COMPUTATIONAL SUPPORT FOR THE COLLABORATIVE DESIGN, ROUTING AND MANIPULATION OF CABLE HARNESSES

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Andrew B. Conru

February 1997

© Copyright 1997 by Andrew B. Conru

All Rights Reserved

Abstract

This thesis discusses the use of a cooperative, agent-based search in the cable harness routing problem (CHRP). They hypothesis behind this work is that a human designer can be greatly assisted in the intricate task of routing cables through a complex space if he is empowered to interact naturally with other autonomous agents as well as with the design itself. A collection of automated search and routing agents, which make use of domain knowledge and which are able to communicate with the human designer, is discussed. In addition, this thesis presents a novel interactive routing environment in which each agent is able to communicate design fragments with others via an adaptive, genetic-based blackboard.

Acknowledgments

This thesis is the culmination of many years of pleasant association with the Mechanical Engineering and Computer Science Departments of Stanford University. I am deeply indebted to my advisor, Professor Mark Cutkosky, for his invaluable insight and support during my Ph.d. study. I would also like to thank Professor Jean-Claude Latombe for his encouragement and advice. I would also like to thank Fritz Prinz for his review and comments.

The Center for Design and Research was a fantastic place to do research – filled with so many wonderful and supportive people. I'm especially honored to have met Hisup Park, Larry Edwards, George Toye, Greg Olsen, Padmanabh Dabke, and Maria Yang, etc. Their friendship added much to my life at Stanford.

Special thanks go to Rachel Fong for her many hours of assistance in writing a readable document. Last but not least, I would like to extend my gratitude to all the members of my family for their love and support.

The financial support of this work by the NSF is greatly appreciated.

Table of Contents

ABSTRACT ..... **IV**

ACKNOWLEDGEMENTS ..... **V**

1. INTRODUCTION **1**

1.1 OVERVIEW OF THE CABLE HARNESS DESIGN PROBLEM 2
1.2 APPROACH 7
1.2.1 *Strategy # 1 – Combat the large search space by using a team of search heuristics and reusing parts of promising designs* 9
1.2.2 *Strategy # 2 – Use designer input to direct the search process and to address unmolded design issues* 11
1.2.3 *Strategy # 3 – Abstract the harness and environment representations differently to reflect process and agent related needs* 12
1.3 APPLICATION 16
1.4 CONTRIBUTIONS 16
1.5 ORGANIZATION OF THE DISSERTATION 17

2. THE CABLE HARNESS ROUTING PROBLEM (CHRP) **19**

2.1 THE CABLE HARNESS ROUTING PROBLEM 20
2.1.1 *Cable Harness Components* 20
2.1.2 *Definition of the Routing Objectives* 23
2.1.3 *Attributes of a Good Routing* 26
2.2 RELATED ROUTING PROBLEMS AND PREVIOUS WORK 28
2.3 INPUTS TO THE CABLE HARNESS ROUTING PROBLEM 35
2.3.1 *Routing Environment* 36
2.3.2 *Wiring List* 37

2.4 SIZE OF THE CABLE HARNESS CONFIGURATION SEARCH SPACE 40

3. THE ENVIRONMENT REPRESENTATION FOR THE CHRP 46

3.1 INTRODUCTION 46
3.2 ENVIRONMENT REPRESENTATION FOR THE DRI 47
3.3 ENVIRONMENT REPRESENTATION FOR THE AUTOMATED ROUTING AGENTS 50
3.4 GENERATING THE APPROXIMATE MEDAL AXIS GRAPH 55
3.4.1 *Generating the AMA surface* 55
3.4.2 *Abstracting the AMA Graph from the AMA* 57
3.4.3 *Using the AMA Graph for obstacle distance calculation* 61
3.5 MODIFYING THE AMA GRAPH CAPACITY DURING ROUTING 62
3.6 COMPUTATIONAL COMPLEXITY OF VOXEL TO AMA GRAPH CONVERSION 63

4. COLLABORATION IN THE CHRP 65

4.1 INTRODUCTION 65
4.2 APPROACH 68
4.3 COLLABORATIVE DECOMPOSITION OF THE CHRP 70
4.3.1 *Topology Search Problem* 72
4.3.2 *Topological Routing Search Problem* 73
4.3.3 *Cable Harness Design Fragments* 74

5. AUTOMATED ROUTING TOOLS 77

5.1 INTRODUCTION 77
5.2 CREATORS 78
5.2.1 *Random Topology Generator* 78
5.2.2 *Thick Bundle Router* 79
5.3.3 *Wire Bundling Router* 81
5.3 MUTATORS 86
5.3.1 *Copy Best & Copy Some Operators* 86
5.3.2 *Move Branch Operators* 86
5.3.3 *Human Designer* 94
5.4 COMBINORS 94

6. DESIGNER'S ROUTING INTERFACE 99

6.1 OVERVIEW 99
6.2 INTERACTIVE BUNDLE REPRESENTATION 101
6.3 TRANSFORMING THE AMA INTO THE SPHERE REPRESENTATION 104
6.4 PHYSICAL MODEL OF BUNDLE PATHS 109
6.4.1 *Environmental Loadings* 111
6.4.2 *Sphere-to-Sphere Axial Forces* 111
6.4.3 *Sphere-to-Sphere Bending Forces* 113
6.4.4 *Sphere-to-Sphere Repulsion* 115
6.4.5 *Obstacle Repulsion Field* 117
6.4.6 *User defined loads* 119
6.5 HIGH LEVEL INTERACTION 120

6.5.1 *Move (Clamp) Bundle* 120
6.5.2 *Move Branch Operation* 121
6.5.3 *Transition Operations* 127

7. EXPERIMENTAL RESULTS 128

7.1 INTRODUCTION 128
7.2 EXPERIMENTAL SETUP & PROCEDURE 129
7.2.1 *Resource Allocation between Routing Agents* 130
7.2.2 *Resource parameters for the Topology and Topological Routing search processes* 130
7.3 ROUTING AGENT OPTIMIZATION 133
7.3.1 *Move Branch Routing Agent* 133
7.3.2 *Combinor Routing Agent* 138
7.4 COLLABORATIVE ANALYSIS 139
7.4.1 *Effect of Problem Type on Agent Performance* 140
7.4.2 *Effect of Average population Fitness on Agent Performance* 143
7.4.3 *Collaboration between the Designer and Routing Agents* 143
7.4 SUMMARY OF EXPERIMENTAL RESULTS 145

8. CONCLUSIONS AND FUTURE WORK 147

8.1 CONCLUSIONS 147
8.1.1 *Decomposition of the Routing Problem into Subtasks* 148
8.1.2 *Evolutionary Search Strategy* 148
8.1.3 *Development of Specialized Environment Representations* 150
8.1.4 *Integration of Designer Input* 150
8.2 DIRECTIONS FOR FUTURE WORK 152
8.2.1 *More aggressive use of designer input in the search process* 152
8.2.2 *Extensions to the flexible harness representation and its use* 152
8.2.3 *Reusing concepts in other routing domains* 153
8.2.4 *Adaptive division of computational resources* 154

A. EVOLUTIONARY STRATEGY TO EVALUATE TOPOLOGY ROUTING 155

A.1 TRANSITION LOCATOR INTRODUCTION 155
A.2 OVERVIEW OF GENETIC ALGORITHMS 157
A.3 TRANSITION LOCATOR GENETIC REPRESENTATION AND OPERATORS 160
A.4 GENETIC PARAMETER SETTINGS 160
A.5 SEEDING THE POPULATION USING THE RUBBERBAND HEURISTIC 161
A.6 COMPARISON WITH RANDOM SEARCH 163

B. OVERVIEW OF SHORTEST PATH TECHNIQUES 164

C. RANDOMLY GENERATED TOPOLOGICAL ROUTINGS 168

D. MULTIPLE ROUTING AGENT RUN DATA 172

E. EXAMINATINO OF AGENT PERFORMANCE 175

REFERENCES 180

List of Tables

2.1	Comparison of routing issues among various routing problems.	29
2.2	Size of exhaustive search space for cable harness topologies and topological routings.	43
7.1	Resource distribution among processes.	131
7.2	Comparison of different fragment selection techniques for a 10 port, balanced wire list harness in the maze environment.	134
7.3	Comparison of different Move Branch techniques for a 10 port, balanced wire list harness in the maze environment.	135
7.4	Comparison of different Move Branch techniques for a 10 port, multi-biased wire list harness in the maze environment.	136
7.5	Comparison of different Move Branch techniques for a 15 port, multiple biased wire list harness in the maze environment.	137
7.6	Results of multiple-biased 10-port harness using the design-based, fragment-based, and mutually-exclusive selection.	138
7.7	Results of single-biased 10-port harness using the design-based, fragment-based, and mutually-exclusive selection.	139

7.8	Results of balanced 10-port harness using the design-based, fragment-based, and mutually-exclusive selection.	139
7.9	Ranking of routing operators for the 10-port harness. The numbers represent the percent decrease in average lowest cost with and without the operator.	141
7.10	Comparison of cost of designs found using various routing methods.	
A.1	Suggested GA parameters for Transition Locator.	161
A.2	Comparison with Random Search.	163
D.1	Multiple-biased 10-port harness in maze environment	172
D.2	Single-biased 10-port harness in maze environment	173
D.3	Balanced 10-port harness in maze environment	173
D.4	Multiple-biased 10-port harness in empty environment	173
D.5	Single-biased 10-port harness in empty environment	174
D.6	Balanced 10-port harness in empty environment	174
D.7	Multiple-biased 15-port harness in maze environment	174
E.1	Comparison of designs found using various routing methods	176
E.2	Sample listing of design inputs for each agent for a single generation	177

List of Figures

1.1	Example 8-port cable harness routed through a cluttered environment.	2
1.2	A cable harness is comprised of many wires protected by an external casing.	3
1.3	Stages of the cable design process.	4
1.4	Schematic diagrams of two topologies that satisfy the same pin-to-pin wiring specifications.	4
1.5	The initial placement of electrical assemblies in the environment typically does not consider routing issues.	5
1.6	The quality of a topology depends, in part, on the relationship between the wiring list and the layout of the ports.	6
1.7	Changes in the environment greatly affect the routing.	7
1.8	Multiple agents can build and reuse designs stored in a centralized pool.	11
1.9	The cable harness representational needs depend on the problem being addressed.	13
1.10	Four different environment representations address the needs of the routing process.	16

2.1	Transition typically have three branches to other ports or transitions each branch is defined with its own offset and orientation.	20
2.2	The topology of a cable harness is different than that of a wire bundling in that cycles are prohibited.	23
2.3	The computational expenses required to route a topology (A) in environment (D) in lessen by approximating the bundles paths (C) using a coarse graph (B).	24
2.4	Routings that double back on themselves are usually a result of poor topology selection (line widths correspond to the number of wires in each bundle.)	27
2.5	The optimal topology is dependent on the wire list.	28
2.6	The CHRP is related to a number of simpler tree spanning problems.	30.
2.7	Close-up of the addition of a Steiner point between three nodes.	31
2.8	The Steiner minimum tree algorithm may produce suboptimal routings for the CHRP.	35
2.9	Plan view of a sample routing environment (the actual obstacles and boundaries are three-dimensional).	36
2.10	Simple and complex routing environments with solid obstacles. (The free-space regions are shown with nodes and arcs of a routing graph, to be discussed in Chapter 3.)	37
2.11	Harness with balanced wire list.	38
2.12	Harness with single-biased wire list.	39
2.13	Harness with single port wire list.	39
2.14	Two potential routings of a 3-port harness.	40
2.15	A gradient search for a 3-port harness may result in a local optimum.	41
2.16	Distinct topologies for 2 to 9 port harnesses.	42
2.17	Four possible routings of a 4-port harness. Thick lines represent the harness; thin lines represent straight paths between nodes in the approximate representation of the free-space (see Chapter 3).	44
3.1	The collapsing wavefront algorithm starts by marking all the voxels next to obstacles and external walls.	49
3.2	Further progress of the wavefront algorithm.	49
3.3	Minimal cost harness routings do not necessarily contact obstacles.	51
3.4	Two example cell decompositions of simple routing environment (transitions drawn as squares, end ports drawn as black circles).	52

3.5	Voronoi diagram and Approximate Medial Axis of a simple routing environment.	53
3.6	Using voxels to represent bundle paths is prone to underestimate the amount of the free- space occupied by the bundles.	54
3.7	Spherical approximation of Voronoi surface.	55
3.8	The AMA surface is constructed by recording the intersection of voxel wavefronts from multiple faces.	56
3.9	Fully connected AMA.	56
3.10	Redundancy and Connectiveness is based on sphere overlap, S_{12}	58
3.11	The AMA Graph of the AMA of Figure 3.9.	59
3.12	The grey spheres are determined to be redundant due to the black sphere.	59
3.13	If the small center sphere is removed, the remaining smaller spheres may over-estimate the available free space.	60
3.14	The spheres in the AMA Graph can be used to approximate the free-space as well as the distance between any point and the nearest obstacle.	61
3.15	The safe routing area is shown in gray.	63
4.1	GA-based Design Flowchart.	68
4.2	CHRP is divided into its topology and topological routing search processes.	71
4.3	Some agents are only able to create a topology while others output both the topology and the topological routing.	72
4.4	Each topological routing has good and bad fragments.	72
4.5	A better offspring is found by combining parts of two different topological routings.	73
4.6	Offspring may take on positive attributes of both parents.	74
4.7	Fragment 2-3 and 1-6 order 1 whereas 4-5 is order 2.	75
4.8	The cost of a fragment of order 2 (fragment 4-5) or more is dependent on its complements.	76
5.1	A tree representation for a cable harness topology.	78
5.2	Topology Template.	78
5.3	In the Thick Bundle agent, ports are connected to a routing under-construction. Ports are selected in descending order of the number of wires at each port.	80
5.4	Optimal routing for the four-port harness.	80
5.5	First two steps of the Wire Bundling agent.	82

5.6	The links are removed one by one.	83
5.7	The U-shape routing may be more costly than the X-shape routing.	83
5.8	The order of the wire routing affect overall routing costs.	84
5.9	10 port harness with a circled fragment.	87
5.10	Fragment 1-2-3 and its complement 4-5-6-7-8-9-10.	87
5.11	Results of the move branch operation with reattachment at transbundle A and termbundle B.	88
5.12	Any topology can be transformed into another by sequentially moving term-bundles.	88
5.13	Distribution of fragment fitness and the number of ports.	91
5.14	The fragment 1-2-3 on the left is to be grafted onto the topology on the right.	96
5.15	Fragment 1-2-3 and its complement from the second topology.	96
5.16	Resulting topology from the combination.	97
5.17	Bundles remaining after redundant bundles are removed are rerouted Before the reattachment location process.	98
6.1	The graph based routing is converted into connected spheres.	103
6.2	A small amount of open space is assumed between wires.	105
6.3	The deletion/adding process depends on the spacing of adjacent spheres.	106
6.4	The center sphere is considered redundant as the outside spheres adequately represent the bundle.	107
6.5	Strand of point masses showing various loadings.	109
6.6	The tension in the bundle changes with the length of the bundle as spheres are inserted or deleted.	112
6.7	The bending radius is quickly estimated from the angle between two links.	114
6.8	The bending moment is applied as a couple on each node.	114
6.9	Two different repulsion profiles produce different routing properties.	117
6.10	Sphere next to obstacle require special attention.	119
6.11	An illustrative example of a large move bundle operation.	121
6.12	Multiple harness may compete for space.	122
6.13	Initial Routing of 12 Port Harness.	123

6.14	Designer Selects End of Transbundle.	124
6.15	Designer pulls harness fragment.	124
6.16	Results of the move-branch operation.	125
6.17	Another transbundle is cut.	126
6.18	Transbundle is moved to new location.	126
6.19	Harness settles to another local minimum.	127
7.1	Population diversity is an important factor to the performance of the search process.	132
7.2	Lowest cost harness routing with multiple-biased wirelist.	136
7.3	Suboptimal harness routing with multiple-biased wirelist.	137
7.4	The Wire Bundling agent works well in cluttered environments.	142
A.1	Routing and topology of a 3-transitioned cable harness.	156
A.2	Binary Genetic mutation changes only a single bit to create a “new” design.	158
A.3	Binary genetic crossover swaps parts of two genetic strings.	158
A.4	Standard genetic algorithm performs crossover and mutation in series.	159
A.5	Routing Chromosome and Fitness.	160
A.6	The rubber band process starts by choosing transition locations randomly then moving them iteratively to minimize cost.	162
A.7	The transitions are then moved to the closest AMA nodes and moved locally along the graph to a local minimum.	162
B.1	Single point-to-point routing.	166
C.1-6	Plots of randomly generated routings.	169
E.1	The relationship between agents can be examined via design genealogies	175

Note: this copy of thesis was restored from previous work and may contain minor typos.

Chapter 1

Introduction

The traditional process of designing complex products typically involves making decisions without fully considering their downstream ramification. This often presents problems as conflicts between decisions may require redesign. Without rapid feedback, redesign becomes more costly as upstream decisions may progress further before the downstream problems are discovered. Concurrent engineering is a well-documented approach for ensuring that downstream issues are considered early in the design process. The need for greater concurrency occurs in a wide range of engineering design problems where the factors on which upstream and downstream decisions are based are tightly coupled.

This dissertation focuses on one specific example called the cable harness routing problem (CHRP), where electronic assemblies are constructed to provide electrical connectivity between multiple locations in a 3D environment (e.g., engine compartment or computer enclosure). It is

desirable to address the routing issues of the CHRP early in the design process while the placement and geometry of components can be changed relatively easily. However, doing so requires cable harnesses to be routinely re-routed each time a change occurs in the routing environment. At issue is that this is currently a costly and time consuming process often taking weeks to finish. This forces designers to create environment designs with extra clearance for routings and to wait until the final stages of the design process to address the issue of cable harness routing – at which point the electrical requirements, component geometry, and spatial layout have already been fully committed. This strategy not only seriously compounds the problems of redesign if no feasible harness routing can be found but also leads to suboptimal designs. As the cost of designing and fabricating cable harnesses is considerable, by addressing the cable harness design earlier in the overall design process, significantly cheaper and lighter harnesses can be obtained [Johnson 94].

This chapter will briefly discuss the cable harness routing problem (CHRP), overview the strategies developed for addressing the problem, and list the contributions of this work.

1.1 Overview of the Cable Harness Design Problem

Cable harnesses carry a combination of signals and power between electronic packages and subassemblies and come in many sizes and varieties. Figure 1.1 shows an eight-port cable harness and its routing through an environment consisting of three blocks and a cylindrical object. Its shape is typical of those used in automotive and aerospace applications.

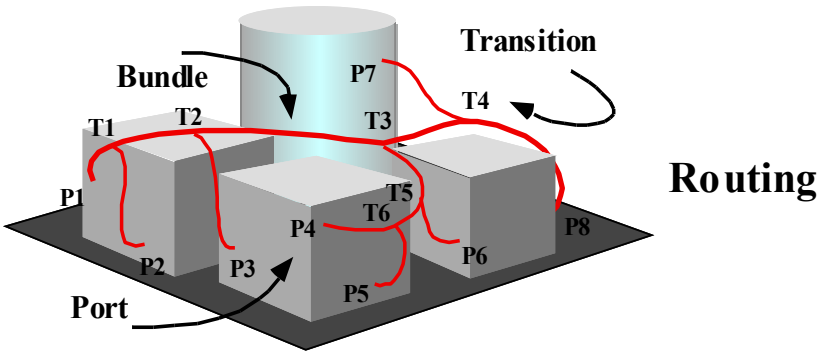


Figure 1.1 - Example 8-port cable harness routed through a cluttered environment.

Figure 1.2 shows, in more detail, the internals of a typical harness assembly, consisting of wires joined with transitions and terminated with the end-connecting ports.

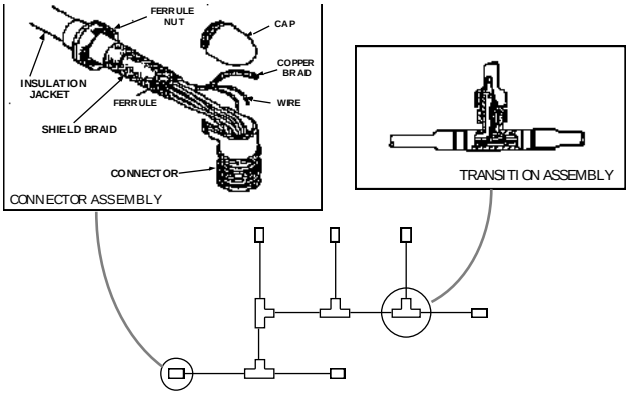


Figure 1.2 – A cable harness is comprised of many wires protected by an external casing

Plainly stated, the cable harness routing problem is to find the least cost routing for a cable harness in a cluttered, 3D environment. The definition of cost may vary depending on the domain. For most, the overall reliability of the harness is of highest concern. However, other factors such as harness weight, weight distribution, material and production costs are often used as quantifiable measures.

The cable harness design process itself consists of several interrelated stages as shown in Figure 1.3. The process begins with a set of electrical specifications and environmental constraints. A wiring list specifies the pin-to-pin connectivity requirements for the electronic packages. The electrical specifications may also include constraints such as limits to the impendance and signal-to-noise ratio. The environment specifications include the geometric models of various packages and structural elements as well as the locations of ports that the harness is to connect. They also may define regions where clamping is permitted and hazardous zones wherein any routing incurs additional costs.

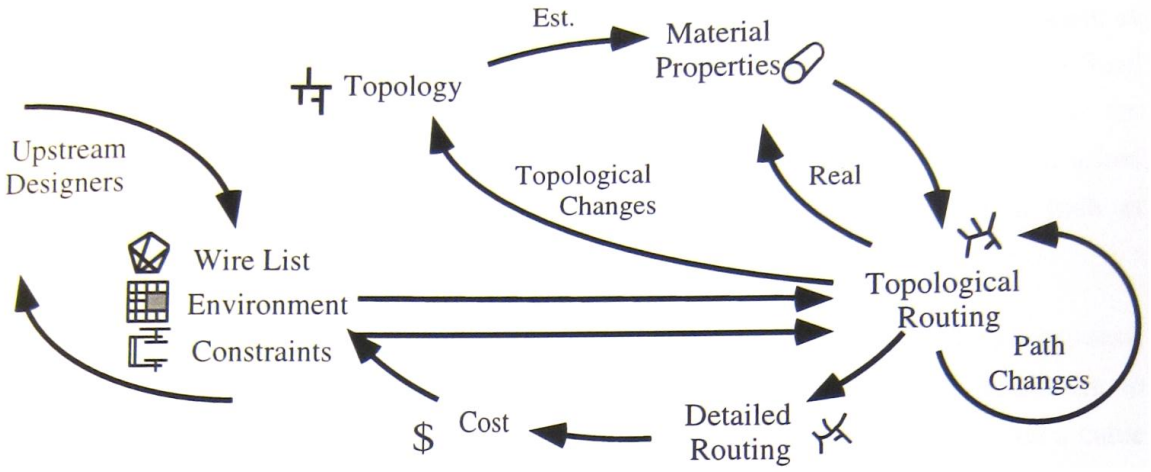
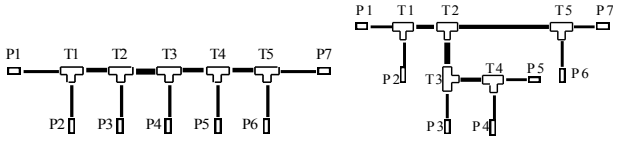


Figure 1.3 – Stages of the cable design process

The harness may be routed once the electrical and environmental specifications are available. This process requires the simultaneous specifications of the topology and topological routing of the harness. The topology of the harness dictates the overall shape of the harness (see Figure 1.4) whereas the topological routing defines the areas through which the harness is routed. For any given set of ports, there are many possible topologies (the number grow factorially with respect to the number of ports as will be discussed in Chapter 2, Section 2.5). For example, both of the two harnesses shown schematically in Figure 1.4 satisfy the same pin-to-pin wiring specifications. However, the wire count in each bundle, and consequently the thickness, material properties and cost per unit length, is different in each case. One of the difficulties of the CHRP is that it is impossible to determine *a priori* which is found in the environment. This occurs as both the actual length of each routed bundle and the required insulation and shielding type can only be determined after the topological routing is known.



Topology A. *Topology B.*

Figure 1.4 – Schematic diagrams of two topologies that satisfy the Same pin-to-pin wiring specifications.

Choosing a good topological routing for a harness is almost always an iterative process as the quality of the overall routing depends, among other issues, on the bundle costs defined by the topology. Once the topological routing is specified, the designer can refine the routing to produce a detailed routing. The detailed routing of the harness defines the actual paths through the environment for each of the bundles and addresses constraints such as those for minimum bending radius and clamping.

The following simple example illustrates some of the steps of the cable design process. Consider the case of a designer whose task is to design a small electronics enclosure to house two separate electronic assemblies shown in Figure 1.5. Assume further that a cable harness is to be designed to connect seven ports containing fifty wires in total. The first decision the designer must make is in defining the location of the assemblies in the enclosure. As it is difficult to determine beforehand what paths the routing will eventually take, the designer typically uses rules of thumb governing assembly placement that do not consider the harness routing; for example, the designer may maximize the clearance between assemblies.

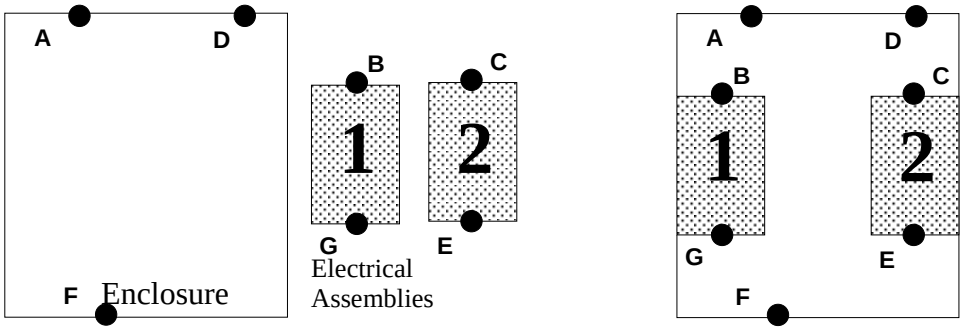
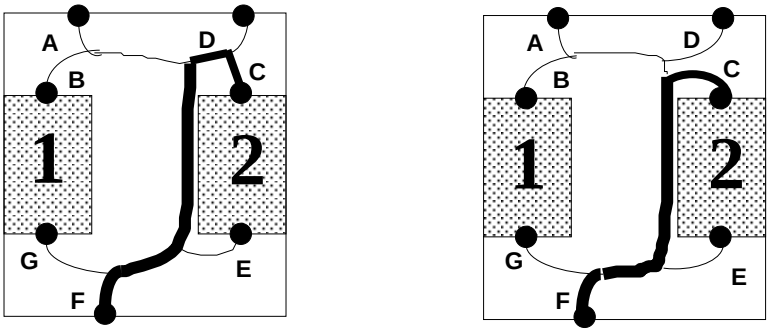


Figure 1.5 – The initial placement of electrical assemblies in the environment typically does not consider routing issues.

Once the environment is defined, the designer can find a topology and topological routing for the harness. A common heuristic is to visually group ports that are close together and then combine these fragments to form the complete routing. Design 1 of Figure 1.6 shows the topology and topological routing that result from using this heuristic. As the process of testing and documenting each routing is time consuming, the designer often stops at this point. More industrious designers may examine the distribution of wires over the defined harness routing and look for inefficiencies in the topology. For example, in Design 1, a disproportionate number of

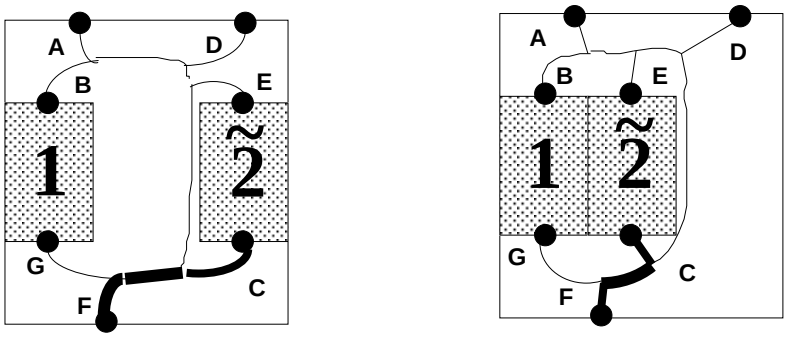
wires connects ports C and F while the paths of these wires are longer than they would be if a different topology, such as the one in Design 2, were used. Fortunately, as a large part of Design 1 may be reused in creating Design 2 (e.g., fragments connecting ports A and B and ports E, F and G), the amount of rework is minimal. As we shall see, this characteristic – namely that improved designs can be assembled from fragments of other designs – make the cable harness design process especially well suited to evolutionary methods such as genetic algorithms.



Design 1 *Design 2*

Figure 1.6 – The quality of a topology depends, in part, on the relationship between the wiring list and the layout of the ports.

Upon closer inspection, the designer may choose to change the environment layout itself to further reduce the inefficient routing of wires between ports C and F. Changing the assembly placement usually involves more rework than a topological change. The designer must compare the additional cost of redesigning the harness and routing environment with the cost savings to be realized. Obviously, the sooner the changes can be made (ramifications of poor assembly placement seen), the lower the chances that other decisions will have been made that depend on the original placement. In this example, the following designs in Figure 1.7 would be an improvement over the previous two designs, but at the cost of moving assembly 2.



Design3 (Rotate Assembly 2) *Design 4 (Rotate & Translate Assembly 2)*

Figure 1.7 – Changes in the environment greatly affect the routing.

1.2 Approach

As the simple example in the previous section reveals, the cable harness routing problem (CHRP) has several characteristics that affect the approach taken for providing computational support. It belongs to a class of engineering search problems characterized the by the following characteristics:

1. Good designs are dispersed in an extremely large search space.

The CHRP is a member of an intractable class of combinatorial optimization problems which are not only NP-complete [Gilbert, 1968], but which also possess a solution landscape characterized by a small number of exceptional designs dispersed in a field of suboptimal designs. This profile can be seen in plots of the cost of randomly generated designs (Appendix C) in which the vast majority of designs are grouped several standard deviations from the best designs. A closer examination of the best designs often reveals a number of strikingly different topologies along with slight variants of each. As the problem is discrete, the solutions pace doesn't possess an underlying structure, such as a gradient, that can be easily exploited by a search technique.

As with many NP-complete problems, many specialized heuristics have been formulated using domain-specific assumptions. However, the nature of the search space limits the effectiveness of using a single search heuristic that has a propensity to return designs in a single, and perhaps suboptimal, region of the search space. Also, as only a small number of samples can be made relative to the size of the search space, there is strong incentive to extract as much information as possible from each design and the relationships between sampled designs – especially as the chances are small that designs found early in the search process are of high quality.

2. It is difficult to integrate subjective objectives in an automated search process.

In addition to deterministic objectives, such as reducing material cost or total harness weight, the routing specification often has complex, judgmental requirements pertaining to clamping, maintainability and installability that complicate the automation of the search process. For example, a customer may ask for a cable to be kept from contacting a hot power supply. As constrains such as this are not intrinsic to the problem formulation, the automation solution must provide a mechanism for a human interaction with the cable harness routing.

3. The individual components of a design have interacting relationships that affect the overall quality of the design.

As will be discussed in Chapter 4, the CHRP can be viewed partly as a configuration design problem [Welch & Dixon 1989] in that a harness is assembled using a number of standardized parts. However, the way in which the parts are assembled affects the overall quality of the design. This presents problems as designers often use local design "cues" or heuristics in creating a design wherein the global interactions procedure unwanted side-effects. For example, designers often think of the cable harness at a high level of abstraction, working in terms of bundles and branches. While this abstraction has the advantage of allowing rapid design changes, it sometimes leads the designer to make poor decisions because the abstraction obscures the details, such as the number of wires in each bundle, that affect the quality of a design. A designer may inadvertently make local changes that significantly diminish the overall quality of the design. The computational solution must therefore either clarify the relationships of the individual components or provide an automated search solution that doesn't rely heavily on localized decisions.

4. Designs can be decomposed into fragments that tend to retain "localized goodness"

Despite the possibility that local decisions may compromise the overall quality of a design, locally good solutions do tend to appear as components of globally good solutions. Therefore, there exists an incentive to identify and reuse locally good components in the search process. This is, once a good fragment is determined, focusing efforts on designs that share that fragment is generally more effective than a more random approach. This is especially advantageous in the case of the CHRP as the size and complexity of the search space precludes exhaustive search. The solution process, however, must be able to balance between focusing on a particular design and exploring new territory, as better designs may exist that do not contain pre-existing fragments.

The approach taken in this work was to use a federation of automated routers working in concert with a human designer to explore the search space of possibilities. The designer interacts with each design candidate in a natural 3D representation to improve it while the automated routers continue to test new designs in the background. Once a satisfactory solution is found, the designer adds additional constraints such as clamping locations. The following sections elaborate on the strategies that were used in this approach to provide efficient and effective support for people engaged in cable harness design and the reasoning behind the strategies.

1.2.1 Strategy #1 – Combat the large search space by using a team of search heuristics and reusing parts of promising designs

The cable harness routing problem can be viewed conceptually as a search within a search. At the highest level, the problem is to choose the topology of the harness. The number of possibilities increases factorially with the number of points to be connected. At the lower level, the problem is to find a routing for a given harness structure that makes best use of the layout of the routing environment. The complexity of this task increases exponentially with the complexity of the environment.

Since the search space is far too large to adequately sample, it is imperative that the search quickly focus on areas with good designs and fully utilize information gained during the search process itself. Typically, a single search technique, such as a router using a specific heuristic, can only return designs in a limited region of the search space. The first impulse is to create a collection of heuristics and apply them sequentially, keeping the best design found. While this would improve the robustness of the system, the strategy would never produce a better design than that of the best heuristic. The idea behind the work in this thesis is that better designs can be found by reusing parts of previous designs. That is, if the findings of multiple agents, assumed likely to be in different areas of the search space, can be traded among the agents and reused in an evolving search framework, the framework will have a greater chance of finding near optimal solutions than the set of individuals.

