An intelligent system for routing automation

Christian Van der Velden\textsuperscript{a}, Cees Bil\textsuperscript{a}, Xinghuo Yu\textsuperscript{b}, Adrian Smith\textsuperscript{c}

\textsuperscript{a}School of Aerospace Mechanical and Manufacturing Engineering, RMIT University, Melbourne, Australia
\textsuperscript{b}School of Electrical and Computer Engineering, RMIT University, Melbourne, Australia
\textsuperscript{c}GKN Aerospace Engineering Services Pty. Ltd., Melbourne, Australia

Abstract

We present an outline of an intelligent routing environment developed for design automation of aircraft electrical wiring harness and pipe routing. The system interfaces with domain experts for capture of engineering knowledge and rules. System users specify only a few parameters to define individual routing problems. Geometry obstacles are specified using a discrete Finite Element (FE) mesh. Resultant paths are output in IGES form as CAD-readable geometry, and as an FE mesh consisting of geometry, routed path and knowledge layers. The knowledge layer provides detail of the rules and knowledge implemented throughout the routing process.

Keywords: Intelligent Systems, Knowledge Based Engineering, Routing, Computer Aided Design (CAD), Design Automation

1 Introduction

Increasing complexity of aircraft electrical systems has led to an increase in the number and size of electrical harnesses required to connect various equipment throughout the airframe. Electrical harnesses routing is a complex task with hundreds of rules and best practices to be satisfied. Wiring looms are typically comprised of thousands of cables which are generally manually routed, with engineers using personal knowledge and experience of the problem domain. Software tools exist which assist designers in creating digital models of harnesses, however, the exact path to be taken is still determined by human operators.

In large scale projects, subsystem design including wiring systems is often conducted alongside principle structural design. Therefore changes in structure which occur during the development cycle can impact on subsystem design and placement, requiring rework for affected harnesses. This can lead to lengthy delays in the aircraft development, as has been seen recently with the Airbus A380 wiring systems [1].

This paper discusses the development of an intelligent routing system for automating electrical harness and pipe routing. The software outputs a CAD model and FE mesh of a path connecting source and target terminals through a set of obstacles, while satisfying design rules and best practices.

2 Problem Background

The Federal Airworthiness Requirements for transport category aircraft (FAR-25) is the main document governing civil aircraft design and certification. Currently FAR-25 does not have a section solely addressing wiring design and routing. Instead governance is provided in numerous subsections, making it difficult to single out and implement all applicable rules [2]. Major aerospace companies often have their own set of rules and standards which are followed for individual aircraft development programs. A separate military specification for wiring of aerospace vehicles, MIL-W-5088L, provides more detailed and centralised guidance for wiring design for military aircraft [3].

The routing problem is commonly encountered in numerous fields ranging from electronics, data flow in computer networks, navigation systems, and artificial intelligence (AI).
Methods and tools used for automated routing in microprocessors or Very Large Scale Integrated (VLSI) circuits provide a good starting point for addressing electrical loom and pipe design problems in aerospace vehicles. Microprocessors consist of millions of logic components interconnected within a very small space. Early algorithms for circuit design were based on a multi-layered two dimensional approach with one of the objectives to minimise the number of layers due to limitations in component manufacturing. Improvements in technologies and manufacturing process for electrical components have increased the number of layers that can be used, leading to a reduction in chip size. However, VLSI routing automation is still not a completely three dimensional problem.

Many approaches to the VLSI routing automation problem have been developed including maze, channel and switchbox routers [4]. However, is not the scope of this paper to focus on these.

### 3 System Structure

The intelligent routing environment is comprised of five main elements including: user interface, expert interface, data storage, main routing system and output layer. The structure and interrelationships of these five elements are shown in Fig. 1.

