910, -1800, 1350); (1375, -1150, 2230) ~ (2115, -1800, 1590); (1535, -1600, 2530) ~ (2385, -1800, 2230); (2115, -1600, 2230) ~ (2385, -1800, 1930) |
| Steam boiler | (-3600, -2000, 1700) ~ (-5000, 600, 3000); (-4650, 800, 2550) ~ (-3950, 600, 2150); (-3900, -2000, 1200) ~ (-4670, -1300, 1700) |
| Hot water boiler | (3300, 200, 2540) ~ (4600, -2000, 1640); (3550, -200, 2720) ~ (4350, -2000, 2540); (3750, 360, 2200) ~ (4150, 200, 1980); (4300, -1500, 1240) ~ (3600, -2000, 1640) |
| Marine main engine | (-900, -1400, 1350) ~ (400, -2000, 650); (-600, -1850, 2690) ~ (100, -2000, 1690); (-700, -1670, 1690) ~ (200, -2000, 1350); (-950, -1150, 2690) ~ (350, -1850, 1690); (-650, -1790, 2910) ~ (50, -1490, 2690); (-850, -910, 2690) ~ (250, -1150, 1690); (-850, -1310, 2970) ~ (250, -910, 2690); (250, -1310, 1410) ~ (-850, -910, 1690) |

in Fig. 19. The influence of routing sequence can be seen by comparison with Fig. 15.

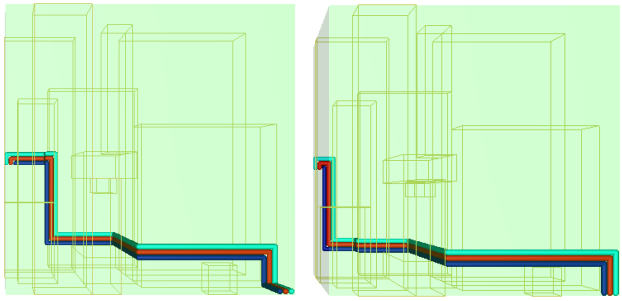



**FIGURE 19.** The layouts of Case 3 with sequence P3→P2→P1. (a) Result 1. (b) Result 2.

### B. CASE STUDY 2

#### 1) CASE DESCRIPTION

In this case study, the fuel piping system of a ship engine room is employed. Fig. 20 shows the schematic diagram of this fuel piping system. The main equipment includes fuel oil tank, fuel transfer pumps, fuel oil storage tanks, marine main engine, diesel generators and boilers. To connect the related equipment, eight fuel oil pipelines are involved.

According to the schematic diagram and the actual installation locations of the equipment, the 3D model of the routing space is created by using SolidWorks, as shown in Fig. 21. The routing space is set as 10000mm (length) × 4000mm (width) × 6000mm (height), and the coordinate origin is at the center of the space. The diagonal coordinates of the cuboids composing the main parts of the simplified equipment are available at Table 5. In accordance with the minimum diameter value of the pipelines and the allowable distance between pipes, the size of the grid is set as 50mm. Thus, the routing space is decomposed into 200 (rows) × 80 (columns) × 120 (layers) cubic grids.

The basic information of the pipes and connection points are summarized in Table 6.

#### 2) PARAMETER SETTINGS AND PRETREATMENT

The proposed algorithms for routing multiple pipes and branch pipes need to call the single pipe routing algorithm internally. And for this case, the parameters of each single pipe routing algorithm are set as follows.

The parameters for A*-Router: the factors in evaluation function $a = 1$, $b = 10$, $c = 10$ (consider the energy constraint) or $c = 0$ (ignore the energy constraint), $w = 1$, the minimum bending length $L_{min}$ = side length of a grid.

The parameters for A*-GA-Router: the constant and factors in the fitness function K = 1000000, $\alpha = 0.2$, $\beta = 0.4$, $\gamma = 0.4$ (consider the energy constraint) or $\gamma = 0$ (ignore the energy constraint), $\delta = 0.3$ (take effect when routing pipes in parallel); the number of intermediate connection points $pts\_num = 3$; acceptable factor $acept\_factor = 1.4$; population size $pop\_size = 20$, the number of generations

