International Conference on Computer Applications in Shipbuilding 2011 (ICCAS2011), Vol.3, pp.153—158.

An Automatic Piping Algorithm Including Elbows and Bends

Y Ando and **H Kimura,** Kyushu University, Japan

SUMMARY

The pipe arrangement has been enabled to be more efficient and economical by recent development of CAD(Computer-Aided Design). However, it is difficult to design a piping layout automatically because of many constraints. We propose an automatic routing method for simple pipes considering both elbows and bends. In practical piping layouts, there are many bends connecting straight eccentric pipes which have gaps within the pipes' diameter. However, no precedence automatic piping algorithm has been taken into account pipelines with such bends. In order to solve the piping design problem including bends, we consider it to a routing problem in a directed and weighted graph. The nodes in the proposed graph have state variables not only locations but directions of the pipe. In addition, the presented method has specifications that sizes of each cell decomposing a design space are not restricted. The efficiency of the proposed method is demonstrated through several experiments.

1. INTRODUCTION

The pipe arrangement in ships has been more efficient by recent progress of CAD. However that work is obliged to demand on experiences of designers because there are many regulations or requirements to be concerned. In this paper, we propose an automatic routing method which is more practical and includes bends that diagonally pass in the local design area. In our knowledge, no existing algorithm can cope with bends. In addition, our algorithm is quite different from previous works in a point of decomposing a design space into free size meshes.

Many previous works about an automatic piping system have been done in many approaches to solve the piping problem [1], [2], [3]. And some of these works, such as Ito [4], Park et al. [5], Asmara et al. [6], [7], and Paulo et al. [8], applied for Cell Decomposition approach. It is comprised to divide the design area into meshes and connect them from the start point to the goal point. There are two main advantages of this approach. The first is possible to apply maze solving algorithms to find solutions. In the maze algorithms, there exist algorithms that assure of finding optimum solutions such as the Dijkstra's method. The second is possible to set different cost values in each cell. With this scheme, the algorithm can draw pipelines at near the hull, while avoiding aisle areas as possible.

In previous works that used Cell Decomposing approach, the mesh size was restricted to be bigger than the pipe diameter. In this paper we propose a new algorithm of which mesh sizes are not restricted by the pipe diameter.

2. FORMULATION OF THE PIPING

In order to deal with the piping arrangement, some assumptions are given, as described below.

[Design space] It is assumed that a space to be designed is provided, and this space is in the shape of a box. All pipes are arranged in this space.

[A target pipeline] Any branches are not included in a pipeline. The pipes' diameter is invariable at any places on the pipeline. Proposed algorithm arranges pipelines one by one, while regarding arranged pipelines as obstacles.

[Direction of pipes] All pipes except bends are parallel to each axis of the design space. Where pipes make turn, manufactured elbows are used. These assumptions are basic configurations of the pipe arrangement from point of maintenance management.

[Start point, goal point, and pipes' diameter] Both start point and goal point include coordinates of the points and the vectors indicating the direction of the pipe. These points and the diameter of the pipe are provided in advance.

[Obstacles] Structures and equipments in ships are regarded as obstacles. The geometric information of obstacles existing in the design space is provided as triangles or boxes. It is not allow for arranged pipes to interfere with these obstacles.

[Aisle spaces] Aisle spaces are spaces for passages. It is assumed that pipes are not allowed to pass through these spaces unless routes make an extreme detour. The geometric information of aisle spaces is given in advance.

[Pipe rack areas] Let it be assumed that some spaces for pipe racks are prepared. The pipe rack is a spot for setting pipes. It is prefer that pipes go through this space unless routes are not so long. The geometric information of pipe rack areas is given in advance.

With these assumptions, the supposed algorithm searches parameters as described below.

[Piping routes] The piping route without branches is represented as lists of coordinates indicating the start point, the goal point, elbows, and bends.
The proposed algorithm arranges pipes with following design objectives:
(1) to minimize the total length of pipes,
(2) to minimize the number of elbows and bends,
(3) to avoid passing the aisle as possible,
(4) to pass through the pipe rack area as possible.

