# A tool for automatic routing of auxiliary circuits in ships

Paulo Triunfante Martins[1], Victor J.A.S. Lobo [1,2]
[1] Portuguese Naval Academy
[2] ISEGI - Universidade Nova de Lisboa
vlobo@isegi.unl.pt

**Abstract.** This work proposes a method and software for the automatic layout design of auxiliary circuits within a compartment of a ship, including pipes and cable trays. This task is normally part of the detailed design phase and so it is expected that the ship's structure, general arrangement and equipment layout has already been decided. Accordingly, knowing the start and goal points for each circuit path, all of them are routed in a 3D space taking into consideration, obstacles, ramifications and several kinds of constraints. The method is implemented using routing algorithms and genetic algorithms in order to find the best possible group of paths considering all the above.

**Keywords:** Genetic Algorithms, Piping, Routing.

## 1 Introduction

The ship design and construction process goes through several stages, starting by its concept design, basic design, detail design and design for production. Typically, it is on the basic design stage that the ship structure, piping and electrical circuits are defined, and the various equipments are selected. Then, in the detail design stage, everything has to come together in a unique design, taking into account the specific set of solutions found.

Amongst the various tasks of the detailed design stage, the pipe layout design (PLD) is one of the more time-consuming and prone to error, and thus can originate more reworks. This comes to be even more important for medium size ships, such as small tankers, research and military vessels, due to lack of space. In fact, bad routing can affect the ship's local structure, as well as her operation and maintenance.

This problem is similar to the routing problem in circuit design, but there are significant differences: the PLD, unlike circuit design, is truly 3-dimensional, and the type of restrictions and desirable characteristics are quite different.

Nowadays, there are several software tools available, that integrate the design and manufacturing systems. In most cases, they have a design module for 3D layout, piping and HVAC (heating, ventilation and air conditioning), complemented with a tool for checking collisions between different objects.

However, we do not know any tool that provides a fully automated system to solve the PLD. This problem consists on finding the layout of pipe and cable routes minimizing the length of the path connecting two or more objects (equipments), using only one path or a main path with ramifications (branch handling), while avoiding obstacles such as ships structure, equipments and other circuits, in accordance with several shape and location constraints (maintenance, ergonomics, structural integrity, etc). Fig. 1 shows the pipe layout of a vessel, as taken from NUPAS Cadmatic home page (http://www.nupas-cadmatic.com/)[1].

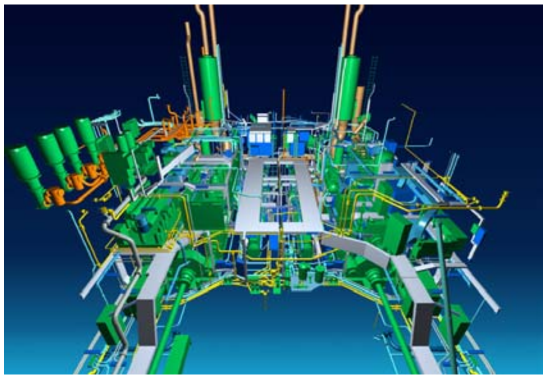

**Fig. 1.** Pipe layout example (taken from NUPAS Cadmatic internet home page).

Without such tool, the outcome of the overall process relies on the designer's knowledge and experience. Further, it is not possible for any person to define all pipe and cable routes at the same time, which raises integration and optimization problems. This is in fact a serious problem for a large number of new constructions.

This text describes a pipe and cable routing tool that can be used in the detail design stage in order to reduce man-hours or during basic design stage to identify potential problems that usually are only detected during construction. The aims of such tools have been defined previously by several authors [2] as follows:

(a) to minimize user input and decision;
(b) to make the system easy to use;
(c) to incorporate both pipe and cable routes;
(d) to be used in real shipyard design process.

## 2. Literature review

The need for a tool to help design the piping layout has been well-known for a long time. Back in the 80's decade shipyards usually built small models where straws were placed to check for collisions between different system components. With the 90's the first software tools that provide 3D visualization and collision check between objects started to appear, integrating a large amount of corporate knowledge and a way to automatically produce material lists and several documentation such as isometric drawings for production. Nevertheless, this is a different problem from the one this paper is concerned with, since no commercial fully automated system to support the designers in the routing selection of the different circuits is known to exist.

Some of the first research on PLD, as defined previously, was dedicated to the design of power plants and chemical refineries, as well as ships and submarines, though the problems are slightly different. During the last two decades several studies have been presented about this subject, trying to solve 2D and 3D problems, with and without pipe ramifications, using different algorithms and constraints.