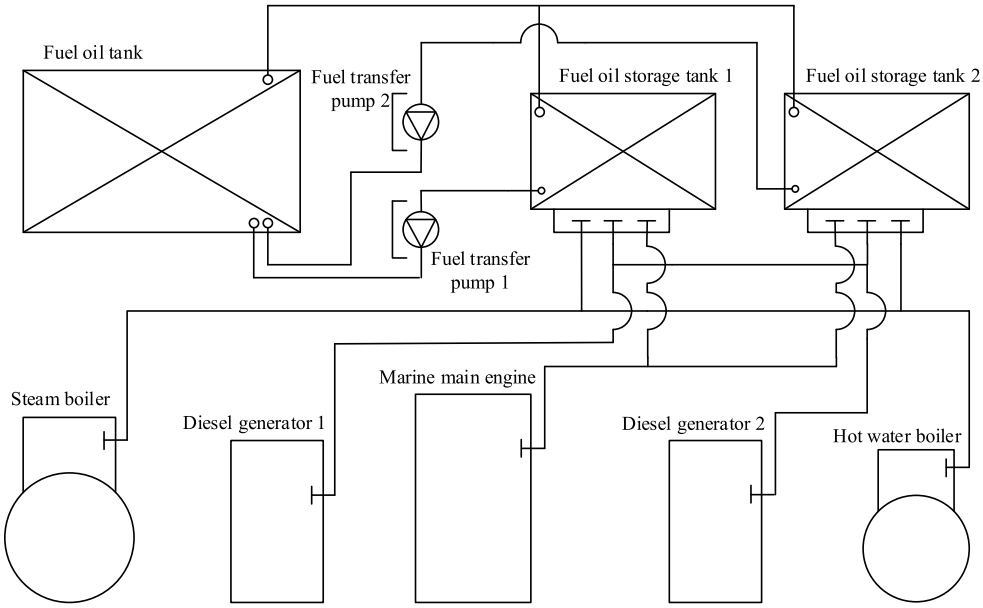



**FIGURE 20.** Schematic diagram of the fuel piping system in a ship engine room.

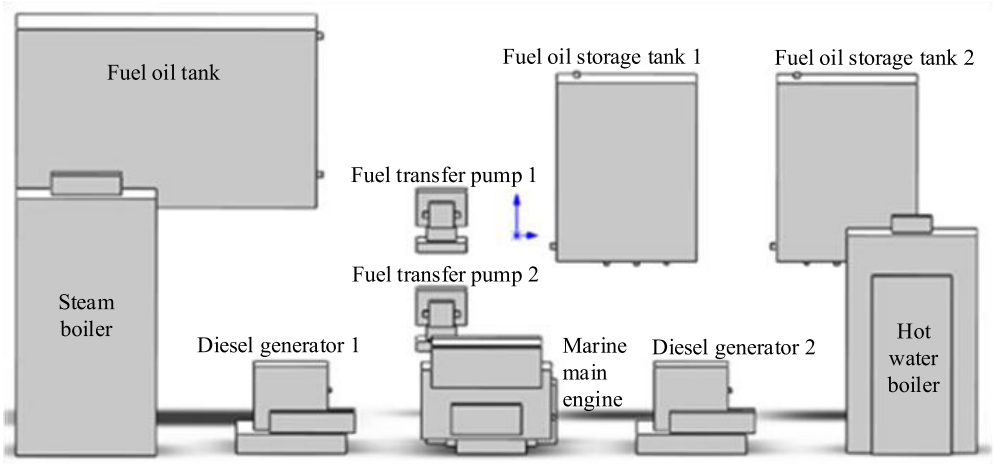



**FIGURE 21.** The CAD model of the fuel piping system constructed by SolidWorks.

*max_gen* = 20, selection probability *select_prob* = 1.0, crossover probability *cross_prob* = 0.8, mutation probability *mutate_prob* = 0.05.

Parameters for LEE-Router: No special settings required.

Several pretreatments on the routing case are performed as follows:

1) Pretreatment on the routing sequence. Piping layout depends on the routing sequence of pipelines and the connecting sequence of subbranches. In this case, the routing spaces of P1, P4, and P5 are relatively isolated, and their order has little impact on the final result; as for P2 and P3, the nozzles on the oil tank are adjacent, and the other nozzles on the transfer pumps are in a similar direction, thus, they are expected to be routed in parallel, in addition, the installation path of P3 is longer than that of P2, so it is better to route P3 prior to P2; the routing spaces of P6, P7 and P8 are adjacent or shared under the oil storage tanks, and then their routing sequence is determined by the diameter values of their main branches. Consequently, a reasonable sequence for routing these pipes is {P1, P4, P5, P3, P2, P6, P7, P8}. Moreover, the connecting sequence of branch-pipe nozzles is arranged in descending order by the diameter values, and the sequence of nozzles with the same diameter value is generated by algorithm.

2) Pretreatment on the energy distribution in the routing space. In practical engineering, pipes are routed close to the surfaces of walls, floors and equipment, so that the pipes can be installed with lower cost and taken less working space. Therefore, the energy of grids locating within a certain distance from the support can be set to 0, e.g., in this case, set 3 grid-side lengths as the distance, which should be larger

**TABLE 6.** Information of the piping paths and nozzles.

| Pipe | Type | Connected equipment | Outer diameter (Ø) of each pipe nozzle (mm) | Grid coordinates of each pipe nozzle (row, column, layer) | Extension distance of each pipe nozzle (grid number) |
|---|---|---|---|---|---|
| P1 | equal-diameter branch pipe | oil tank | Ø60 | (60,76,4) | 2 |
| | | oil storage tank 1 | Ø60 | (112,68,10) | 2 |
| | | oil storage tank 2 | Ø60 | (156,68,10) | 2 |
| P2 | single pipe | oil tank | Ø48 | (60,48,5) | 2 |
| | | transfer pump 1 | Ø48 | (81,40,12) | 2 |
| P3 | single pipe | oil tank | Ø48 | (60,48,4) | 2 |
| | | transfer pump 2 | Ø48 | (81,20,12) | 2 |
| P4 | single pipe | transfer pump 1 | Ø48 | (88,40,12) | 2 |
| | | oil storage tank 1 | Ø48 | (107,34,10) | 2 |
| P5 | single pipe | transfer pump 2 | Ø48 | (88,20,12) | 2 |
| | | oil storage tank 2 | Ø48 | (151,34,10) | 2 |
| P6 | unequal-diameter branch pipe | oil storage tank 1 | Ø64 | (118,31,19) | 7 |
| | | oil storage tank 2 | Ø64 | (170,31,19) | 7 |
| | | steam boiler | Ø46 | (22,1,88) | 2 |
| | | hot water boiler | Ø46 | (186,1,88) | 2 |
| P7 | unequal-diameter branch pipe | oil storage tank 1 | Ø62 | (124,31,19) | 4 |
| | | oil storage tank 2 | Ø62 | (164,31,19) | 4 |
| | | diesel generator 1 | Ø44 | (63,11,100) | 2 |
| | | diesel generator 2 | Ø44 | (143,11,100) | 2 |
| P8 | equal-diameter branch pipe | oil storage tank 1 | Ø44 | (130,31,19) | 2 |
| | | oil storage tank 2 | Ø44 | (158,31,19) | 2 |
| | | marine main engine | Ø44 | (107,6,105) | 2 |