In order to regard the pipe arrangement as the single-purpose optimization, routing costs which are proportional to the total length of routes are provided. Moreover, an elbow cost and a bend cost are given in advance. The proposed algorithm searches optimal routes with minimized sum of these costs.

3. PIPING ALGORITHM

3.1 GRAPHING OF STRAIGHT PIPES AND ELBOWS

Our approach involves Cell Decomposition approach which is usually used some automatic piping systems. It consists of decomposing the design space into cells and regarding piping routes as lists of cells.

The algorithm divides the design space into meshes and regards intersection points of meshes as nodes. In previous works, the properties of nodes are only the coordinates of the point. However, our approach regards not only the coordinates but direction of the pipe as properties of nodes. With this approach, mesh sizes are not demand on the diameter of the pipe. Consequently the proposed algorithm can generate more practical designs.

The algorithm searches a piping route with minimum costs by regarding the pipe arrangement as a routing problem in a directed and weighted graph. This algorithm uses the Dijikstra's method during searching the pipe route. The Dijkstra's method is an algorithm that solves the shortest path problem in a directed graph with nonnegative edge weights. The Dijkstra's method guarantees a path with minimum costs between the start point and the goal point.

The Dijkstra's method is composed to four steps, as described below.
(1) Set current node and mark the other nodes as unvisited.
(2) Consider all its unvisited neighbours and calculate their distance for current node.
(3) The next current node will be the node with the lowest distance in the unvisited set.
(4) If goal node has been visited, finish. Otherwise, set the unvisited nodes with the smallest distance as the new current node and continue from step (2).

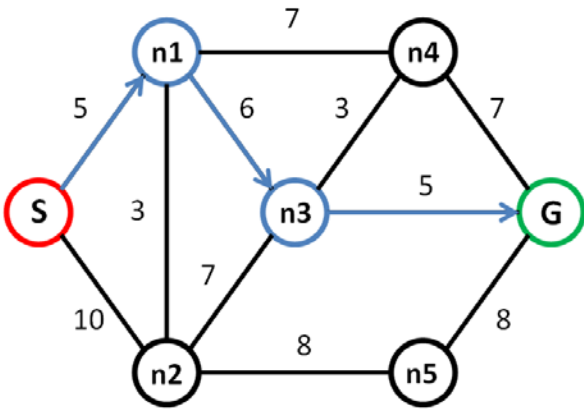

Figure 1: A shortest pass solved by the Dijkstra's method.

Figure 1 illustrates a simple example using the Dijkstra's method. The start node is node S and the goal node is node G. The root between node S and G is found by the Dijkstra's method repeating four steps as described. In case with n nodes and m edges, the running time for this method is $O(n^2)$ at worst, and this running time is the best possible complexity to solve the routing problem in a directed and weighted graph. The proposed algorithm can search solutions efficiently from this merit.

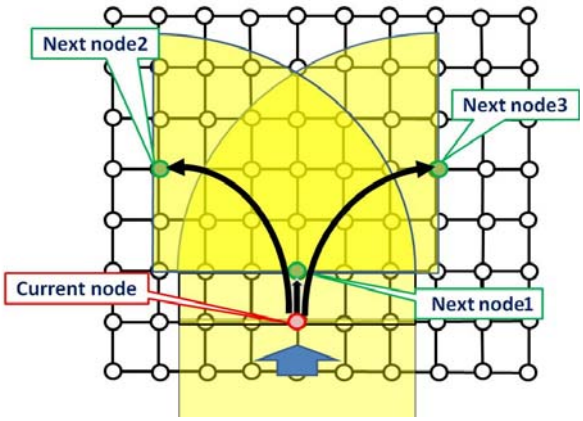

Figure 2: A gird partitioning network model in case of straight pipes and elbows.

Figure 2 shows examples of nodes transitions during search. The current node at Figure 2 is the node located a head of the pipe during search. The rectangle under the current node means the straight pipe, and a direction of the pipe is up. The diameter of the pipe is bigger than the width of mesh dividing the space. The coordinate and the direction of the current node are state variables in this approach.