As mentioned earlier, the CHRP has the property that the complete design can be created by assembling fragments under prescribed rules. Since the interrelationships of these components affect the contribution of each component to the overall goodness of the design, it is impossible to predict exactly which combination of fragments will result in high quality designs. However, even simple heuristics can be used to estimate which fragments will have a high chance of being useful. This unique property lends itself to solution methodologies that capitalize on finding and reusing design fragments.

Consider the case where a number of designs are generated randomly. As the quality of these designs may be greatly inferior to ones created with heuristics, it may first appear to be a waste of computational resources. However, the decomposable nature of the cable harness enables even poor designs to provide useful information. Each fragment, or design component, represents a subset of the design parameters and can be selected based on its relative merits to isolate and reuse the best parts of each design. As good designs and fragments are chosen more often than poor ones, computational resources are focused on regions of the search space with high probabilities of success.

Because the search space has many regions with relatively good designs, it is important to assemble a team of agents capable of exploring many designs as possible. In this work, two classes of search agents, *mutators* and *combinors*, are developed that are able to extract and reuse fragments to produce new designs. These agents work in parallel with others, called *creators*, that generate designs from scratch using either random methods or heuristics. Although

heuristic-based creators can be used to quickly produce above-average designs, they tend to narrow the search prematurely to a small number of regions.

The mutators and combinors, however, by substantially altering pre-existing designs, keep the search process from stagnating.

Figure 1.8 gives an overview of an architecture capable of allowing these routing agents to work in parallel. All three agent classes can read and write designs in a common representation using a design repository or pool. Since each agent works independently from the others and shares designs only as desired using the design pool, the search process can progress asynchronously.

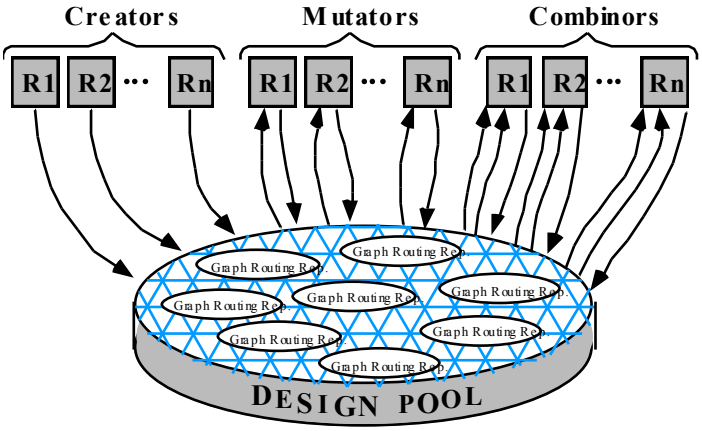


Figure 1.8 – Multiple agents can build and reuse designs stored in a centralized pool.

1.2.2 Strategy #2 - Use designer input to direct the search process and to address unmodeled design issues

One of the early goals of this work was to completely automate the routing of the cable harnesses. However, it was found that while automated routers are highly effective in finding near-optimal routings, several issues prevented them from being used exclusively. First, the objective function for the industrial cable harness routing problem contains parameters and constraints that are difficult to quantify and model. This may make a numerically "optimal" solution inferior when unmodeled issues are taken into consideration. For example, consider a

harness that is used where maintenance must be performed frequently. An automated router may favor a routing that, while optimizing a weigh-based objective function, occludes a maintenance access panel and therefore is "suboptimal." While some constraints can be addressed implicitly or explicitly in the search process, it is virtually impossible to anticipate all such constraints before-hand. It is not uncommon for new constraints or issues to be discovered and imposed on a case-by-case basis, mid-way through the cable design process. Therefore, creating a totally automatic system that searches for a globally "optimal" solution without provisions for human guidance, is impractical.

A second limitation arising from relying solely on automated routing agents is that as the difference in quality between designs becomes smaller, it becomes increasingly difficult for the routing agents to pick the better ones. An example is where a harness is routed well with the exception of a single poorly connected branch. In this case, the overall quality of the routing might be high enough that the automated routers would continue to use it. However, an experienced cable designer could spot the bad branch and fix it.

These issues are addressed by incorporating the designer in the automated search process. The designer's inputs are formulated as a mutators agent because the designer is able to interact with and selectively modify any design in the design pool. To ensure that the designer's work is not undone, the designer's solution(s) may be retained in the pool. By changing the members of the design pool, the designer is able to direct the search process. As the search agents can work asynchronously, the designer can adjust his or her level of involvement needed.

1.2.3 Strategy #3 - Abstract the harness and environment representations differently to reflect process and agent related needs.

One of the biggest challenges of the cable harness routing problem is using a limited amount of computational resources to solve a large problem. Fundamental to the success of this process is determining a set of representations that can prune, via abstraction, large regions of the search space while still meeting the needs of the problem. Abstraction exploits the fact that if one abstracted design is superior to other abstracted designs, it is likely that the corresponding detailed designs will also be superior to those derived from the others. In addition to reducing the effective size of the search space, abstraction reduces the computational resources needed to evaluate each design because fewer constraints need to be addressed. In this work, separate cable

harness and environment representations were developed for different stages of the design process so that each could capitalize on representation-specific characteristic at the highest appropriate level of abstraction.

The harness can be viewed at various levels of abstraction: topology, topological routing, and detailed routing as shown in Figure 1.9 below. The set of abstraction enables routing information to be successively refined thus reducing the overall computational burden. For example, the wire count and bundle diameter can be determined using only the topology represented using nodes and links. The approximate length, and thereby costs, of each bundle can be determined using the topological routing (a coarse approximation consisting of the topology routed over a graph representation of the environment). This graph representation is sufficient to estimate the environment capacity needed for the routings, that is, the amount of free-space required for ample clearance with the cable harness. While the topological routing enables rapid evaluation and comparison of designs, it doesn't provide the detail needed for the final design. A more flexible routing representation, that is capable of bending naturally, is used to mimic the shape and paths of real bundles.

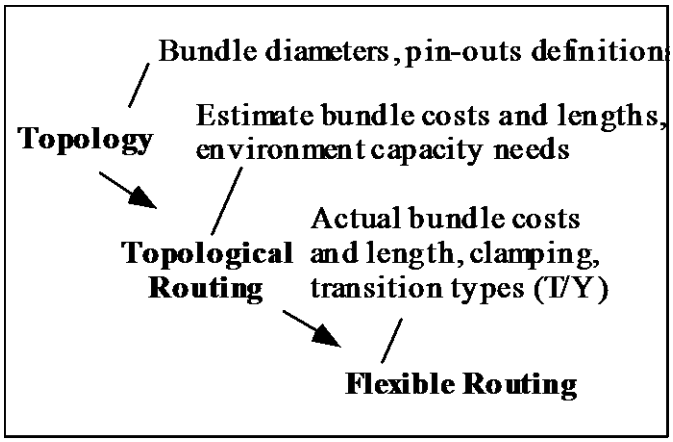


Figure 1.9 – The cable harness representational needs depend on the problem being addressed.

The different representations also reflect the varied needs of the routing agents. Because the automated routing agents typically evaluate thousands of harnesses, they require a representation that facilitates rapid design evaluation. The topological routing, by using a graph-based estimate of the cables' overall lengths, works well for quickly approximating the cost of a design. The

human designer, however, requires a different representation – one that offers more realism and flexibility. As discussed in Chapter 6, each bundle in the detailed routing representation is modelled as a string of spheres connected by springs. This representation, by being able to represent smooth, flexible curves that react naturally to forces and constraints, is suitable for human interaction. A consequence of the need to provide different representations of the harness for use by the automated agents and by the human designer is the need to translate back and forth between them.

Multiple environment representations were also developed to address the different geometric needs of the routing problem. Cable harnesses are typically used in applications where the components are closely packed together, and this in turn leaves minimal space for routings. The environment is characterized by a collection of open spaces connected by a network of narrow passageways. Since the amount of available free space changes as each harness is routed, the environment representation must make it possible to keep track of how much room is available during the routing process. In addition, as most input environment representations derive from detailed CAD models, the representation must abstract these models to facilitate rapid path planning.

As routing, in general, is a computationally intensive task, each aspect of the process must exploit abstraction where possible. To the end, several different representations are used for both the routing environment and harness to satisfy each of the different routing information needs. The environment representations need to:

- Display the environment
- Determine if the harness intersects with the environment
- Maintain the total bundle capacity of a given region in the environment
- Determine the connectivity of the different routable regions

Three general classes of environment representations — roadmaps, exact decomposition, and approximate decomposition — are candidates for the CHRP. Roadmap methods reduce the dimensionality of the search space by abstracting the free space with a sparsely connected

network, or graph. This abstraction has the obvious advantage of reducing the amount of computational resources needed to produce general paths for routings (approximate routes can be generated by traversing the network). It was therefore used by the automated routing agents to determine the topological routing, and to a lesser extent for the environment representation for the designer's interface, to maintain information about the capacity of various regions of the free space. The topological routing generated by the automated routing agents is represented as points on the free-space graph.

Exact methods are those that define the free space precisely using analytical functions. While these methods are good for use in domains requiring precise detail, such as robot path planning with interference checking, they are cumbersome and computationally too expensive for the cable harness routing problem, in which many solutions must be examined quickly.

Approximate methods, as the name suggests, approximate the environment to definable levels of detail. Approximate methods trade off accuracy for speed (and memory) by representing the free space using a set of uniform or hierarchical cells. By labelling which cells are devoid of obstacles, it is possible to determine if a point is in the free space by checking the cell in which it resides. The approximate (voxel) decomposition is well suited for representing the cable harness environment in that it gives a conservative estimate of the free space and facilitates rapid clearance calculations needed for cable-obstacle collision detection. It is therefore used as a supplement to the graph representation in the environment representation for the designer's interface.

Figure 1.10 summarizes the representations used to address the four environment needs of the CHRP – a boundary representation is used for the visual model, an approximate cell decomposition (voxels) representation is used to determine collisions, and approximate medial axis (AMA) is used to determine the maximum capacity of a region, and a graph abstracted from the AMA is used to determine the connectivity or channel information of the environment.

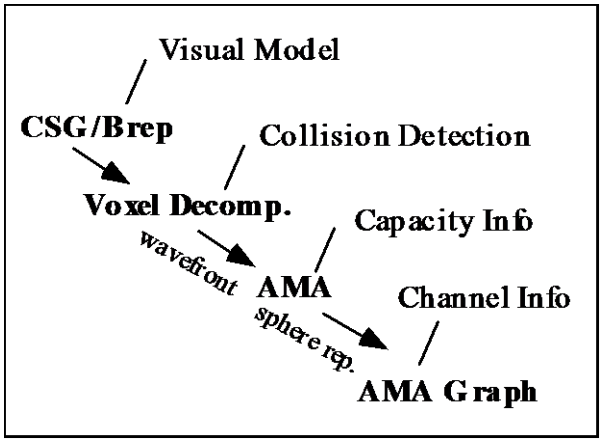


Figure 1.10 – Four different environment representations address the needs of the routing process.

1.3 Application

One of the tangibles of this work is the creation of a routing package capable of illustrating the use of the collaborative search methodology. This system, called CALIBER, has been used as a testbed to analyze the performance of the designer's interface, the automated search methods, and their ability to work as a team. It has been able to successfully route harnesses with up to sixty connectors and hundreds of wires.

The user-interface focuses on the interactive needs of the designer. The designer is able to view the harness in 3D, interactively manipulate the harness in real-time, and perform high level topological modifications. In addition to these interactive design operations, the designer is able to selectively share designs with the set of automated routing agents.

This software has been successfully tested in practice by designers at a large aerospace company [Crocker 1996]. Preliminary feedback has shown that the design time for cable harnesses using this system can be reduced from months to just a few hours.

1.4 Contributions

This work addresses the questions of "How can automation be used to best solve the CHRP," "How can a designer be immersed in the design process to handle unmodeled issues," and "What

environment and harness representations are natural matches for the needs of the designer and the automated solution." The solution presented was evaluated by testing it over a wide range of routing environments and harness types to examine the advantages and limitations of the routing system in terms of general robustness and performance. The results of these tests show that implicit collaboration between routing agents effectively produces designs that are superior to ones created by the designer or automated agents working alone. The system is able to evolve multiple designs simultaneously, reusing parts of previous designs to make new ones. The results validate three contributions of this work:

- The genetic-based collaborative architecture uses computational resources efficiently by using quality components of pre-existing designs to produce better designs.

- The designer-based, interactive design methodology enables both local and global modifications of the design in an intuitive manner and thus promotes rapid prototyping and management of unmodeled issues.

- The multiple levels of abstraction defined for this problem facilitate efficient hierarchical search of a huge search space.

1.5 Organization of the dissertation

The remainder of this dissertation is divided into seven chapters. Chapter 2 details the cable harness routing problem (CHRP), its representations in this work, and the attributes of good routings. A discussion of routing fragments, which are critical for effective design communication between routing agents, is also presented along with a review of previous research.

Chapter 3 presents the environment representations used for both the automated agents and human designer. The representations are tailored to the needs of both of these agent types. It also discusses various classes of routing problems that divide the problems according to the type of wirelists, the number of connectors, and the environment complexity.

Chapter 4 presents the collaborative architecture used to allow many different routing agents to share their design experiences. Also discussed is the decomposition strategy of separating the problem into topology design and routing and its ability to reduce the size of the research space.

Chapter 5 describes each of the three major classes of automated agents used in this work and details the issues related to design selection and reuse.

Chapter 6 presents the harness representation that enables the human designer to interact naturally with the design. This representation takes into account the relationship between bundle radius and bending stiffness, collisions with obstacles in the routing environment, and the bookkeeping required to enable high-level interaction.

Chapter 7 presents the experiments done to test the effectiveness of the collaborative architecture. The goal is to show that a combination of simple routing agents can feed off of each other's strengths to produce designs that are consistently superior to ones produced if they worked alone and to do so using less computational resources than if they worked sequentially. Several different classes of routing environments and harnesses are described that illustrate the diversity of the problem.

Chapter 8 summarizes this work and its contributions and then discusses potential future directions.

Chapter 2

The Cable Harness Routing Problem (CHRP)

Plainly stated, the cable harness routing problem is to find the least cost routing for a cable harness in a cluttered, 3D environment. This chapter builds on this description by defining the components of the CHRP and qualities of good routings as well as discussing related work. It then defines a number of problem classifications that will be used in later chapters to evaluate the robustness of the developed architecture and routing agents.

2.1 The Cable Harness Routing Problem

2.1.1 Cable Harness Components

Given a sufficient degree of abstraction, a cable harness can be decomposed into four basic components—end connectors (ports), transitions, bundles, and the cable harness configuration (also referred to as the harness topology). These parts are shown in Figure 1.2 and described below.

Ports, labelled P1-P7 in Figure 1.4, connect the cable harness to the electrical assemblies. Each of the harness' nP ports, $P_i \in \mathbf{P}=\{P_1,P_2,...P_{nP}\}$, contains a subset of the total nW wires, $\mathbf{W}^{Pi}=\{W^{Pi}_1, W^{Pi}_2, ... W^{pi}_{nw}\}$. A 4x4 homogeneous transform, P_i^{xyzo}, is used to define each port's x, y, z location and orientation in Cartesian space.

Transitions, sometimes called junctions, are labelled T1-T5 in Figure 1.4, and they form the branching points among the bundles. Each of the harness' nT transitions, $T_i \in \mathbf{T}=\{T_1,T_2,...T_{nT}\}$, connects to n (typically 3) ports or other transitions. The x, y, z location and orientation Cartesian space of each transition is defined by a homogeneous transform, T_i^{xyzo}. Each of the n branches of a transition is also specified by a homogeneous transform T_{ij}^{xyzo}, relative to T_i^{xyzo} as shown in the figure below.

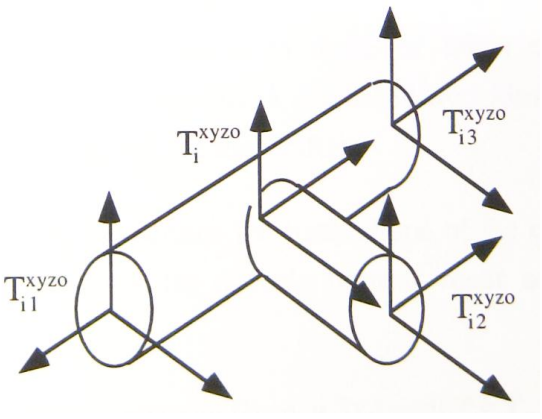


Figure 2.1 – Transitions typically have three branches to other ports or transitions – each branch is defined with its own offset and orientation

The set {**P**,**T**} contains all the *nodes* of the harness. All neighboring ports and transitions (nodes) can be recorded in an n-tuple, $N^{Ti} = <N_1^{Ti}, N_2^{Ti},...,N_n^{Ti}>$. As many transitions are

standardized parts that can only have a discrete number of wires, the actual number of wires going towards each neighbor is also recorded in an n-tuple, $W^{Ti} = < W_1^{Ti}, W_2^{Ti},...,W_n^{Ti} >$, for part selection purposes.

Bundles are defined as the segments of the cable harness that connect any two adjacent nodes (ports or transitions). It is straightforward to show that a harness with nP ports contains 2nP-3 bundles. Each of the bundles, $B_i \in \mathbf{B}=\{B_1,B_2,...B_{nB}\}$, contains W^{Bi} wires and connects two harness nodes; these nodes are recorded as $N^{Bi} = < N_1^{Bi}, N_2^{Bi}>$. Bundles that connect two transitions are called *transbundle*s, while those connecting a transition to a port are called *termbundles*. Bundles also record path information between their end nodes.

The term "path" is a term requiring further discussion as its definition depends on which routing representation is being used. For example, a distinction is made between a path of transitions within a harness and a harness' own physical routing. In this work, a wire path, $p_{harness}(W_i)$, connecting two ports, P_i and P_j, within a harness, is considered to be a connected series of n distinct transitions, $T_k \in N$, such that $P_i \in N^{T1}$ and $P_j \in N^{Tn}$. On the other hand, $p_{route}(W_i)$ is the actual path for the wire as it is routed in the environment's free space. It is geometrically constrained in that its orientation must match that of the ports and transition branches it connects. A third path, the wire path, denoted by $p_{graph}(W_i)$ is similar to $p_{harness}(W_i)$ but consists of nodes of a graph (see Chapter 3) instead of nodes of a harness. Paths of bundles are analogous to those of wires and are represented as $p_{harness}(B_i)$, $p_{route}(B_i)$, and $p_{graph}(B_j)$. Appendix A discusses algorithms used in this work for the point-to-point planning component of the CHRP.

The *Cable Harness Topology*, C, defines the basic shape of the cable harness as if it lay flat on a table. A valid topology for a cable harness must adhere to the following constraints:

1. Degree(T_j) is bounded (typically fixed at 3) for all $T_j \in \mathbf{T}$
2. $p_{harness}(P_i,P_j)$ exists for $P_i, P_j \in \mathbf{P}$, $P_i \neq P_j$
3. $p_{harness}(P_i,P_j)$ is unique for $P_i, P_j \in \mathbf{P}$

The third constraint is illustrated by the following figure. The routing on the left shows a valid cable harness topology in that there is only a single path between any two ports. The routing on the right shows a related problem to the CHRP, called wire bundling, where wires are first routed individually and then bundled together for support and maintainability. In this case, the third constraint is relaxed to allow cycles – multiple paths between points on the routing.

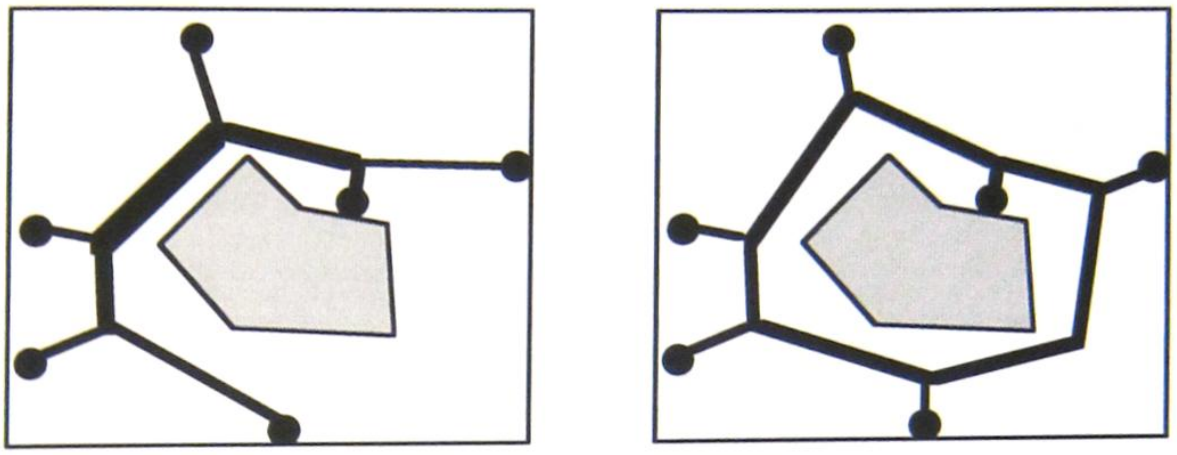


Figure 2.2 – The topology of a cable harness is different than that of a wire bundling in that cycles are prohibited.

Beyond satisfying the above constraints, an ideal topology representation should:

- Be capable of defining every possible topology.
- Be unbiased in the sense that each topology is represented only once.
- Have a bi-directional mapping with the physical harness.
- Posses the property that small changes in the representation correspond to small changes to the topology and vice-versa.

As discussed in Chapter 1, the solution methodology for the CHRP relies on isolating parts of designs (called fragments) and reusing them in the search process. Isolating fragments requires a topology representation that maintains locality (i.e., the ability to store information regarding contiguous components of a design in contiguous components of an encoding).

At present, there are several existing tree representations [Palmer 1994], including characteristic vectors (list of edges in the tree), rooted trees that record parentage of each node, Prüfer numbers [Moon 1967], and Biased Link/Nodes [Palmer 1994]. Of the different techniques, the Biased Link/Node representation possesses the greatest amount of locality. The biggest problem with these encodings, however, is that they isolate the structure of the topology from the factors on which the quality of the fragments depend. That is, these encodings do not retain any information pertaining to the topological routing in space. This work avoids the problem by operating directly on the physical representation of nodes and links (called the phenotype in genetic problems) instead of an abstract encoding (called the genotype in genetic problems) [Goldberg 89].

2.1.2 Definition of Routing Objectives

Designers of cable harnesses must juggle many constraints and objectives during the routing process. Aside from the issues of reliability, the objective is typically to minimize a weighted average of a collection of cost functions. The cost functions usually include factors for physical weight and material and/or manufacturing costs. The objective function used in this work is defined in equation 2.1.

$$F = \sum_{i=1}^{i=nB} cost(nW^{B_i}) \cdot len(l^{B_i}) \qquad \text{Equation 2.1}$$

where nW^{B_i} = number of wires in bundle i

l^{B_i} = length of bundle i

cost = unit cost for bundle, e.g;

$$cost(nW^{B_i}) = \alpha + \beta nW^{B_i}$$

len = adjusted length of bundle i, e.g.;

$$len(l^{Bi}) = \begin{cases} l^{Bi} & l^{Bi} \geq l^{Bi}_{min} \\ (S+1)l^{Bi}_{min} - Sl^{Bi} & l^{Bi} < l^{Bi}_{min} \end{cases}, S > 1$$

where $l^{B_i}_{min}$ = min. bundle manufacture length of bundle i.

The cost function offers an estimate of the actual dollar costs of a cable harness based on the number of wires passing through each bundle and the economies of scale associated with adding

additional wire to a bundle. This occurs, in part, due to standardized sizing of bundle conduits. That is, as the wire capacity of the conduit is fixed, the bundle has a large initial cost. The objective function also uses a penalty function on short bundles since they present an assortment of manufacturing problems. The use of a penalty function to impose the minimum length constraints, while not guaranteeing that the constraint will be satisfied, greatly simplifies the search process in that all designs are feasible and therefore are not discarded in the search process. This is important as the search technique in this work, discussed in Chapter 4, can reuse fragments of previous designs regardless of the design's feasibility.

The most computationally expensive facto of the objective function is the determination of the bundle lengths. This is addressed by using a coarse approximation of the path during the evaluation process. The path is generated over a sparce graph that represents the free-space of the environment as illustrated in Figure 2.3 below (The graph is derived from an approximation of the free-space, as discussed in Chapter 3). The first step is to map the spatial coordinates of the ports of the topology (A) to nodes in the graph (B). Next, graph nodes are chosen for each of the transitions and bundles are routed over the graph (C). Although the routings created in this manner do not share the same shape as the final routing (D) and may have longer bundle paths, the computational savings are substantial. A more realistic path is generated only after a near-optimal routing is found.

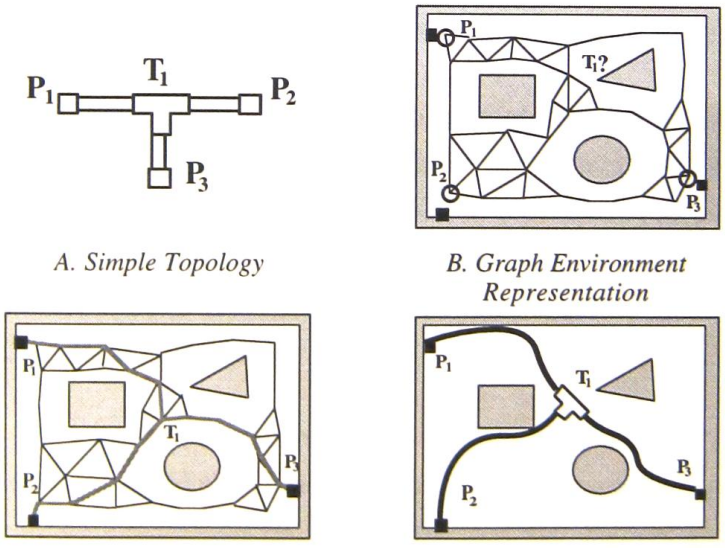


Figure 2.3 – The computational expenses required to route a topology (A) in an environment (D) is lessened by approximating the bundles paths © using a coarse graph (B).

The graph-based harness routing problem is defined as follows:

> Given a bi-directional graph in three-dimensional Cartesian space, **G**, of nG nodes and nE edges, a harness topology, C, of nP ports all mapped to graph nodes on **G,** and a list of nW wires each connecting two ports, choose a subset, **G'**, of the edges of **G** which minimizes an objective function while ensuring that there is a single, connected path in **G'** for each of the wires, the maximum degree of any graph node in **G'** is bounded, and that the topology of **G'** matches the topology of the harness C.

Since the shortest path between the end nodes of each bundle, as determined by a point-to-point routing technique, will always be used to define its length, finding the harness routing boils down to finding the topology that has the lowest cost when its transitions are placed in **G**. In this case, **G'** is then the union of all the links within the shortest paths found between the end nodes of all the bundles.

The objective function parameters mentioned so far are easily quantified and are therefore well-suited to computerized search. Nonetheless, as in any real-world engineering problem, there are other, unmodeled objectives which are much more qualitative in nature and therefore more easily handled by a human designer. Examples of such objectives are:

- Avoiding hazardous regions, such as those with high temperature or electrical fields, if possible, to eliminate the need for additional cable insulation and shielding. The difficulty here is in determining at what point it is better to cross a hazardous region rather than to circumnavigate it. While a penalty function could be used, determining whether or not the costs are consistent with those used in the other objective cost parameters would be problematic.

- Securing cable harnesses to the routing environment. The determination of "good" clamping regions typically requires external knowledge of the routing domain (e.g., which obstacles are candidates for clamping).

- Ensuring that cable harnesses are easy to install. This requires a versatile understanding of the installation procedures, computationally intensive harness installation testing routines, and additional qualitative cost function parameters.

- Ensuring that cable harnesses do not block components that require frequent maintenance and repair. Again, more external information is required to automate this factor.

- Making sure that cable harnesses are not routed directly across an open space. Cable harnesses routed across an open space not only reduce the accessibility of a region but also tempt maintenance personnel to use the harness as a hand hold - a dangerous action that could cause functional failure or worse.

2.1.3 Attributes of a Good Routing

Before reviewing relevant items from the literature, it is useful to conclude our discussion of the nature of the CHRP with some observations about the characteristics of good versus inferior solutions In general, good designs share these common properties:

- They tend to be as short as possible while still avoiding dangerous regions. Since the cost to add one more wire in a bundle decreases as the number of wires increases (i.e. there are economies of scale), low cost harnesses often have short thick bundles.

- They minimize a composite objective function consisting of material costs, manufacturability, serviceability, and survivability.

- Most have short "term bundles" (bundles connecting transitions to ports). This occurs when the "backbone" of the harness passes close to each of the ports.

- Transitions connecting two ports are usually placed near the pair when the ports, themselves, are near one another.

- The set of transitions stay within the convex hull of port locations.

- The bundles usually do not double back on themselves.

The following figure shows a topology that doubles back on itself and one that doesn't. Designs such as the one on the right are *typically* of lower cost than designs like the one on the left. There are some occasions, however, when the topology and routing of the design on the left is actually more cost effective (e.g., if a disproportionate number of wires pass along the switchback). This illustrates the fact that the overall cost of the routing is dependent on the wiring list as well as on the arrangement of ports and obstacles.

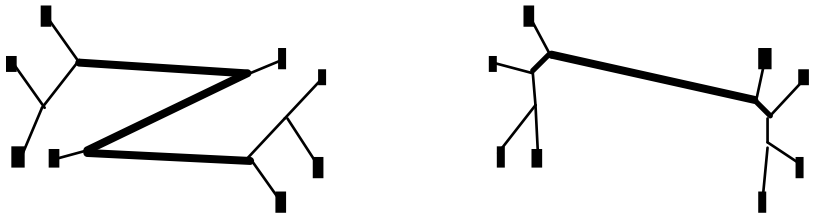
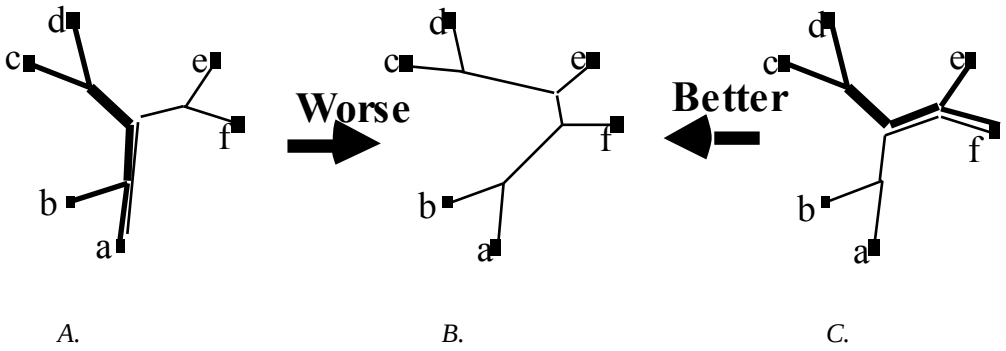


Figure 2.4 - Routings that double back on themselves are usually a result of poor topology selection (line widths correspond to the number of wires in each bundle).

- The wires tend not to be much longer than the straight line distance between the ports.

This property depends on the topology of the harness, the shape of the environment, and the allocation of wires between ports. Figure 2.5 shows an example where the relative costs between two topologies change depending on their wiring distributions. The two outside topologies (A & C) are identical and the harnesses only differ in their wire lists. Of the first pair of topologies (A & B), the one on the left is superior since most of the wires connect the fragment ab to the fragment cd and the path between them is shorter. The opposite is true for the second pair of topologies (B & C). As most of the wires connect fragment cd to ef, the middle topology has the lower cost.



A. *B.* *C.*

Figure 2.5 - The optimal topology is dependent on the wire list.

2.2 Related Routing Problems and Previous Work

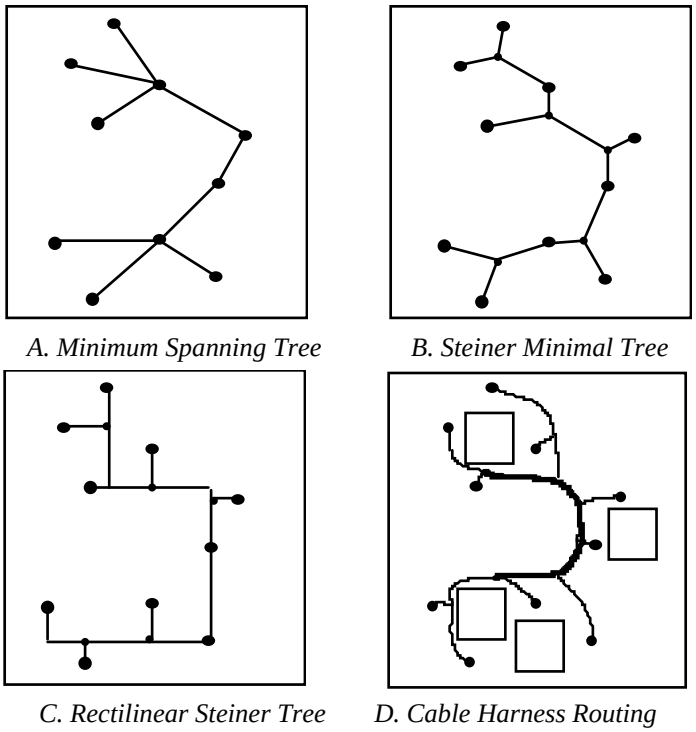
The CHRP shares similarities with a number of other routing problems that have an underlying tree topology. Table 2.1, below, compares the issues related to an illustrative sampling. VLSI refers to the routing of multiple-terminal-nets (i.e., trees), wire routing refers to the routing of individual wires into larger assemblies. The VLSI routing problem is strongly constrained in that the paths of the links must bend at prescribed angles and use predefined routing channels. Heavily researched, the most closely related work to the CHRP is in the routing of such trees among obstacles. A few examples will be discussed later in this section. One difference from the CHRP is that the unit cost per length of all the links in a VLSI routing are almost always the same. Wire routing, and to a larger extent, wire bundling share the most similarities to the CHRP. Both of these routing problems shave similar routing environments, clamping and bending constraints as the CHRP. However, both have less stringent bundling, installation and manufacturing constraints. Pipe and duct routing can be divided into two problem categories; point-to-point routing and one-to-many routing. Point-to-point routing, where two points are to be connected by a single routing, is closer related to wire routing than the CHRP. However, one-to-many routing, where many locations are to be connected via a tree topology to a central source or sing, share many similarities with the CHRP. In addition to having similar tree structures, the capacity (i.e., cost) of each link of the trees is depend on the topology and load between the points being connected.