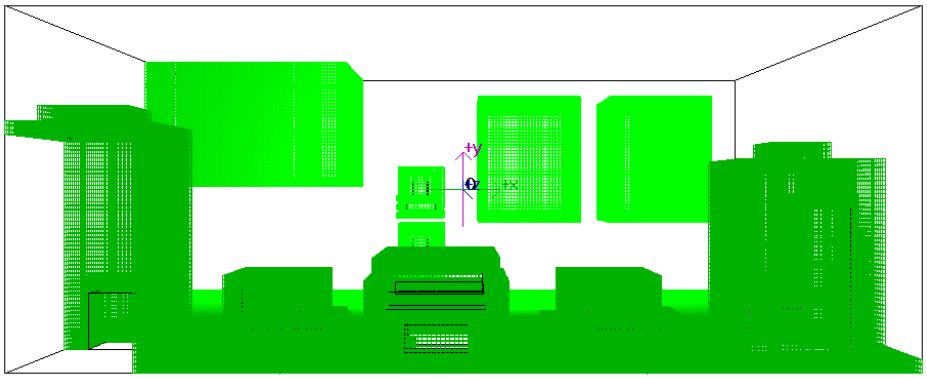
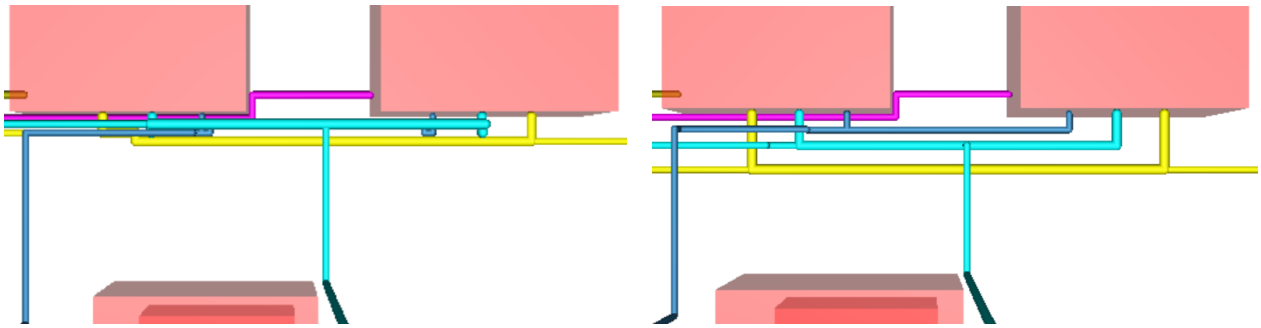


**FIGURE 22.** The grids with zero energy value in the routing space.

than the maximum distance of obstacle extension (1 grid-side length). The energy value of other free-grids is set to 25. Fig. 22 shows the grids with zero energy. The reason for setting zero energy to these grids is that the installation cost is not much different in these areas. Moreover, the same energy setting can better meet the direction constraint on pipe nozzles and reduce the bending numbers.

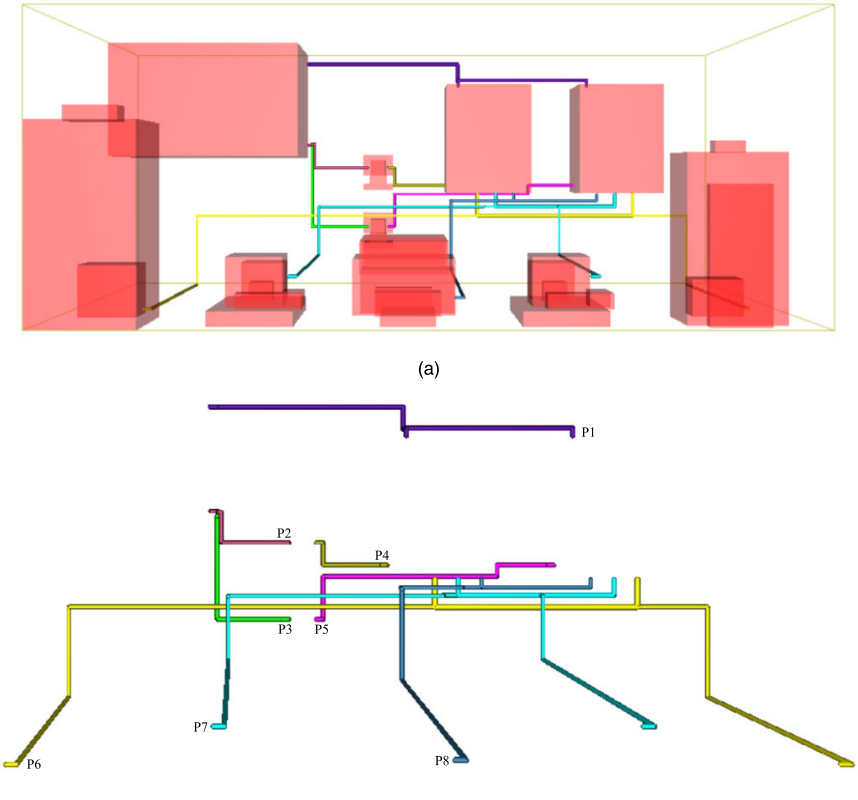
3) Pretreatment on the direction constraint of the pipe nozzles. The initial pipe nozzles may be covered by the obstacle extension, to eliminate the impact and meet the direction constraints on pipe nozzles, the locations of the connection points will be extended outward appropriately in advance. The grid paths between the initial nozzles and new nozzles are straight, which should be kept and marked as obstacles. When finishing the routes between the new pipe nozzles, both kinds of paths will be connected together to get a complete layout. Note that the extension distance must be greater than the maximum distance of obstacle extension (1 grid-side length). In general, the same extension distance will be specified for all pipe nozzles at first, to get better result, the extension distances of some nozzles could be adjusted according to the feedback of initial layouts. In this case, when the extension distance of each nozzle is set to 2 grid-side lengths. The obtained layout of LEE-Router under the oil storage tanks is shown as Fig. 23 (a). As can be seen, to avoid collisions between pipes, many bending parts are involved, which is not allowed in practical engineering. For this reason, the nozzle-extension distances of P6 and P7 under oil storage tanks are adjusted to longer values as shown in Table 6. Consequently, the new layout is shown as Fig. 23 (b).

After these pretreatment, the pipes can be arranged in turn by following steps: (a) extend the obstacles outward by a



(a) (b)

**FIGURE 23.** The routing effects under oil storage tanks obtained by LEE-Router. (a) The extension distance for all pipe nozzles is set to 2 grid-side lengths. (b) The extension distances for the nozzles of P6, P7, and P8 under oil storage tanks are set to 7, 4, and 2 grid-side lengths, respectively.




(a) (b)

**FIGURE 24.** The optimal layout of LEE-Router. (a) The full view. (b) The pipe view.

distance calculated with formula (1); (b) search the path of current pipe with the proposed algorithms; (c) put the grids on the path of current pipe into the obstacle set; (d) restore the temporary-obstacle-grids marked in step (a) to free-grids.

### 3) RESULTS AND CONCLUSION

Based on the parameter settings and the pretreatment, the optimal CAD layouts of the fuel piping system obtained by different algorithms with sequence of P1→P4→P5→P3→P2→P6→P7→P8 are shown in Fig. 24 ∼ Fig. 28, and the statistical results are summarized in Table 7.