In case of going straight ahead from the current node, the current node will transit to the next node1 at Figure 2 when there is not any interference with some obstacles. The place of the next node1 is next to the current node in a same direction of the node. When transiting to the next node1, some costs that are proportional to a length between the current node and the next node1 will be added.

In case of making turn to right or left, the current node will move to the next node2 or the next node3. These two nodes exist on the closest place defined by the diameter of straight pipes and the elbows' size. In these cases, an interference check will be done as same way as in case of going straight. When the interference check will be cleared, some costs corresponding with a price of elbows will be added.

The algorithm search a piping route from start point while repeating these processes at all linked nodes. The algorithm is able to search the route without a restriction of mesh sizes because of calculating a distance between the current node and the next node at each time.

3.2 GRAPHING OF BENDS

The proposed algorithm can search the piping route including bends. Figure 3 shows how to compose a bend into three primitives in order to deal with the bend easily.

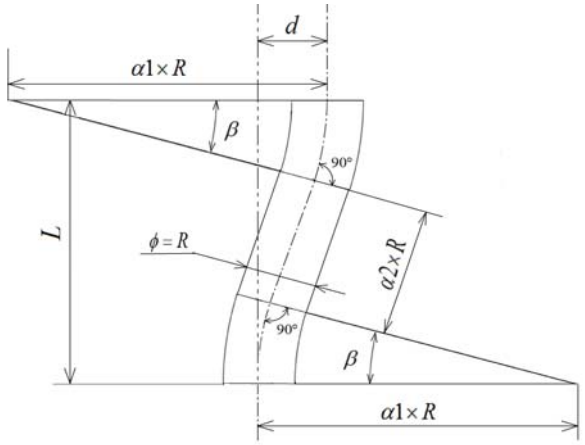

Figure 3: A bend part composed of three primitives.

When dealing with a bend, the degree of bending can be computed by

$$\beta = \arcsin\left\{\frac{d - 2\times\alpha 1\times R}{\sqrt{(2\times\alpha 1)^2 + (\alpha 2)^2}}\right\} + \arcsin\left\{\frac{2\times\alpha 1}{\sqrt{(2\times\alpha 1)^2 + (\alpha 2)^2}}\right\} \quad (1)$$

where d is the horizontal distance between the current point and the next point, R is the radius of the bend, $\alpha 1$ is the coefficient of a bending radius, and $\alpha 2$ is the coefficient of a straight part.

The minimum vertical length of the bend is computed by

$$L = (2\times\alpha 1\times R\times \sin\beta) + (\alpha 2\times R\times\cos\beta), \quad (2)$$

where β is the value calculated through equation (1). In our experiments, the value of $\alpha 1$ and $\alpha 2$ are set at 5, 0, respectively, and the value of d is limited in the diameter of the pipe.

In case of searching the next node through the bend, the algorithm searches nodes which exist in the place away from the length L calculated through equation (2). From these nodes, the algorithm chooses the closest node as the next node.

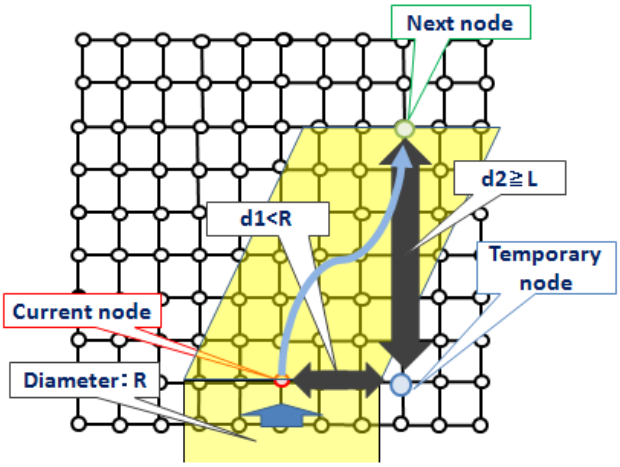

Figure 4: A grid partition network model in case of a bend.

Figure 4 shows an example of a node transition by using a bend. In Figure 4, the current node is the node located a head of the pipe during search, and the diameter of the pipe is larger than the size of meshes constructing the design area.