	Cable Harness	VLSI (SMT)	Wire Routing	Wire Bundling	Pipe Routing	Duct Routing
Clamping is important	Yes	N/A	Sometimes	Yes	Yes	Yes
Separate routings can be bundled	Yes	No	Yes	Yes	No	No
Installability a factor in path planning	Yes	N/A	No	No	Sometimes	Yes
Has constraints on curvature of paths	Yes	Typ. Bends at 45 or 90°	No	No	Typ. Bends at 45 or 90°	Typ. Bends at 90°
Shortest path of indiv. wires always preferred	No	Yes	Yes	Yes	N/A	N/A
Routing parallel to obstacles/walls	No	N/A	No	No	Yes	Yes
Routing complexity increases with topological complexity of the environment	Yes	Yes	No	Yes	Usually	Yes
Desirable route along channels	Yes	Yes	Yes	Yes	Yes	Yes
Require a tree-shaped topology	Yes	Yes	N/A	No	Sometimes	Usually

Table 2.1: Comparison of routing issues among various routing problems

The search for a good cable harness topology has similarities to a number of spanning-tree problems. The simplest of which is the Minimum Spanning Tree (MST) problem (Figure 2.6A shows an example MST for a set of ten points). The goal of the MST problem is to find the least cost tree that connects all of the vertices in a graph, **G**. It is solvable in O(E logE) time where E is the number of edges in the graph [Prim 1957; Kruskal 1956]. This greedy algorithm works by sorting the links (i.e., edges) in ascending order with regards to cost. Links are then added to a list of desired links so long as their inclusion does not result in cycles with other selected links. The problem of using this technique for the CHRP is that the cost of links is a function of the complete routing of the harness. This results in a "Catch-22" because it is impossible to sort the

links by cost until the topology is routed. Even if links are selected based on length, the resulting tree would not necessarily be the minimum cost as the bundle cost is a function of both its length and number of wires it contains. Harness routings using the MST also frequently violate the maximum allowed order for a transition (number of bundles emanating from a transition) as well as minimum bundle lengths constraints.



A. Minimum Spanning Tree *B. Steiner Minimal Tree*

C. Rectilinear Steiner Tree *D. Cable Harness Routing*

Figure 2.6 - The CHRP is related to a number of simpler tree spanning problems

If additional points are introduced to shorten the length of the MST (not necessarily the length of wires in the harness), the new points are called Steiner points and the resulting tree is called a Steiner tree (SMT). This is illustrated in the close-up of figures 2.6A and 2.6B shown below. While the total length of the tree (lines in the figure) connecting points A,B, and C is *always* shorter in the Steiner Tree, the distances between A-B, A-C, and B-C are not.

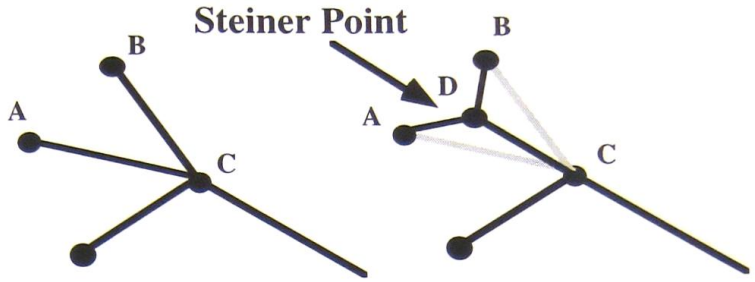


Figure 2.7 – Close-up of the addition of a Steiner point between three nodes

Steiner tree problems can be divided into three formulations based on the distance metric for each path : Euclidean (Figure 2.6B) , rectilinear (Figure 2.6C), and graph-based. The graph-based formulation, in which a tree is found over a graph, is most interesting with respect to the CHRP in that it more readily addresses obstacles. The graph-based Steiner minimum tree problem is formally defined as follows:

> Given a graph, **G,** with costs associated with each link and a subset of vertices called demand points, **D**, **D**$\in$**G**, find the tree of minimum total cost that connects each of the demand points.

The SMT problem degenerates to the MST problem when **D**=**G** and to the well-known shortest path problem when **D** contains only two points. The SMT, of which the CHRP is a special case, has been shown to be NP-complete [Karp; Garey & Johnson 1977]. Cayley's Formula specifies the number of distinct spanning trees on a complete graph of n points to be n^{n-2}. [Cayley 1889, Even 1973].

While it will be shown that the CHRP is sufficiently different from the SMT to preclude direct applicability of previously developed search methods, these methods offer ideas which may be tailored to the CHRP. Traditional search methods for the SMT problem can be categorized as enumeration and heuristic-based. While enumeration techniques that may evaluate every tree permutation can be used for small graphs (nG < 30), heuristic methods must be used for larger graphs.

Examples of enumeration include the enumeration of minimum spanning trees (STEA), topology enumeration (TEA), dynamic programming (DPA) and a host of IP formulations including a set

covering algorithms (SCA), implicit enumeration algorithms (IEA), Lagrangean relaxation algorithms (LRA), and the dual ascent algorithm (DAA). In the spanning tree enumeration algorithm (STEA) [Hakimi 1971], the minimum cost Steiner tree is found by enumerating all minimum spanning trees (MST) of subnetworks of **G** induced by subsets of **W** of **D** such that **D** $\subseteq$ **W** $\subseteq$ **G.** As the time complexity of STEA is $O(p^2 2^{n-p}+n^3)$, it is only appropriate for very small networks ($n<25$). Topology enumeration algorithms,, like the one formulated by Melzek [Melzek 1961], use a restrictive definition of the topology of the spanning tree of **W** to reduce the total number of trees to be examined. However, it requires a preprocessing step in the tree generation process. Dynamic programming has been used by Dryfus and Wagner [1971] to exploit the optimal decomposition property where optimal solutions of smaller problems are found and retained for us in solving larger subproblems. Suboptimal subproblems can be discarded and not reconsidered when solving larger problems. While the time complexity is smaller, $O(n3^p+n^2 2^P+n^3)$, it is only applicable for Steiner problem (i.e., not the CHRP) as how the subproblems are reused do not affect the quality of the subproblems.

Aneja [1980] formulated the Steiner problem as a set covering problem (SCA) as follows. First, a partition $\{W,\bar{W}\}$ of **G** is defined such that $D \cap W \neq \emptyset$ and $D \cap \bar{W} \neq \emptyset$. Next, all possible cut-sets (set of graph links that span $(\{W, \bar{W}\}$ for all such partitions are enumerated as $C_1, \ldots C_q$. The links of **G** are then defined as $e_1, e_2, \ldots e_{nG}$ and a matrix A as,

$$a_{ij} = \begin{cases} 1 & \text{if } e_j \in C_i \\ 0 & \text{otherwise} \end{cases}, \; i = 1, 2, \ldots q \text{ and } j = 1,2, \ldots nG$$

With this definition, the Steiner tree problem can be formulated as a linear IP problem, shown below, provided that the cost per unit length of each link is constant for all lengths.

$$\min \sum_{j=1}^{nG} c_j x_j$$

$$\text{subject to} \qquad \sum_{j=1}^{nG} a_{ij} x_j \geq 1, i = 1,2, \ldots q$$

$$x_j \in \{0,1\}, \; j = 1,2, \ldots nG$$

While the number of constraints in this formulation grows exponentially with the size of the problem (nG), the SCA is able to handle the constraints implicitly. The main problem with using the SCA on the CHRP is that the number of nodes in the graph routinely surpass 1000 and therefore overwhelms any simplex routine. Additionally, the factor, $c_i x_i$, for the CHRP requires additional constraints to accommodate the variable nature of the cost function c_i.

Many other IP enumeration formulations have been developed with the goal of reducing the effective size of the search space with the use of problem-specific knowledge in branch and bound methods. The primary problem of using this class of algorithms on the CHRP is that the optimization of each partition assumes that how it is connected to the rest of the solution does not affect the qualities (i.e., lower bound) of the partition's solution. The first formulation is the Implicit Enumeration Algorithm (IEA) of Shore [1982]. In this algorithm, the set of all feasible solutions, trees that span the set of ports, is systematically separated into smaller subsets based on partitions of **G.** The subtrees of each of these subsets are then analyzed using upper and lower bounds to determine if further testing is warranted. Beasley [1984] developed an enumeration algorithm (LRA) which differs from Shores' in the way it calculates the lower bounds of the Steiner tree that connects the ports in each graph partition. Formulated as a 0-1 LP problem, the connectivity constraint of the tree solution is maintained by using commodity flow constraints.

While computational comparison for most of these algorithms are rare, Fouolds and Rayward-Smith [1983] ran comparative tests on STEA, DPA, and IEA. Several randomly sized (nP) trees were routed on complete graphs (i.e., a link connects all pairs of nodes) of nG=10,20, and 30. They found that for cases where nP<<nG, DPA worked the best. IEA worked the best when 2nP≅nG and DSTEA work best when nP≅nG. For Steiner problems, the most promising technique for large nG is LRA. Using a CRAY-1S, they were able to solve a nG=500 sized problem in twenty minutes.

Many heuristics have been defined for the Steiner problem [Winter 1981], and most build the Steiner tree incrementally. For example, Wang & Liu [1991] generated a Steiner minimum tree by choosing an un-routed demand point closest to a partially routed tree and then finding the shortest path from that point to the partially routed tree; they repeated this process until all the ports had been routed. The time complexity of their technique is $O(kmn^2)$ where k = the number of neighbor vertices of ports, m = the number of ports, and n = the number of vertices.

Another class of heuristics begins with other tree structures and then converts them into Steiner trees. G. Ho and C.K. Wong [1990] describe two algorithms for converting a minimum spanning tree into a rectilinear Steiner tree (a rectilinear Steiner tree is similar to the Euclidian Steiner tree except that the bundles run along right angles as shown in Figure 2.3C) in an obstacle-free environment. The conversion is done by taking each edge of the MST and turning it into a staircase made up of edges on the underlying grid. This grid is defined by all the horizontal and vertical lines intersecting all the demand points. Ho and Wong show that the Steiner trees generated are at most 1.5 times as costly as the minimum spanning trees.

A number of techniques address the problem of obstacles in the routing environment. Miriyala and Sherwani [1991], for example, used an approach that starts with a simplification of the problem and then adds complexity incrementally to find Steiner minimum trees when all the ports lie on the inside of a rectangle. They begin with an optimal routing (found via enumeration) for the case where only one rectangular obstacle is in the middle of the rectangle. They then add obstacles one-by-one and re-route all the paths which intersect the obstacle. As this approach is reactive in nature instead of using the global layout of the obstacles, the quality of the routings may be poor, especially as the complexity of the routing environment increases.

While these heuristic methods are able to produce routings quickly, they do not consider the "load" (number of wires in the case of the CHRP) between Steiner points [Lee 1980; Winter 1987] and therefore may not be directly appropriate for the CHRP. The following figure shows an example where SMT heuristics fail to produce near-optimal solutions when applied to the CHRP. The figure on the left shows the SMT solution (where the sum of the lengths of the links in the routing is minimized). However, since the majority of the wires must be routed the long way around the obstacle, the cost of the harness is much higher than the cost of the routing on the right. Of course, other CHRP objective parameters, such as installability and maintainability, may favor the routing on the left as the overall length (called the tree diameter) is shorter and the bundles are more sturdy.

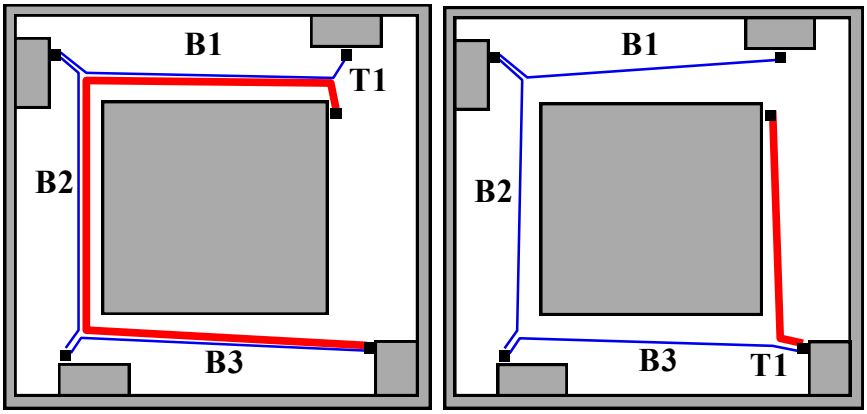


Figure 2.8 - The Steiner minimum tree algorithm may produce suboptimal routings for the CHRP.

A number of genetic approaches for searching for minimal Steiner trees have been recently formulated in environments without obstacles. The first is Hesser, Männer, & Stucky [1989] who used a standard genetic (binary) encoding of the x-y real-value locations of all the Steiner points in the routing. The connectivity of these nodes was found by applying a MST algorithm. They were able to show that the genetic approach performed as well as simulated annealing and a specialized SMT technique. Julstrom [1992] developed an genetic formulation for the Rectilinear Steiner Tree problem. He showed that better designs could be found faster by performing cross-over at the phenotype level instead of the genotype level. Both formulations did not use domain knowledge to actively predict which parts of the design should be retained for future generations. Actively predicting which parts of the design to reuse will be shown in Chapter 7 to be effective in genetic searches.

2.3 Inputs to the Cable Harness Routing Problem

The CHRP begins with the consideration of several raw inputs—the port locations, routing boundary, obstacle information, and the pin-to-pin wiring list. This section discusses several classes of routing environments and wiring lists that are typically seen in practice. Classifying the problem will be shown to be important in Chapter 7 as the solution performance is found to be dependent, in part, on the class of problem being tackled.

2.3.1 Routing Environment

The environment, E, through which the harness is routed, is initially defined by a constructive solid geometry (CSG) or boundary-rep datafile. Figure 2.9 shows an example routing environment consisting of boundaries, obstacles, and port locations. The environment is converted into both an approximate cell decomposition [Latombe 1989]. and a graph network as discussed in more detail in Chapter 3.

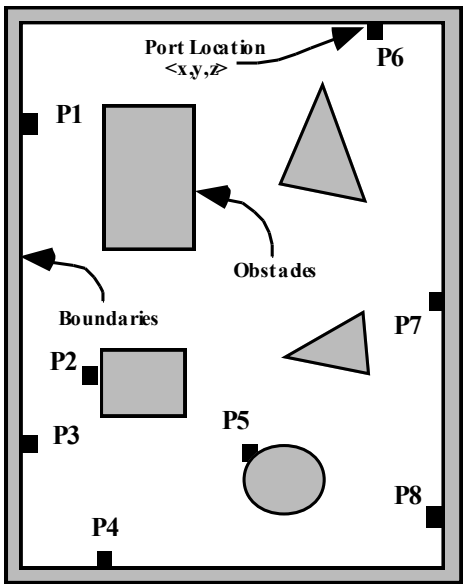


Figure 2.9 - Plan view of a sample routing environment (the Actual obstacles and boundaries are three-dimensional)

The complexity of the free-space of the routing environment is roughly related to the connectedness of the free-space's topology. That is, "complex" environments typically have may topologically different paths between any two points in the free-space. The simplest environment contains no obstacles. The optimum routing for each harness topology in this environment can be found wit gradient-based search so long as no minimal bundle length constraint exists. While the objective function in the CHRP does have such constraints, the chances for the routing to become stuck on a local optima are far less than for environments with obstacles. As it is more probable that a near optimal routing of a particular topology will be found for this environment type, it is used as a limiting case to test a routing agent's or routing team's ability to generate good topologies.

More complex routing environments, like the ones shown below in Figure 2.10, have many regions that are connected by paths that may pass through different channels. The example on the right is considered far more complex as there is a high probability that multiple paths exist between any two points. Most real-world routing environments, such as engine compartments and computer assemblies, fall into this category. Since randomly placed transitions of even good topologies will not settle down to good routings using gradient methods, complex environments will favor routers that both produce good topologies and that are able to place transitions well.

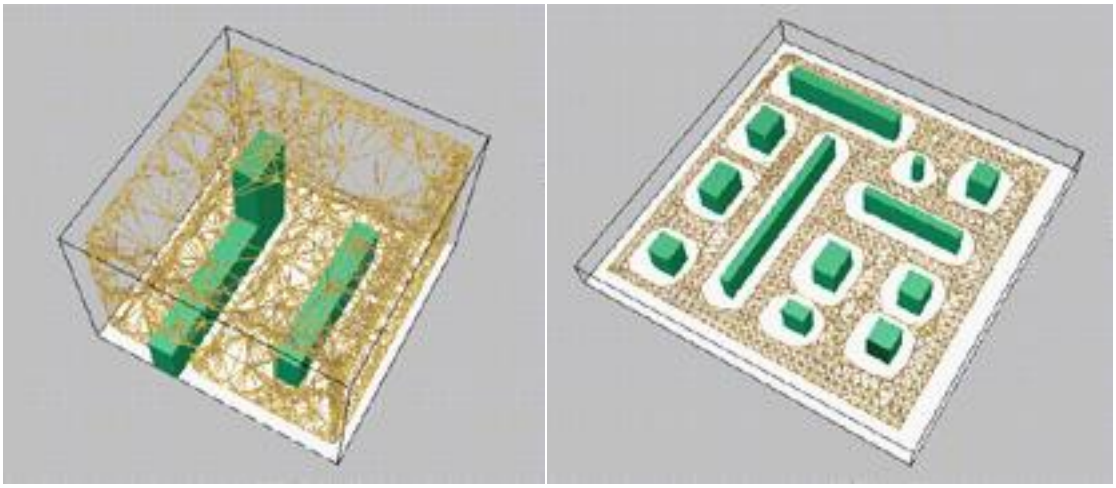


Figure 2.10 – Simple and complex routing environments with solid obstacles. (The free-space regions are shown with the nodes and arcs of a routing graph, to be discussed in Chapter 3).

2.4.2 Wiring List

The wiring list defines how many and what kind of wires are to be connected between ports of the harness. The number and distribution of the wires in the list are key considerations. The bias in the wiring lists, defined as the extent to which the number of wires varies from port to port, greatly influences the overall shape of the harness as well as the performance of various routing methods. The following subsections describe several different levels of wire list bias which are used later to examine each routing method's ability to account for bias.

The most common distribution of wires in a harness is the normal or balanced wire list. In this case, no single port or pair of ports has a clear majority of wires. A wiring list in which all the ports have the same number of connectors to all other ports would have the highest degree of

balance. In practice, however, these *perfectly* balanced wiring lists are rare. Harnesses with balanced wire lists may be problematic to some heuristic routing techniques that pick a starting point based on wire count and build a routing incrementally - each starting point is equally attractive.

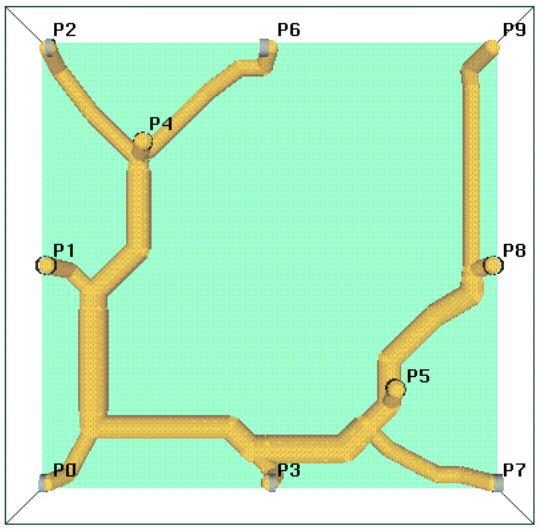


Figure 2.11 - Harness with balanced wire list

A harness is called biased when the wire list favors a subset of the ports. Figure 2.12 shows a case where two ports (P1 and P5) contain the majority of wires. In this case, the topology of the harness fragments containing the remaining ports (e.g., the fragment containing P2, P4, and P6) has little effect on the overall cost of the harness. Some wire lists have several pairs of ports that have a disproportionate number of wires and are classified as multibiased. Multibiased wire list are interesting as they often lead to optimal designs that are difficult for designers to find. This occurs as the optimal routing is more likely to have bundles that criss-cross. Harnesses with these kinds of biased wire lists can present difficulties to genetic approaches, discussed in Chapter 4, during the design selection phase. This occurs as the difference in fitness between designs that share the same routing "backbone" is usually negligible.

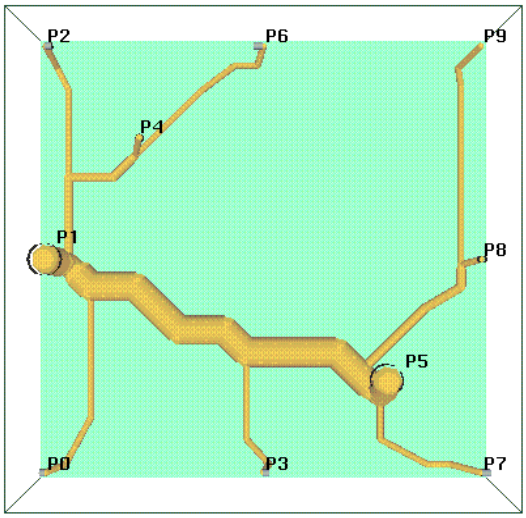


Figure 2.12 - Harness with single-biased wire list.

A wire list in which all the wires share a common port, as shown below, is the extreme opposite of the balanced wire list. Examples of this class of wire list include control and power harnesses that carry signals from a single location to all the other port locations.

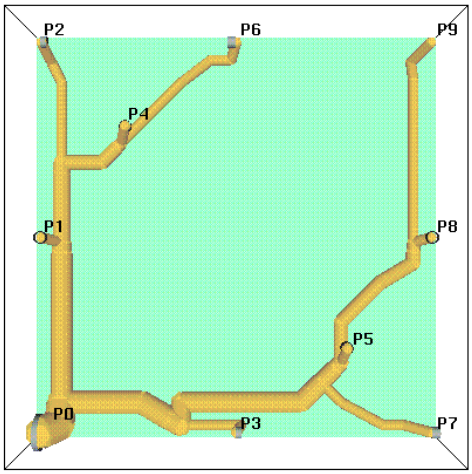


Figure 2.13 - Harness with single port wire list

2.4 Size of Cable Harness Configuration Search Space

As previously mentioned, the size of the search space for this NP-complete problem quickly grows out of control. The problems can be seen by first examining the most simple harness and adding complexity incrementally. The simplest possible harness has three ports and a single transition and hence only one unique topology as shown in Figure 2.3A above.

Since finding the shortest path between two points is straightforward, the only problem is in determining where to place the lone transition. Figure 2.14 shows two possible placements for a transition.

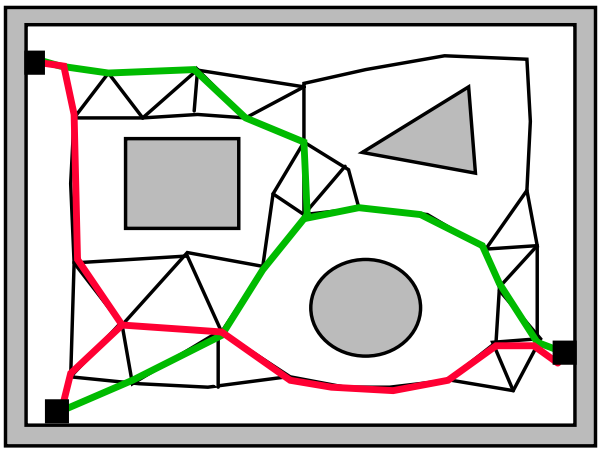


Figure 2.14 - Two potential routings of a 3-port harness.

At first glance, determining the lowest cost harness appears as simple as dropping the transition in the graph and moving it locally to the global optimum. If this were true, then the problem would be solved in linear time with respect to the number of transitions and graph nodes.

However, it can be easily illustrated that the search space has multiple local minima as shown in Figure 2.15 below. Here the cost of the harness, resulting from placing the transition at each node, is shown for each node of the graph. (The nodes correspond to a decomposition of the free space as discussed in Chapter 3. For the purposes of the present discussion, they are simply a finite set of locations in the free space at which the transition can be placed and the cost of the harness computed). If the transition is first placed at any of the nodes with a large dot and moved

locally to neighboring nodes, the harness will become stuck at a local minimum of 151 (transition at A) instead of the global minimum of 122 (transition at B).

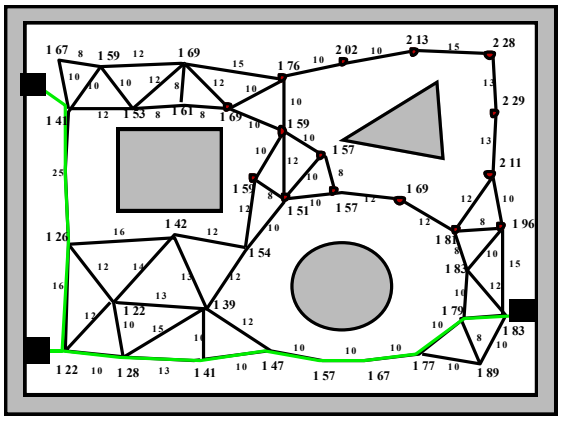


Figure 2.15 - A gradient search for a 3-port harness results in a local optimum.

In this simple case, only two local minima exist, but many nodes (i.e., locations) must be tested to improve the chances of finding the global minimum. This presents more problems as the number of ports increases. For example, finding the minimum cost location for the transition could become as computationally costly as checking the cost of every permutation of the harness with every transition placed at every one of the graph nodes. Table 2.1 shows how the number of topologies and routings for each topology increases as the number of ports increases. Essentially, two things happen: First, the number of topologies grows factorially as defined by Hyafil [1973], in the equation below

$$O(\mathrm{nP}) = \frac{(2\mathrm{nP}-5)!}{2^{\mathrm{np}\text{-}2} \cdot 2(\mathrm{nP}-2)!} = 1 \cdot 3 \cdot 5 \cdots (2\mathrm{nP}-5)$$

where nP is the number of ports of the harness. This growth can be seen in Figure 2.16 which lists all the different topology shapes for harnesses with two to nine ports. The number below each shape is the total number of topologically distinct permutations (i.e., considering symmetry properties) of port layouts possible for the shape.

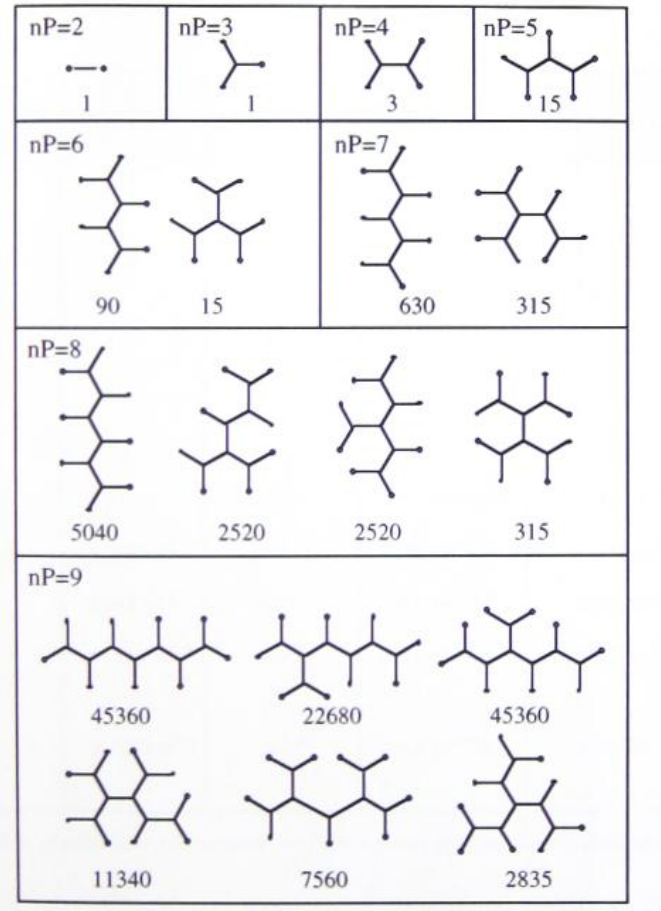


Figure 2.16 – Distinct topologies for 2 to 0 port harnesses

Second, the number of possible routings for each of these topologies increases at a rate of nG^{nP-2} where nG is the number of possible locations for the transitions. This quickly makes exhaustive search impossible. The final column in Table 2.2 shows the number of distinct designs (unique topologies and topological routings) possible for a modest sized environment with 1000 graph nodes. As the number quickly surpasses even the fastest computers, it is therefore important to devise a solution that does not need to exhaustively examine every design possibility.

# Ports	# Topologies	# Routings/ Topology	# Topological Routings	# Topological Routings (nG=1K)
2	1	-	1	1
3	1	N	1N	$1X10^{3}$
4	3	N^{2}	$3N^{2}$	$3X10^{6}$
5	15	N^{3}	$15N^{3}$	$1.5X10^{10}$
6	105	N^{4}	$105N^{4}$	$1.0X10^{14}$
7	945	N^{5}	$945N^{5}$	$9.4X10^{17}$
8	10395	N^{6}	$1.0X10^{4}N^{6}$	$1.0X10^{22}$
9	$1.3X10^{5}$	N^{7}	$1.3X10^{5}N^{7}$	$1.3X10^{26}$
10	$2.0X10^{6}$	N^{8}	$2.0X10^{6}N^{8}$	$2.0X10^{30}$
15	$7.9X10^{12}$	N^{13}	$7.9X10^{12}N^{13}$	$7.9X10^{51}$
20	$2.2X10^{20}$	N^{18}	$2.2X10^{20}N^{18}$	$2.2X10^{74}$

Table 2.2 - Size of exhaustive search space for cable harness topologies and topological routings.

Figure 2.17 below further illustrates the problem of multiple topologies by showing four routings for a 4-port harness. A&B and C&D each share the same topology, differing only in the placement of the transitions.

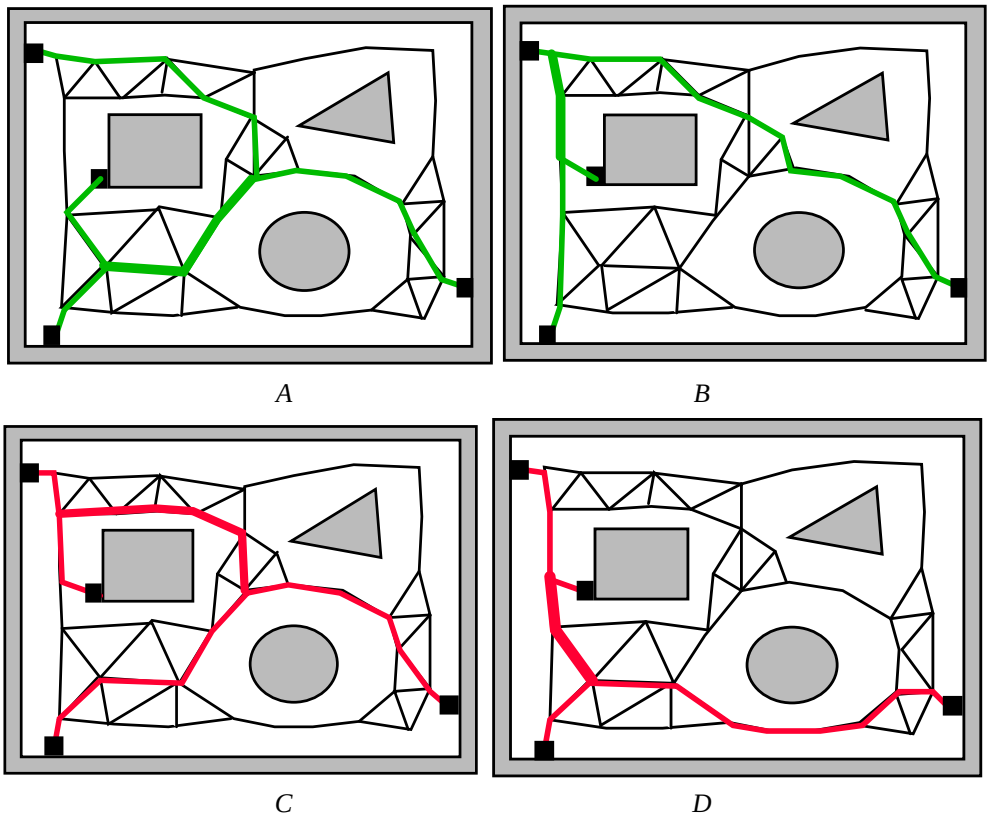


Figure 2.17 - Four possible routings of a 4-port harness. Thick lines represent the Harness; thin lines represent straight paths between nodes in the approximate Representation of the free-space (see Chapter 3).

It is difficult to determine which of the configurations in Figure 2.17 is optimal as the cost depends on both the length of each bundle and the number of wires in each bundle. This situation illustrates the coupling between the problem of the finding the optimal topology and its topological routing.

In practice, aerospace cable harnesses usually have between three and twenty ports with most having around eight ports. As different solution methodologies have different sensitivities to problem size, it is important to understand where the transition exists between easy and hard problems. Cable harness problem sizes can be broken into three general classes:

Class 1 - Small Harnesses (3-7 ports)

Here, the number of possible topologies is small enough to allow for the determination of optimal solutions via exhaustive search. While the goal in this case is to find the optimal solution faster than enumeration, the probabilistic nature of the collaborative and evolutionary solution proposed in this work makes such an event unlikely.