Where $L_{\emptyset=D}$ denotes the length of the pipeline with diameter value $D$, $B$ denotes the number of bends, $R$ denotes the number of branch points, $E$ denotes the grid energy of the pipe path, and $T$ denotes the time cost for routing pipe by a specified algorithm.

From the analysis of the results, the following conclusions can be drawn.

1) All of the proposed algorithms are able to generate the connective paths between the pipe nozzles, as well to meet some constraints that must be satisfied, e.g., arrange pipelines orthogonally with the rigid structures, maintain direction constraint on the nozzles, and avoid collision between pipes and layout environment.

2) Based on the framework of Branch-Pipe-Router and the single pipe routing algorithms, the optimal layouts of

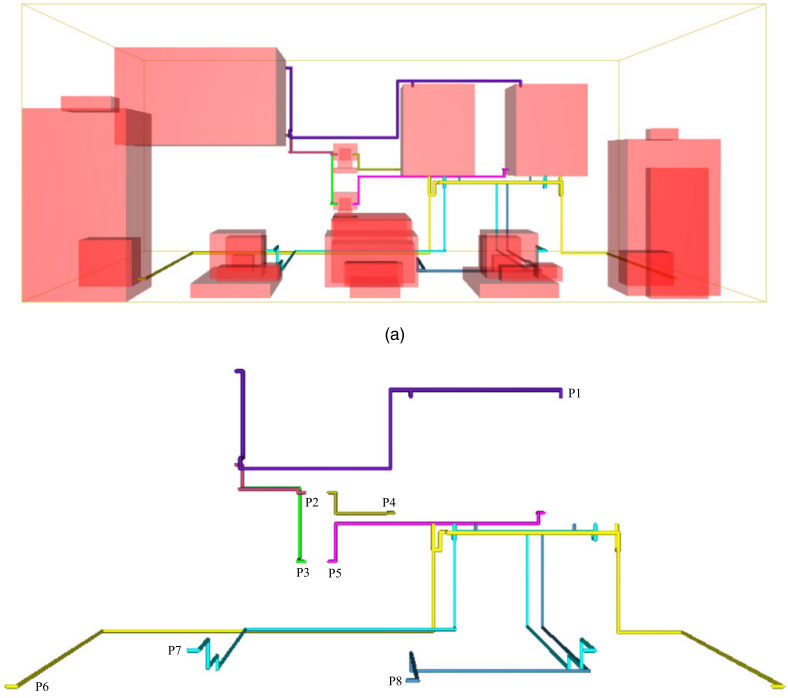



**FIGURE 25.** The optimal layout of A*-Router that considers the energy constraint. (a) The full view. (b) The pipe view.

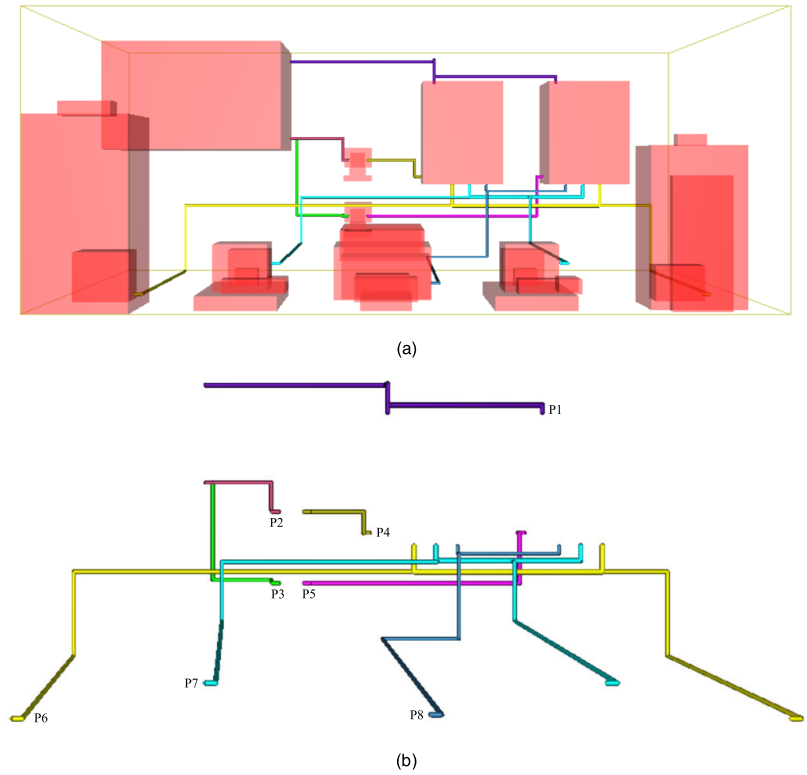



**FIGURE 26.** The optimal layout of A*-Router that ignores the energy constraint. (a) The full view. (b) The pipe view.

equal-diameter branch pipes (P1, P8) and unequal-diameter branch pipes (P6, P7) can be obtained. All the layouts are feasible, except for the layout obtained by A*-Router considering the energy constraint (Fig. 25). The reason has been explained in the pretreatment section. The number of branch points in each layout is 6.

3) LEE-Router takes path length as the optimization objective during the search, and the reduction of pipe

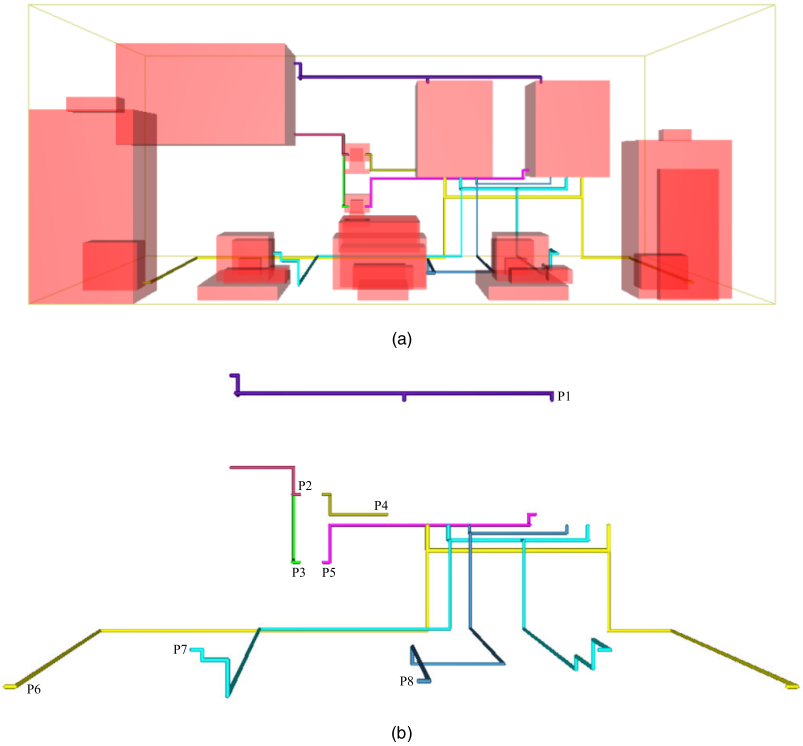



**FIGURE 27.** The optimal layout of GA-A*-Router that considers the energy constraint. (a) The full view. (b) The pipe view.