In case of using the bend which links the current node and the next node, the algorithm will search the temporary node as shown in Figure 4. The length between the current node and the temporary node is larger than the pipe diameter. Once the temporary node will be found, the algorithm will search the next node in Figure 4. The next node will places farther than the L calculated through equation (2) from the temporary node.

In this case, the interference check will be done as same way as in other cases, and the current node will link with the next node if the check will be cleared. This new link includes costs of the bend which corresponds with a price of it.

4. EXPERIMENTS

4.1 TEST CASE SETTING

To verify the useful of this algorithm, some computer simulations are conducted. As tests case, the design area which extends 16[m], 3[m], and 3[m] in each x, y, and z direction is set and this area includes ten obstacles. The start point and the goal point located at (0.5[m], 1.75[m], 1.5[m]) and (15.75[m], 1.5[m], 1.5[m]), respectively. The pipe will start at the start point to the positive direction on the x-axis, and end at the goal point from the negative direction on x-axis. This area is divided meshes and the dimension of the mesh is 0.25[m] with each directions.

In this demonstration, eight cases are set and the diameter of the piping routes without any ramifications is 0.2[m], 0.3[m], 0.4[m], 0.5[m], 0.6[m], 0.7[m], 0.8[m], 0.9[m] in each cases. The location of obstacles, the start point and the goal point do not change at all cases. The cost of the straight pipe is set 1.0 per 1[m] extension of the straight pipe and the cost of the elbow and the bend is set 0.1 and 0.3 respectively. Table 1 shows locations of each obstacle in this experiment.

Table 1: The detailed location of obstacles.

	x_min [m]	x_max [m]	y_min [m]	y_max [m]	z_min [m]	z_max [m]
Area	0	16	0	3	0	3
Obstacle1	3	4	0	3	0.8	1.1
Obstacle2	3	4	1.8	2.1	0	3
Obstacle3	7	8	0	3	1.9	2.2
Obstacle4	7	8	0.9	1.1	0	3
Obstacle5	11	12	0	3	1.9	2.2
Obstacle6	11	12	1.9	2.2	0	3
Obstacle7	5	6	1.3	1.6	0	1.7
Obstacle8	5.3	5.6	0	3	1.6	3
Obstacle9	9	10	1.3	1.6	1.6	3
Obstacle10	9.3	9.6	0	3	0	1.7

4.2 RESULT

As the computing environment, we used Windows-Vista, with Intel Core2 Duo 2.5Ghz and 2.00GB of memory. As the programming language, Java version 1.6 was used.

The results of the demonstration are displayed in Figure 5 through Figure 12. The yellow or orange parts are pipes and gray boxes are obstacles in Figure 5 through Figure 12. Table 2 shows the total cost, the number of elbows and bends, and the searching time.

Some bends, which are illustrated as orange parts in the pipe line, appeared in Figure 6 to Figure 9 and Figure 12. These solutions are new ones in terms of involving the bend. Without considering bends, these routes will make a detour.

Table 2; Features of obtained solutions with 0.25 [m] meshes of x-axis.

Diameter [m]	Num. of Elbows	Num. of Bends	Total Costs	Time [s]
0.2	9	0	17.9	1285
0.3	7	1	19.0	447
0.4	7	1	19.5	387
0.5	8	1	19.6	373
0.6	7	1	21.5	80
0.7	9	0	22.4	68
0.8	9	0	22.4	63
0.9	14	1	26.7	45

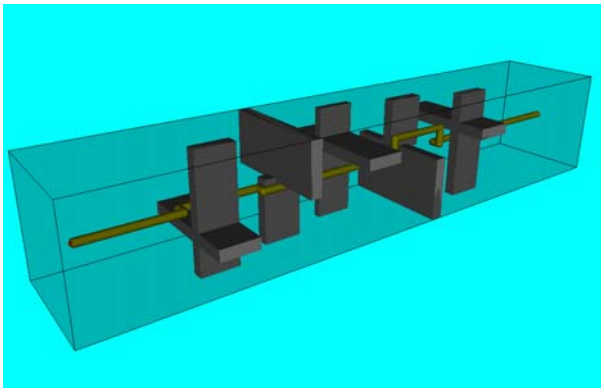

Figure 5: An obtained pipeline with 0.2 [m] diameter.