Class 2 - Medium Harnesses (8-11 ports)

Here, the number of topology permutations becomes large enough that enumeration doesn't make sense. However, the search space is still manageable for most heuristic or evolutionary formulations as the probability that any randomly selected fragment is of high quality is still relatively high compared to larger harnesses.

Class 3 - Large Harnesses (12+ ports)

Here the size of the problem becomes so large that only a minute fraction (<0.0001%) of all possible configurations can be tested.

Aside from harness size, the actual distribution, or amount of clustering, of the ports in the environment affects the relationship between harness complexity and the number of ports. Clustering is the extent by which subsets of the total number of ports are "clustered" in a given area. For example, a harness may connect two or more clusters of ports that are each located far apart. Clustered routing environments tend to reduce the effective overall complexity of the routing problem. This occurs since clustered ports quickly become spanned by harness fragments of high quality. This also occurs because the difference in quality between these fragments and any randomly selected fragment is much greater than for uncluttered cases and therefore is easily isolated by intelligent router agents.

For experimentation purposes, the number of ports must be large enough to provide a challenging problem but small enough to keep the problem tractable. To that end, harnesses with 10 and 15 scattered (i.e., not clustered) ports were tested as discussed in Chapter 7.

Chapter 3

Environment Representations for the CHRP

3.1 Introduction

As mentioned earlier, a number of different routing techniques are combined in this work to tackle the CHRP. As might be expected, the designer's routing interface (DRI) and the automated routing agents differ in their environmental representation needs. Of primary importance to the environment representation for the DRI is the ability to perform rapid distance calculations between a part of a harness and its nearest obstacle. This is needed as the designer interacts with a virtual representation of a cable harness that must not intersect any obstacle. The actual path planning of the cable harness routing is performed either by one of the automated routing agents or by explicit, direct involvement by the designer. Therefore, the environment representation for the DRI does not need to contain comprehensive information about all possible routing channels.

The automated routing agents, however, need an environment representation that provides more information regarding the structure and capacity of the routing space. This is needed as these agents must rapidly create thousands of test routings during the search process. The need is

compounded in domains with tight routing environments or when more than one harness is to be routed in the same location, as the routable region becomes more constrained. The environment representation must therefore be an abstraction of the environment that extracts the connectivity and capacity from the environment. As many routings will be created for the same environment, spending computational resources upfront in an environment preprocessing routine is justified. The appropriate choice of abstraction can greatly affect the speed and performance of the routing tools.

Several considerations must be addressed while picking the right abstraction for the environment representation of the automated agents, including the accuracy of the final representation, the reduction computational complexity for routing, and the computational complexity of the abstraction process. Any one of an infinite number of abstractions could be used for the cable harness routing problem so long as it satisfies the following criteria:

- Sharing the general topology of the free space
- Promoting rapid, coarse routing between two points.
- Having sufficient granularity to allow all possible harness topologies to be routed.
- Maintaining capacity information of the free space

This chapter discusses two environment representations. The first is used exclusively by the designer's routing interface (DRI) and the second by the automated routing agents.

3.2 Environment Representation for the designer's routing interface

As mentioned earlier, the environment representational needs of the designer's routing interface are driven mainly by the need to know how far any point on the harness is from the nearest obstacle. There are many representations which have been developed [Latombe 1989] that could be used for distance calculations, divided roughly between exact and approximate representations. Exact methods calculate the distances using algebraic representations of the obstacles. While the result is accurate to the precision of the machine, these methods can be slow especially if the routing domain is complex. Exact methods were not used in this work as

the designer does not require an exact routing during the early stages of the design process where many routings are being experimented with.

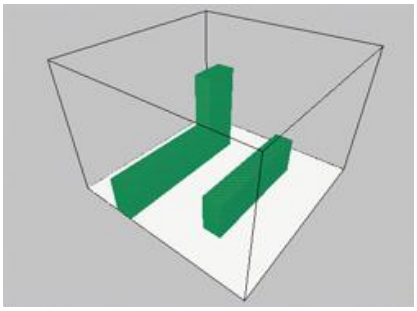
Approximate methods typically use a simplification of the environment that promotes rapid distance calculations at a cost of increased memory requirements and a loss of accuracy. In this work, an approximate cell decomposition [Brooks & Lozano-Pérez 1983] was used to approximate the routing environment. This decomposition approximates the environment as a three-dimensional grid. Each cell of the grid, called a *voxel*, is marked with a non-zero number if it contains any obstacles or a zero if unoccupied. The occupied cells constitute the *obstacle-space* of the environment, and the remaining *free-space* is the region through which the cable harness can be routed.

The clearance, distance to the nearest obstacle, for each voxel is pre-calculated using a collapsing voxel wavefront algorithm, such as the one described by Barraquand, J., Langlois, B., and Latombe,.J.C. [1989], starting from the surface of the obstacles. The procedure starts by marking all the voxels next to an obstacle or boundary with the value of 1. The wall of marked voxels, called the wavefront, is then expanded away from the obstacles and boundaries until all the voxels are marked using the procedure shown in the following pseudo-code:

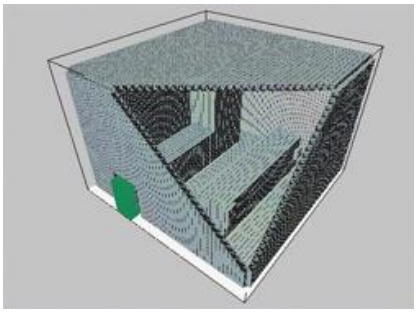
1. Until all of the voxels are marked non-zero do
2. For all zero-valued voxels adjacent (1-neighbors) to a non-zero voxel
3. Mark with the lowest adjacent non-zero voxel's value plus one

Figure 3.1A shows a simple environment with two obstacles contained within an external bounding box and Figure 3.1B- shows the voxels that are marked as being next to the obstacles and external walls (a cut-away is used to view the interior)

.



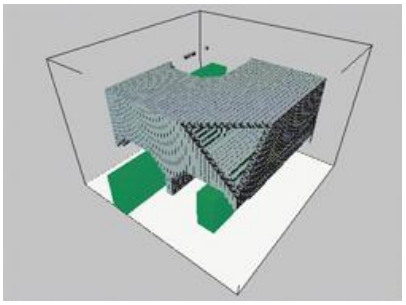
A. Simple Environment of two obstacles within an external bounding box



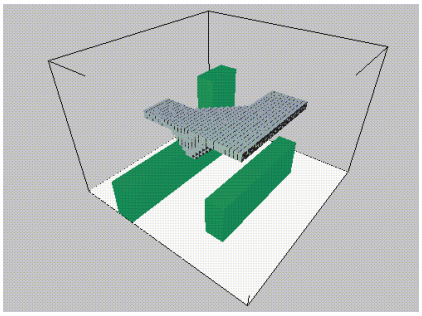
B. Cross-section of the cells which are next to obstacles and external walls

Figure 3.1 - The collapsing wavefront algorithm starts by marking all the voxels next to obstacles and external walls.

Figure 3.2 shows further progress of this algorithm.



A. Voxels which are five voxels away from The nearest obstacle or boundary wall



B. Voxels which are ten voxels away from the nearest obstacle or boundary wall

Figure 3.2 - further progress of the wavefront algorithm.

The result of this procedure is a grid of voxels each marked with the distance in terms of voxels in a Manhattan sense (i.e., rectilinear) from the nearest obstacle or boundary wall. This distance estimate for a voxel, **Grid**[i][j][k], is determined quickly using the following equation.

distance = e_{vox}(**Grid**[i][j][k] - 0.5) where e_{vox} is the edge length of the voxels

The precision of this calculation is e_{vox} and dependent on the resolution used for the approximation of the free space. Since memory resources increase with the cube of the resolution, the resolution needs to be as coarse as possible. This compromise is governed by the routing domain. For example, if the ability to discern narrow passageways is important, then the resolution must be fine and the designer must accept the price of larger memory requirements. However, if the features of the environment are large, the representation may be coarsely abstracted without greatly affecting the quality of the routing.

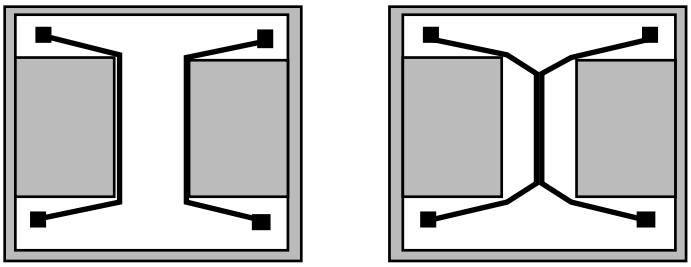
3.3 Environment Representation for the Automated Routing Agents

As the automated routing agents' primary task is to quickly explore large regions of the routing search space. Doing so requires an environment representation that reduces the computational burden required to find and evaluate feasible routings. As the output routings of the automated routing agents will only e used as starting points by the DRI, the detail and accuracy of the routing is less important than the overall shape. This means that the environment may be coarsely abstracted so long as the following exist:

- The basic topology of the free space is maintained in the abstraction.
- The capacity of each routing region is defined and updated as each wire or bundle is routed.
- Defined routing paths or channels do not intersect obstacles and resolve all potential paths between regions of the free space.
- The representation possesses sufficient granularity to allow transitions to be spaced apart as necessary to satisfy minimum bundle length requirements.

A number of different environment representations have been developed for the 2D and 3D point-to-point routing problems [Latombe 91] as overviewed in Appendix B. They basically fall into two general categories: roadmap methods (visibility graphs, freeway and retraction methods) and cell decompositions (exact and approximate). Of the various roadmap representations, visibility graphs produce the shortest paths between two points.

They use the fact that the shortest path between two points in a 3D space will either be a straight line or touch the edges of obstacles in the environment. Visibility graphs exemplify some of the problems of using roadmap methods for cable harness routing. First, the visibility graph continuous problem becomes NP-hard [Canny, 1988] in 3D spaces although some approximations [Papadimitriou, 1985] have use in very simple environments. Second, the CHRP differs from the point-to-point case as the shortest paths are less likely to run along the obstacles. The following figure illustrates this point. In Figure 3.3A, the two independently routed wires are in contact with the obstacles for almost half of their journey. When the wires are combined to form a cable harness, shown in Figure 3.3B, the center region of the harness is pulled into the free space between the obstacles.

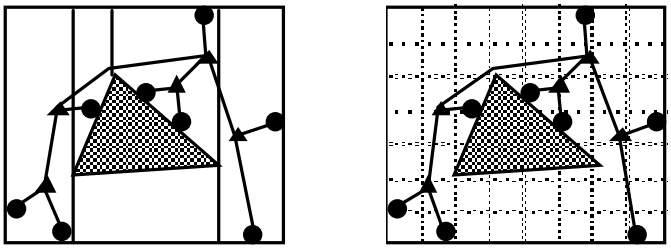


A. *B.*

Figure 3.3 - Minimal cost harness routings do not necessarily contact obstacles.

The key point is that while the routing *is* the least costly routing for this topology, the path for each of the wires is not. As is it impossible to predict how the bundling will affect the optimal bundle paths before they're routed, the environment representation need not be precise.

Cell decomposition methods reduce the dimensionality of the free space into a network of one-dimensional curves (typically straight lines). In general, these techniques approximate the free space using uniform or polygonal cells or regions which are connected to their adjacent cells. Figure 3.4 shows several simple examples of cell decompositions that will illustrate some of the issues of the routing problem.

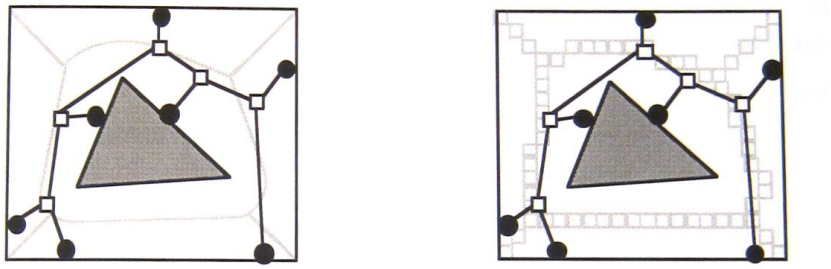


A. Exact Cell *B. Uniform Approximate Cell*

Figure 3.4 - Two example cell decompositions of simple routing environment (transitions drawn as squares, end ports drawn as black circles).

Figure A shows a typical exact cell decomposition that divides the free space into precise trapezoidal regions (While a 2D example is shown, 3D analogs exist). These five regions are connected in a sparse graph to their nearest neighbors. Superimposed on the figure is a seven port harness showing possible locations for each of the five transitions. A difficulty with this representation is that there are several cells that contain more than one transition or port. This presents problems as it is impossible to determine the topology of the harness from its routing on the sparse graph. Also, it is difficult to determine the maximum number of bundles that can fit in a region. The figure on the right shows the approximate cell decomposition of the region that is used in the DRI. While it is much easier to find distinct cells for each of the transitions and ports, the representation fails to promote rapid point-to-point routing. It also fails to maintain the capacity information of the free space as multiple channels (connect paths in the environment graph) exist in the same region of the free space. If multiple routing channels exist, determining the capacity of the entire region would be difficult since the capacities of each of the channels are interdependent. For example, consider the case where a particular region that can hold a maximum of 100 wires is represented by four separate channels (predefined paths). One would have to know how many wires had been routed in all four channels in order to determine how many more wires could be routed in each channel. An ideal environment representation would contain just enough nodes to specify the topology of the harness while still limit the number of paths between regions of the free space. The number of paths can be minimized by assuming that the paths always fall on the surface which is furthest from the obstacles and boundary walls. This approach greatly simplifies capacity computation.

The environment representation used in this work is based on a retraction roadmap method in which the free space is collapsed onto its Voronoi surface as shown in Figure 3.5A. This surface is the loci of the centers of all maximal spheres contained in the free-space. These points are, by definition, equidistant from two or more surfaces. As a visual aid, think of spheres centered around points on the medial axis each having the largest radius possible without intersecting an obstacle. These spheres should then touch at least two surfaces of the obstacles, thus making the medial axis surface as far as possible from all the obstacles. The grey lines of the following figure shows the 2D equivalent to a Voronoi surface.



A. Voronoi Diagram *B. AMA*

Figure 3.5 – Voronoi diagram and Approximate Medial Axis of a simple routing environment

As determining the algebraic Voronoi surface in complex 3D spaces is computationally impractical, the approximate medial axis is used instead. The approximate medial axis (AMA) [Blum 1967], otherwise known as the symmetric axis transform or skeleton, is a discrete set of points, or voxels, that approximates a surface in the middle of the free space shown in Figure 3.5B. The generation of the AMA surface will be discussed in the next section.

An advantage of defining paths along the AMA is that these routings provide maximum clearance for the routing since they are as far from the obstacles as possible. This is important later in the design process where material properties of the cable harness are included in the routing process. At that stage, the design process requires the initial voxel routing, typically consisting of many sharp bends, to be converted into smooth paths that are C^4 continuous. By starting with paths with maximum clearance, the smooth paths have a higher chance of not intersecting obstacles.

The voxel-based AMA has several shortcomings undermining its ability to route cable harnesses effectively. First, as tens of thousands of AMA voxels are commonly used to represent an environment, finding short paths over this surface using an A* algorithm would be costly—a point exacerbated by the fact that sometimes hundreds of wires need to be routed. Some routing techniques even require these wires to be routed several times during the bundling process. Second, the voxelized AMA is unable to represent the amount of available free space when cables are routed through a given region. Consider the case of a bundle with a diameter equal to the width of four voxels. When this bundle is routed as a series of connected voxels, it will overlap other nearby voxel paths of bundles on the Voronoi surface. This is illustrated by the following figure where a close-up view of two thick bundles (the one on the left has a diameter of four voxels while the one on the right has a width of two voxels) reveals that while the voxel paths do not intersect, the bundles that they represent do intersect.

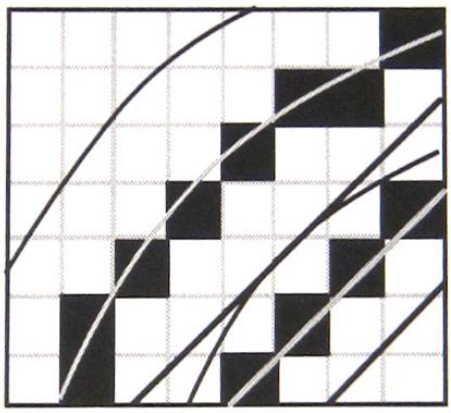


Figure 3.6- Using voxels to represent bundle paths is prone to underestimate the amount of the free-space occupied by the bundles

Third, problems arise when the paths of several wires are to be bundled. If each wire is routed independently over the vocalized AMA, very few wire routings would have voxels in common thus making it difficult to determine a common routing channel.

These issues are addressed by further abstracting the AMA into a sparce graph network as shown in Figure 3.7 below. In this case, points of the AMA are retained so long as their corresponding maximal spheres do not greatly overlap their neighbors as will be defined in Section 3.4.1.

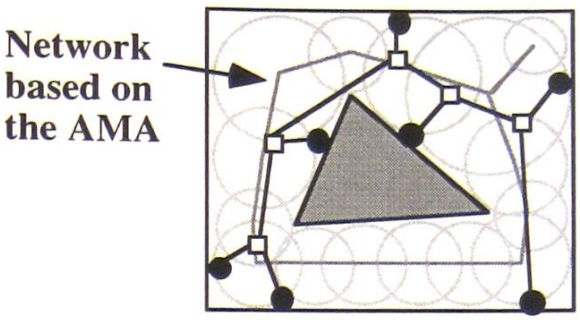


Figure 3.7 – Spherical approximation of Voronoi surface

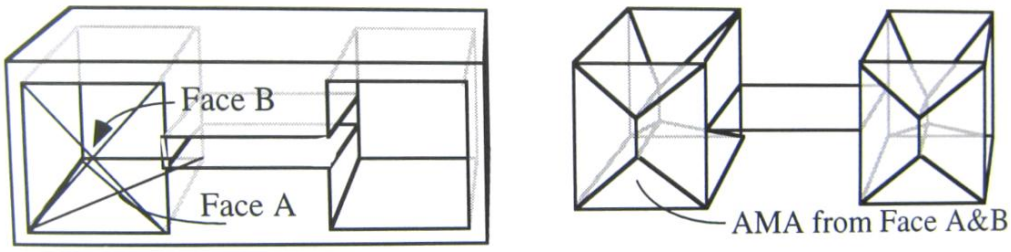
This representation shares similarities with both the roadmap and cell decomposition environment representations in that the free space topology is abstracted into 1-D curves and that the free space volume is represented by the set of maximal spheres. The sphere-based environment representation also produces a balanced number of graph nodes making it an attractive representation for the cable harness routing problem. The following section will discuss the specifics of the conversion process.

3.4 Generating the Approximate Medial Axis Graph

3.4.1 Generating the AMA surface

Many techniques have been developed for generating various flavors of the AMA for 2D [Lee 1982] [Srinivasan & Nackman 1987] [Patrikalakis & Gursoy 1989] and for 3D environments [Hoffmann 1990] [Yu, et. al.1991] [Sudhalkar 1993] [Reddy 1995]. These approaches can be categorized into algebraic and approximate methods. The algebraic methods are capable of producing the medial axis surface explicitly; however, they all require solving an extensive number of algebraic equations. The approximate methods, on the other hand, are somewhat simpler to deal with, and start from some form of an approximate cell decomposition of the free space.

In this work, the AMA is generated using an approximate method algorithm that collapses a voxel wavefront identical to the one used in the designer's router interface (DRI). The only difference is that the procedure labels each voxel which an id of the nearest obstacle or boundary face. The following figures show the AMA surface for a simple environment.



A. Sample Routing Enclosure *B. Associated AMA*

Figure 3.8 – The AMA surface is constructed by recording the intersection of voxel wavefronts from multiple faces

The following pseudo-code shows the procedure for AMA generation.

1. Label each face of the obstacles and boundaries with a unique face id.
2. Label all voxels in contact with the obstacles and boundaries with the id of their corresponding face
3. Mark all the voxels in the free-space with 0, 1 otherwise
4. Until all of the voxels are marked non-zero do
5. For all zero-valued voxels adjacent (26-neighbors) to a non-zero voxel do
6. If the test voxel is adjacent to non-zero voxels with only one unique face id,
7. Assign the face id to the test voxel.
8. If the test voxel is adjacent to two or more voxels with different face labels,
9. Record that the voxel is in the AMA
10. Mark with the lowest adjacent non-zero voxel's value plus one

The output of the above procedure on the environment in Figure 3.8 is as follows (two-dimensional view only)

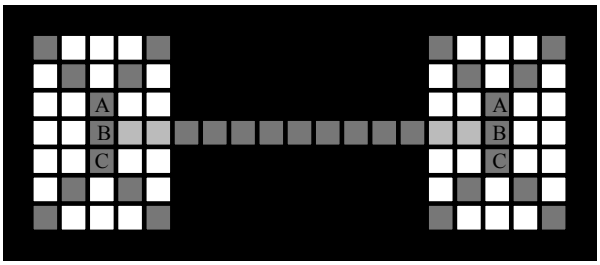


Figure 3.9 - Fully connected AMA.

3.4.2 Abstracting the AMA Graph from the AMA

The AMA Graph is an abstraction of the AMA that maintains the topology of the free space while requiring far fewer nodes. The goal is to sufficiently represent the entire topology of the AMA while simultaneously creating a data structure which promotes rapid path routing.

The total number of nodes and links of the AMA graph can change dramatically based on the amount of abstraction desired. On one end, every voxel on the AMA could be connected to its nearest neighbor. This would produce a graph which matches the approximate medial axis identically but which would be prohibitive to searching for the shortest paths. On the other extreme, only a small number of voxels on the AMA could be selected and connected to form a sparse graph. While this graph could be used to quickly route paths, these paths may be far away from the AMA, may fail to share the same topology of the AMA, or may even pass through obstacles. So clearly, some compromise in the number of nodes used for the AMA Graph is needed.

This compromise is found by having the graph density be inversely proportionate to the size of the free space. The logic is that tightly packed regions require a denser graph representation than those regions were ample clearance exists. Consider three adjacent voxels aligned on the AMA. If a sphere with a radius equal to the distance between the voxel and its nearest obstacle is centered on each of the voxels, neighboring spheres would most likely overlap considerably and therefore be redundant.

A greedy algorithm, shown as pseudocode below, is used which selectively removes redundant spheres. The algorithm first starts by choosing the voxel furthest from any obstacle corresponding to the largest sphere. It then propagates a wavefront along the AMA starting from the voxel. All the cells that are within the sphere are tested for redundancy with the starting voxel. If the overlap between the starting sphere and test sphere (centered at a test voxel) is above a critical level and all the spheres connected to the test sphere sufficiently overlap the starting sphere, then the test sphere is removed.

1. Connect all the spheres with adjacent AMA voxels.
2. Sort all the spheres in decreasing order of radii.
3. Label all the spheres as unmarked.

4. Until all the spheres are marked, do:
5. S_1 = largest unmarked sphere.
6. Label S_1 as marked and propagate a wavefront from S_1 over all the unmarked spheres with centers contained in S_1.
7. For each sphere on the fringe of the wavefront, S_f , do:
8. If S_f is not connected to any unmarked sphere that is not itself in the fringe and *Redundant* (S_1, S_f)
9. Remove S_f.
10. If *Redundant* (S_1, S_f) and *Connect* (S_1, S_u), for all unmarked spheres, S_u, not in the fringe and connected to S_f,
11. Remove S_f.
12. Connect all S_u to S_l.
13. Label S_f as marked.
14. The remaining set of spheres is the AMA Graph.

The function, *Redundant* (Sphere1, Sphere2), returns true when the two spheres overlap sufficiently so that Sphere2 is considered redundant. The function, *Connect* (Sphere1, Sphere2), returns true when the spheres overlap sufficiently to be considered connected. The determination as to how much overlap constitutes redundancy or connectives is subjective. For this work, the determination is based on the dimensionless variable called the scaled overlap, $\sigma_{12} = \frac{\delta_{12}}{\min(r_1, r_2)} = \frac{r_1, + r_2 - d_{12}}{\min(r_1, r_2)}$ where r_1 is the radius of Sphere1, r_2 is the radius of Sphere2 and d_{12} is the distance between them as shown below.

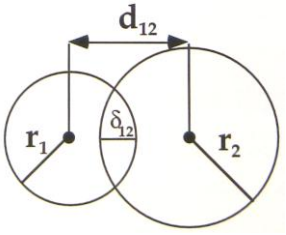


Figure 3.10 – Redundancy and Connectiveness is based on sphere overlap, σ_{12}

Two spheres are considered connected if $\sigma_{12} > 0.268$ and redundant if $\sigma_{12} > 1.3$. The following shows the AMA Graph for the above AMA.

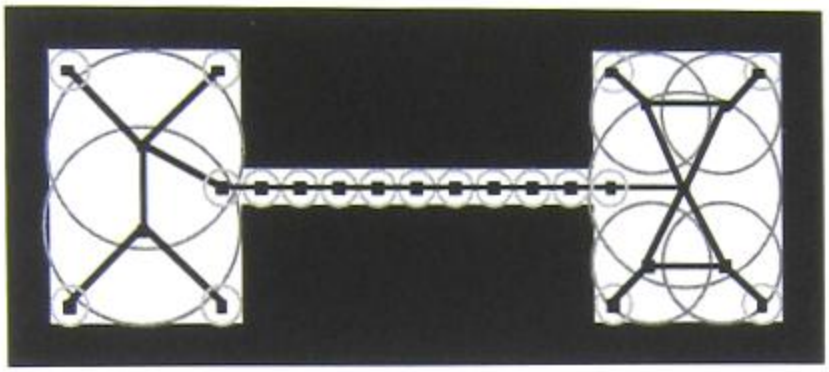
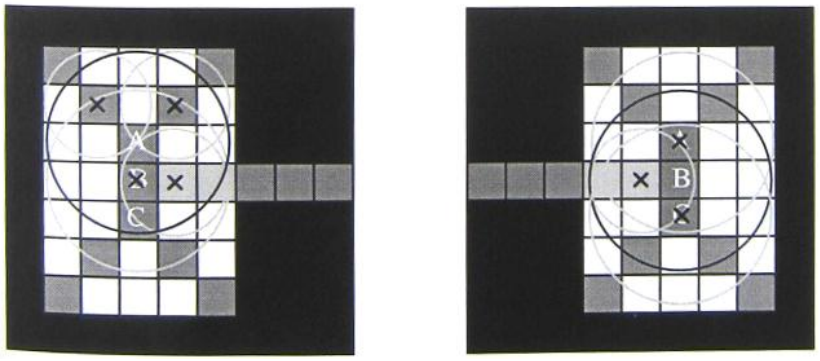


Figure 3.11 – The AMA Graph of the AMA of Figure 3.9

The left and right hand sides of the graph have different shapes due to the order in which spheres where tested for redundancy. The following figure illustrates the first step in the voxel deletion process for each half of Figure 3.11. Since the three large spheres, corresponding to voxel A, B, C in Figure 3.9, are the same size, one is chosen randomly. In this example, the sphere corresponding to voxel A is chosen first and used to determine that voxel B and three others are redundant, as shown in Figure 3.12A. In the case of the right region of Figure 3.11, the sphere corresponding to voxel B is tested first as shown in Figure 3.12B.



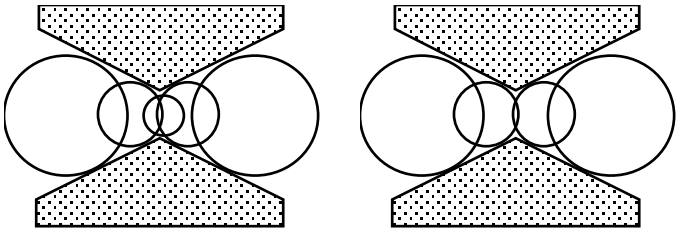
A. The X's mark redundant spheres due to excessive overlap with the sphere at voxel A

B. The X's mark redundant spheres due to excessive overlap with the sphere at voxel B

Figure 3.12 – The grey spheres are determined to be redundant due to the black sphere.

There are two concerns when it comes to deleting voxels and their associated spheres. The first is that when the "redundant" sphere is removed, its neighboring spheres must still overlap. If they do not overlap, the connectivity mapping between the spheres and environmental free space will

be lost. This concern is addressed by requiring that at least two neighboring spheres of any deletion candidate sphere are considered connected. The other concern is that the radius of the "redundant" sphere should not be too much smaller than that of the remaining spheres. This presents a problem as the remaining spheres will be linked together and each link assigned a maximum capacity equal to the smaller radius of the two spheres. Now if the free space constricts near the small redundant sphere, then the capacity of the link could be too large. For example, Figure 3.13 shows a case where the spheres are representing a narrowing channel. If the small sphere in A is removed and the medium spheres connected as in B, the available capacity of the channel is overestimated. While not implemented in this work, this potential problem can be addressed by defining the link capacity between spheres that are connected after a sphere deletion to be the smaller of the two previous link capacities.



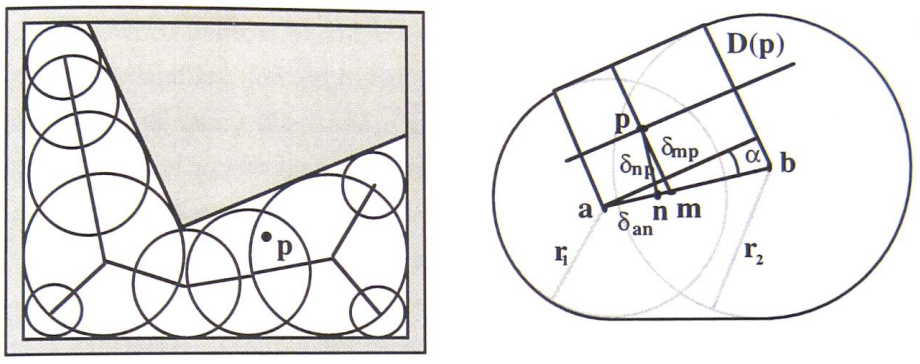
A. *B.*

Figure 3.13 - If the small center sphere is removed, the remaining smaller spheres may over-estimate the available free space.

The greedy nature of this algorithm results in several interesting properties of the AMA graph that make it an ideal candidate for routing problems. First, the larger regions of the free space will be represented with the fewest number of links. Since the final paths for the harness bundles need not be determined in the preliminary phases of the design process, having sparsely placed graph nodes in large open areas makes efficient use of search resources. Second, most of the wires will be routed along links that lie along edges which connect three planes of the AMA since these edges are further from the obstacles than the voxels on the planes and thus have the largest capacity. This is useful in that the fewer the number of spheres representing a region there are, the better is the ability of the heuristic methods to estimate the capacity of a region.

3.4.3. Using the AMA Graph for obstacle distance calculation

As mentioned earlier, The AMA graph is useful for describing the connectivity information of the various regions of the free-space. However, it can also be used to estimate the distance between any point within a sphere and its nearest obstacle. Figure 3.14A below shows a simple free-space represented by ten spheres and an arbitrarily chosen point, p.



A. Sample Environment

B. Free-Space Approximated by Two Spheres

Figure 3.14 – The spheres in the AMA Graph can be used to approximate the free-space as well as the distance between any point and the nearest obstacle.

Using the assumption that the free-space can be approximated by the volume contained in the set of minimal cones that encapsulate two adjacent spheres (see Figure 3.14B), the distance from a point inside a pair of adjacent spheres (cone) to the nearest obstacle and the distance gradient at that point can be approximated as:

$$D(p) = \begin{cases} r_1 - \delta_{ap}, \text{ if } -r_1 < \delta_{am} < 0 \\ r_1 + \delta_{an}\dfrac{r_2 - r_1}{\delta_{ab}} + (r_2 - r_1)\dfrac{\delta_{np}}{\delta_{ab}}\tan\alpha - \dfrac{\delta_{np}}{\cos\alpha}, \text{ if } 0<\delta_{am} < \delta_{ab} \\ r_2 - \delta_{bp}, \text{ if } \delta_{ab} < \delta_{am} < \delta_{ab} + r_2 \end{cases}$$

Where $\alpha = \sin^{-1}\left(\frac{r_2 - r_1}{\delta_{ab}}\right)$ and $\delta_{ij} = \left\| point_i - point_j \right\|$

$$\Delta D(p) = \begin{cases} \dfrac{a-p}{\|a-p\|}, if\ -r_1 < \delta_{\text{am}} < 0 \\ \dfrac{m-p}{\|m-p\|}, if\ 0 < \delta_{\text{am}} < \delta_{\text{ab}} \\ \dfrac{b-p}{\|b-p\|}, if\ \delta_{\text{ab}} < \delta_{\text{am}} < \delta_{\text{ab}} + r_2 \end{cases}$$

While not implemented in the software, the above equations eliminate the need to retain the whole voxel representation, defined in Section 3.2, for use in obstacle collision checking. The key advantage of using the AMA Graph representation is that it requires far less memory than the voxel representation. Further, it may be possible to generate the AMA Graph for specific regions of the free-space and then splice the AMA Graph sections together. This would require less memory as the sections could be voxelized sequentially. The drawbacks of using the spheres for distance and gradient estimation are that the calculations are more computationally intensive and conservative (except for special cases as shown in Figure 3.13) than that of the voxel representation and finding the spheres which contain the point to be tested requires additional computation.

3.5 Modifying the AMA graph capacity during routing

One of the requirements of a routing environment representation is its ability to determine how much room is available for new or additional routings at a given time. The size of the spheres at each node of the AMA graph gives some information as to how much room is available in the free space. The available space remaining in each node can be estimated by having each node keep track of the amount of wires routed through it. However, only an estimate can be made as a sphere may be connected to multiple, and perhaps smaller, spheres. The estimate is conservative (except for the issues raised in Figure 3.13) as the radius of the smaller of the two connected spheres is used to define the capacity of the link connecting the spheres and that the spheres, by nature, give a conservative estimate of the free space. Figure 3.15 below shows this estimated area in gray for a simple environment.