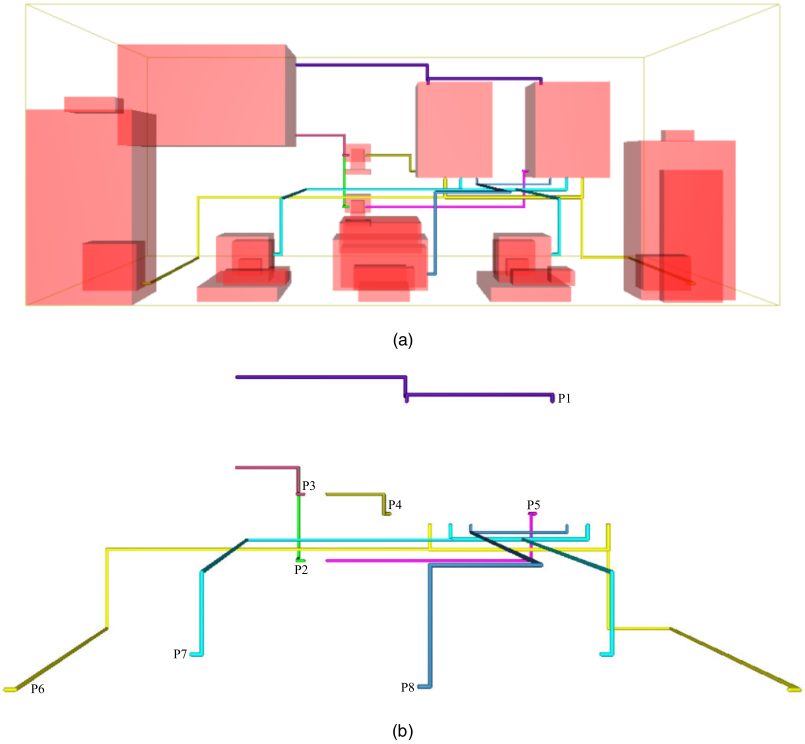



**FIGURE 28.** The optimal layout of GA-A*-Router that ignores the energy constraint. (a) The full view. (b) The pipe view.

bends depends on the backtracking process of path exploration. Therefore, the length of its layout is the shortest (58000), and the number of bends is at the medium level (42), while the path energy is high (18150), which means the pipe paths near the supports are less, as shown in Fig. 24.

4) When considering the energy constraint in A*-Router and GA-A*-Router, the energy values of their layouts

**TABLE 7. Routing results of the fuel piping system obtained by different algorithms.**

| Pipe | Objectives | Results of Lee-Router | Results of A*-Router (consider the energy constraint) | Results of A*-Router (ignore the energy constraint) | Results of GA-A*-Router (consider the energy constraint) | Results of GA-A*-Router (ignore the energy constraint) |
|---|---|---|---|---|---|---|
| P1 | $L_{\emptyset=60}$ (mm) | 5,700 | 8,000 | 5,700 | 5,700 | 5,700 |
| | $B$ | 3 | 7 | 3 | 4 | 3 |
| | $R$ | 1 | 1 | 1 | 1 | 1 |
| | $E$ | 1,625 | 900 | 1,775 | 1,300 | 1,625 |
| | $T$ (ms) | 109 | 47,922 | 2,266 | 46,437 | 44,828 |
| P2 | $L_{\emptyset=48}$ (mm) | 1,850 | 1,850 | 1,850 | 1,850 | 1,850 |
| | $B$ | 4 | 5 | 4 | 3 | 3 |
| | $E$ | 550 | 350 | 425 | 425 | 425 |
| | $T$ | 31 | 188 | 94 | 20,656 | 20,313 |
| P3 | $L_{\emptyset=48}$ (mm) | 2,900 | 2,900 | 2,900 | 2,900 | 2,900 |
| | $B$ | 4 | 5 | 3 | 3 | 3 |
| | $E$ | 875 | 400 | 875 | 475 | 475 |
| | $T$ (ms) | 63 | 203 | 78 | 23,656 | 20,297 |
| P4 | $L_{\emptyset=48}$ (mm) | 1,400 | 1,400 | 1,400 | 1,400 | 1,400 |
| | $B$ | 3 | 4 | 3 | 3 | 3 |
| | $E$ | 300 | 300 | 300 | 300 | 300 |
| | $T$ (ms) | 32 | 437 | 94 | 21,235 | 20,750 |
| P5 | $L_{\emptyset=48}$ (mm) | 4,000 | 4,000 | 4,000 | 4,000 | 4,000 |
| | $B$ | 5 | 5 | 3 | 5 | 3 |
| | $E$ | 600 | 600 | 1,625 | 600 | 1,625 |
| | $T$ (ms) | 187 | 1,000 | 78 | 20,485 | 21,235 |
| P6 | $L_{\emptyset=64}$ (mm) | 3,350 | 4,050 | 3,350 | 3,350 | 3,350 |
| | $L_{\emptyset=46}$ (mm) | 15,100 | 15,000 | 15,100 | 15,100 | 15,100 |
| | $B$ | 6 | 12 | 6 | 6 | 6 |
| | $R$ | 2 | 2 | 2 | 2 | 2 |
| | $E$ | 5,425 | 1,900 | 5,425 | 2,625 | 4,975 |
| | $T$ (ms) | 860 | 6,438 | 3,484 | 89,828 | 83,485 |
| P7 | $L_{\emptyset=62}$ (mm) | 2,450 | 2,850 | 2,450 | 2,450 | 2,450 |
| | $L_{\emptyset=44}$ (mm) | 12,950 | 14,950 | 12,950 | 14,750 | 12,950 |
| | $B$ | 10 | 19 | 8 | 15 | 7 |
| | $R$ | 2 | 2 | 2 | 2 | 2 |
| | $E$ | 6,800 | 1,800 | 6,800 | 2,475 | 7,000 |
| | $T$ (ms) | 937 | 5,985 | 2,672 | 84,109 | 81,610 |
| P8 | $L_{\emptyset=44}$ (mm) | 8,300 | 9,650 | 8,300 | 8,700 | 8,300 |
| | $B$ | 7 | 7 | 5 | 7 | 4 |
| | $R$ | 1 | 1 | 1 | 1 | 1 |
| | $E$ | 1,975 | 900 | 2,550 | 900 | 2,975 |
| | $T$ (ms) | 438 | 35,094 | 4,593 | 51,782 | 48,922 |
| Total | $L_{Total}$ (mm) | 58,000 | 64,650 | 58,000 | 60,200 | 58,000 |
| | $B_{Total}$ | 42 | 64 | 35 | 46 | 32 |
| | $R_{Total}$ | 6 | 6 | 6 | 6 | 6 |
| | $E_{Total}$ | 18,150 | 7,150 | 19,775 | 9,100 | 19,400 |
| | $T_{Total}$ (ms) | 2,657 | 97,267 | 13,359 | 358,188 | 341,440 |