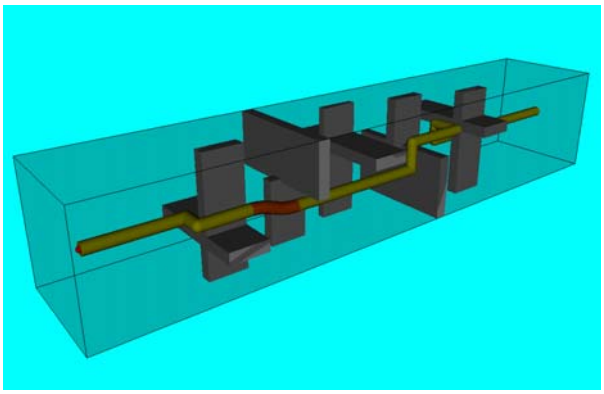

Figure 6: An obtained pipeline with 0.3 [m] diameter.

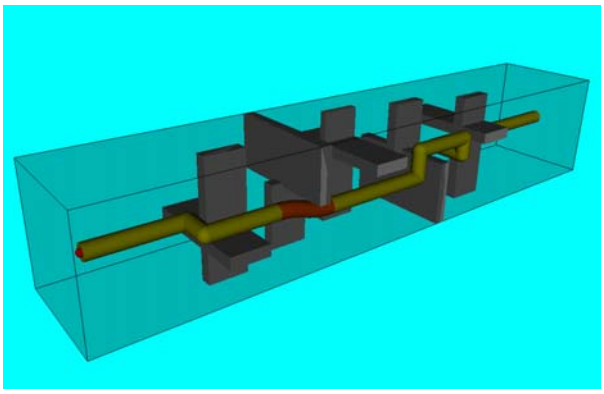

Figure 7: An obtained pipeline with 0.4 [m] diameter.

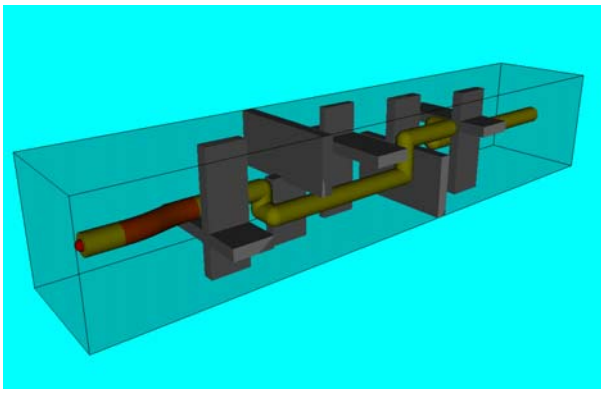

Figure 8: An obtained pipeline with 0.5 [m] diameter.

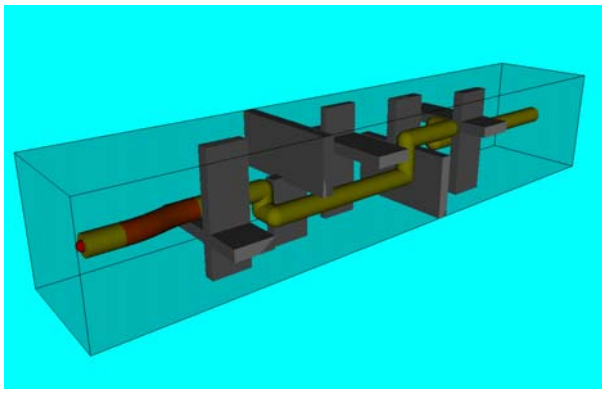

Figure 9: An obtained pipeline with 0.6 [m] diameter.