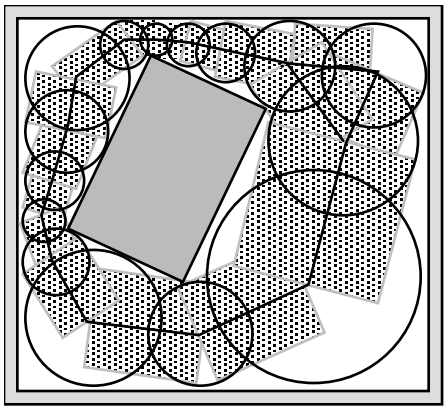


Figure 3.15 - The safe routing area is shown in gray.

The capacity of each of the links must be modified each time a wire is routed. As each wire is routed, its cross-sectional area is subtracted from the capacity of each link along its path. If the capacity of any link becomes smaller than the wires to be routed, the wires must take alternative routes.

3.6 Computational Complexity of Voxel to AMA Graph Conversion

The voxel to AMA graph conversion process can be divided into three steps. The first is the construction of the AMA using the wavefront algorithm discussed in Section 3.4.1. As each voxel of the free-space must be touched by the wavefront, the complexity is O(n) where n is the total number of voxels. The second step is the connectivity repair process for the AMA. Once again, as a single wavefront is constructed, the complexity is O(n). The final step is the abstraction of the AMA by a sparce network of spheres. After a sorting operation of O(s logs) where s is the number of spheres (i.e., equal to the number of voxels on the AMA), the spheres are ready to be connected. In the worst case of the abstraction process, none of the spheres is found to be redundant. This means that each sphere must test all the spheres whose centers are located in the sphere for redundancy . While this case is unlikely, it does produce an upper bound complexity of O(s·c) where c is the average number of sphere centers contained in a sphere. However, if one assumes that p percent of the spheres contained in the test sphere will be determined to be redundant, then only $\frac{s}{p \cdot c}$ test spheres need to examine the c sphere centers each they contain. This results in a complexity of O(s).

Even though the complexity of each step is linear, except for the sphere sorting process, it must be noted that n may be large (e.g., a 100x100x100 voxel environment has n=1,000,000 voxels) and the number of test spheres appears to be approximately $2n^{2/3}$. Fortunately, this is not a grave burden because the generation of the AMA graph is a one-time environment preprocessing step that expedites the routing and cost computation operations which are repeated thousands of times.

Chapter 4

Collaboration in the CHRP

4.1 Introduction

Product design is typically a process of educated trials and errors in which one or more designs are modified, enhanced, or scrapped altogether in the quest to find a better design. It is not uncommon to hear designers refer to their work as an evolutionary process. Evolutionary processes share the ability to reuse parts of previous work to produce new—and hopefully better—designs. While evolutionary design processes can be used by a single designer, they can be even more powerful when used by a team a designers—especially if the problem is complex. This is advantageous not only because the work load can be distributed, but also because specialists can be used. This team approach can be effectively applied to the CHRP in that while many heuristic solution strategies exist to tackle the large search spaces, none of them works

well in all cases. This chapter will discuss an evolutionary strategy for the CHRP, the decomposition of the problem required, and present an overview of the specific team members.

Three independent approaches could possibly be used with the CHRP. The first is to simply enable the designer to explore the search space more effectively by providing computer-based, interactive tools. But while the designer is effective in making local adjustments to designs, he often produces suboptimal designs by exploring only a handful of designs, by using an abstraction that hides the details that affect the quality of a design, and by an inability to predict how components of a design will interact when combined. The second approach is to create a number of heuristics and to apply each to the problem. However, in this case, the best design found would be no better than the best design found by each technique used independently—even though the robustness of the solution process might be improved. The third approach is to develop an algorithm to randomly compare thousands of designs in the search space. But since the solution space is sparsely populated with good designs, random search would require a prohibitively large number of cases to be examined. This method would have little chance of finding a near optimal solution. However, by combining each of these approaches, there exists a potential to radically improve the overall performance of the system. Heuristics can be used to initiate the search process while the designer and other evolutionary search mechanisms can be used to direct the search and build upon previous work.

The goals of the system architecture presented here are to allow a number of design tools to share their findings with one another and to enable the allocation of resources commensurate with each tool's performance. Collaboration is facilitated by the use of a shared work repository of finite size, similar to a blackboard, into which λ complete designs are stored. This set of stored designs, called a population, is used as input to a set of nA routing agents, $\mathbf{A} = \{A_1, A_2, \ldots A_{nA}\}$, to produce a new population of hopefully better designs. The heuristic-based, evolutionary, and designer-based routers fall into three main categories: *creators*, *mutators*, or *combinors*.

1. Creators:

   Creators do not require any inputs other than the problem specifications to generate a complete design, i.e., D_i^t = Creator(*raw inputs)* where i is the design number and t is the generation number. They are most effective at the beginning of the design process to "seed"

the initial design population. They can also be used as the design evolves to add new ideas to stale designs.

2. Mutators:

   Mutators take a single design and alter a small subset of the design parameters, i.e., $D_i^t = \text{Mutator}(D_j^{t-1}), 1 \le i, j \le \lambda$. Typically, the new design shares many of the properties of the original design. Simple gradient search or simulated annealing procedures can be classified as mutators. In general, mutators that leverage domain specific knowledge outperform those that do not. Mutators use domain knowledge to retain useful design fragments and are often used to improve designs that have some poor components, as well as to introduce additional ideas.

3. Combinors:

   Combinors take two or more designs as inputs and attempt to extract the best fragments from each of them to produce a new design, i.e., $D_i^t = \text{Combinor}\,(D_j^{t-1}, D_k^{t-1}), 1 \le (i, j, k) \le \lambda$. In order for a combinor to be successful and find compatible matches, it must be able to determine which fragments have high value and to understand the relationships between design fragments of different designs. Combinors, therefore, require far more domain knowledge than mutators.

There are two basic steps required to implement a collaborative system and to enable the different agents to work synergistically. The first is to modify each of the agents to allow design information to be input and output. The second is to decide what information to pass between agents and what control mechanism to put in place. Most recent work on multi-agent systems for engineering applications focuses on either the conflict resolution issues required for a team to find a feasible design [Belmonte, et al 1990; Lander et al, 1993; Birmingham et al, 1993] or how to combine multiple optimizing agents [Souza 1991; Hogg 1993; Talukdar 1993].

Talukdar classifies collaborative systems using four properties: data flow, control flow, activity constraints, and modification constraints. Data flow describes the amount of looping of design data among agents. Control flow relates to the degree of central supervision required by the agents. Activity constraints describes the data input and output constraints of each agent in terms of other agents. The modification constraints are used to determine the interdependency of the

agents. The modification constraints are used to determine the interdependency of the agents. The modification constraints define the structural modifications needed whenever an agent is added to or deleted from the system. The approach used in this work is classified by Talukdar as an "A-Team" in that the data flow is strongly cyclic, the control flow allows all the agents to work independently, the activity constraints are such that the agents can work asynchronously, and the modification constraints allow any agent which can read and write designs in a prescribed format to be added or deleted without requiring structural changes to the system.

4.2 Approach

The specific goals of the communication architecture for this problem is to eliminate all overhead for translation between representations and for design communication, and to enable each of the agents to build on previous designs. To this end, it is required that all of the agents be able to communicate their designs in a common representation to a central repository. Each agent is allowed only to write over the design slots allocated to it, but all the designs in the central repository are readable by all the agents. The figure below illustrates the flow of the design process in that designs from a previous generation are selected and reused to create the next generation of designs.

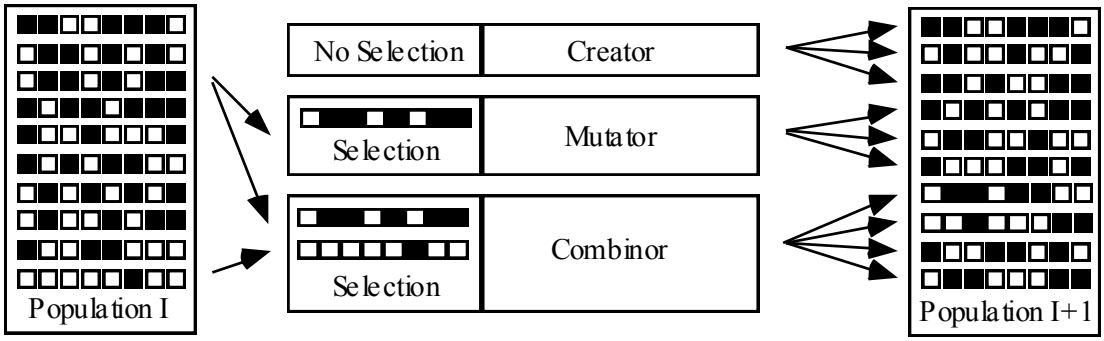


Figure 4.1. GA-based Design Flowchart

It is assumed that the design process starts with an initial population of λ designs that has been generated by one or more Creator agents. For each generation of designs, each of the nA agents is allocated λ_i designs of the next population ($\Sigma\lambda_i = \lambda$) and each posts its new designs for that sub-population independently. The mutators and combinors are then able to select any of the designs in one generation and then report their findings into the next generation. By applying this approach to the evolutionary process of engineering design, computational resources spent on one generation of designs can be reused during future design interactions.

One of the results of this approach is that the design process has the ability to retain new designs that may be of lower quality than their predecessor(s) and thus avoids the problem of local minima. This is an extremely useful property in that it is often necessary to modify a design and turn it into an intermediate design of inferior quality to get to a new, better design. Since the designer rarely has the time nor the desire to consider designs that are no better than the best-as-yet-found design, this enables the automated agents to pursue design possibilities that would be otherwise missed.

As with other probabilistic approaches, the genetic solution process will find the global optimal solution given an unlimited amount of computational resources. However, practical limitations require a termination condition be defined. In this case, the search process was allowed to run for a fixed number of topological routing tests. This number was simply defined by the amount of time allocated by the designer. As the process is able to return a design anytime, the designer is able to balance the desired solution quality with available search time. This is especially important in the conceptual phase of the overall harness design process when only a crude estimate of routing quality is needed to compare design possibilities. While not implemented in this work, a termination condition of exiting whenever the fitness of the best designs does not improve over a set number of generations.

At first glance, the strategy proposed here seems very similar to the standard genetic algorithm [Koza 1992; Goldberg 1989] (See appendix A.2 for an overview of the genetic algorithm). While the genetic algorithm has been used extensively for finding good solutions to difficult search problems [Goldberg 1989,1995], it routinely fails to find near-optimal solutions. This typically occurs for two reasons.

The first is due to the landscape of the search space. The search spaces for problems where the genetic algorithms are used typically have many locally optimal solutions of similar quality that are far superior to the vast major of possible designs. This presents a problem as the genetic algorithm has a tendency to converge on an above-average design instead of searching for even better designs.

A second reason that the standard genetic algorithm has difficulties finding near optimal designs is due to the way designs are encoded. Most of these problems stem from the genetic representation of the design (genotype) and the naiveté of the selection strategies and operators. Genetic algorithms typically use a binary encoding of a design, called a genotype, that can be mapped into a representation that more closely resembles the design itself. The second representation is called the phenotype. For the harness topology, it is a data structure that maintains the connectivity relationship between transitions and ports. The best genotype representations are those in which the alleles (analogous to bits of a binary encoding) corresponding to related features of designs are close to one another in the encoding**.** This increases the chances that even the most blind genetic operators will be able to retain substrings of high fitness when they cut or mutate the design's encoding. However, finding a genetic representation (genotype) for a design that also naturally breaks into logical substrings is extremely rare. Davidor [1991] shows that the greater the epistasis (the degree in which the design variables are coupled) of the domain, the more difficult it becomes to have encodings that allow standard genetic algorithm formulations to isolate fragments. In short, there are many limitations to the standard genetic algorithm that make it inappropriate for solving the CHRP.

In this work, the problems related to genotype encoding are avoided by using the phenotype representation of the cable harness as the genetic representation. Domain-specific operators can then be used to extract good fragments from designs. The following section will discuss why this is important.

4.3 Collaborative Decomposition of the CHRP

As stated before, the CHRP can be viewed of as a problem of two levels, shown in Figure 4.2. At the higher level, a harness can be viewed simply in terms of its topology (sometimes referred to as its configuration), C. The top part of figure 4.2 shows three potential topologies for a six port harness. At a lower level, the problem is to find this topological routing, R. Here, the

topology and its port locations are defined and all that remains is to determine locations for the transitions in the AMA graph. The process is analogous to what would be done by a path planner except that the path planning has been abstracted to placing transitions on a graph. The solution process is to formulate the transition location process as a topological routing evaluation separate from the topology search. The Transition Locator (GA formulation discussed in Appendix A) operates on each topology to search for the optimal routing, $R^t_i=TL(C^t_i)$.

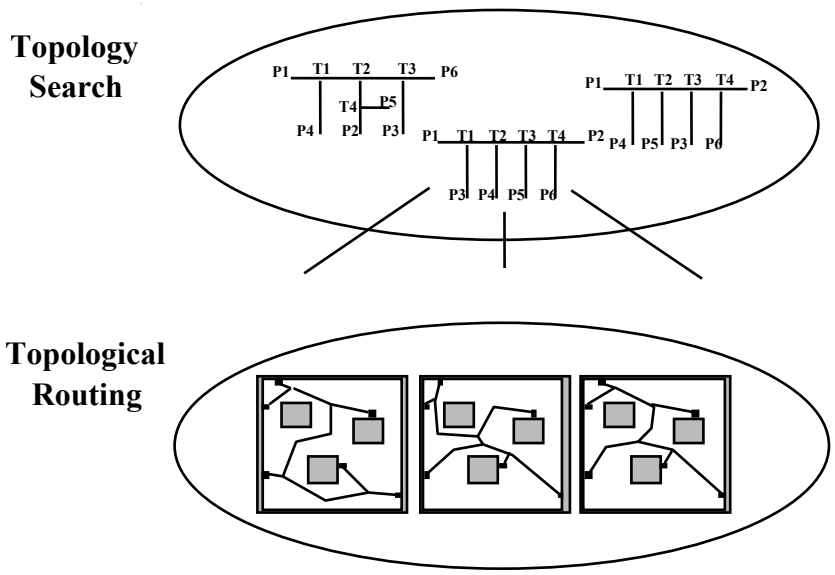


Figure 4.2. CHRP is divided into its topology and topological routing search processes.

Since each of the routing agents discussed in this work is able to generate a topology, but not all can output a topological routing, R, the topology representation is used as the least common denominator between the routers. This is illustrated in Figure 4.3. The transition location process then becomes just an evaluation algorithm that takes a topology as input and that outputs the transition locations and therefore its topological routing.

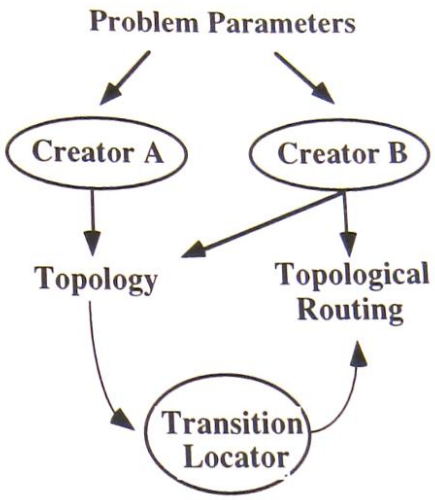
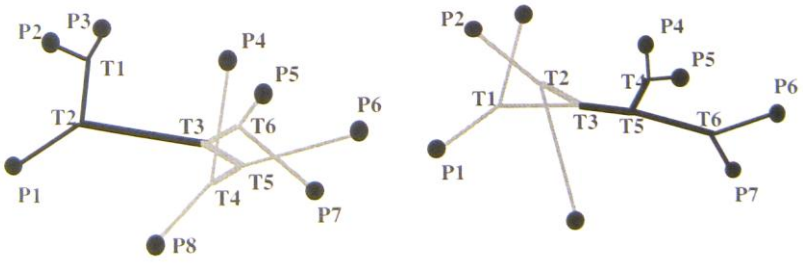


Figure 4. 3 - Some agents are only able to create a topology while others output both the topology and the topological routing.

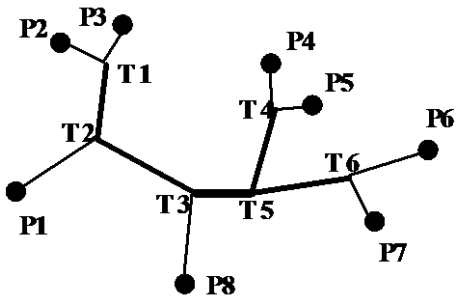
4.3.1 Topology Search Problem

The CHRP is well suited for a collaborative architecture since it is possible to effectively combine routings produced by different techniques to form new routings of higher quality. Consider the following example of two 6-transitioned cable harnesses each produced by different heuristic or genetic techniques. While each of them produces suboptimal routings, they both have parts (shown with solid lines) which match the geometry well. If the best part of Parent1 is grafted onto the best part of Parent2 of Figure 4.4, then the resulting routing would retain the beneficial qualities of both parents as shown in Figure 4.5.



Parent 1 *Parent 2*

Figure 4.4 - Each topological routing has good and bad fragments



Better Offspring

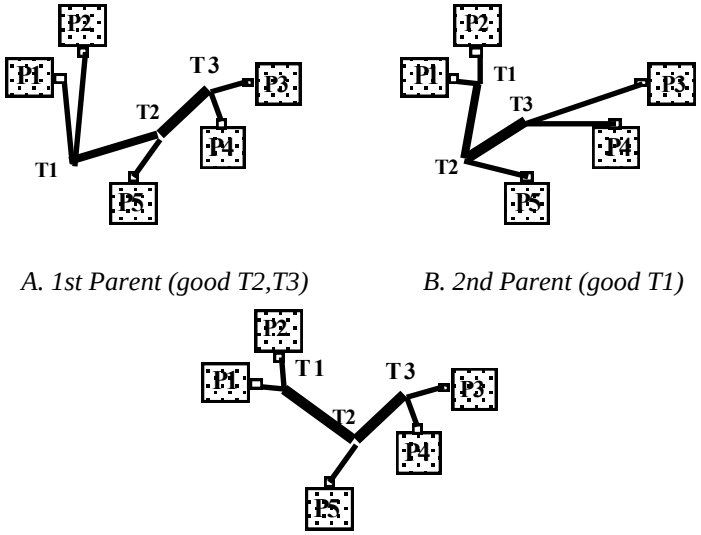
Figure 4.5 - A better offspring is found by combining

Parts of two different topological routings

4.3.2 Topological Routing Search Problem

The Transition Locator, discussed in greater detail in Appendix A, has been formulated as a genetic problem to capitalize on its ability to search large spaces and on the inherent moderate levels of epistasis of the transition-location (topological routing) component of the CHRP. The later can be illustrated by Figure 4.6A in that the quality of the placement of transition T1 is independent from transition T3. While the genetic algorithm has become a panacea for tough search problems, it is particularly applicable here as the decomposable nature of the topological routing problem aligns with the strengths of genetic approach – most notably, the ability to combine or reuse components of pre-existing designs.

The genetic formulation takes a population of designs, in this case, different routings of a given topology, and applies a combination of mutation and combination operators to create the next generation. It is therefore able to combine parts of multiple routings to create a new routing which is better than its parents. Figure 4.6 below shows an example of two parents, A and B, which if they exchange the location of transition T1, produce an offspring C which has a lower cost than either parent.



A. 1st Parent (good T2,T3) *B. 2nd Parent (good T1)*

C. Offspring (good T1,T2, and T3)

Figure 4.6 - Offspring may take on positive attributes of both parents.

4.3.3 Cable Harness Design Fragments

Design fragments are the smallest units of information passed between agents and share the following properties:

- They contain sufficient information to be evaluated for local fitness.
- They can be combined into complete designs.
- They can be used to isolate good and bad elements of a design

Complete routings are first decomposed into fragments so that these routers can analyze the fragments and then select those that, when combined, create good complete designs. By selecting and reusing only the best fragments, better designs can then be created.

Each design fragment for the CHRP is a connected sub-graph of a harness topology, C. Any piece of a topology which results from the cutting of a termbundle is considered a fragment. Each fragment, f_i, contains nP^{f_i} ports and has a complement, $f_i^{'}$, which is C / f_i. The order of a fragment is defined to be the number of cut ends that it has. For example, if the topology was

only cut once then it would have two fragments of order 1. If the topology was cut twice, as shown in Figure 4.7, then it would have two fragments of order 1 and one fragment of order 2.

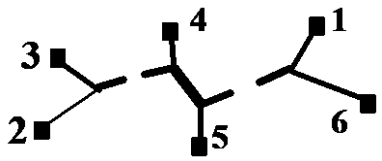


Figure 4.7 - Fragment 2-3 and 1-6 are order 1 whereas 4-5 is order 2.

The transbundles that are cut are called the free ends of the fragment and are considered to be attachment points to other fragments. A particular topology with nT transitions has 2(nT-1) fragments of order 1 and (nT-1)(nT-2)/2 fragments of order 2. Fragments of orders higher than 2 are possible when nT becomes greater than 3, but the actual number depends not only on nT but also on the shape of the topology.

A fragment is most useful when its qualities do not change as it is combined with other fragments. This property is a function of the method used to evaluate the qualities of a fragment. For example, in the cable harness case, a variable cost of a fragment is the material costs of the wires that it contains. This cost is a function of the number of wires in the bundles of the harness and the lengths of the bundles. While the number of wires in fragments of order 1 are fixed, the number of fragments of a higher order is not. For example, the center fragment of the topologies in Figure 4.8a and Figure 4.8B, f^{45}, may have a different wire count if it is to connect f^{23} to f^{16} or f^{2} to f^{136}. This means that for fragments with orders higher than one, any localized fitness for the fragment will change depending on what is attached to its free ends. This is the key reason why only fragments of order 1 are used by the CHRP routers. In addition, the more free-ends a fragment has, the harder it becomes to combine it effectively with other fragments**.**

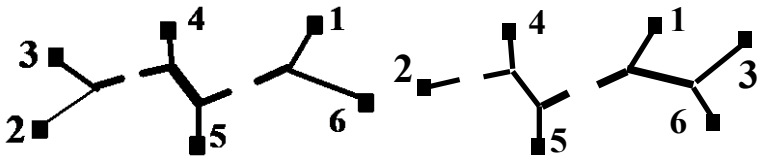


Figure 4.8 - The cost of a fragment of order 2 (fragment 4-5) or more is dependent on its complements.

The actual encoding for the fragments of the topology search records the ports and transitions contained in the fragment, the connectivity of these nodes, the cost per length of each of the bundles, and the transition of the free end. The encoding for the topologically routing records the associated topology encoding and the AMA graph node locations for each of the transitions.

The following chapters will discuss each of the topology routers developed for this work. Chapter 5 discusses the automated routers that are able to work together but independently from the human designer, and Chapter 6 discusses a novel routing representation that enables a designer to work interactively with a 3D virtual representation of a cable harness.

Chapter 5

Automated Routing Tools

5.1 Introduction

Automated routing tools are critical for the effective exploration of the search space of the cable harness routing problem. This section discusses a number of automated routers developed in this work explicitly for the CHRP which are specially formulated to generate topologies and topological routings, to improve existing designs, and to combine elements of multiple designs. These routers are categorized as creators, mutators, and combinors. The first of these, creators, are able to generate designs using only the raw inputs (i.e., the port locations, wiring list and routing environment) while mutators modify a single design to create a new one and combinors create new designs by combining two or more previous designs. Although most of these automated routers have been implemented as procedural calls or operations, they are referred to as "agents" here because the search formulation lets them operate independently.

5.2. Creators

The goal of all the CHRP creators is to find topologies that map well to the routing environment (i.e., topological routings that minimize the objective function). Three different creators—the Random Topology Generator, the Thick Bundle Router, and the Wire Bundling Router—are developed for this work. The Random Topology Generator does not require any inputs other than the number of ports in the harness, and the other two use the AMA graph, port locations, and the wiring list.

5.2.1 Random Topology Generator

The Random Topology Generator's task is to increase the design diversity of the topology population. By generating feasible topologies without using heuristics, it is able to create new design fragments without bias. Another advantage of this router is that it is able to produce every feasible topology for a given number of ports.

A directed tree is used to represent the topology. For any topology, a unique tree can be defined by recording only the children for each transition. An example of such a tree can be found in Figure 5.1, where an arbitrarily chosen transition, labelled T1, is considered to be the root of the tree for a 3-transitioned harness. A unique tree can also be represented as a list of n-tuples as illustrated in an example shown below in Figure 5.2.

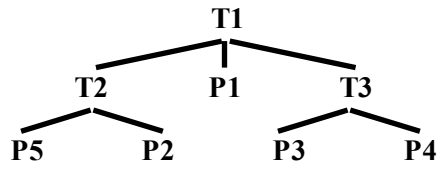


Figure 5.4 - A tree representation for a cable harness configuration

T1	T2	P1	T3	T2	P5	P2	T3	P3	P4

Figure 5.5 - Topology Template

In Figure 5.2, the gray cells are simply labels for each of the three transitions and the white cells are the topology nodes that the transitions connect. Notice that all of the topology nodes are represented twice, with the exception of T1. Finding valid topologies consists of filling in the blanks of the topology template (Figure 5.2) with each of the topology's nodes—except for the implicitly defined transition, T1—subject to the following constraints:

1. T1 must connect to at least one other transition
2. Transitions must not connect to themselves.
3. The path of transitions between two ports $p_{harness}(P_i,P_j)$ is unique for $P_i, P_j \in P$

5.2.2 Thick Bundle Router

The Thick Bundle Router (TBR) extends Takahashi and Matsuyama's[1980] technique for the Steiner Minimum Spanning Tree problem by addressing the loading between ports. It starts by finding the pair of ports with the most wires in common and then routes the shortest path between them. Next, the remaining ports are sorted in decreasing order by the number of wires emanating from them and a path is routed from each of the remaining ports to nodes in the graph that are already a part of the routing in progress. The following pseudocode defines the algorithm more formally.

1. Determine P_1,P_2 with the most wires in common.
2. Construct a subtree, t_1, of **G** consisting of the links in $p_{graph}(P_i,P_j)$
3. Create the set, **USED**=$\{P_1,P_2\}$.
4. Set counter, k = 2.
5. While k<nP, do:
6. Determine $P_i \in$ **P/USED** with the maximum number of wires.
7. Construct t_{k+1} by adding the minimum cost path joining P_i to t_k
8. k = k+1.
9. t_k is the topological routing, R.

The following example, Figure 5.3, of a four-port harness illustrates this heuristic. The figure on the left shows the initial subtree, t_1, whereas the figure on the right shows the entire routing, R. The ports are labelled in the order in which they were added to the routing.

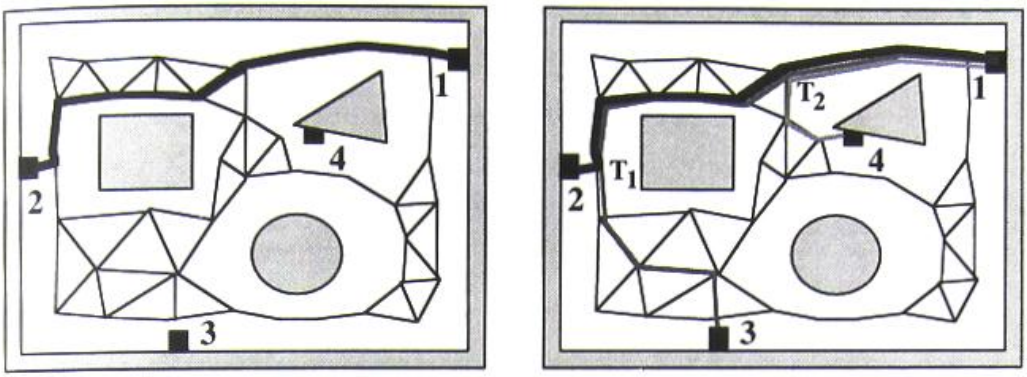


Figure 5.6 - In the Thick Bundle agent, ports are connected to a routing under-construction. Ports are selected in descending order of the number of wires at each port.

Harnesses routed with the Thick Bundle Router tend to connect ports that are spatially close to one another. And while the thickest bundle technique is capable of quickly producing both a topology and topological routing, it is by no means guaranteed to be optimal. For example, Figure 5.7 below shows an optimal routing for the four-port harness routed in Figure 5.6. The problem here is that the quality of the final routing changes depending on the order in which subsequent ports are connected**.** This means that it is unlikely that an arbitrarily chosen routing order will always result in the best routing possible using this technique.

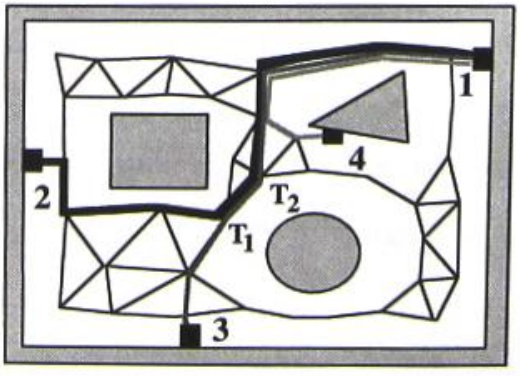


Figure 5.7 - Optimal routing for the four-port harness

At most, this heuristic requires $O(PN^2)$ time, where P is the number of ports in the harness and N is the number of nodes in the graph. This is true since determining the shortest path for each of the bundles takes at most $O(N^2)$ time. In their work, Takahashi and Matsuyama were able to prove that with the Steiner case, the result of this algorithm would be no more than 2*(1-1/P) times as costly as the optimal solution, regardless of the graph and port locations. However, as the cost of each bundle is non-uniform and dependent on the routing order, this upper boundary does not hold for the CHRP.

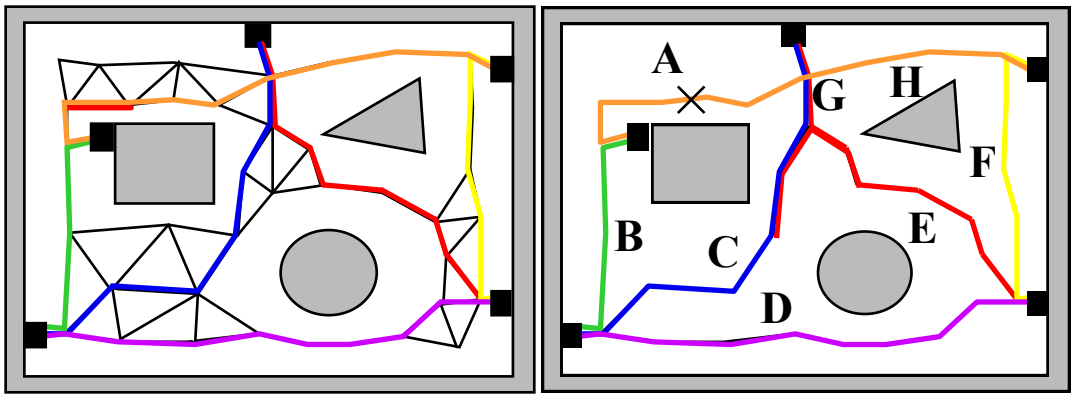
In conversations with cable designers, it appears that they often use the TBR heuristic when they design harnesses manually.

5.2.3 Wire Bundling Router

The Wire Bundling Router (WBR) takes the stance that the routing of all the wires should be weighed together to predict the major channels. In this way, collections of multiple ports whose wires share common shortest paths are bundled. This is in contrast to the TBR where the main branch of the harness is determined by the path between the two ports connected with the most wires. Like the TBR, however, the Wire Bundling Router is able to generate both the topology and its topological routing simultaneously. The process is formalized as follows:

1. Route each wire, $W_i \in \mathbf{W}$ along $P_{graph}(P_1^{Wi}, P_2^{Wi})$ in the graph, **G.**
2. Remove the edges in **G** that do not contain wires.
3. Define an empty set, **Taboo**={}.
4. While **G** contains at least one cycle, do
5. Choose the subpath (series of connected graph nodes of order 2), $\mathbf{S} \in \mathbf{G}$, $\mathbf{S} \notin \mathbf{Taboo}$, that minimizes a subpath cost function.
6. If there exists a path, $P_{graph}(P_1^{Wi}, P_2^{Wi})$ for all $W_i \in \mathbf{W}$, on G/S then
   $\mathbf{G} \leftarrow \mathbf{G/S}$
   else
   $\mathbf{Taboo} \leftarrow \mathbf{Taboo+S}$
7. R=**G** is the solution.

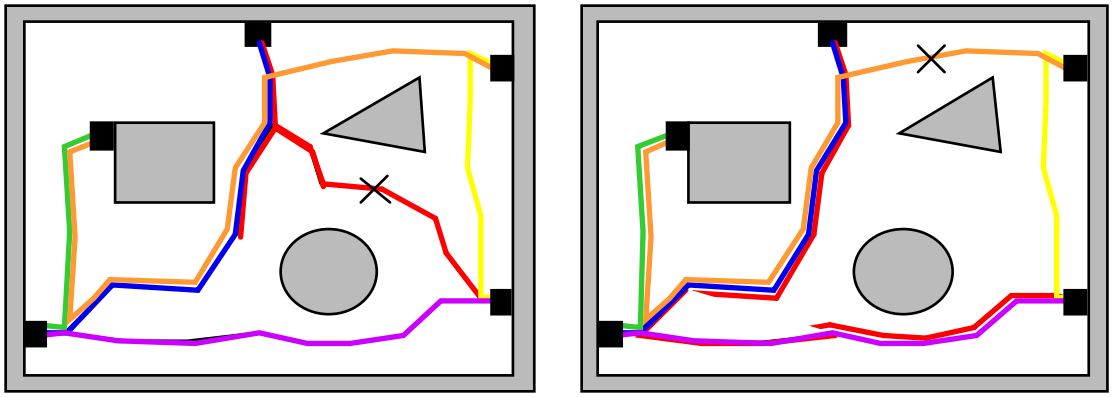
The following example illustrates the process on a five port harness. It starts by finding the shortest path routing for each of the wires in the harness over the AMA graph. The resulting routing shown in Figure 5.8.A, albeit not necessarily a cable harness itself, can be used as a lower bound estimate of the harness routing with minimum wire lengths (recall that this is not necessarily the same as the least cost harness). Next, all of the unused edges in the graph are removed, leaving only edges with one or more wires as shown in Figure 5.8.B.