(7150, 9100) are significantly lower than those obtained by other algorithms which ignore the energy constraint (18150, 19775, 19400). Low energy means that pipelines can be arranged closer to the equipment and the floor for installation. However, it should be pointed out that in this case, only the grids around the equipment and the floor are set with low energy value (0), and the grids in other locations have higher energy value (25). A*-Router considering energy constraint excessively pursues the low-energy layout, which leads to get arrangement with unsatisfactory length and bending number, as shown in Fig. 25. For this reason, it is not recommended to use A*-Router considering the energy constraints alone in the space with similar characteristics. In comparison, GA-A*-Router considering the energy constraint only generates intermediate connection points in the specified areas, thus, avoids excessive searching for the low-energy layouts, so its result has better overall performance, in particular, the pipes under the oil storage tanks and near the floors are compact for space saving, as shown in Fig. 27.

5) When ignoring the energy constraint in A*-Router and GA-A*-Router, the obtained layouts get high energy values (19775, 19400), but they are the best in path length and bend numbers. Both of them have the shortest path length (58000). GA-A*-Router has the least number of bends (32), and A*-Router takes the second place (35), as shown in Fig. 26 and Fig. 28. The two algorithms are suitable for use when the straightness of piping layout is desired.

6) GA-A*-Router employs the connection points to guide the generation and evolution of the chromosomes, so the obtained layouts show good parallel effect on pipeline P2 and P3, as shown in Fig. 27 and Fig. 28, while LEE-Router

and A*-Router only consider the optimization of the current routing pipe, so their results do not present the best parallel effect on P2 and P3, as shown in Fig. 24 ∼ Fig. 26.

7) For routing the fuel piping system, LEE-Router takes about 3 seconds; A*-Router takes about 13 seconds when ignoring the energy constraint, and about 97 seconds when considering the energy constraint; GA-A*-Router takes about 341 seconds when ignoring the energy constraint, and about 358 seconds when considering the energy constraint. The explanations for the computing performance are as follows: (a) GA-A*-Router is based on evolution process and it calls the single pipe routing algorithms repeatedly during the processes of population generation and evolution, so it costs more computation time than LEE-Router and A*-Router. (b) LEE-Router takes the shortest time because it can explore the paths from one specified grid to many other grids in one execution, while A*-Router only explores the path between two specified grids at a time, so for looking branch points in branch pipe, LEE-Router is superior to A*-Router in terms of time performance (Section III-E). (c) When the energy constraint is involved in A*-Router and GA-A*-Router, the energy objective ($E_{cost}$) in the evaluation function of A* algorithm will play a role, resulting in the exploration of more grids during search, Therefore, it is more time-consuming when A*-based algorithms considers the energy constraint, particularly, in a large layout space. Note that the chromosomes of GA-A*-Router are based on the proposed connection-point strategy, and the spatial distances between the adjacent connection points are relatively short. With this improvement, A* algorithm can find the sub-paths between connection points in less time. As a result, the impact of energy constraint on computation time of GA-A*-Router is not so obvious as that in A*-Router.

Currently, the piping algorithms still depend on user's input, e.g., the runtime parameters, the routing sequence of pipes and nozzles, the connection strategy of subbranches, the distribution of grid energy, as well as the extension distances for nozzles, which will affect the final results. By utilizing the algorithms, a designer could arrange pipelines by following steps: (a) select a proper algorithm according to the actual piping requirements; (b) generate the initial layout by using the algorithm and the user input; (c) adjust the parameters on the basis of feedback; (d) re-run the algorithm to improve the current layout; (e) repeat steps (c) and (d) until the stop criteria is met. Compared to the manual design which costs a large amount of time and energy, it is much easier to learn the characteristics of the routing algorithms and master the skills on parameter settings. The proposed algorithms can provide reasonable references in short time for the users. In general, only minor modifications are required for the practical application.

## V. CONCLUSION

In this article, four practical and effective pipe routing methods have been developed based on A* algorithm and GA. Firstly, the improved A* algorithm (A*-Router) is presented, in which the requirements such as path length, number of bends, and routing along supports are considered. Meanwhile, the algorithm framework using priority queue as Open table has been implemented. Compared with the maze algorithm, A*-Router is better in time performance and optimization quality. Then, the improved genetic algorithm (A*-GA-Router) is proposed. It initializes the chromosomes with A*-Router and connection-points strategy. Moreover, a novel repair operator is proposed and embedded in A*-GA-Router, which improves the performance of the new GA. Simulations show that A*-GA-Router outperforms A*-Router at optimization ability and layout diversity. Based on the connection-points strategy of A*-GA-Router, the algorithm for arranging pipes in parallel (Multi-Pipes-Router) is put forward. The chromosomes of current pipe are initialized with the connection points generated around the routed pipe, and then have been evolved with the iterative process of GA. Simulations show that Multi-Pipes-Router can produce ideal parallel layouts for the piping cases in different problem sizes. To cope with branch pipe routing, the Branch-Pipe-Router is proposed, in which the single pipe routing algorithms, i.e., A*-Router, GA-A*-Router, and LEE-Router, are embedded into the modified Steiner Tree framework to explore the branching points and the paths of subbranches. In the end, the simulation on the fuel piping system of a ship engine room shows that the proposed algorithms are able to provide satisfactory layouts within acceptable computation time.