![Fig. 1. Knowledge Based router system structure.](image)

### 3.1 User Interface

The main interface for system users is a simple form with a small number of inputs required for specifying the model to be solved. These inputs include:

1) Filename and path for input geometry
2) Input node type (Primary structure / subsystems / payload / etc.)
3) Filename and path for results output
4) Output format (FE mesh / IGES wireframe model)
5) Node type to route (cable / pipe / path category)
6) Rule library to follow (electrical, hydraulic, pneumatic, etc.)
7) Terminal locations (x, y, z Cartesian coordinates)

### 3.2 Expert Interface

Domain experts interface with the system through a knowledge editor and rule editor. The knowledge editor reads and writes entries to a knowledge database containing numerous options and settings for different routing applications including import and export options, solver settings, and built-in rules.

The rule editor is used for creating domain rules, implemented by the solve engine as "IF [condition], THEN [action]" statements. Rules are created in the editor by specifying attributes including rule subject, condition for rule to be implemented and action to be taken. Families of rules for different routing domains and different path types are stored in separate libraries in Comma Separated Variable (CSV) format. Rules currently supported by the system include: path profile, bend radii, clamping and attract / repel rules (discussed below).

### 3.3 Data Storage

Data used by the system is of two types; data specific to individual routing problems, and domain data. Both types are stored externally to the system. Domain data is specified by the expert as described above, and consists of rule libraries and the knowledge database, accessed by the solve engine.

Data for individual routing problems consists of model data and case data. Model data contains input geometry to be traversed, represented by a dense FE mesh. Case data includes inputs specified by the user, listed above.
### 3.4 Routing system

The routing system element consists of three components: a model interpreter, solve engine and results writer. Each is outlined below:

#### 3.3.1 Model interpreter

The model interpreter reads the geometry mesh and generates a 3D maze object. The maze itself is represented by a regular 3D grid with each node described by an x, y, z integer address and node type (e.g. empty space, wall, start / finish location, routed path, excluded zone, etc.).

Multiple parts can be read into the maze object representing different sections of geometry falling within the solution space, allowing rules to be applied independently to each section. For example an FE mesh of primary structure may be imported into the maze and represented by a given node type. A second mesh of subsystems may then be imported into the maze, represented by a different node type. This arrangement will now allow rules to be applied to primary structure and subsystems independently.

#### 3.3.2 Solve engine

The search algorithm in the solve engine uses a best-first search technique, employing a cost function to determine path movements in a similar way as the A* algorithm [5]. In A*, a simple cost function, \( f(n) \), is evaluated which is the sum of the shortest distance from the source node to the node searched \( g(n) \), and an estimated cost to the goal using a heuristic function, \( h(n) \) (Eqn. 1). Lower cost nodes are favoured, thus the search will generally find the target more quickly than breadth first searches. Provided the heuristic which calculates the remaining distance does not overestimate the distance to the target, the algorithm is both optimal and complete [5]. Three examples implemented include the straight line (shortest possible, but usually not achievable), diagonal (shortest practical) and Manhattan (no diagonal) distance.

\[
 f(n) = g(n) + h(n) \tag{1}
\]

The algorithm begins at the source node and expands each adjacent node (a total of 26 nodes, excluding those on boundaries of the solution space). The cost for each node is determined according to equation 1, and the lowest cost node is selected as the new search point. The status of all nodes searched this step is changed to “searched”. The previous node then becomes the new node’s parent and its status is changed to “processed”. The algorithm then searches all unprocessed nodes from this new point, and selects the lowest cost node, repeating the process. If the algorithm reaches a dead end, it will search back through the list of expanded nodes until it finds the lowest cost unprocessed node and will resume the search. When the target is reached, a backtracking process traces through the list of node parents until it reaches the starting point. The list of parents then defines the path.