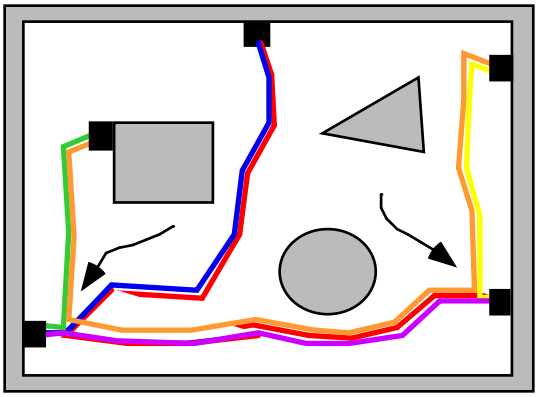
A. Wire routed individually *B. Unused links removed*

Figure 5.8 - First two steps of the Wire Bundling agent

Next, the remaining subpaths are cut sequentially, so long as the affected wires can be rerouted over the remaining edges of the graph (Figure 5. 9). The subpaths are removed in the order of increasing cost value (several cost functions are discussed later in this section), and by always cutting the lowest-cost subpath first, the wires tend to be bundled along the same thick, short paths. This procedure always results in a spanning tree.



A. Routing After First Link Cut *B. Routing After Second Link Cut.*



C. Final Routing – Transitions are close to port locations

Figure 5. 9 - The links are removed one by one.

Harnesses routed with the wire bundling agent tend to have transitions that lie close to the port locations. This leads to designs with short termbundles that may therefore be suboptimal. For example, consider the simple 4 port harness, shown below, connecting ports that are spaced 1 unit apart. In this case, only 1 wire connects each pair of ports (6 wires total). The routing on the left requires 10 units of wire where as the one on the right only requires 6 * sqrt(2)=8.46 units.

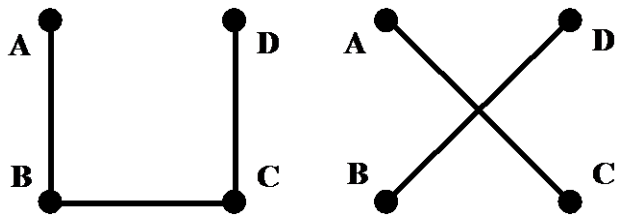
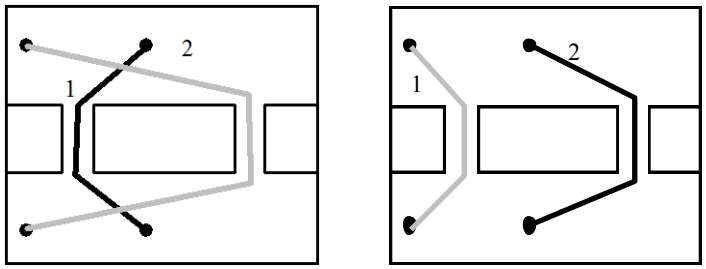


Figure 5.10 - The U-shape routing may be more costly then the X-shape routing.

However, this technique gives very good results when a large percentage of wires connect a subset of ports. Since the determination of the harness topology is influenced by the distribution of all the wires instead of just the ones in the principle port pairs, the routings of the WBR tend to out perform the TBR in cases where the wire distribution has multiple primary backbones.

Since wires are originally routed sequentially, the order in which they are routed may affect the final routing cost. For example, a low capacity channel may become filled first with wires that have a lower opportunity cost associated with not routing them through the channel than other wires, and may result in a suboptimal routing. This can be illustrated by the following figures.



A. *B.*

Figure 5.8 -The order of the wire routing affect overall routing costs.

In Figure 5.8A, the black wire is routed first along its shortest route and occupies the left channel. This causes the grey wire to take a much longer-than-necessary route through the right channel. In Figure 5.8B, the opposite occurs. In this case, the grey wire is routed through the left channel first, causing the black wire to take the right channel. As the added cost of routing the black wire through the right channel instead of the left is smaller than the added cost for the grey wire, the overall cost of the routing (sum of the length of the wires) is lower. This means the order in which wires are routed can affect the overall cost. Fortunately, most routing problems in the aerospace domain that motivated this work have sufficient clearance and this issue occurs infrequently.

The order in which graph links are cut also greatly affects the topology of the final routing. For example, if link B was cut first in Figure 5. 9B, the final routing would have both a different topology and routing. Since the number of different cut sequences is factorial in order, it is impractical to try every cut sequence to find the least cost routing. Therefore, only two heuristics

are used in the selection process for link removal. Several of the heuristics tested in this work are outlined below.

The first heuristic is to always cut the subpath containing the fewest number of wires. The logic is that since there is an economy of scale associated with putting more wires in the same bundle, a routing with many long bundles of few wires is typically costly. Additionally, re-routing a few wires over a perhaps suboptimal path is less costly than re-routing many wires. Ties are broken by choosing the longer link. This heuristic typically results in harness routings that have relatively thick transbundles**.**

The second heuristic is to cut the subpath that has the smallest diameter-to-length ratio. This results in harness routings that favor short, thin bundles over long, medium thickness ones. The motivation behind this heuristic stems from the fact that the incremental cost to route a second wire over a given length is usually far less that the cost to route the first. This means that a long stretch of cable containing only a few wires typically results in higher cost harnesses.

The formulation of the WBR may be improved by removing unused links more slowly during the wire cutting phase. This can be accomplished by deleting the links containing no wires a few at a time instead of all at once. This has the effect of preserving the opportunities of finding shorter paths for wires that are being rerouted.

The WBR can also be used for another problem related to the CHRP, the wire bundling problem, where the routing may include cycles. In this case, wires are routed independently directly in the physical routing environment, and then grouped in a rather ad hoc fashion. In the WBR, the degree to which the wires are bundled can be set by simply stopping the wire removal process before the routing's topology becomes a tree. As each subpath is rerouted, the marginal routing costs may be weighed against aesthetics and maintenance concerns until the desired balance is achieved. The CABLER system developed in this work uses this property to allow the designer to incrementally remove subpaths in wire bundling problems.

5.3 Mutators

A mutators is a router that takes a design and alters it to create a new design. Mutators may change any subset (including the empty set) of the input design's parameters. The quality of the mutator depends on its ability to discern between good and bad design components. This section will discuss three types of mutators: copy operators, the move branch operator, and the human designer.

5.3.1 Copy Best & Copy Some Operators

Just as in traditional design, good ideas and designs must be remembered while trying out new ideas. The "memory" of this design process is maintained in the population of designs. Two "copy" operators retain entire designs in this collaborative architecture. The first is the Copy Best operator. As its name suggests, this operator just copies the best of the previous generation into the next. The second operator is the Copy Some operator, and it first selects designs probabilistically based on their relative fitness and then copies them into the next generation. Together, these agents produce an elitist strategy that ensures that the best designs are not lost in the evolutionary search process.

5.3.2 Move Branch Operators

The Move Branch operator is modeled after the way designers currently edit harness designs in practice. The goal is to make a change to the overall topology while maintaining the good elements of the design information. The process proceeds by selecting a design, isolating a fragment of the design, then reattaching the fragment to the rest of design, thereby reusing the good fragment. The questions to be answered when deciding to use the Move Branch operator are "how to select the input design?", "how to choose the best fragment", and "how to reuse the fragment while guaranteeing a feasible routing?". The specifics of the mutation process will be discussed first for contextual purposes.

Mutation Process

The mutation process simply cuts a design into two parts and grafts the smaller onto the larger. The process is formalized as follows:

1. Select a bundle $B_i(N_1^i,N_2^i)$
2. Divide the topology, C, at B_i to produce f_i and f_i' where $f_i^{'nP} \leq f_i^{'nP}$.
3. Remove N_2^i from f_i' and connect the two remaining topology nodes.
4. Select a bundle $B_j(N_1^i, N_2^j) \in \mathbf{B}^{f_i'}$
5. Disconnect N_1^i, N_2^j
6. Create a new N_2^i and connect it to N_1^i, N_1^j, and N_2^j.

The move branch operation is guaranteed to produce a new topology but does not create a topological routing. The following examples show the move branch operation on a 10-port harness shown in Figure 5.9.

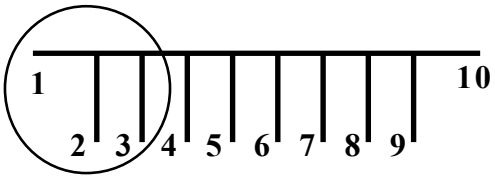
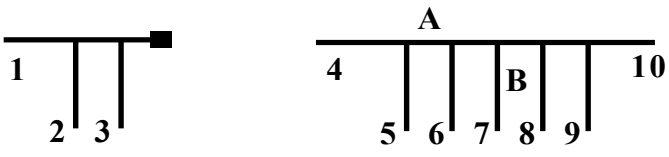


Figure 5.9 - 10 port harness with a circled fragment.

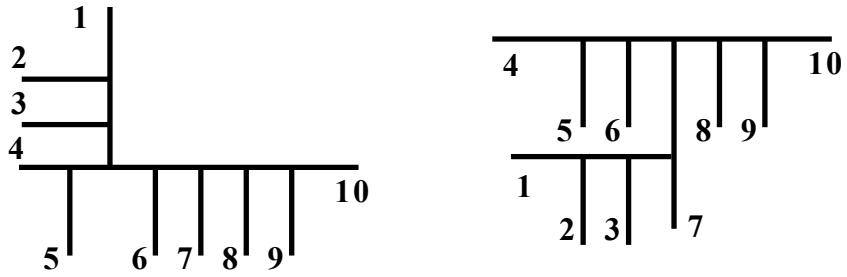
If the transbundle between port 3 and 4 is cut, the result is the fragment 1-2-3 and its complement, 4-5-6-7-8-9-10, shown below.



Fragment *Complement*

Figure 5.10 - Fragment 1-2-3 and its complement 4-5-6-7-8-9-10.

The complement has $2f_i^{nP} - 4$ possible reattachment locations that will result in new topologies. Of the 10 possible in this example, the results of two reattachments , labeled A and B in Figure 5.10, are shown in Figure 5.11 below.



Mutation A *Mutation B*

Figure 5.11 - Results of the move branch operation with reattachment at transbundle A and termbundle B.

It can easily be shown that any topology can be changed to any other by a series of mutation operations. Thus, this operation has the ability to create designs with every possible configuration in the search space. The ability to do this centers on two assumptions: any term-bundle (bundle connected to a port) can be removed and reattached to any other bundle and still result in a valid topology and any topology can be constructed by sequentially adding term-bundles (attaching ports). The following figure illustrates how a new topology can be constructed by sequentially removing ports F and B and reattaching them on the term bundle of port D. While only term bundles are moved in this example, larger fragments could have been used in the same way.

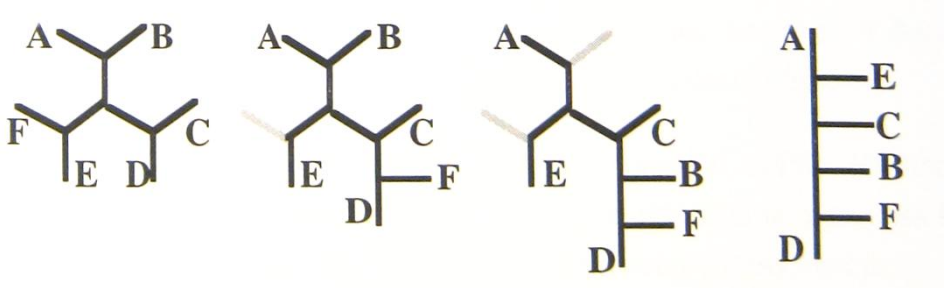


Figure 5.12 – Any topology can be transformed into another by sequentially moving term-bundles.

The next section will discuss several heuristics that help direct the move branch operator in making good design selection decisions.

Design Selection

The design selection method plays the important role of directing the search process and maintaining diversity in the design population. Choosing a selection method involves balancing a tradeoff between rapidly converging to a single solution and having a high probability of finding the global optimum. While it is tempting only to use the very best designs, this strategy usually results in suboptimal solutions. This occurs since focusing on reusing only the best designs reduces the chances of generating new designs. This decrease in population diversity typically results in stagnating the search process.

The design and fragment selection procedures for the mutator are based on the premise that the fragment, and to a lesser extent its complement, should reasonably route well in the environment. Two general class of selection techniques can be used. The first is to select between fragments based on high fragment fitness. However, a problem with this approach is that the complements of the selected fragments may not have a high fitness. The second approach is to choose a design of relatively high overall fitness and then to select a fragment-complement pair from it such that both parts route well The second selection approach is used in this work.

Several specific design-level selection strategies were tested. The first and most straightforward method is random selection as defined by $p(D_i^t) = \frac{1}{\lambda}$ where $p(D_i^t)$ is the probability that design *i* will be chosen among λ designs during generation *t*.. In this case, all the designs have an equal probability of being selected. This method has the greatest chance of producing truly unique offspring and eliminating bias; however, it does not use any of the readily available discriminatory information such as design fitness.

Another technique tested, called Fitness Proportional Selection (FPS) [Holland 1975; Goldberg 1989], uses the design fitness value to bias the selection in a process Here, the probability that a design is chosen is proportionate to its relative fitness, that is:

$$p(D_i^t) = \frac{f(D_i^t)}{\sum_{j=0}^{\lambda} f(D_i^t)}.$$

These first two design selection processes are preservative [Back 91] in that all designs have a chance to be selected. Preservative techniques, which have been shown to decrease the convergence rate, are most effective in search spaces such as the CHRP's where even poor designs may be close (in the sense that a single mutation operator may be able to vastly improve its fitness) to better designs.

A different class of selection methods, called Rank-Based Selection [Baker 1985][Whitley 1989], focuses on the ordering of the designs instead of their fitness values. Two flavors of rank-based selections were tested in this work. The first is Uniform Ranking [Schwefel 1981] where the top μ designs are chosen equally as shown below.

$$p(D_i^t) = \begin{Bmatrix} \frac{1}{\mu} & 1 \le i \le \mu \\ 0 & \mu \le i \le \lambda \end{Bmatrix}$$

The second is Linear Ranking [Baker 85], defined below, where the designs are chosen with a frequency inversely proportionate to their ranking.

$$p(D_i^t) = \frac{1}{\lambda}(\eta - 2(\eta - 1)\frac{i-1}{\lambda - 1}) \text{ where } 1 \le \eta \le 2$$

Another interesting approach to the selection process, which was not tested in this work, is to utilize a sharing method. Sharing Methods [Mahfoud 1994] are used to increase the diversity of the design population when only a limited number of designs are tested. The approach is to scale the fitness of a design by its uniqueness and to use the scaled fitness in place of the original fitness in the selection methods discussed above. The scaling is shown below.

$$f'(D_i^t) = \frac{f(D_i^t)}{\sum_{j=1}^{\lambda} sim(D_i^t, D_j^t)}$$

where $sim(D_i^t, D_j^t)$ is the sharing function that estimates the difference between two designs. Determining *sim* is not a trivial matter. Most sharing methods are used in domains where the designs are binary encoded. The sharing function is usually directly related to the number of bits that are different between the two encodings (assuming a binary encoding). Since small changes

in the cable harness encoding may result in large design changes, any sharing function would have to look at similarities between the topologies.

Fragment Selection

The goal is to chose fragments that closely match the environment while not containing too few or too many ports. If the fragment contains too few ports it doesn't retain enough information. If it contains too many ports then the gains from combining it with other fragments would often be small. The following plot shows the cost per port for fragments of six different populations of 10 port harnesses.

3:00	227.630569			
4:00	334.138458			
5:00	453.532745			
6:00	555.579956			
7:00	675.552612			
8:00	787.773193			
Gen	0		0	1
2:00	67.849388	2	67.849388	64.438576
3:00	85.761032	3	85.761032	78.940437
4:00	98.818275	4	98.818275	95.542236
5:00	105.209679	5	105.209679	97.089409
6:00	110.683327	6	110.683327	98.273537
7:00	116.251411	7	116.251411	106.408562
8:00	115.22036	8	115.22036	106.800842
Gen	1			

Figure 5.13 - Distribution of fragment fitness and the number of ports.

The optimal fragment size depends on three factors: fragment cost/port, fragment specialization, and fragment information. Obviously, the less ports a fragment contains, the less topological information it contains. A number of test runs were made that showed that runs that favored using small fragments (less than 40% of the total number of ports) produced better routings than runs where larger fragments were chosen.

Fragment selection procedures fall into two categories, Bundle-Based (BB) and Fragment-Based (FB). To choose which bundle to cut, BB methods look at the properties of all of the bundles. The bundle to be cut is chosen either randomly or based on bundle cost (function of the number of wires in the bundle) or on bundle length. When a bundle is cut, the smaller of the two fragments is used as the fragment to be reattached to the larger fragment. For the cost and length based selections, bundles are chosen probabilistically based on their relative cost and length. That is, the probability that a bundle is chosen is directly proportionate to its cost or length. The premise is that long and costly bundles are the result of poor topologies and should not be maintained in the resulting topology. By cutting a harness at these points, the resulting fragments will more than likely not have long or costly links.

Fragment-Based fragment selection methods choose fragments based on the number of ports a fragment has and the cost associated with the fragment. The goal is to maintain fragments that effectively span a number of ports throughout the move branch operations. The selection process works by defining a fragment level objective function for all the possible fragments of all the designs. This function promotes fragments of low cost that span a large number of ports. The fragment is then chosen on a fragment fitness-weighted probability.

Various objective functions for fragments were tested empirically because of the difficulty in defining the relationship between a good fragments and the corresponding complete design. The first fragment selection method (FSM1) tested uses the global objective function but is restricted to only use the bundles in the fragment under consideration.

The function, shown below, is also scaled to the number of ports the fragment services, n_p.

$$C_f = \frac{1}{n_p} \sum_{i=1}^{\#bundles} f_i(nWb_i) \cdot g_i(Lb_i) \quad i \in \{Bundles in fragment\}$$

The goal with this function was to select fragments that spanned a large number of ports for a small cost per port serviced.

The next selection function (FSM2) uses the difference in cost between routing wires along the fragment and routing the wires directly between the ports themselves. The motivation is that if the difference in cost is small then the fragment matched the graph well. The following function

defines FSM2 where $d_h(N_i, N_j)$ is the distance a wire must travel over the fragment between ports N_i and N_j, and $d_g(N_i, N_j)$ is the same distance in the context of the entire AMA graph.

$$C_f = \frac{1}{n_p(n_p - 1)} \sum_{j=i}^{n_p} \sum_{i=1}^{n_p} \left\{ d_h(N_i, N_j) - d_g(N_i, N_j) \right\} d_g(N_i, N_j)$$

The final function tested (FSM3) was to simply use the closeness of the ports in the fragment as a measure of the goodness of the fragment topology. The motivation was that if the ports are close together in a graph distance sense, then the ports are likely a part of a cluster and should be combined in a fragment. The following function was used.

$$C_f = \frac{1}{n_p(n_p - 1)} \sum_{j=i}^{n_p} \sum_{i=1}^{n_p} d_g(N_i, N_j)$$

Chapter 7 compares the effectiveness of the various selection methods.

Reattachment Location Selection

Once the fragment is chosen, the remaining issue is where to reconnect the fragment to its complement. Both random and heuristic techniques were tested. For the random case, all of the bundles in the complement fragment are equal candidates for reconnection points. For the heuristic case, the fragment is reconnected to the nearest (geometrically) bundle of the complement. It was quickly determined that the second case, used exclusively, reduces the effectiveness of the move branch operation in the first few generations since most of the early designs have many long bundles due to poor harness topologies. When the reconnect process was forced to always choose the closest bundle, the chances for a suboptimal reconnect were high. As the design process continues and the routings, in general, improve, the contribution of such a reconnect strategy improves considerably.

General Comments on the Move Branch Operator

The Move Branch operation is most effective where f and f' are of high local fitness as its ability to eliminate poor-fitness fragments from a design is limited. This occurs since the operator only modifies two bundles at a time and retains most of the connectivity information in f and f'. It is therefore most applicable on harnesses which are already close to the optimal.

5.3.3 Human Designer

The human designer is by far the most imaginative mutator agent. As discussed earlier, the designer, by working at the topology level, is partial to choosing designs that have short bundles without considering the actual wiring within, and this may result in suboptimal routings that look more like MSTs. However, it is assumed that as the designer has the ability to creatively generate new designs and "repair" slightly suboptimal designs that may have one or two poorly connected fragment. The designer also plays a necessary role in the search process to address unmodeled constraints.

5.4 Combinors

A combinor is an operator that is able to take two or more designs and combine them to create new designs. Combining multiple designs is a high-level design process that can be extremely effective. However, it is rarely used in most design processes for a number of reasons. First, the work involved with keeping track of previous designs and recognizing complementary qualities of several designs from a population becomes prohibitive for human designers. Therefore, designs are typically viewed individually instead of as various pieces that can be combined. Second, the logistics of operating on and combining two or more designs can become a nightmare without automation. Automating the search process for combining designs with good fragments creates an opportunity for much faster design improvements.

When combining designs, the goal is to make the most of the population design fragment pool with a limited computational budget. One key challenge is that it is difficult to estimate the qualities of a design by only looking at its fragments since the quality of the fragments is highly dependent on global topology information. It is even more difficult to predict the final fitness of

a design resulting from the combination of fragments from different designs. Because of this, the combinor agent developed in this work focuses on finding initial fragments that have complementary qualities and combining them spatially without disrupting previous routing efforts. This section discusses both the options to consider when performing a combination as well as the specific procedures used in this work. The questions to be answered are "how to select the input designs?", "how to choose which fragments should be used?", and "how to combine the fragments while guaranteeing a feasible routing and reusing previous routing information?" The specifics of the combination process will be discussed first for contextual purposes.

Combining Designs

The approach to combining two designs is to graft one part of a design onto that of another. The combination process takes a fragment from topology C_1, $f_i^{C_1}$, and a second topology, C_2, as inputs and follows the following procedure.

1. For all the ports, $P_i \in f_i^{C_1}$
2. Find the node, N_i of C_2 where $\exists$ a bundle, $B_{j(N_i,P_i)}$
3. Define $B_1 = B(N_i,N_1 \neq P_i), B_2 = B(N_i,N_2 \neq P_i)$ where $N_1 \neq N_2$
4. $C_2 \leftarrow C_2 / N_i,P_i,B_1,B_2$
5. Create $B(N_1,N_2)$
6. Select a bundle, B_{att}, of C_2
7. Create a new node, N_{att}, and bundle, $B(N_{att},N_{f1})$
8. $C_2 \leftarrow C_2 + f_1$ is the combined topology

For example, suppose the following two 10-port topologies are to be combined where a 3-port fragment of the left harness is to be attached to the topology on the right (the next section will discuss several procedures in choosing which design and fragments to combine).

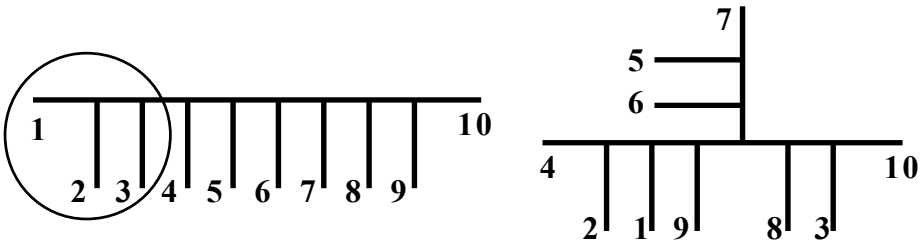


Figure 5.14 - The fragment 1-2-3 on the left is to be grafted onto the configuration on the right.

For the resulting configuration to be feasible, it must not contain duplicate ports. To ensure this, the links to ports of the harness on the right, which are duplicated in the fragment, are removed to produce the following complementary fragments.

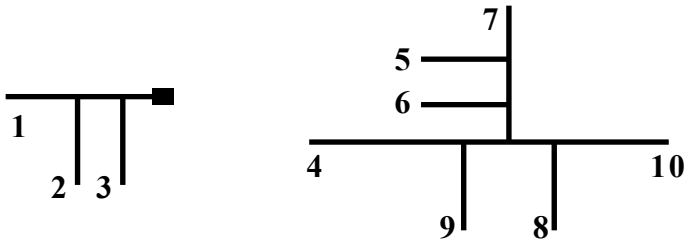


Figure 5.15 - Fragment 1-2-3 and its complement from the second topology.

At this point, the procedure must decide the best location for attaching the 3-port fragment onto the 7 port fragment. The problem is similar to the problem of re-attaching a branch in the move-branch operator and the same heuristics apply. While using any bundle location would be feasible, a better approach would be to find the shortest path between the transition connecting port 3 of the 3-port fragment and the 7-port fragment. This, of course, has the minor drawback of reducing the number of possible fragments. The resulting feasible topology is shown below.

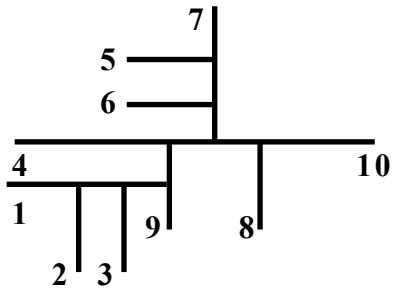


Figure 5.16 - Resulting configuration from the combination.

Selecting Designs and Fragments

Appropriate selection of the designs and fragments is critical to keeping the combinor procedures from becoming a random process. The goal in selecting designs is to find two or more designs that have complementary high quality fragments. As in the automated move branch operator, designs of medium to high fitness are more likely to contain at least one fragment that matches the environment and wire list well. A principle difference for the combinor is that the designs need only have a single fragment of high quality - an event more likely to occur early on in the search process.

The simplest selection strategy is just to pick a pair of designs each having relatively large global fitnesses. However, this technique may fall prey to the deceptive nature of the problem. Deception in evolutionary processes occurs when two fragments of different designs of high quality are combined, but the resulting offspring is inferior. This occurs when the fragments have conflicting routings. A more sophisticated selection procedure is to pick designs based on their relative fragment fitness. The process works by defining a fragment level objective function, similar to those used in the move branch operator, for all the possible fragments of all the designs. The fragment is then chosen on a fitness-weighted probability. This technique, while isolating the high quality fragments, suffers from the same problems as selecting designs based on the complete designs' fitnesses. Both of these techniques fail to consider the interdependencies between the fragments.

A still more sophisticated selection process is to first find a fragment of high quality and then scan the remaining designs for a match. The motivation is to increase the chances that a candidate with a high fitness would fill in the ports missing in the first fragment. As a simple

means of discrimination, the other designs can be scanned for high-quality fragments that share very few ports with the fragment being considered. The fitness for the each candidate fragment, $f_2 \in C_2$, is scaled using the following formula:

$$F_{f_2} = F_{f_2} \frac{f_2^{nP} - SP}{f_2^{nP}}$$

where SP corresponds to the number of ports shared between the fragment candidate and the first fragment.

Reattachment Location Selection

Once the two fragments are chosen, the remaining question is where to reconnect the fragment to its complement. This process differs from the that of the move branch operator's in that it is unclear which bundles of the second topology will remain, or which paths they will take, after the redundant ports have been removed. To circumvent this problem, the bundles of the second topology that remain after removing the duplicated ports are rerouted between the transition locations. This is illustrated in the following example.

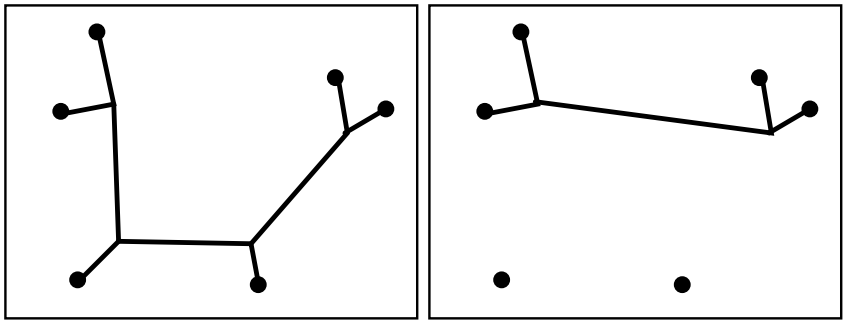


Figure 5.17 - Bundles remaining after redundant bundles are removed are rerouted before the reattachment location process.

Once these bundle are rerouted, the same reattachment bundle selection techniques as for the Move Branch operator are used for the combinor operator.

Chapter 6

Designer's Interactive Routing Interface

6.1. Overview

No matter how much automation is used to generate cable harness routings, there will always be cases where specialized, unmodeled constraints and circumstances will require direct human involvement. This chapter discusses both the issues of and the methodologies for immersing a designer in the cable harness routing process. It focuses on a design environment in which a designer may naturally interact with a virtual design representation as it is being developed.

While several CAD tool manufacturers have developed spline-based tools [ProEngineer 1996, SDRC 1996] for the documentation and display of cable harness routings, they have failed to give designers the intimacy they need to rapidly and intuitively modify the designs by requiring

them to meticulously define spline points by hand. This problem is exacerbated during the conceptual and redesign phases where each revision requires the spline points to be moved.

As tools that shorten the cable harness design and routing process are not available, designers today often resort to building a physical mock-up of the routing environment and then use segments to rope or tubing to represent the cable harness to facilitate intuitive and interactive modification of designs [Johnson & Jensen, 1994]. An obvious limitation to physical mock-up is that no more than a handful designs can be tested. On complicated routing projects, designers often quit searching after the first feasible design is found and before the best one is discovered.

For a computer-aided tool for cable harness design to be accepted in practice, the user-interface must provide the same visual and physical cues as the traditional mock-ups, along with a comparable level of interaction. Facilitating this sensation requires a harness representation that:

- Shares the shape of a physical harness with regards to diameter, curvature, topology, and overall path.
- Automatically avoids collisions with obstacles in the environment.
- Facilitates rapid path and topological modifications by the designer.
- Responds in a natural way to high-level designer interaction and environmental loads.
- Automatically reduces excess length in the routings.
- Reflects the balance between the accuracy and the detail of the model and the computational resources available.

The different design factors can be balanced by defining the bundle paths as a point-mass-spring system. Since the loading on this system defines the eventual shape and consistency of the individual harness, the selection and choice of parameters of the loading must be carefully discerned. The above representational requirements are satisfied by defining the loading on each point as a summation of three different loads:

1. An internal axial spring pulls neighboring points in the bundle closer together. This has the effect of keeping the bundle length as short as possible.

2. An internal bending spring tries to keep the bundles from making sharp bends.

3. An obstacle potential field repels the bundles away from the obstacles and other cable harnesses. In most cases, the magnitude of the repulsion drops off the further the bundle is from the obstacles.

Each of these loads balance the others to produce routings that are both realistic in shape and sufficiently malleable for designer interaction. It should be noted that, since the motivation of this work has been to provide computational support during the creative, and thus revision prone phase of the cable harness design process, a greater emphasis was put on speed than on the accuracy of the model. That is, the capacity to quickly modify the shape and topology of a harness was determined to be a more important factor than the ability to know the precise shape of the bundles within a cable harness.

The specific topics in this chapter include the sphere representation of paths, the conversion of the AMA graph to the sphere representation, the dynamic model of bundle paths, the numerical modelling of the dynamic system, and the high level of interaction that is gained from the representation.

6.2 Interactive Bundle Representation

While the AMA graph representation is convenient for quickly routing wires into a harness, it fails in several ways as an interactive representation. First, it does not produce a realistic routing capable of intuitive, interactive design and thus it cannot convey an impression of a real cable harness to designers. Second, the graph-based paths of the harness bundles of the AMA graph are neither the shortest possible nor do they reflect the material properties of the bundles (e.g., minimum bending radius constraints). Third, the representation of the AMA graph is static in that each time an obstacle is moved, the AMA graph must be recalculated. A representation other than the AMA graph is clearly necessary.

This need for a practical, dynamic representation of curves in space has been recognized in the past, primarily in the fields of computer animation and mechanical design. Terzopoulos, Witkin, Kass et.al. [1988] first introduced the use of deformable models for extracting shapes from video images and for simplifying the generation of animations. In their work, the features of video images were converted into a force field that acted on a string of spline points called a snake. The spline minimized bending and stretching energies to produce a smooth curve.

Later, Terzopoulos et. al. [1988] defined a model which used FEA to describe curves and surfaces, producing smooth paths with realistic properties. By defining a series of loading functions, they were able to create objects that mimic visco-elasticity, plasticity, fracture, and inelasticity. However, this model fell short in its ability to facilitate large changes in the structure of the objects.

Thingvold et al. [1990] discussed a method to move path points in space using a simple energy model of axial and bending springs. The goal of their work was to allow a designer to modify a surface using only a few coarse control points and to have the physical model automatically determine the location of a dense grid of control points. These points, modeled as point masses, were acted on by a network of springs that ensured that the points were both separated as well as consistent with the shape of the surface. These points were then used as control points to a tensor product B-spline.

Celniker and Gossard [1991] demonstrated the use of energy-based, finite-element deformation models for sculpting curves and surfaces. They represented a curve with Hermite polynomials so that boundary conditions could be easily defined. The bending and stretching energies were modeled and minimized using calculus of variations. The final shape was determined using a finite element method. With a dynamic time stepping algorithm, the curve was allowed to deform until the external loading matched the internal energies.

Similarly to Celniker and Gossard's model, Shimada [1992] used the finite-element form for curves to design components using high level commands. In Shimada's model, the designer was able to specify just a few regions of the boundary of a 2.5D part and connect those regions with an elastic band of points. Repulsion and attraction fields then moved the curve "near" to the desired regions.