Asmara and Nienhuis [3], divide the approaches into: roadmap search, cell decomposition approaches, potential field methods and mathematical programming methods. Probably, the most common one is the cell decomposition approach, consisting in the workspace division into different cells, complemented with mathematical programming methods, which deal with the path layout as a standard optimization problem with constraints.

This approach was followed by Zhu and Latombe [4], where the 2D and 3D problems were solved considering that each path had only one start point and one goal, though several paths had to coexist in the same workspace. In a first stage, the workspace was divided into rectangular cells taking into consideration existing obstacles. Afterwards, a search for the shortest paths connecting all start and goal points was performed using the A* (Branch and Bound) algorithm [5]. The order by which the paths are calculated is critical, and some of branches of A* will fail to find any solution at all. In order to solve this problem and to implement shape and location constraints, a path evaluation technique was implemented to decide the order in which the different routes should be defined, and then obtain sequentially a possible solution for all paths.

Ito [6] uses genetic algorithms to define and evaluate possible paths in a uniform cell decomposed space. His procedure, though containing many constraints, introduces the spatial potential energy concept, attributing a score to each cell depending on whether the path is intended to go through it. As an example, if it is established that pipes should go along walls, a lower value of potential energy is attributed to the cells near them, while the cells containing objects have a very high potential energy. In the end of the process, the possible paths are evaluated using a fitness function, choosing the solution where the sum of the potential energy for all routes in the problem is the smallest, which in fact already includes the shortest path requirement.

Park and Storch [7] differ from the previous references by proposing a cell decomposition method of non-uniform cells, whose shape and size is defined by grouping different pipes into common routes and considering pipe ramifications. This idea stems from the analysis of fabrication costs and operability, which are also used in the evaluating process of candidate paths, implemented by a decision tree.

More recently, [3, 8] presented the DelftPipe tool that aims to solve the PDL for pipes in a ship. The tool consists of an interface with commercial CAD systems, a pipe routing tool and an optimizer tool. As far as the pipe routing tool is concerned, it starts by implementing a non-uniform cell decomposition method, and then implements Djikkstra's shortest path algorithm to select the different candidate routes taking into account pipe ramifications and the order in which each circuit is routed. The third stage uses the discrete particle swarm optimization algorithm to find a solution that complies with the performance criteria of reducing the path length and number of bends, as well as the standard criteria imposed by international rules or by the rules of the classification society according to which it is being built.

## 3. AISROUTE – decision support tool for pipe and cabling layout design

The AISROUTE is the decision support tool that aims to help the designer in the task of routing pipe circuits and electrical cabling, free from collisions, along a new vessel, once its general arrangement, structure and equipments locations have been defined.

This tool was built using Matlab and is similar to the one presented by the Delft University group, in the sense that it is modular; however the algorithms and solutions found are quite different from the ones presented in [3, 8].

Figure 2 shows the decision support tool block diagram, where all the different stages of the process can be seen:

(1) The "User Interface" imports the geometry from a commercial CAD program, where the different objects (equipments and structure) are placed. This action defines the workspace and some of the location constraints to paths;

(2) Next, the designer must specify all start and goal points (requirements) for the different paths, including pipe, cable routing and accessibility paths. On the other hand, for each path the different constraints (shape and location) should also be set, in order for them to be implemented afterwards;

(3) Routing begins by finding several "shortest path" solutions that fulfill the start and goal points' requirements and the workspace constraints, by changing the order of the paths to define first;

(4) The different solutions are then used as the initial population for a genetic algorithm implementation that evaluates each solution as far as all requirements and constraints are met. This evaluation is done by a fitness function that attributes penalties when any constraint is not fulfilled and makes use of the spatial energy concept of Ito [6] to reward the best paths;

(5) Finally, once a good solution is found, the "User interface" allows for its visualization and its export.

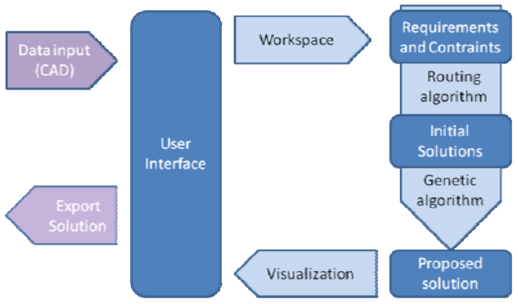

**Fig. 2.** AISROUTE block diagram

## 3.1 User Interface and workspace definition

The user interface allows the user to introduce the necessary data for routing, including the workspace geometry where the paths have to be defined, as well as their requirements (start and goal points) and different constraints. It was built in a windows environment using Matlab and Java objects, and it is able to import ASCII STL files that almost all CAD software is able to export.