 Whereas the majority of A* applications are 2D, the algorithm used by the routing system is extended to 3D. The algorithm also includes additional terms in the cost function through which rules from the rule library are implemented (Eqn. 2).

\[
 f(n) = g(n) + h(n) + \sum i(n) \tag{2}
\]

The \( g(n) \) and \( h(n) \) terms are determined in the same way as A*. Additional terms \( i(n) \) modify the node cost, depending on distance from particular node categories, termed subject nodes. This has the effect of either shifting the path towards or away from subject nodes. The cost of each additional term is determined by performing a local search within a given radius around each node searched. If a subject cell is encountered within this radius, the cost of the rule term is evaluated according to Eqn. 3. Variables include: rule weighting, \( W \), distance from subject cell, \( T \) (value of 1 for a node adjacent to a subject cell), and decay rate, \( D \).

\[
 i(n) = W \times (1 - (T - 1) \times D) \tag{3}
\]

Path attract rules have a negative weight factor, reducing overall node cost, \( f(n) \), causing the algorithm to search in the direction of reduced cost. The decay rate ensures nodes closer to subject cells have a higher influence on the rule term cost. Examples of attraction rules include conforming to structure and existing harnesses and attachment points.

Rules that repel the path away from subject cells have a positive weight factor, \( W \), such that node cost, \( f(n) \), is increased. Example rules of this type include: routing away from hot areas and areas containing fluids, clearance from moving parts, and Electro-Magnetic Interference (EMI) rules specifying that certain wire types must not lie within a given distance of others due to electromagnetic interference.
3.3.3 Results writer

Results from the path finding process are output as either a wireframe IGES model and/or an FE mesh. These are detailed below.

3.5 Output layer

Resultant paths are output in IGES format as a CAD-readable wireframe model consisting of straight segments connected with a user defined bend radius. The path model can be imported and integrated into the existing CAD geometry assembly. Operations using tools in the CAD software package can be performed as necessary, adding detail to the path for a complete digital representation of the routed part (e.g. adding thickness, adding detail to terminals, etc.).

The second output method is a FE mesh consisting of geometry, routed path and knowledge layers. The geometry layer, represented by hexahedral elements, contains all different forms of obstacles encountered within the search space (such as primary structure, subsystems, etc.). Routed paths are represented in wireframe using bar elements. The knowledge layer provides details of rules and methods used in the routing process, providing justification for the path placement. Knowledge represented includes a map of where various rules were accessed, their influence on the routed path, and a 3D map of the area searched by the algorithm.

4 Test case

Weapon bays are critical areas of new generation fighter aircraft, consisting of a large number of subsystems, electrical harnesses and pipes, together with payload within a limited space. A view inside one of the F-35 Lightning II, or Joint Strike Fighter (JSF), weapon bays is given below [6].

It can be seen from this view that a large number of different cable and pipe types are present within the search space. This adds to the complexity of the routing problem due to interaction between various structural components, and cables of different types.

Master CAD models for complex aircraft structures such as these are typically assemblies of a large number of smaller parts and sub assemblies, including structural components, subsystems, fasteners, etc. No-go zones are defined for designers such that components are not placed in areas that will interfere with the payload that is intended to be carried.

Fig. 2. Electrical wiring and pipe loom in weapons bay of F-35 JSF. [6].

4.1 Generating geometry

A simplified version the weapon bay geometry (excluding bay doors) was created using CAD software and is shown in Fig. 3. This simplified version consists of a principle structural frame, some arbitrary subsystems, as well as a sample payload of bombs. Terminal locations are defined in the model as point objects and the x, y, z coordinates for each are extracted.

For this test case three different cable types will be routed, with for sets of source / target terminals defined for each. Rules governing interaction between the three types will be implemented as well as a rule specifying that paths follow structure as closely as possible.

Fig. 3. Test case geometry
4.2 Discretising geometry

For input into the grid-based algorithm, model geometry is to be converted to a discrete form. This step is performed by extracting geometry from the CAD software, and meshing in FE pre and post analysis software. To achieve a good representation of the geometry in the solver, a very fine mesh is required. Wherever possible, a solid tetrahedral automatic meshing tool is used to generate the required mesh. In some cases with complex geometry, surfaces may become damaged through the geometry extraction from CAD to FE software. In this case a shell mesh is used. The three sections of geometry to be represented uniquely within the software are meshed and exported separately. Terminal locations can be specified in the FE model by creating an entity set for both source and target terminals consisting of nodes defining their x, y, z coordinates, or later by typing coordinates into the solver.