Together, Shimada and Gossard [Shimada 1992] focused on developing a dynamic curve representation for 2D detailed shape design. The methodology was to have the designer specify boundary conditions and environmental potentials for a FEM-based deformable curve. The curve itself sought a smooth shape by minimizing a functional defined as a sum of energy weighted for bending and stretching deformation as well as for potential energy related to geometric constraints. The functional was solved as an interactive dynamic problem.

Quinlan [1993] extended the concept of snakes to provide for collision avoidance for the domain of 2D robot motion planning. His approach represented the path of a robot as a series of points at the center of a string of spheres. The radii of the spheres were maximized so long as the spheres stayed within the free space. The number of spheres required to represent paths was directly related to the tightness of the routing region. The spheres were modeled as points in constant tension and were animated using a time stepping procedure. The final path of the routing was defined as a cubic Bézier curve that interpolated between control points related to the sphere's centers.

In this work, these approaches are extended by combining a physically-based model of the harness with a spherical collision detection capability. This model shares some similarities with Quinlan's "sphere" model for 2D robot path planning in that each bundle is represented by a series of spheres connected by virtual springs. However, the sphere size is fixed as the diameter of the bundles it represents. Figure 6.1 below shows a transition with three bundles (left) converted into the sphere representation (right).

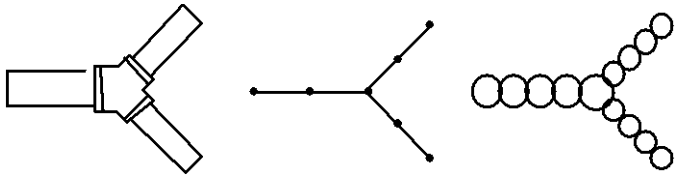


Figure 6.12 - The graph based routing is converted into connected spheres.

The harness occupies the space defined as the union of all the spheres, that is:

$$H = \sum_{i}^{nS} S(s_i) \text{ where } S(s_i) = \{q : |s - q| \le r(s)\}$$

Each sphere can be connected to either 1, 2, or 3 other spheres (more in the case of ganged transitions) corresponding to port spheres, bundle spheres, and transition spheres respectively.

Several different primitives, including superquadratics [Terzopoulos 1991], were also considered for the bundles' representation. In fact, using the original cylindrical model of each link of the AMA graph was considered for a routing primitive. This technique was not pursued since the cylindrical representation requires more calculations than is practical for collision detection and since it does not readily mimic the shape of the bundles. The sphere-based model developed for this work, by contrast, is a representation that gives the designer a feel for the path of the harness while still being tractable for high speed.

The following section discusses how the AMA links are converted into spheres, how the original spacing of the spheres is chosen, and how spheres are added or subtracted as the length of each bundle changes.

6.3 Transforming the AMA into the Sphere Representation

The sphere radius, r_i, must first be defined to convert the AMA links into the sphere representation. The sphere radius definition depends on the specifics of the routing domain. For example, in the cable harness domain, most of the wires are encased in a conduit that holds a discrete number of wires. This means that the outside radius of the bundle is only weakly related to the number of wires in the bundle and that it must therefore be determined by a table. In the wire routing/bundling domain, however, individual wires are bundled together and fastened by a tie. In this domain, the outside radius of the bundle is strongly related to the number of wires it contains and therefore it can be determined using a simple equation that uses only the number of wires as input. For simplicity, the wire bundling domain was assumed in this chapter. Under this assumption then, the bundle and likewise the sphere radii are a function of the number of wires inside the bundle and the amount of insulation and shielding, if any, around the bundle, is as defined below:

$$r(s_i) = r(B_i) = \sqrt{\frac{4}{3}\sin\frac{\pi}{3}\sum_{j}^{n} W_j^{s_i}} \quad + \text{insulation thickness}$$

The expression is derived by assuming a large number of wires in a bundle such that the internal spacing between wires is considered to be the open space in the bundle. The following figure shows the open space when three wires are bundled**.**

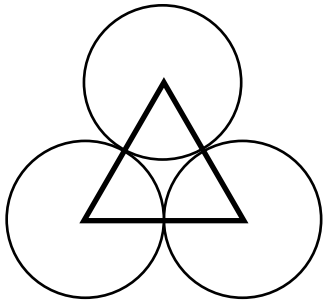


Figure 6.2 - A small amount of open space is assumed between wires

Obviously, this expression is only an estimate since the actual cross-section of the bundle is dependent on many factors.. For example, as the curvature of a path increases, the bundle tends to flatten into an oval to reduce the axial stresses in each of the wires.

A minimum radius is used in the case where a link only contains a few wires. This both reduces the number of spheres required to display the harness as well as increases the time constant used in the dynamic simulation (discussed later).

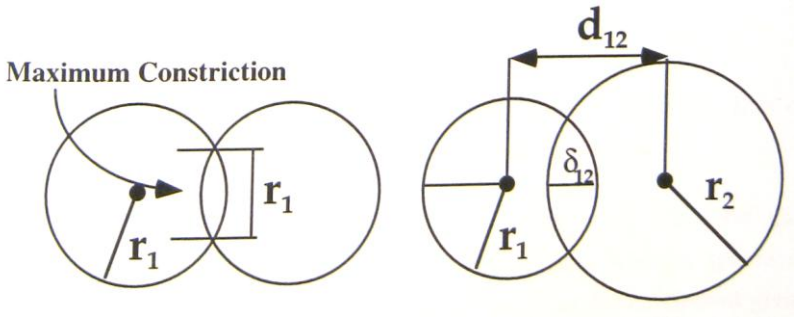
The sphere conversion process starts by replacing every node on the AMA graph that is used by the cable harnesses with a sphere. If the node has two links stemming from it, then it is a bundle node and it is represented by a sphere with the same radius of the bundle. If the node has three links, then it is a transition and has a radius of $r(s_{T_i}) = \frac{\deg}{2}\sqrt{\sum_{j}^{\deg} r(B_j^{T_i})^2}$ where $r(B_j^{T_i})$ is the radius of the jth bundle emanating from transition T_i and the degree, deg, is the number of bundles that T_i connects. If the node has more than three links and a harness topology is desired, then it must first be subdivided into multiple nodes. First, the two links with the smallest angle between them are removed from the node and connected to a new node. These links are chosen for removal since it is assumed that if two links point in the same general direction from a node, then the associated Steiner point would most likely lie in the direction of the link. This new node is then connected to the original node. This process is repeated until the original node has only three links. If a node has N different neighbors, where N>3, then that node will be divided into

N-2 separate nodes. While this operation eliminates the problem of too many links coming from a transition, it does, by definition, cause the minimum length constraint on transbundles to be violated. This can be corrected via an interactive process discussed later in this chapter.

The next step in the conversion process is to replace the AMA graph links with a series of connected spheres. In this step, the spacing, or overlap, of these spheres becomes an important consideration. If the overlap is too great, a large number of spheres must be used to represent the link—which results in problems later when the designer interacts with the design. Similarly, if the spheres are spaced too widely, the spheres may actually separate from each other during animation.

The desired amount of overlap between spheres is subjective. The closer the spacing, the closer better they collectively represent the shape of the bundle. However, closely spaced spheres present modeling and stability problems. On the other hand, if the spheres are spaced far apart, the bundle may then appear to have severe "constrictions" where the spheres overlap. Aside from being aesthetically displeasing, these "constrictions" present problems during animation as sharp obstacles may further separate these spheres. As a compromise, the desired maximum "constriction" rate that the spheres should achieve was defined as 50% (See Figure 6.3A).

The degree by which two spheres overlap is quantized by a dimensionless variable called the scaled overlap, $\sigma_{12} = \frac{\delta_{12}}{\min(r_1, r_2)} = \frac{r_1 + r_2 - d_{12}}{\min(r_1, r_2)}$. A scaled overlap of 0.268 corresponds to a 50% bundle constriction. The Figure 6.3B shows that the scaled overlap is just the amount two spheres overlap divided by the smaller of the two radii.



A. Maximum Constriction *B. Scaled Overlap*

Figure 6.3 - The deletion/adding process depends on the spacing of adjacent spheres.

Because the sphere representation will be used by the designer to modify the bundle paths and topology of the harness in real time, the representation must be able to adjust the length of each bundle. As the number of spheres required to represent a path depends on the path's length, spheres must be added or removed to compensate. All decisions as to when to add or delete spheres are based on the scaled overlap. Adding spheres occurs when the spacing between two spheres becomes too large. "Too large" in this case, is defined when the overlap between two spheres, $\sigma_{insert}=\sigma_{12}$, becomes less than the desired 0.268. A new sphere equal to the size of the smaller of the two spheres is then added between the two spheres. These new spheres will immediately have a scaled overlap of 1.134 with the two spheres.

Redundant spheres, like the center sphere in the figure below, are deleted when the scaled overlap, $\sigma_{delete}=\sigma_{13}$, is greater than the arbitrarily chosen constant, 1.3. A scaled overlap of 1.3 was chosen because it ensures, by being larger than 1.134, that a new sphere will not be immediately inserted in the space where the newly deleted sphere once existed.

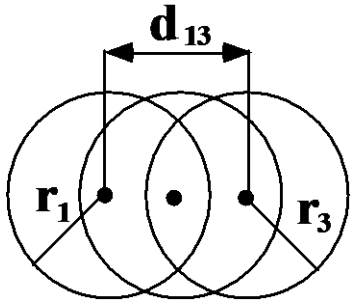


Figure 6.4 - The center sphere is considered redundant as the outside spheres adequately represent the bundle.

Only spheres with two neighboring spheres are candidates for deletion. This constraint prevents the deletion of spheres associated with ports and transitions.

While the concept of adding and deleting spheres seems simple at first glance, the implementation contains some pitfalls. The concern is that when a sphere is added between two spheres, it may in turn overlap its new neighbors by an amount greater than is allowed by the deletion strategy and thereby be instantly removed. This adversely affects dynamic stability during the animation. This problem does not occur so long as $\sigma_{insert} < 2\ (\sigma_{delete} - 1)$. The

constants of $\sigma_{insert} = 0.268$ and $\sigma_{delete} = 1.3$ keep this from happening. The compromise of using these constants is that paths may be produced with more spheres than is necessary.

Spheres associated with transitions may have additional constraints. First, most transitions for harnesses with conduit shielded bundles are built from standardized components that may be defined by the designer. These transitions come in standard shapes —mainly planar T and Y shapes. The breakaway for a T shaped transition (the leg of the T that is not parallel to another leg), usually comes in 30, 45, 60, 75 and 90 degree angles. These orientation constraints are maintained by specifying the angles between the transition sphere, s_T, and its three neighbors, s_1, s_2, s_3, where s_1 and s_3 are furthest apart. The spheres s_T, s_1, and s_2 are used to define the plane that must contain s_3. The spheres, s_T and s_1 are used to define a principle axis that helps specify the location of the third and fourth spheres based on the type and breakaway angle of the transition.

The orientation of ports also can be included in the physical model of the harness. Similar to the spheres in the transition, the sphere associated with a port and its connected neighbor share a spatial constraints in that the location of the neighboring sphere must remain on the axis defined by the port orientation. That is,

$$s_1 = s_p + tn \text{ where } t \in \Re.$$

The specification of transition type and port orientation can be defined anywhere in the design process. Frequently, the port orientations are fixed upstream in the design process. This may result in inferior routings since the upstream designers typically do not consider the actual routing of the harness while defining the port locations and orientations. Even more interesting is the fact that transition types are sometimes picked before the routing is defined as well. Major cost and time savings can be realized by taking the approach of least commitment, by first routing the harness without restricting the orientation and locations and then using the routing itself to specify these constraints. In this way, the constraints are driven by the routing.

6.4 Physical Model of Bundle Paths

Once the spheres have been defined, the next step is to define the interrelationship between them to simulate the harness. The goal in the dynamic simulation of bundle paths for the human designer is *not* to model the shape of a harness precisely but to give it a flexible and interactive feel. Since the designer needs to rapidly interact with the design (e.g., pull a bundle, change the topology, move obstacles), the representation must be able to quickly change accordingly. By allowing the designer to operate on a complete design, many of the spatial and topological problems of the routings can be corrected earlier in the design process.

In the physical-based model of the bundle paths, each sphere of the path is considered to be a point in space that is affected by a composite of simulated loads as shown in Figure 6.5. These loads produce paths that are as short as possible while avoiding obstacles, simulating gravitational fields, and addressing bundle bending constraints. The result is a representation that can define a realistic shape and give the designer the sensation of manipulating a harness made of flexible taffy.

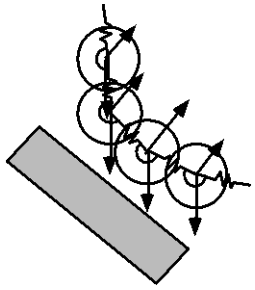


Figure 6.5 - Strand of point masses showing various loadings.

The dynamic formulation depends on the modeling needs. If the designer wants to simulate the motion of the harness during dynamic loading, such as vibration or impulse loads, then a second order model is required. However, a second order dynamic formation is overkill and prone to numerical instabilities during the design phase, where the shape of the harness undergoes rapid and sometimes drastic changes. A first-order model, shown below, is sufficient for most designer interactions in the conceptual phase.

$$\mathbf{C}\dot{\mathbf{X}} + \mathbf{K}\mathbf{X} = \mathbf{F}(\mathbf{X})$$

$$\dot{\mathbf{X}} = \mathrm{G}(\mathbf{X}) = \frac{1}{\mathrm{c}}(\mathbf{F}_{\mathrm{env}} + \mathbf{F}_{\mathrm{axial}} + \mathbf{F}_{\mathrm{bending}} + \mathbf{F}_{\mathrm{sphere}} + \mathbf{F}_{\mathrm{obs}})$$

$$\mathbf{X}_{n+1} = \left(1 - \mathbf{C}^{-1}\mathbf{K}\Delta t\right)\mathbf{X}_n + \mathbf{C}^{-1}\mathbf{F}\Delta t$$

where $\mathbf{C}$=c$\mathbf{I}$ and Δt is the time step.

The tricky component in using the model is choosing the time step. The time step should be as large as possible to produce rapid interaction without causing instabilities. Since the system has several non-linear components, determining the time step is not deterministic. However, to obtain a feel as to the bounds on the time step, the system can be simplified to collection of points in a dampening field, each connected by linear springs of equal stiffness, k. With this simplification, the system becomes uncoupled and the system of particles can be modeled as three separate systems in each of the principle axes. It can be shown that for $\mathbf{C}$=c$\mathbf{I}$, the largest eigenvalue for the system, $\lambda_{\max}$, is no greater than $6\frac{\mathrm{k}}{\mathrm{c}}$. The largest eigenvalue is typically associated with the transition spheres since they are connected to the most number of other spheres. The time step for long term numerical stability for this case would need to be less than or equal to $\frac{\mathrm{c}}{3\mathrm{k}}$.

The real model of the harness differs from this simplification in three ways. First, the springs connecting the spheres are non-linear. This is more true for the bending springs than the axial springs defined later in this section. Second, the spheres are subject to external loads that are a function of their location in the free space. These obstacle repulsion forces are both non-linear and stronger than the intersphere springs when the spheres are very close to the obstacles. Third, additional instabilities may occur when spheres move so far during a time step that enough space is produced between them to allow one or more spheres to be inserted (see Section 6.4.2). As no practical analytical model is seen to account for these issues, the time step was decreased to one-tenth that of the minimum value to avoid numerical instabilities. A limit on the maximum distance a sphere can be moved during a time step was also set to reduce the chances that numerical instabilities occur due to sphere insertions.

The following subsections list the various loadings that are selectively applied to each of the point masses. The actual load parameters depend on the particular feel desired by the resulting routing.

6.4.1 Environmental Loadings

The physical model enables environmental loading to be simulated, depending on the order of the modeling of the dynamic system. For static applications, the primary environmental loading is due to gravity. Gravity forces are added to the system by simply multiplying the mass of each sphere by the gravity field. The mass of each sphere is estimated by dividing the mass per unit bundle length by the number of spheres per unit length.

$$F_{env} = m(s_i)g$$

Since many applications of cable harnesses are for the aerospace industry, second order dynamic models that can simulate the effects of vibration and inertia would be useful. However, as these issues are secondary to routing concerns and as they require a more accurate model of the harness, a second order dynamic model was not developed.

6.4.2 Sphere-to-Sphere Axial Forces

Two conflicting motivations must be balanced regarding axial loading. First the spheres must remain as close to being equidistant from their neighbors as possible. Second, the spheres should exhibit near constant tension as the length of the bundle changes. While pure tension forces are used by Quinlan, springs are used here to ensure that the spheres will try to remain equidistant-distant from one another. This is especially important if the bundles are to be pulled tightly over the corner of an obstacle because the action may result in separated spheres.

The following function defines the loading on particular spheres.

$$F_{axial} = k\alpha \sum_{j=1}^{deg} r(s_i)\left[(s_j - s_i) - d_o\right]$$

where $d_o = \gamma\Delta_{max}$ and Δ_{max} is the maximum displacement between spheres before a new sphere is inserted and γ is a constant around 0.2. As the spring constant, $k\alpha r(s_j)$, is proportionate to the "cost" of the bundle, costlier bundles will be shortened by elongating other bundles until equilibrium occurs. The designer generates strong impulse loading on the dynamic system when interacting with the design. For this reason, a non-linear component was added to the spring model using a free length equal to the original separation of the spheres with a scaled overlap of 0.268 as shown below.

$$F_{axial} = k\sum_{j=1}^{deg} r(s_i)\left[\alpha(s_j - s_i - d_o) + \beta(s_j - s_i - d_o)^2\right]$$

The non-linear component of the spring model, although small compared to the linear portion, tends to soften the effect of large impulse loads.

The effect of the axial springs is to cause the tension of the bundle to change with the length of the bundle. Fortunately, as spheres are inserted and deleted as the length of the bundle changes, the change of the tension of the bundle is mitigated. The following figure shows the worst case scenario of how the tension drops when spheres are inserted into the bundle as it is stretched. This is the worst case because it assumes that the bundle is stretched very slowly, thus letting spheres be inserted only when all the spheres are separated by the maximum allowable distance. As the spacing of the spheres is usually non-uniform in practice, spheres are inserted at more random intervals. This results in a more uniform loading.

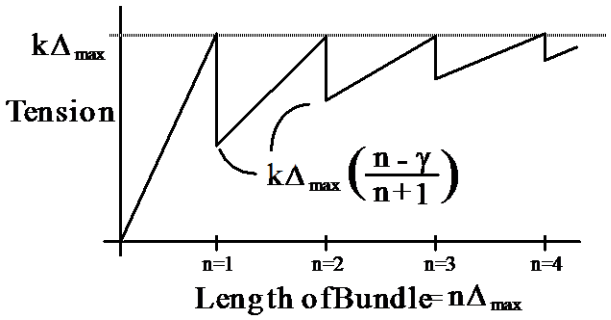


Figure 6.613 - The tension in the bundle changes with the length of the bundle as spheres are inserted or deleted.

The minimum bundle length constraints add complexity to the axial loading. As the axial springs shorten the length of the bundles, they may become shorter than the minimum allowed and therefore require special attention. This situation is addressed by adjusting the spring's free length as a function of the bundle length. The following equation is used in this work to scale the free length.

$$d'_o = d_o \left\lceil \left(\frac{L_{min}}{L} \right)^3 + 1 \right\rceil$$

where L_{min} is the minimum length of the bundle

6.4.3 Sphere-to-Sphere Bending Forces:

If the axial spring loading was used alone for the dynamic model, the paths would have sharp bends as the cables go around corners. Bending forces are modeled to address the minimum bending radius constraint. The bending constraint is maintained indirectly by using a non-linear spring constant that produces large forces as the path of the cable approaches the bending constraint.

Since the bending radius must be calculated many times during the dynamic simulation, a simple estimate is used to determine the bending radius. Figure 6.7 shows an example of three consecutive points in a path. The bending radius, ρ, is determined for this example as follows:

$$\rho = \frac{L}{\tan\left(\frac{\theta}{2}\right)} \text{ where } L = \frac{1}{2}\min(L_1, L_2)$$

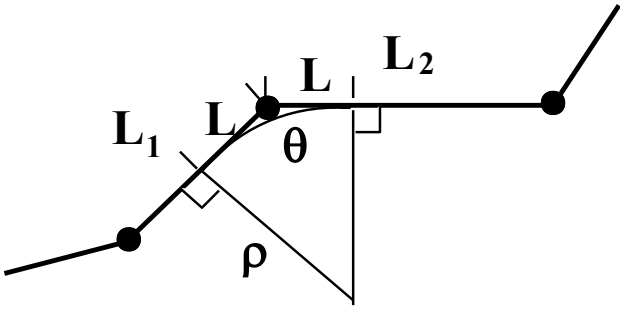


Figure 6.7 - The bending radius is quickly estimated from the angle between two links.

The bending moment associated with the bending radius comes from the well-known expression, $M=EI/\rho$. Since this expression only holds true for Euler beams with minimal deflections and the goal is to simply mimic realistic behavior, only the inverse relationship between M and ρ is used. Once an estimate of M is determined, it is decomposed into two couples acting on the links L_1 and L_2, using the assumption that the links are rigid. This is illustrated in the figure below. The forces required to produce the couples are applied at each of the points.

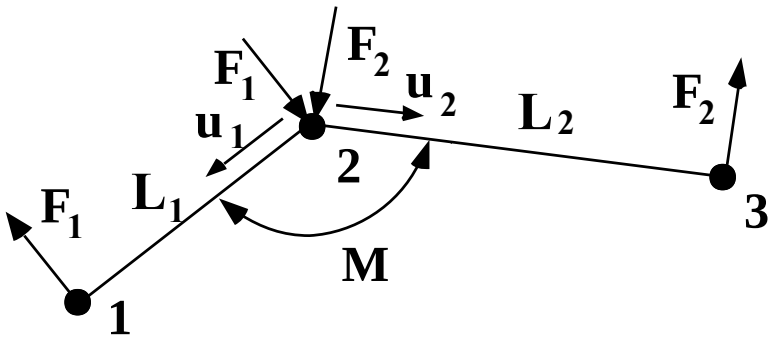


Figure 6.814 - The bending moment is applied as a couple on each node.

The magnitudes of the forces F_1 and F_2 are defined as follows:

$$F_1 = \frac{M}{2L_1}$$

$$F_2 = \frac{M}{2L_2}$$

$$F_{bending}(s_2) = F_1 + F_2$$

By dividing the bending moment, M, into two couples on the links L_1 and L_2, the effects of M on the two links is roughly inversely proportional to the lengths of the links. This makes intuitive sense since in the above figure, if a spline was generated using the points as controls, that spline would most likely have greater curvature between points 1 and 2 than between points 2 and 3, and hence it would exhibit greater bending moments.

6.4.4 Sphere-to-Sphere Repulsion

Since the spheres are representing solid entities, they must not intersect other spheres with the exception of their adjacent spheres. A repulsive force, shown below, is defined for the spheres causing them to avoid contact and enabling multiple harnesses or different sections of the same harness to compete for space when routed in the same region. The function, $F_t(\rho)$, relates the sphere separation to the following forces.

$$F_{sphere} = \sum_{j=1}^{n} Fr(p(s_i - s_j))$$

$$\rho(s_i - s_j) = \rho = \text{separation of spheres } s_i \text{ and } s_j \text{ (See Figure 6.9)}$$

Where $$F_r(\rho) = \begin{cases} A - B\rho & \rho < 0 \\ A(1-\alpha) & 0 < \rho < \rho_{max}, \\ 0 & \rho_{max} < \rho \end{cases} \quad \alpha = \frac{\rho}{\rho_{max}}, B \gg A$$

This loading function is one of many possible (see Figure 6.9)

Computational complexity is a pressing issue of sphere-to-sphere collision detection since harnesses represented by thousands of spheres are typical. The number of collision checks during each loading cycle could be of the order $O(B^2)$ where B is the total number of spheres in the system. This complexity is greatly reduced by using hierarchical bounding boxes that track which spheres are in which large cells of the environment. The procedure is to cast a coarse grid of cubical cells over the environment. The edge length of these cells are approximately equal to

five times the diameter of an averaged sized sphere. This number is a tradeoff between memory constraints and the number of collision checks needed. Once all the spheres are divided into their corresponding cells, each sphere need only be checked for collisions with the other spheres in the cell and the spheres in that cell's neighboring cells. This reduces the theoretical time complexity to O(B) and in practice to around 30B.

Several special cases for sphere collision checking cases must be addressed for the CHRP. The first is that spheres that are near to one another in the same harness should not be checked for collisions since they intersect by design. Programmatically this is accomplished by not checking for collisions between spheres that are either adjacent or separated by only one other sphere. Collision detection is also not performed for spheres that are within three spheres from a fixed sphere (port spheres and spheres fixed by the designer in a clamping operation are considered fixed). This avoids the problem of spheres being inserted near fixed spheres and quickly being pushed away.

The definition of the relationship between the separation of the spheres and the sphere-to-sphere repulsion forces can greatly change the shape of the final routings— especially if more than one cable harness is routed in the same environment. For example, consider the following two force profiles in Figure 6.9. Profile A starts with no load for $\rho > \rho_{max}$, then increases linearly as ρ approaches 0. This profile produces bundles that are always far apart from each other. Profile B is similar to A but has an attraction region where the repulsion force is negative. This profile promotes bundling between different bundles since spheres settle within an attraction well. Profile B is very useful for a domain in which it is desirable to have bundles that use similar routing channels. Using this loading profile, different bundles will appear to have magnets that cause them to stick together.

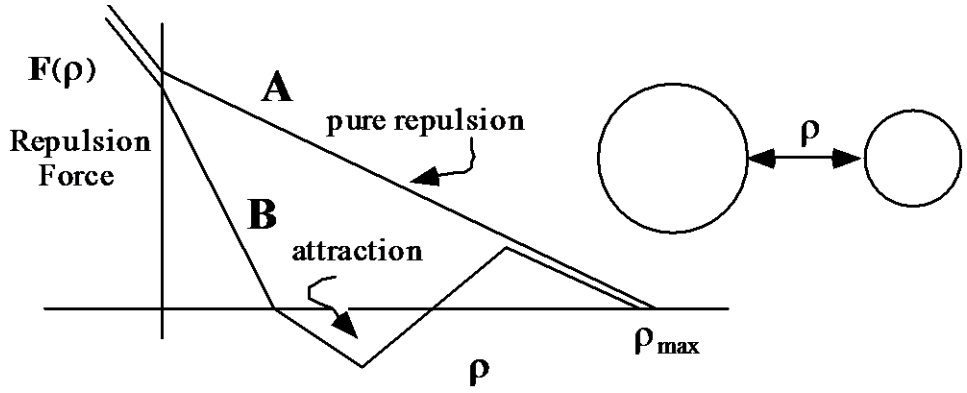


Figure 6.9 - Two different repulsion profiles produce different routing properties.

Both profiles have a large repulsive force for negative ρ. This causes bundles that are initially routed along the same AMA graph link to quickly separate. To keep the bundles from separating in random directions, all the spheres in each bundle of each harness routings are initially offset by a small amount from the other bundles.

6.4.5 Obstacle Repulsion Field

One fundamental requirement of the harness routing automation is that it must not produce routings that intersect obstacles. Most collision detection schemes for CAD either use analytical models or approximate methods. Since rapid interaction is crucial for interactive harness design, an approximate method is used. In this case, the voxel representation used for the creation of the AMA is reused to produce a lookup table of the minimum distance between a point and an obstacle. A repulsive force field was used to keep the routings from intersecting the obstacles. In general, a repulsive force field applies a load on a given point in the direction away from the closest obstacle provided that the point is within a minimum distance from the closest obstacle. The distance, ρ(s), from the surface of a sphere to its closest obstacle as well as its direction, u, are shown below. **Grid**[i][j][k] is the voxel that contains the center of the sphere and e_{vox} is the edge length of the voxels.

$\rho(s) = e_{vox}(\mathbf{Grid}[i][j][k] - 0.5)$ - Sphere Radius

$$\vec{u} = \begin{vmatrix} \mathbf{Grid}[i][j][k] - \mathbf{Grid}[i-1][j][k] \\ \mathbf{Grid}[i][j][k] - \mathbf{Grid}[i][j-1][k] \\ \mathbf{Grid}[i][j][k] - \mathbf{Grid}[i][j][k-1] \end{vmatrix}$$

The mapping between $\rho(s)$ and the repulsion force determines how the spheres will react when coming in contact with the obstacles. This function needs to produce a large force when $\rho(s)$ is small. Just as with the case of sphere-to-sphere repulsion, different functions can be used depending on whether the bundles are suppose to run along the obstacles or if they simply need to avoid obstacles.

While the linear pure repulsion shown in profile A of Figure 6.9 keeps the spheres from intersecting the obstacles, a quadratic loading profile given below works better by letting the spheres come closer to the obstacle before experiencing a heavy loading. A is an arbitrary constant and ρ_{max} is the maximum effectual range of the repulsion field.

$$F_{obs} = \begin{cases} A(1-.1\alpha^2) & \text{for } \rho < 0 \\ A(1-\alpha^2) & \text{for } 0 < \rho < \rho_{max} \\ 0 & \text{for } \rho_{max} < \rho \end{cases} \text{ where } \alpha = \frac{\rho^2}{\rho_{max}^2}$$

Since the designer is able to pull the bundles at will, bundle spheres may be pulled into obstacles. In this case, the repulsion field must be able to push the spheres back into the free space. This is done by applying the AMA generation algorithm in the obstacle space regions of the environment using negative numbers instead of positive. These negative numbers then estimate the distance to the nearest free-space.

Spheres which are near those connected to obstacles, such as in Figure 6.10, need special consideration during obstacle collision checking.

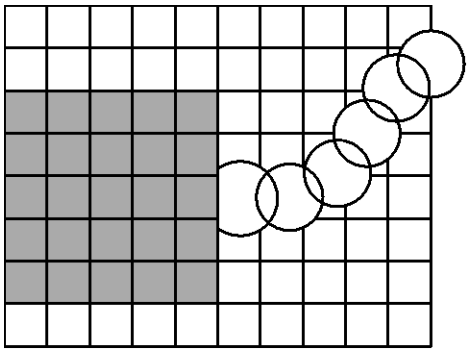


Figure 6.15 - Sphere next to obstacle require special attention.

The main difficulty with these spheres is that they are strongly affected by the obstacle repulsive field. This field causes spheres near ports, for example, to be blown away while new spheres are being added to the end. This results in a stream of spheres being created like spaghetti coming out of a spaghetti press. The way this problem is addressed is by reducing the strength of the repulsion field on the spheres near the obstacles. This is done by scaling the strength of the repulsion field by the distance from the obstacles as follows:

$$F'_{obs}(s_i) = \begin{cases} \dfrac{l(s_j)}{l_{\max}} F_{obs}(s_j) & l(s_j) < l_{\max} \\ F_{obs}(s_j) & \text{otherwise} \end{cases}$$

$$l(s_j) = \sum_{1}^{j-1} d_o(s_i, s_{i+1})$$

6.3.6 User defined loads

The designer is able to affect the shape and routing of the harness in several ways. First the designer can pull a bundle by attaching a virtual spring from a sphere to a 3D mouse point. The spring simply adds an additional load on the sphere. The designer can also define the location of a sphere or attach it to the point by a spring. In some cases, the designer can also add a point load that simulates direct loading on a sphere.

6.5 High Level Interaction

The designer is now able to engage in a much higher level of interaction with a physical model of the harness than would be possible with a spline model without complex CAD commands to learn and remember. The difference between the user interface using a physical model and other modeling packages is that the designer only needs to use a mouse to perform major design modifications. This simplicity is very well received in practice since it enables designers with even minimal computer skills to be productive. This section will discuss three of the more useful interactions used in cable harness design to create every possible topology and topological routing.

6.5.1 Move (Clamp) Bundle

Once the designer is satisfied with the topology and layout of the harness, he/she can then focus on specifying the detailed path planning of the bundles. This is done by having the designer click on the bundle to be moved (clamped) and having the closest sphere attach to the 3D mouse as illustrated in Figure 6.11 below. The designer can then move the sphere until clicking the mouse again. This action releases (fixes) the bundle. If the bundle is to be clamped, the bundle model will fix the location of the bundle and automatically generate a smooth routing through the point by animating the routing. While this type of interaction mimics what one can do with a physical harness, it lacks the power to redesign the shape of the harness or move the bundles around topologically complex environments. It is for these cases that the Move Branch operation was developed.

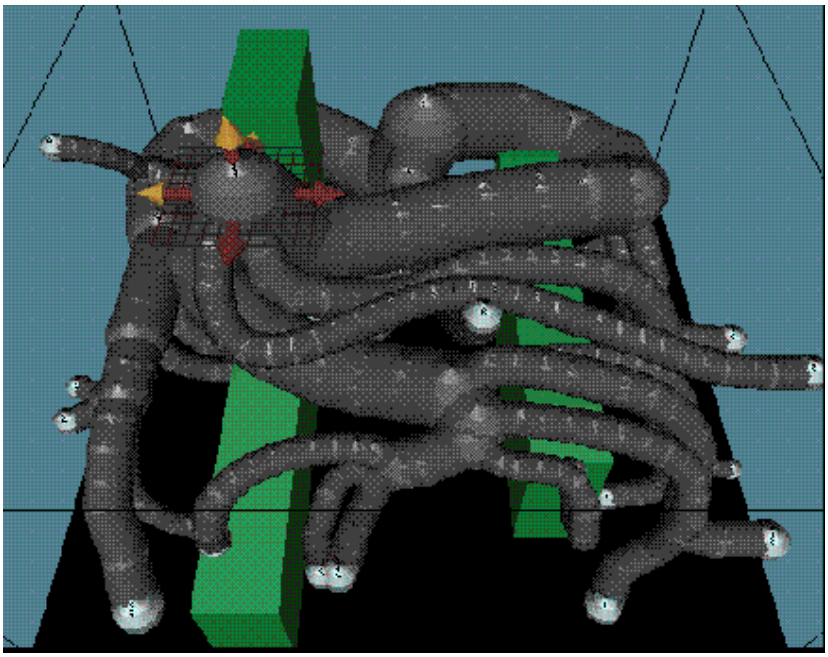


Figure 6.1116 -An illustrative example of a large move bundle operation

6.5.2 Move Branch Operation

The designer may want to change the topology of the harness or move a bundle in a way that is impossible if the bundle is not first cut in two (e.g., a cable that routes to one side of a pillar must be cut before it can be rerouted to the other side). The move branch operation is most useful in complex routing environments or when multiple harnesses are to share the same space. This occurs as it is more difficult to modify the routing of a harness when it weaves around obstacles or is covered by others. The following figure shows a typical example of the crowded environments in which harnesses may be routed. In this case, if a buried harness needs to be rerouted, it must often be cut in two before being reconnected.