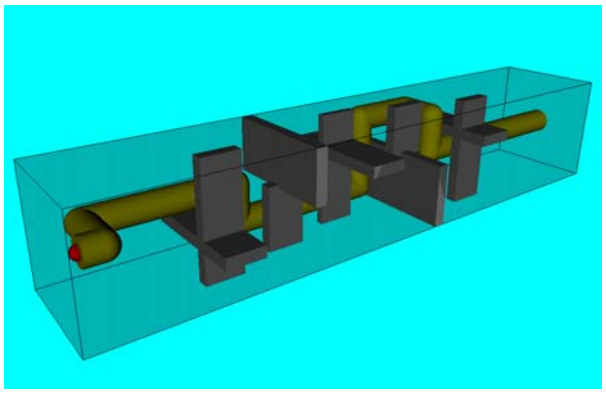

Figure 10: An obtained pipeline with 0.7 [m] diameter.

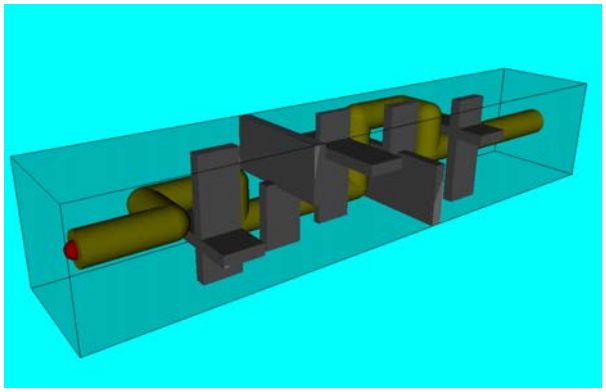

Figure 11: An obtained pipeline with 0.8 [m] diameter.

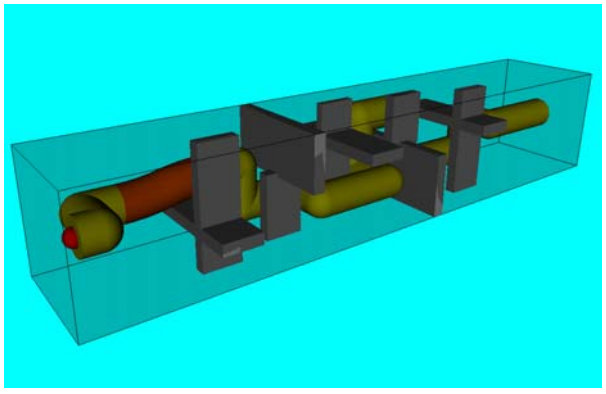

Figure 12: An obtained pipeline with 0.9 [m] diameter.

5. DISCUSSION

5.1 ABOUT THE TEST

In all cases, there are no interference between any of pipes and any of the obstacles and these routes are guaranteed the minimum costs because the algorithm used the Dijkstra's method. That means this algorithm succeeded finding the optimal piping route involving bends while keeping restriction with each pipe diameter.

As shown in Table 2, the smaller the diameter is, the longer the searching time takes. Therefore, it is conceivable that the decrease of the diameter causes the increase of the searching area. This increase of the searching time can be prevented by an improvement of the algorithm.

The Dijikstra's method is classed as the breadth-first search. But this style of searching needs huge memories to find a solution because this method searches all nodes neighbouring the searched node.

In concrete terms, we plan to use the A* search which is improved version of the Dijkstra's method as future plans.

5.2 ABOUT MESH SIZES

To investigate the suitable mesh sizes, we demonstrated other test case that was changed the mesh size only in x-axis to 0.5[m]. Table 3 shows the result of the demonstration.

Compared with Table 2 and Table 3, obtained solutions in both cases are the same. However, it was confirmed that the searching time was improved significantly. Especially in case of the diameter 2.0[m], the searching time changed into less than twenty percent of it.

This result means that the suitable setting of the mesh size can reduce the searching time dramatically because it reduces the number of nodes to be dealt by the algorithm. In order to find the suitable value of the mesh size, we need more experiments in diverse case setting.

Table 3: Features of obtained solutions in 0.5[m] meshes of x-axis

Diameter [m]	Num. of Elbows	Num. of Bends	Total Costs	Time [s]
0.2	9	0	17.9	213
0.3	7	1	19.0	146
0.4	7	1	19.5	126
0.5	8	1	19.6	126
0.6	7	1	21.5	24
0.7	9	0	22.4	23
0.8	9	0	22.4	22
0.9	14	1	26.7	12

5.3 ABOUT OTHER TEST CASE

In other case setting, it was confirmed that an obtained pipeline interfered with itself as shown in Figure 13. It is impossible to implement this solution. Therefore, the algorithm needs to improve in making sure to reject such solutions interfered with itself.