Future work will focus on the follows: (a) convert the conventions and requirements of actual pipe design to the constraints utilized by routing algorithms; (b) investigate smart strategies for generating the routing sequence of pipes and nozzles; (c) verify and improve the algorithms by using more routing cases which cover the arrangements for internal circulation pipe and external circulation pipe; (d) Implement an intelligent ship piping system that utilizes the proposed routing algorithms.

## REFERENCES

[1] *Product Data Representation and Exchange—Part 217, Application Protocol: Ship Piping*, Standard 10303-217, SCRA, New York, NY, USA, 1996.
[2] J.-H. Park and R. L. Storch, "Pipe-routing algorithm development: Case study of a ship engine room design," *Expert Syst. Appl.*, vol. 23, no. 3, pp. 299–309, Oct. 2002.
[3] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Math.*, vol. 1, no. 1, pp. 269–271, Dec. 1959.
[4] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. SSC-4, no. 2, pp. 100–107, Jul. 1968.
[5] C. Y. Lee, "An algorithm for path connections and its applications," *IEEE Trans. Electron. Comput.*, vol. EC-10, no. 3, pp. 346–365, Sep. 1961.
[6] D. W. Hightower, "A solution to line routing problems on the continuous plane," in *Proc. Papers 65th Years Electron. Design Autom. 25 Years DAC*, 1988, pp. 1–24.
[7] J. Soukup, "Global router," in *Proc. 16th Design Autom. Conf.*, Jun. 1979, pp. 481–484.
[8] D. Zhu and J.-C. Latombe, "Pipe routing-path planning (with many constraints)," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 1991, pp. 1940–1947.
[9] H. Richert and G. Gruhn, "A numeric-heuristic system for plant wide pipe routing," *Comput. Chem. Eng.*, vol. 23, Supplement, pp. 735–738, Jun. 1999.