4.3 Routing

The intelligent routing environment is then used to route paths between the terminals. The user interface is a familiar Microsoft Windows form consisting of file browser dialogue boxes for specifying model locations, drop down boxes for specifying geometry/route types and a text box for defining terminal locations. The user is required to specify a number of parameters to set up the routing job. Each set of seven parameters discussed earlier (section 3.1) appears as a row in a central table termed the “solution space”. Each row in the solution space can perform one or more of three actions including: import geometry, route a path, and output results. Geometry section of each unique type are represented by a row in the solution space, consisting of file location, file format (FE mesh or local maze type), and geometry type (principle structure, subsystems, payload, etc.). Additional rows are added for each net to be routed, defined by terminal coordinates, type of path to be routed, and rule set to follow. A row for outputting results is also added, defined by output path and filename, and format.

In this test case, a number of cables of different types were routed in the solution space, with various rules governing the interaction between them.

4.4 Routing output

System outputs are delivered in three ways: Firstly, a simplified three view preview is given within the routing environment. This gives the user a very quick indication as to whether the routing task was successful.

Secondly, a multilayered FE mesh is written consisting of geometry, routed path and knowledge layers, allowing users to analyse results. The top half of Fig. 5 shows the geometry layer consisting of the three geometry sections defined previously, and the three routed paths, represented in wireframe by bar elements.

The knowledge layer is shown in the lower half of Fig. 4. It consists of a number of layers which detailing several aspects of the solution process. These include a series of colour coded elements following the routed paths, representing rules accessed to determine each path step. A map of nodes searched is given to show areas of the model that were considered by the algorithm, a map of where rules were accessed on various parts of the model where each individual rule was implemented.

Fig. 4. FE output
Thirdly, a wire frame IGES model of the path is written. Each path appears as a separate IGES wire frame model. Each model is imported into the existing geometry assembly. Detail is added to the paths by creating a circular profile normal to the path and extruded along the paths length.

Each cable category routed is represented by a different colour and results are shown in Fig. 5

![Fig. 5. CAD output](image)

### 4.5 Evaluation of results

The library used by the solver for this case implemented several rules. The first rule was specified that each cable type shall follow structure as closely as possible. It can be seen that this rule was followed very closely. Paths tended to follow stiffeners in the model as the algorithm found this to be a lower cost path.

Rules specifying that cables of the same type should be followed as closely as possible were successfully implemented, evident in Fig. 5.

Finally, rules were implemented for each cable specifying that if a cable of a different type is encountered, to either stay away as much as possible or follow closely. With reference to Fig. 5, the red and blue cables were permitted to follow each other, but stay away from the yellow cable wherever possible. This rule was generally well followed, with the exception being a single red cable which had to pass the yellow cable to reach its target.

In areas of rule conflicts, the behaviour of the routed paths tends to twist and double back on itself slightly. Improvements to the algorithm in terms of intelligently applying rules in a hierarchy will greatly improve handling in these areas.

Considering these preliminary results, overall system performance in terms of path quality with respect the test rules was found to be very good.

### Conclusion

The intelligent routing environment outlined in this paper is successful in implementing knowledge based methods and techniques together with path finding principles used in other domains to produce routed paths satisfying design rules.

The knowledge and rule editors simplify the knowledge capture process, giving domain experts flexibility to implement new routing methods and rules, extending the application to new problem domains (e.g. water pipes, air conditioning ducting).

The quality of routed paths in terms of relevance to particular problem domains will be improved with addition of detailed domain specific knowledge to the knowledge base and rule libraries. The software has proved useful so far in providing designers with new ideas for path placement that had not been previously considered.

An improved process for accessing and managing rules within the solver will greatly enhance software outputs, reducing unwanted turns in routed paths.

### References