The first step to work with AISROUTE is to define the space geometry of a "compartment" containing all obstacles (equipment and structure), that will stand for the location restrictions for the different paths. Each one of the obstacle must then be imported from CAD software. Figure 3 shows the user interface window, where some solids and three routes have been added.

Internally, the space is decomposed into cubic cells, whose faces have the same area as the smallest section area of the circuits to be routed. The coordinates of each cell form a matrix of X × Y × Z (longitudinal/ transverse/ vertical positions). Afterwards, when the obstacles are imported, the 3D matrix cells related to their position are marked as occupied.

This matrix is named the workspace, and it will be the platform over which the routing process is going to take place. During this process, the routes connecting the different starts and goals are searched through the matrix elements and the ones selected as part of each path are filled up. Once the process is completed, a new solution matrix, that includes not only the workspace but also the routes, is found. This procedure is going to be described in the next chapters.

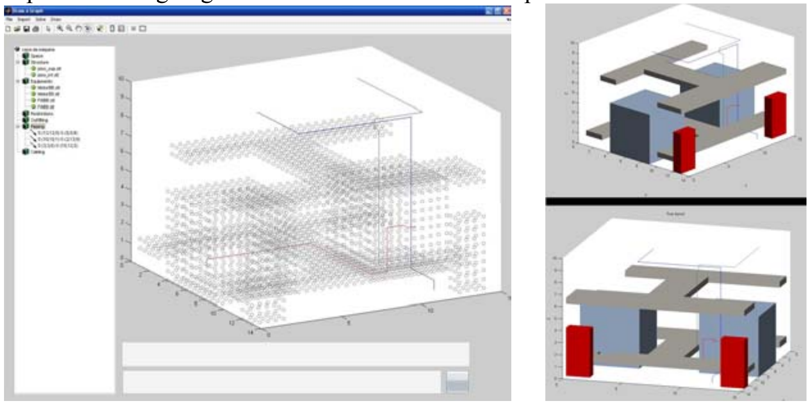

**Fig. 3.** AISROUTE user interface window (a), and visualization window (b)

Though this cell-decomposition procedure produces a large number of cells and causes an extra computer effort, these disadvantages may be overlooked in comparison with the advantages that they bring, namely:

(a) there is a direct relation between the spatial coordinates and the cells of the workspace and solution matrices;

(b) obstacles positions are marked in the workspace matrix and so the routes found will be collision free without need for subsequent collision check;

(c) it enables the use of maze solving algorithms for circuits' routing with and without ramification;
(d) it enables the implementation of different kinds of constraints and to select areas where there are advantages for the paths to go through;

### 3.2 Requirements and constraints analysis

Once the workspace is known there is the need to identify the start and goal points for each route, to identify the cells which can't be crossed by the path, and good areas for the path to go through. In other words, it is necessary to define the requirements and constraints for each path.

The constraints can be divided according to their nature into shape and location constraints [4], but to be able to implement them they were grouped into three different types:

(a) Type 1 – common location and shape constraints to all circuits, related to the obstacles of the workspace, including other paths, and to the fact that only 90º curves are allowed by the algorithm;
(b) Type 2 – shape and location constraints that exist only for a specific route, such as the case of gravity flow pipes, or the impossibility of going through some areas of the workspace (e.g. cables running through the floor);
(c) Type 3 – location constraints originated by the proximity to other routes. These ones are quite difficult to address since they are dynamic, and dependent upon previous path choices. As an example, it is intended that most pipes run together to be able to use common supports, however there are fluid systems that have to keep some distance between them due to heat sensibility, or electromagnetic interference between electric cables.

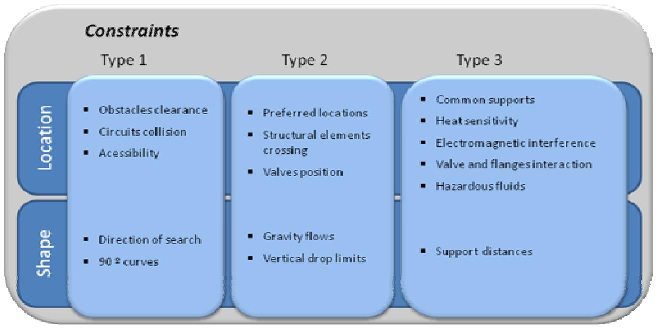

**Fig. 4**. Different kinds of constraints divided by their nature and type

The type 1 constraints will be present in all stages of the process, i.e. in both routing algorithm and genetic algorithm applications. On the contrary, the type two and three constraints are only going to be considered in the genetic algorithm implementation.