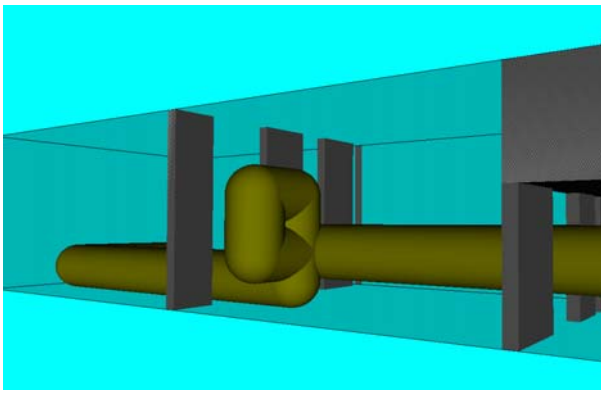

Figure 13: An obtained self interfered route.

6. CONCLUSIONS

This paper presents an automatic piping algorithm including not only elbows but bends for full automation of the piping route design. The proposed algorithm uses the Dijikstra's method to minimize the cost of the route. Therefore the proposed algorithm ensures that the obtained route is the optimum.

This algorithm also has special features that the size of meshes dividing the design area is free. In previous works, it is restricted that mesh sizes must be larger than the pipe diameter when that research applies for Cell Decomposition approach. This feature leads to obtained solutions more practical.

In addition, the proposed algorithm is also able to search piping routes while involving bends on a pipeline. It can help to generate more optimal and efficient solutions because practical pipelines usually involve bends.

In order to improve the algorithm, we revolve about the many practical aspects of the piping routes such as searching time, the pipe rack area, the aisle area, the order of pipes to set, and constraints of piping constructions.

7. ACKNOWLEDGEMENTS

This paper is the result of work sponsored by Japan Society for the Promotion of Science (JSPS) Grants-in-Aid for Scientific Research (B) 23360388. This support is gratefully acknowledged.

8. REFERENCES

1. W. CHEN., 'Gaprus-Genetic Algorithm Based Pipe Routing Using Tessellated Objects', *Journal of Computers in Industry*, 1998.

2. S. IKEHIRA, H. KAJIWAEA, and H. KIMURA, 'Automatic Design for Pipe Arrangement using Multi-objective Genetic Algorithm', *International Conference on Computer Applications in Shipbuilding (ICCAS)*, 2005.

3. H. KIMURA, and S. IKEHIRA, 'Automatic Design for Pipe Arrangement Considering Valve Operationality, *International Conference on Computer Applications in Shipbuilding (ICCAS)*, 2009.

4. T. ITO and S. FUKUDA, 'A genetic algorithm approach to piping route path planning', *Journal of Intelligent Manufacturing*, 1999.

5. J. H. PARK, and R. L. STORCH, 'Pipe-routing Algorithm Development: Case Study of a Ship Engine Room Design', *Journal of Expert Systems with Application*, 2002.

6. ASMARA, A. and U. NIENHUIS 'Automatic Piping System in Ship', *International Conference on Computer and IT Application (COMPIT)*, 2006.

7. ASMARA, A. and U. NIENHUIS 'Automatic piping system Implementation', *International Conference on Computer and IT Application (COMPIT)*, 2007.

8. T. M. PAULO, and V. J. A. S. LOBO, 'A Tool for Automatic Routing of Auxiliary Circuits in Ships', *Portuguese Conference on Artificial Intelligence (EPIA)*, 2009.

9. AUTHORS BIOGRAPHY

Yuto Ando holds the current position of a candidate of M. A. in Department of Maritime Engineering, Graduate School of Engineering, Kyushu University.

Hajime Kimura holds the current position of Associate Professor at Department of Maritime Engineering, Kyushu University. His previous experience includes optimization in shipbuilding, etc.