[10] T. Ito, "A genetic algorithm approach to piping route path planning," *J. Intell. Manuf.*, vol. 10, no. 1, pp. 103–114, Mar. 1999.
[11] S. Sandurkar and W. Chen, "GAPRUS—Genetic algorithms based pipe routing using tessellated objects," *Comput. Ind.*, vol. 38, no. 3, pp. 209–223, Apr. 1999.
[12] G. Thantulage, T. Kalganova, and W. A. C. Fernando, "A grid-based ant colony algorithm for automatic 3D hose routing," in *Proc. IEEE Int. Conf. Evol. Comput.*, Jul. 2006, pp. 48–55.
[13] Q. Liu and C. Wang, "A modified particle swarm optimizer for pipe route design," in *Proc. 11th IEEE Int. Conf. Comput. Sci. Eng. Workshops*, Jul. 2008, pp. 157–161.
[14] T. Ren, Z.-L. Zhu, G. M. Dimirovski, Z.-H. Gao, X.-H. Sun, and H. Yu, "A new pipe routing method for aero-engines based on genetic algorithm," *Proc. Inst. Mech. Eng., G, J. Aerosp. Eng.*, vol. 228, no. 3, pp. 424–434, Mar. 2014.
[15] Q. Liu and G. Jiao, "A pipe routing method considering vibration for aero-engine using Kriging model and NSGA-II," *IEEE Access*, vol. 6, pp. 6286–6292, 2018.
[16] Y. Qu, D. Jiang, G. Gao, and Y. Huo, "Pipe routing approach for aircraft engines based on ant colony optimization," *J. Aerosp. Eng.*, vol. 29, no. 3, May 2016, Art. no. 04015057.
[17] Q. Liu and C. Wang, "Pipe-assembly approach for aero-engines by modified particle swarm optimization," *Assem. Autom.*, vol. 30, no. 4, pp. 365–377, Sep. 2010.
[18] Q. Liu and C. Wang, "A discrete particle swarm optimization algorithm for rectilinear branch pipe routing," *Assem. Autom.*, vol. 31, no. 4, pp. 363–368, Sep. 2011.
[19] Q. Liu and C. Wang, "Multi-terminal pipe routing by Steiner minimal tree and particle swarm optimisation," *Enterprise Inf. Syst.*, vol. 6, no. 3, pp. 315–327, Aug. 2012.
[20] S. S. Kang, S. Myung, and S. H. Han, "A design expert system for auto-routing of ship pipes," *J. Ship Prod.*, vol. 15, no. 1, pp. 1–9, 1999.
[21] X.-Y. Shao, X.-Z. Chu, H.-B. Qiu, L. Gao, and J. Yan, "An expert system using rough sets theory for aided conceptual design of ship's engine room automation," *Expert Syst. Appl.*, vol. 36, no. 2, pp. 3223–3233, Mar. 2009.
[22] A. Asmara, "Pipe routing framework for detailed ship design," Ph.D. dissertation, Dept. Marine Transp. Technol., Univ. Delft Univ. Technol., Delft, The Netherlands, 2013.
[23] S.-H. Kim, W.-S. Ruy, and B. S. Jang, "The development of a practical pipe auto-routing system in a shipbuilding CAD environment using network optimization," *Int. J. Nav. Archit. Ocean Eng.*, vol. 5, no. 3, pp. 468–477, Sep. 2013.
[24] X.-N. Fan, Y. Lin, and Z.-H. Ji, "Ship pipe routing design using the ACO with iterative pheromone updating," *J. Ship Prod.*, vol. 23, no. 1, pp. 36–45, 2007.
[25] X.-N. Fan, Y. Lin, and Z.-S. Ji, "A variable length coding genetic algorithm to ship pipe path routing optimization in 3D space," (in Chinese), *Shipbuilding China*, vol. 48, no. 1, pp. 82–90, Mar. 2007.
[26] X.-N. Fan, Y. Lin, and Z.-S. Ji, "Multi ant colony cooperative coevolution for optimization of ship multi pipe parallel routing," (in Chinese), *J. Shanghai Jiaotong Univ.*, vol. 43, no. 2, pp. 193–197, 2009.
[27] W.-Y. Jiang, Y. Lin, M. Chen, and Y.-Y. Yu, "A co-evolutionary improved multi-ant colony optimization for ship multiple and branch pipe route design," *Ocean Eng.*, vol. 102, pp. 63–70, Jul. 2015.
[28] W.-Y. Jiang, Y. Lin, M. Chen, and Y.-Y. Yu, "An ant colony optimization-genetic algorithm approach for ship pipe route design," *Int. Shipbuilding Prog.*, vol. 61, nos. 3–4, pp. 163–183, 2014.
[29] Z.-R. Dong and Y. Lin, "A particle swarm optimization based approach for ship pipe route design," *Int. Shipbuilding Prog.*, vol. 63, nos. 1–2, pp. 59–84, Jan. 2017.
[30] Z. Dong and Y. Lin, "Ship pipe routing method based on genetic algorithm and cooperative coevolution," *J. Ship Prod. Design*, vol. 33, no. 2, pp. 122–134, May 2017.
[31] H.-T. Sui, W.-T. Niu, Y.-X. Niu, C.-K. Zhou, and W.-G. Gao, "Pipe-assembly approach for ships using modified NSGA-II algorithm," *Comput. Aided Drafting, Des. Manuf.*, vol. 26, no. 2, pp. 34–42, 2016.
[32] W. Niu, H. Sui, Y. Niu, K. Cai, and W. Gao, "Ship pipe routing design using NSGA-II and coevolutionary algorithm," *Math. Problems Eng.*, vol. 2016, Jan. 2016, Art. no. 7912863.
[33] H. Sui and W. Niu, "Branch-pipe-routing approach for ships using improved genetic algorithm," *Frontiers Mech. Eng.*, vol. 11, no. 3, pp. 316–323, Sep. 2016.
[34] Y.-L. Wang, C. Wang, F. Peng, C.-G. Jin, Y. Lin, and Z.-S. Ji, "Intelligent layout design of ship pipes based on genetic algorithm with human-computer cooperation," (in Chinese), *Shipbuilding China*, vol. 56, no. 1, pp. 196–202, Mar. 2015.
[35] Y.-L. Wang, Y.-Y. Yu, K. Li, X.-G. Zhao, and G. Guan, "A human-computer cooperation improved ant colony optimization for ship pipe route design," *Ocean Eng.*, vol. 150, pp. 12–20, Feb. 2018.
[36] J. Fan, M. Ma, and X.-G. Yang, "Research on automatic laying out for external pipeline for aeroengine," (in Chinese), *J. Mach. Des.*, vol. 20, no. 7, pp. 21–23, Jul. 2003.
[37] A. Asmara and U. Nienhuis, "Automatic piping system in ship," in *Proc. 5th Int. Conf. Comput. IT Appl.*, Maritime Industries, Leiden, The Netherlands, 2006, pp. 269–280.
[38] J. Wu, Y. Lin, Z.-S. Ji, and X.-N. Fan, "Optimal approach of ship branch pipe routing optimization based on co-evolutionary algorithm," (in Chinese), *Ship &Ocean Eng.*, vol. 37, no. 4, pp. 135–138, 2008.
[39] Z.-R. Dong and Y. Lin, "Method of ship routing based on co-evolution and parallel computing," (in Chinese), *J. Dalian Univ. Techno.*, vol. 56, no. 4, pp. 367–374, Jul. 2016.
[40] M.-X. Lin, K. Yuan, C.-Z. Shi, and Y.-T. Wang, "Path planning of mobile robot based on improved A* algorithm," in *Proc. 29th Chin. Control Decis. Conf. (CCDC)*, Chongqing, China, May 2017, pp. 3626–3632.
[41] J. Peng, Y. Huang, and G. Luo, "Robot path planning based on improved A* algorithm," *Cybern. Inf. Technol.*, vol. 15, no. 2, pp. 171–180, Jun. 2015.
[42] L. Niu and G.-B. Zhuo, "An improved real 3D A star algorithm for difficult path finding situation," in *Proc. Int. Arch. Photogramm., Remote Sens. Spatial Inf. Sci.*, Beijing, China, vol. 37, Jul. 2008, pp. 927–930.
[43] H.-C. Wu, J.-H. Liu, C.-T. Tang, L.-J. Xu, and J.-S. Liu, "Automatic pipe layout design and optimization method based on improved A* algorithm," (in Chinese), *Comput. Integr. Manuf. Syst.*, vol. 22, no. 4, pp. 945–954, 2016.
[44] C.-J. Li, Z.-P. Yin, and T. Xiong, "Study on the auto-routing method of electromechanical products based on A* algorithm," (in Chinese), *Sci. Technol. Eng.*, vol. 11, no. 7, pp. 1474–1479, 2011.
[45] M. L. Fredman and R. E. Tarjan, "Fibonacci heaps and their uses in improved network optimization algorithms," *J. ACM (JACM)*, vol. 34, no. 3, pp. 596–615, Jul. 1987.
[46] Z.-R. Dong, "A research on auto-routing methods and applications of ship piping," (in Chinese), Ph.D. dissertation, Dept. Comput. Sci. Technol., Univ. Dalian Univ. Technol., Dalian, China, 2017.

**ZONGRAN DONG** received the B.S. and M.S. degrees in computer science from the Shenyang University of Technology, China, in 2004 and 2007, respectively, and the Ph.D. degree in computer science from the Dalian University of Technology, China, in 2017. From April 2007 to April 2012, he was a Teacher with the Department of Computer Science and Technology, Dalian Neusoft University of Information, China. From April 2012 to August 2018, he was a Senior Engineer with the Rockwell Automation Dalian Software Research and Development Center, China. Since September 2018, he has been an Associate Professor with the School of Software, Dalian University of Foreign Languages, China. His research interests include ship pipe route design, optimization algorithms, and software engineering.

**XUANYI BIAN** received the B.S. degree in ocean engineering and technology and the M.S. degree in design and manufacture of ships and marine structures from the Jiangsu University of Science and Technology, in July 2015 and July 2018, respectively. He is currently pursuing the Ph.D. degree in design and manufacture of ships and marine structures with the Dalian University of Technology. His research interests include automatic ship pipe routing and intelligent ship.