As far as the type two constraints are concerned, they are dependent upon the workspace and the obstacles within and though some are pre-defined within the program, they can be easily changed or new ones can be set by the user.

Nevertheless, the quality of the routing process is dependent upon the type three constraints. These are the ones that will allow for an easier operation of the ship and a more reasonable layout, though they are the most difficult to implement and to define. Unfortunately these are also the ones that are not yet fully developed in this work. Some of the constraints that should be considered are shown in **Fig. 4**.

### 3.3 Routing implementation using shortest path algorithms

The first approach to finding a good layout for all circuits is made using a shortest path algorithm presented by Lee [9] who, in its definition, is able to deal with the type one constraints.

To explain it briefly, it starts by flooding the neighbor cells of the start point in all six main directions (3 dimensional) up to when the goal point is reached, taking into account if the cells are already occupied by any obstacle. Then, the path is backtracked to the start point following a pre-defined search pattern. Although this method requires a high level of computation time, it was selected because it is easily changed to cope with different pipe/ cable tray sections (more than one cell) and main path ramifications. Additionally, the method always finds the shortest path, though it can find more than one, as presented in figure 5 (a).

The algorithm change to search for routes of circuits with several sections' shapes and sizes is trivial, once it is considered that to belong to the path, not only the target cell has to be free, but also her neighbors.

To implement the ramification case, it was assumed that besides the two points to be connected, let's say S1-G1, there is an extra one G2 that will be considered as a new start point. From G2 the flooding process is done up to the time when any of the cells of the path S1-G1 is reached (figure 5(b)).

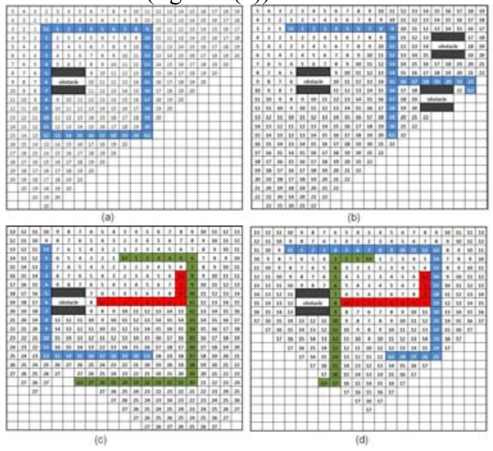

**Fig. 5.** Lee Algorithm routing examples for two similar shortest paths (a), path ramification (b), and different solutions for multiple routing dependent upon the order of paths (c) and (d)

In the figure 5 (c) and 5(d) examples, it can also be seen that when routing more than one circuit (let say S1-G1, S2-G2, S3-G3 – blue, red, green), there are several different path combinations that meet all requirements and that are found using the same Lee algorithm but changing the order in which the path search is done. As an example the paths found when starting to route S1-G1 and S2-G2 (figure 5(c)) are very different from the ones found starting by S3-G3 (figure 5(d)).

This is the way how the first set of solution matrices are found, taking in account for the definition of the routing order the following:

(a) the first route that should be considered is the "accessibility" that connects all different equipments/ objects defined in the workspace;
(b) the paths with larger sections should be routed first, in order to have less curves and a more basic routing style [7];

### 3.4 Genetic algorithm implementation

The genetic algorithm is used to evaluate an initial set of solutions found by the previous step and then to change them in accordance with the types two and three constraints. Their implementation in this area, as mentioned before, was first presented by Ito [6].

Genetic algorithms are inspired by the principles of the evolution of species, as described in [10] or [11]. The procedure begins by selecting a set of possible solutions named parents, whose characteristics are their chromosomes. These solutions are then used to obtain a group of children (new set of possible solutions), in an iterative form, using techniques called reproduction, which may follow three different processes:

(a) elitism – the best parents chromosomes are fully transmitted to their children;
(b) crossover – a new child is obtained mixing the chromosomes of two parents, using some of the genes of each parent;
(c) mutation – change of some of the genes of one parent in a repeated way through the iteration process.

The evaluation of each parent/ child is done by a fitness function that incorporates all requirements and constraints, using a penalty system for the ones that do not complete the requirements or do not fulfill the constraints. The aim is to have the less penalties as possible, i.e. the smallest fitness value.

In this work, each 3D solution matrix is considered to be the chromosomes with each matrix element (corresponding to a cell) being the gene that can be occupied or not. Accordingly the initial set of solution matrices found using the shortest path algorithm by changing the routing order are the first parents (initial population).

The process starts by attributing a spatial potential energy value, dependent upon the constraints and the circuit that is being routed, to each one of the matrix elements/ cells/ genes [6]. Afterwards, using the reproduction techniques new solution candidates are found and evaluated, aiming to find the paths with the lowest potential energy, found by adding up all values of the cells belonging to the path. Further, if the start and goal point are unconnected an extra very large value of energy is added (penalty).

As an example, figure 6 represents this procedure in 2D. Let it be assumed that the ramified path (blue path) has already been defined. It is intended that the red path does not cross any obstacle (type one constrain - black), stays away from the space limits (type two constraint - brown) and its path should be close to the blue path to use the same supports (type three constraints – light blue). A possible potential energy distribution is presented in figure 6(b) which may result in the solution presented in figure 6(a).

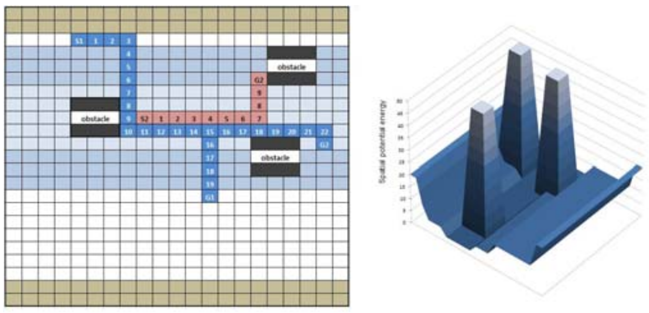

Fig. 6. Use of spatial potential energy concept in the routing of two pipes

So far, a number of preliminary tests have been performed on different configurations of compartments. The results have been very satisfactory, achieving good quality results in reasonable time. We are currently applying this software in a real case, and evaluating the results against the standard available solutions.

## 4. Conclusion

A method for the pipe and cable tray layout design was presented using a combination of routing algorithms [9] and genetic algorithms [10, 11], making use of the spatial potential energy concept [6].

This is thought to be a reasonable method as long as the means to define the potential energy values for each cell depending upon the circuit are well defined. In fact, this is the most difficult and yet less completed part of the work.

All in all, a decision support tool the help designers in defining the auxiliary circuits within a compartment  is working and available to be tested, waiting to find all unavoidable errors in the implementation.

## References

1. NUPAS, Nupas Cadmatic home page - http://www.nupas-cadmatic.com/. 2009, Numeriek Centrum Groningen B.V. and Cadmatic Oy.
2. Kang, S., S. Myung, and S. Han, A Design expert system for auto-routing of ship pipes. Journal of Ship Production, 1999. **15**(1).

3. Asmara, A. and U. Nienhuis. Automatic piping system in ship. in COMPIT - 5th Int. Conf. on Computer and IT Application Mar. Ind. 2006. Leiden.
4. Zhu, D. and J. Latombe. Pipe Routing = Path Planning (with many constraints). in IEEE International Conference on Robotics and Automation. 1991. Sacramento, California.
5. Hart, P.E., N.J. Nilsson, and B. Raphael, Correction to "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". SIGART Newsletter, 1972(37): p. 28-29.
6. Ito, T., A genetic algorithm approach to piping route path planning. Journal of Intelligent Manufacturing, 1999. **10**(1).
7. Storch, R. and J.-H. Park. Pipe-routing expert system. in International Conference on Computer Applications in Shipbuilding. 2002
8. Asmara, A. and U. Nienhuis. Automatic Piping System Implementation: A Real Case. in COMPIT - 6th Int. Conf. on Computer and IT Application Mar. Ind. 2007.
9. Lee, C.Y., An Algorithm for Path Connections and Its Applications. IRE Transactions on Electronic Computers, 1961. **10**(2): p. 364-365.
10. Holland, J., Adaptation in Natural and Artificial Systems: an introductory analysis with applications to biology, control, and artificial intelligence. 1975, Ann Arbor: University of Michigan Press.
11. Fogel, D.B., Evolutionary Computation : Towards a New Philosophy of Machine Intelligence. 2nd ed. 1999: IEEE Press.