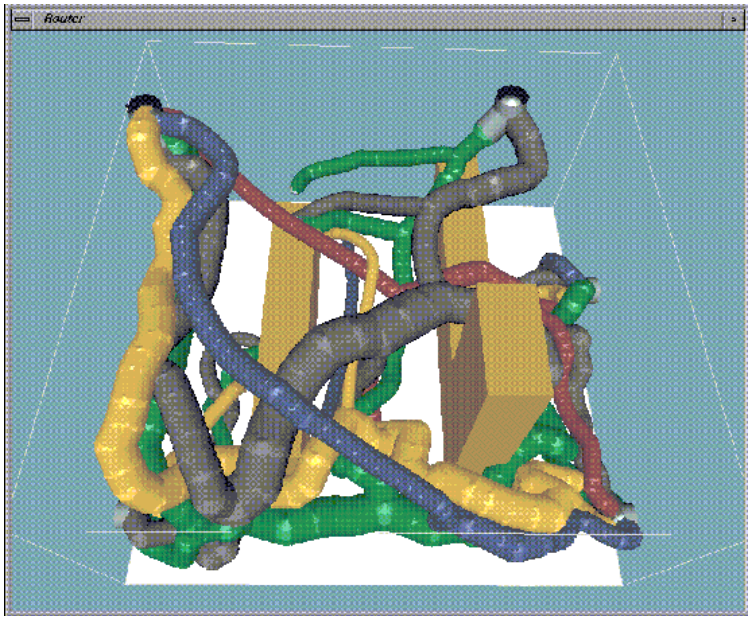


Figure 6.117 -Multiple harness may compete for space

The following describes the steps a designer would take to make a move branch operation on the twelve-port cable harness shown in Figure 6.13Figure 6.. Assume in this example that the designer wants to modify the harness so that it doesn't pass through the area between the two obstacles. (Note: the harness routing in this figure represents the lowest cost routing.)

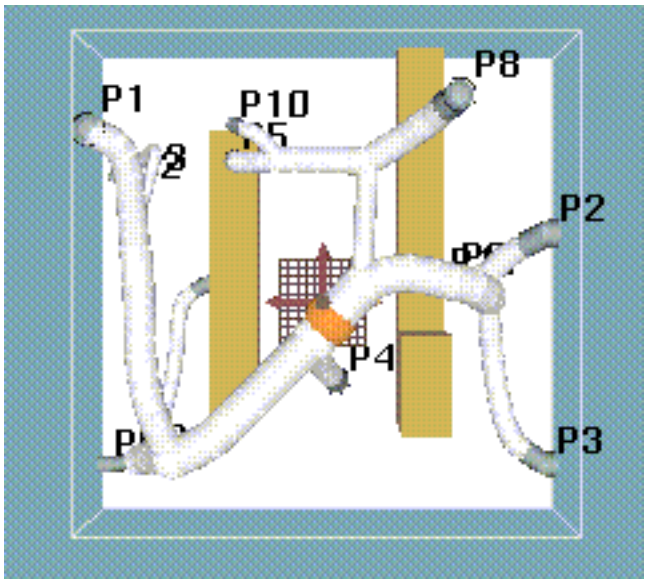


Figure 6.13 - Initial Routing of 12 Port Harness

The first step is to use a 3D cursor to select a bundle to cut. In this case, the designer selects a thick transbundle in the middle of the harness. Since a transbundle is cut, the designer must then choose which end of the transbundle to move as shown in Figure 6.14. The other end is truncated to the nearest transition. The move branch operation for a termbundle is similar to that of a transbundle. The only difference is that when the initial cut is made, the bundle fragment not connected to the port is truncated at the transition.

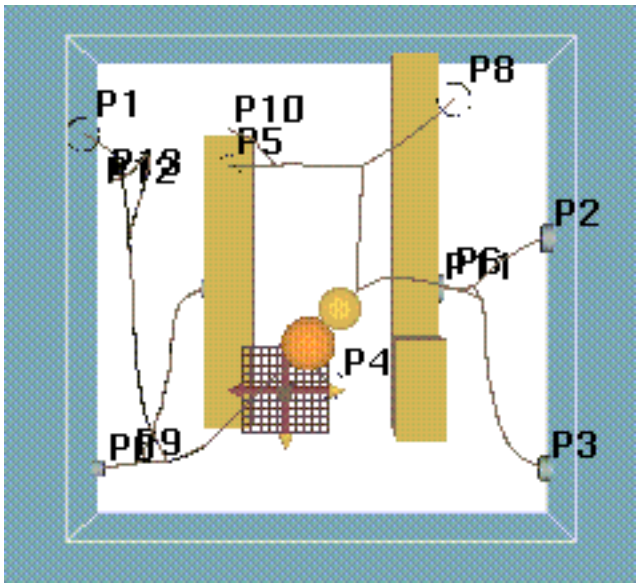


Figure 6.18 - Designer Selects End of Transbundle

Once the end is selected, the designer is able to pull the harness fragment towards any point on the other fragment. Figure 6.15 shows two steps of this process. Notice that the harness adds spheres as necessary as the fragment is pulled to the right.

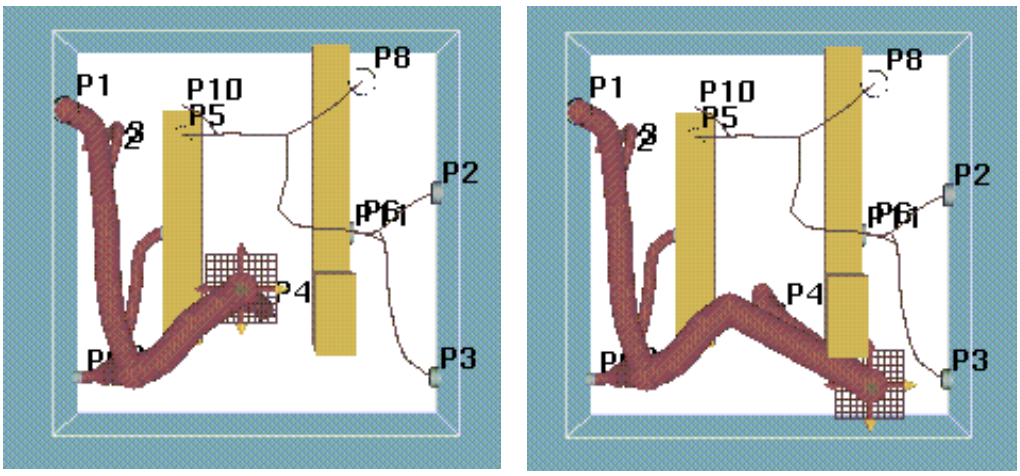
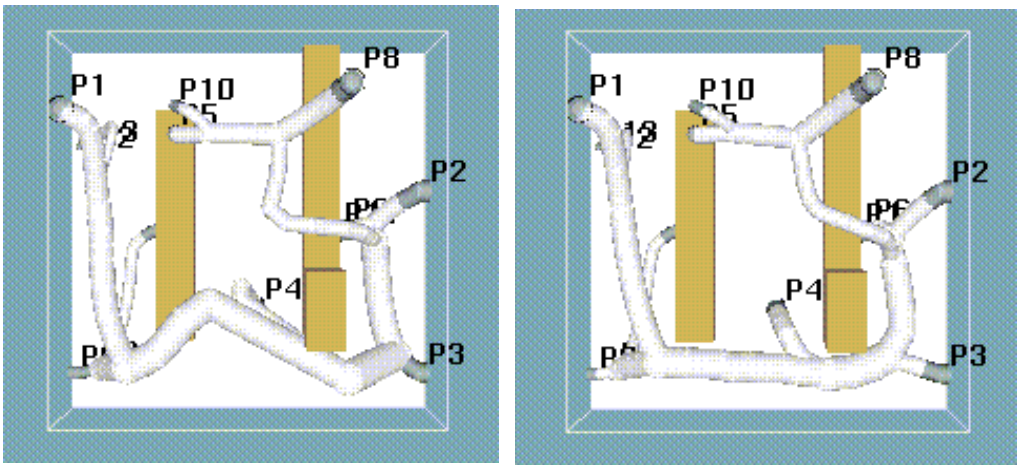


Figure 6.15 - Designer pulls harness fragment

When the end is reconnected, all the wires that passed through the severed transbundle are re-routed over the new topology. The lower-right-corner path connecting P2 and P3 in Figure 6. is

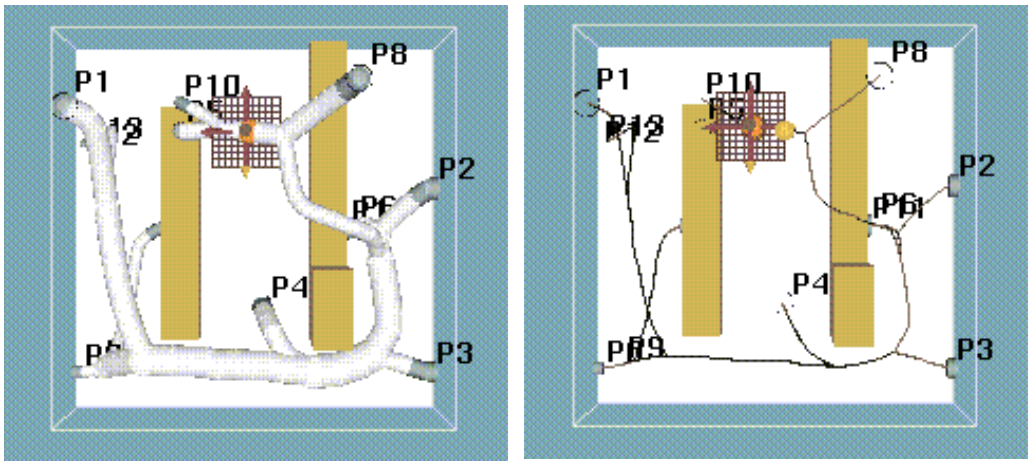
most affected by the move-branch operation in that it becomes much thicker than the original routing as shown in Figure 6. A. After the move-branch operation is performed, the harness dynamics are simulated until the system reaches a local minimum energy as shown in Figure 6.B. The designer may then specify the type (shape) of the transitions based on the angles that the new transition's bundles make.



A *B*

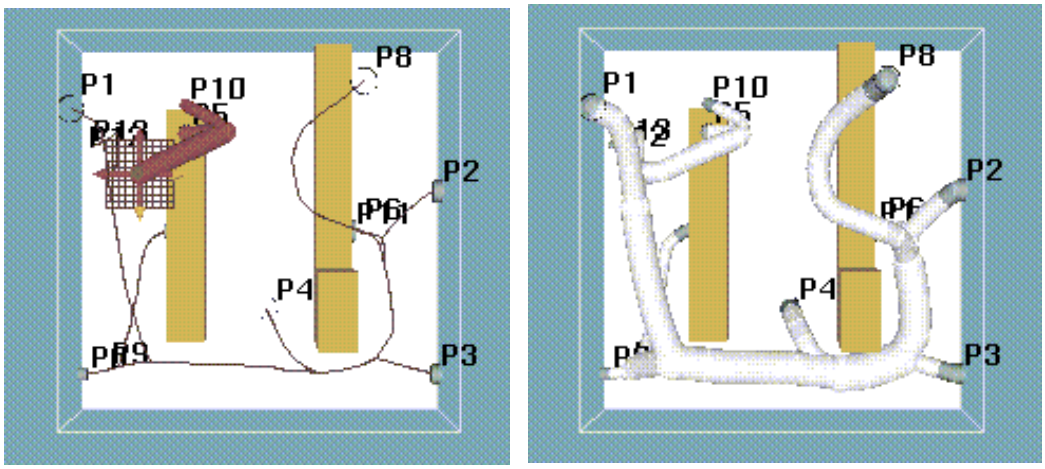
Figure 6.16 - Results of the move-branch operation

Figure 17 & 18 show the move-branch operation again for another transbundle. While this operation clears up the area between the two obstacles (see Figure 19), it adds considerable cost to the harness. This is evident in that most of the bundles in Figure 6.A become thicker in Figure 6.B. This occurs since the wiring list has several wires that connect port 8 with ports 5 and 10. In Figure 6.A, these wires take a direct route between the ports. However, in Figure 6.B, the wires must traverse most of the length of the harness to find their destinations.



A *B*

Figure 6.17 - Another transbundle is cut



A *B*

Figure 6.18 - Transbundle is moved to new location.

Once the designer has finalized the topology, the entire system is simulated until the routing settles on a new local minimum. Typically, this minimum is consistent with the designer's wishes since the routing starts in the potential well of the minimum. Figure 6.19 shows the results of this process.

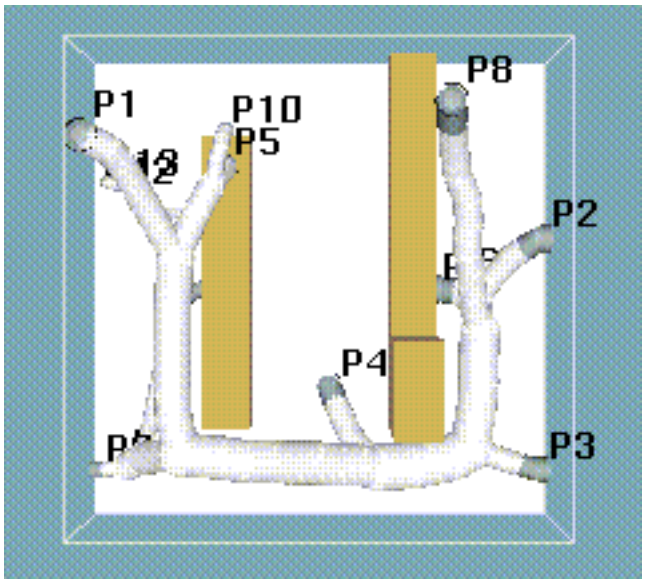


Figure 6.19 - Harness settles to another local minimum

6.5.3 Transitions Operations

The designer is given complete control over the type of each transition. A 2D layout of the harness provides the ability to specify the angle and type of each of the transitions. Once specified, the model of the harness is again simulated to reflect the transition changes. On some occasions the designer may even wish to used a ganged transition that can service more than three bundles. While not implemented in this work, ganging transitions could be specified by simply having the designer select a transbundle and automatically remove the associated spheres and connect the two transitions.

Chapter 7

Experimentation Results

7.1.Introduction

The previous chapters have discussed a search strategy for the CHRP that assists designers by using a set of automated agents in an evolutionary and collaborative system. The effort required to combine different search techniques can only be justified if the end results are significantly better than designs created when only a single technique is used given similar amounts of computational resources. This chapter presents a closer analysis of the individual routing agents as well as their relative and combined effectiveness in solving the problem. Since the nature of the problem precludes comparing the results to globally optimal solutions, only the relative performances of various tests can be measured – and only for a given set of inputs which are representative of the range of possible conditions.

The experiments discussed in this section were devised to understand the nature of the search problem and verify the effectiveness of agent collaboration to exploit it. Specifically, the individual search parameters of the Move Branch and Combinor agents were first optimized independently to isolate them from other search techniques. Each of the agents were tested using various selection and reattachment heuristics to solve a number of different harness routing problems. Next various combinations of the automated routing agents were tested on multiple test problems to compare their performance. The designs created by the automated system were also compared with designs created independently by a human designer. Before discussing the results, it is useful to overview the experimental setup and procedure.

7.2 Experimental Setup & Procedure

Although the system architecture could have been created as a distributed system running on multiple machines, all the routing tools discussed in this work were encapsulated into three programs. They are the collection of automated router agents (ARA), the designer's routing interface (DRI) that was discussed in Chapter 6, and a third program that acted as a blackboard in that it stored the design population. A communication layer using the KQML[1992] protocol was used to pass design information between the other two programs.

Most of the automated tests were run using only the ARA, initiated using a script setup file. In the case where the DRI was also used, the designer first initiated the DRI application and loaded an environment and harness to be routed. In this case, the system automatically generates an initial routing using either the Thick-Bundle or Wire-Bundling routing agents discussed in Chapter 5. The designer is then free to modify this routing to explore other designs using the interactive harness representation described in Chapter 6. When the designer wants automated help, he initiates the blackboard communication system and ARA module using an automatically generated script file. This file specifies the current environment and wirelist as well as the set of desired router tools run parameters. The designer's current design is also sent to the blackboard system to help seed the design population. The ARA then cycles through the evolutionary search process, pausing after each new design is created to post it to the blackboard. After each generation, the ARA scans the blackboard for designs posted by the DRI. In so doing, fragments of cable harnesses can be reused by the automated routers to test areas of the search space unexplored by the designer.

The designer is free to query the blackboard at any time to download the latest designs and compare them with the current design. These designs created by the automated routers are listed in order of quality. The designer can view each design in its sphere representation form described in Chapter 6 and modified it as desired. By using the blackboard system to store multiple designs in progress, a designer can simultaneously work on multiple ideas and thus improve the chances of finding better designs. The DRI module runs indefinitely until the designer terminates the program.

The script file itself is used to define all the search parameters for the ARA module. In addition to defining which routing environment and wirelist to use, the script file also defines the resource allocation for each routing agent and the design population size and number of generations to use for the topology and topological routing (Transition Locator as discussed in Appendix A) search.

7.2.1 Resource Allocation between Routing Agents

The allocation for each routing agent was defined as a percentage of all the designs available. To simplify the analysis process, each agent was either allocated the same number of design slots as the others or none at all depending on whether or not it was used. The only exception to this was for the Copy-Best agent which was always only allocated a single design slot. Under real working conditions, this allocation would be suboptimal as some agents, such as the Thick Bundle and Wire Bundling routers, do not actively explore the search space and need only a few design slots.

7.2.2 Resource parameters for the Topology and Topological Routing search process.

The product of the population size and number of generations define the total number of designs to be tested during the search. As with most large search problems, the results for the CHRP improve as more designs are tested. The limiting factor, therefore is the amount of time available to search. In this case, an arbitrary limit was set at 15 minutes to search for a 10 port harness and 90 minutes for a 15 port harness. The limit is equivalent to 90,000 and 360,000 topological routings for a 10 and 15 port harness respectively using a 100Mhz MIPS R4000 Silicon Graphics machine. The next steps are to divide the resources between the topology search and topological

routing search and to define the population size and number of generations for each. The following chart shows the distribution of resources used for many of the tests in this chapter. The population size and number of generations used in the topological routing search is considerably less than for the topology search as only a rough comparison between designs is needed by the topology search process.

	10-Port Harness	**15-Port Harness**
Topology Search		
Population Size	30	40
# of Generations	25	30
Topological Routing		
Population Size	15	20
# of Generations	8	10

Table 7.1 – Resource distribution among search processes.

Choosing between using a large number of generations or a large population size requires a compromise. On one hand, using a large number of generations enables a good design fragment to be tested in many future designs. On the other, using a large population size increases the chance that good design fragments exist in the first place. Interestingly, the number of the designs in the search space is not necessarily as important as population diversity, measured as the percentage of the total possible fragments contained in the design population.

The following scatter plot illustrates the relationship between search performance and population diversity (which is affected by the nature of the agents used). Four searches of test runs were generated using varying proportions of the Copy-Some and Move Branch agents for a balanced 10-port harness in the maze environment (Figure 2.10B). The maximum fitness (i.e., the square of the inverse of the harness cost function described in equation 2.1. Note, the square was used to increase the chances that a design of low cost is reused – especially important in this work as the difference in the cost between designs is small) and population diversity are plotted for each

generation of the run (starting from upper left and ending at the lower right). When only the Copy-Some agent is used, no new designs are created and the population diversity quickly drops and the fitness stagnates. However, as the amount of designs allocated to the Move-Branch agent (using random fragment and reattachment selection) increases, the diversity does not drop as fast and the maximum fitness improves dramatically. However, increasing population diversity using the random Move-Branch agent faces diminishing returns as the population's ability to retain good fragments decreases.

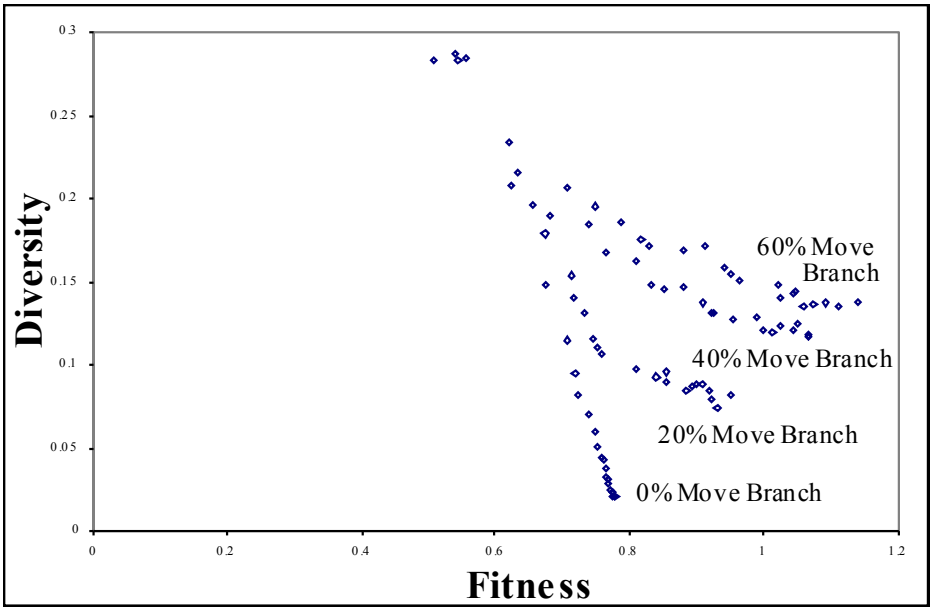


Figure 7.1 - Population diversity is an important factor to the performance of the search processes.

Goldberg's [Goldberg 1989] theoretically optimal population size formulation for genetic algorithms suggest that the optimal amount of computational effort rises exponentially with the problem size. He reports that the optimal population size is approximately $1.65 \cdot 2^{0.21 \cdot b}$ where b is the number of bits required to encode a single design. As the number of bits required to binary encode a harness is approximately 15 bits/transition (10 bits to model position and 5 bits to model topology), the theoretically optimal population size may be several orders of magnitude greater than what is computationally feasible.

The reason why most genetic formulations (binary-type genotype encoding) need such large populations is that the genetic operators work blindly on encodings of a problem where related design features are typically not near each other. This means that the chances that a standard genetic operation, such as crossover, retains a fragment are small and therefore many designs must be retained. However, the problem formulation and execution in this work drastically reduce the size of the population required to find good designs.

7.3 Router Agent Optimization

The goal in testing the individual routing agents is to see if it is possible to exploit the decomposable structure of a harness topology as discussed in Chapter 4. This can be found by testing if the performance of the agents improves if they incorporate heuristics that really on design fragments. A closer look at each of the routing agents can also provide insight as to the nature of the search problem. For example, the Move Branch agent, by acting on a single design, can help distinguish the advantages of good fragment selection from good design selection. On the other hand, the Combinor agent can be used to examine the level of "deception" of the search space (i.e., the extent to which combining high quality designs results in new designs of lesser quality). The following discusses the analysis and optimization of both these agents.

7.3.1 Move Branch Routing Agent

The Move Branch agent's performance depends on its selection of which designs to use, which fragments to retain, and where to reattach the fragments. As it was determined early that the performance of the MB agent is better when it selects designs based on fitness, the Fitness-Proportionate-Selection (FPS) method was used exclusively for design selection. As FPS outperforms random design selection, it is possible to assume that designs with above average fitnesses are more likely to contain fragments of high quality.

The fragment selection process was examined by testing random, fragment-based and bundle-based selection techniques (described in Chapter 5) on a 10 port, balanced wire list harness in the maze environment. Each run was performed eight times using only the Copy-Best, Copy-Some, and Move Branch operators. The fragments were reattached do a randomly selected branch of the complement. Table 7.2 lists the results. The fitness is measured as the square of the inverse of

the cost function defined in chapter 2 and σ_{Fitness} is the standard deviation of the maximum fitness recorded for each of the runs. While it is unclear if the sample distribution is truly normal, σ_{Fitness} gives an idea of the spread of the samples.

Method	Description	Fitness	σ_{Fitness}
Random	Randomly selects a fragment by randomly cutting a bundle.	1.023	0.072
Cost Bundle	Chooses a bundle to cut based on bundle cost.	1.030	0.127
Long Bundle	Chooses a bundle to cut based on bundle length.	0.979	0.103
FSM1	Uses the global objective function scaled to the fragment.	1.104	0.098
FSM2	Uses the difference in cost between routing wires along the fragment and directly between ports.	1.125	0.083
FSM3	Uses the closeness of ports in a fragment.	1.141	0.062

Table 7.2 – Comparison of different fragment selection techniques for a 10 port, balanced wire list harness in the maze environment. (Topology: 25 gen, 30 designs, Topological Routing: 15 gen, 8 designs).

The test results show that it is possible to isolate and reuse fragments using fragment-based selection techniques. Also, the bundle-based techniques failed to produce results better than random selection.

Once the fragment is chosen, the next step is to determine where on the complement to reconnect it. As discussed in Chapter 5, the recombination location can be chosen either randomly or to be the bundle that is the closest to the fragment trunk. For test purposes, the random selection method was compared against runs that used the closest complement bundle for half of the designs and randomly selected bundles for the other half.

The following table displays the results of eight runs various fragment selection techniques and the two recombination techniques. The columns represent the average best of generation (labeled Max Fitness) and their standard deviations, the average of the last generation, and the final population diversity (Div). The numbers show that when the closest bundle was used (50% of the time) as the reconnection point, the quality of the final design increases and the final population as the diversity decreases as compared to random reconnection.

Reconnect Technique	Fragment Selection	Max Fitness	σ_{MaxFit}	Ave Fitness	Diversity	$\sigma_{\text{Diversity}}$
Random	Random	1.0229	0.0715	0.7099	0.1453	0.0160
Random	Cost Bundle	1.0300	0.1273	0.7239	0.1443	0.0200
Random	Length Bundle	0.9791	0.1028	0.6635	0.1553	0.0195
Closest Bundle	Random	1.0993	0.1024	0.8016	0.1176	0.0144
Closest Bundle	Cost Bundle	1.1083	0.0914	0.8015	0.1109	0.0156
Closest Bundle	Length Bundle	1.1277	0.0966	0.8217	0.1074	0.0186

Figure 7.3 - Comparison of different Move Branch techniques for a 10 port, balanced wire list harness in the maze environment (Topology: 25 gen, 30 designs, Topological Routing: 8 gen, 15 designs).

Next, to check how these results are affected by biases in the wire lists, the same runs were made for a 10 port, multiple-biased wire list harness as shown below.

Reconnect Technique	Fragment Selection	Max Fitness	σ_{MaxFit}	Ave Fitness	Diversity	$\sigma_{Diversity}$
Random	Random	2.6431	0.1910	1.8100	0.1299	0.0170
Random	Cost Bundle	2.5782	0.1300	1.7258	0.1375	0.0131
Random	Length Bundle	2.6530	0.2355	1.8469	0.1324	0.0202
Closest Bundle	Random	2.6847	0.1404	1.8778	0.1367	0.0207
Closest Bundle	Cost Bundle	2.6538	0.1544	1.8572	0.1291	0.0240
Closest Bundle	Length Bundle	2.6968	0.1565	1.9729	0.1184	0.0151

Figure 7.4 - Comparison of different Move Branch techniques for a 10 port, multi-biased wire list harness in the maze environment (Topology: 25 gen, 30 designs, Topological Routing: 8 gen, 15 designs).

This time, while the closest bundle approach outperformed the random selection, the difference was much closer. The reason for this is believed due to the topology of "good" designs for harnesses with multiple-biased wiring lists. For these harnesses, a "good" design often has several thick, long bundles that connect a pair of ports as shown below in Figure 20.2.

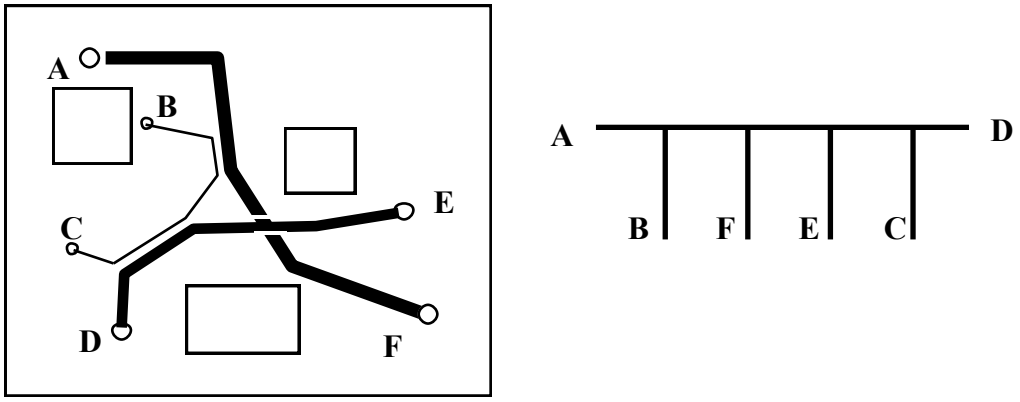


Figure 20.2 - Lowest cost harness routing with multiple-biased wirelist

Now, suppose the term bundle connecting E was cut and reattached to the closest bundle (the term bundle connecting port F). This resulting harness, shown in Figure 7.3, has a higher cost. This would always occur for the closest-bundle selection technique each time the term bundle connecting E or F is cut.

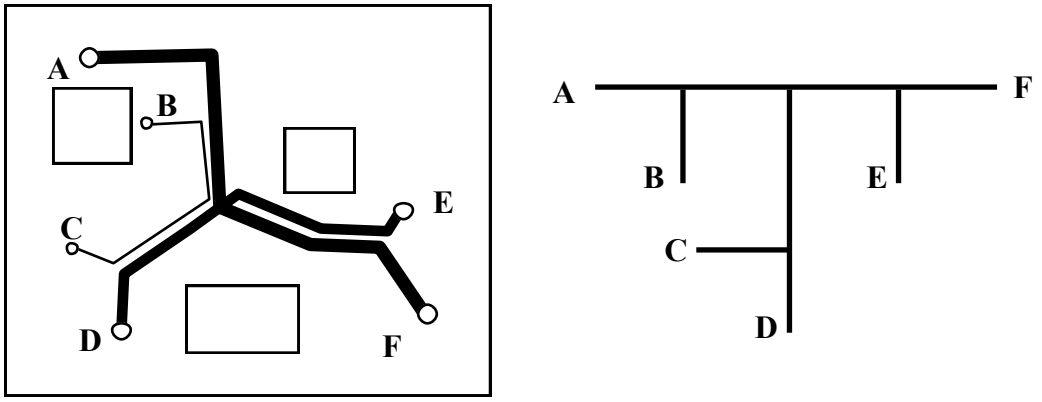


Figure 7.3 - Suboptimal harness routing with multiple-biased wirelist

As a quick check to see how these results are affected by the number of ports, a 15 port harness with a multiple-biased wire list was tested with the results shown below.

Reconnect Technique	Fragment Selection	Max Fitness	σ_{MaxFit}	Ave Fitness	Diversity	$\sigma_{\text{Diversity}}$
Random	Random	0.6268	0.0861	0.0646	0.0085	0.00125
Random	Cost Bundle	0.5765	0.1014	0.0533	0.0080	0.00145
Random	Length Bundle	0.6447	0.1423	0.07010	0.0087	0.00106
Closest Bundle	Random	0.6041	0.1166	0.0822	0.0068	0.00089
Closest Bundle	Cost Bundle	0.6405	0.1326	0.0946	0.0070	0.00059
Closest Bundle	Length Bundle	0.6329	0.0956	0.0680	0.0066	0.00095

Figure 7.5 - Comparison of different Move Branch techniques for a 15 port multiple biased wire list harness in the maze environment (Topology: 40 gen, 40 designs, Topological Routing: 10 gen, 20 designs).

Interestingly, the advantage of reattaching the fragment to the closest point of the complement was even more unclear in this case as some of the random reattachments cases actually outperformed those that used proximity as a guide. This appears to occur because harnesses with larger numbers of ports tend to take longer to converge on designs without criss-crossing transbundles, and therefore proximity information may be of limited value until later in the search process.

7.3.2 Combinor Routing Agent

Using the findings of the Move Branch agent as a guide, most of the effort in optimizing the Combinor agent was focused on the second fragment selection and recombination processes. The first fragment was chosen using the same fragment-based selection method used for the Move Branch agent (FSM3). Three different selection techniques, described in Section 5.4.2, were examined for the second, or complement, fragment. The first was design based (DB) and simply selected the second design using fitness proportionate selection (FPS). The second complement fragment selection technique was fragment-based (FB) and used the same selection method as for the first fragment (FSM3). The third, and most computationally intensive, called mutually-exclusive selection (MES), gave greater weight to designs that had high-quality fragments with many mutually exclusive ports with the first fragment.

All three techniques were tested ten times on several different 10-port cable harnesses in the maze environment. Only the Copy-Best, Copy-Some, and Combinor operators were used with an allocation of 1,14, and 15 designs respectively. The following tables show the results for three different test cases with slightly shorter runs than for the test of the Move-Branch agent. In each case, using domain knowledge to ensure compatibility between the designs to be combined (i.e., mutually-exclusive selection) significantly improved the performance of the Combinor agent (note, the numbers listed are the fitnesses measured as the square of the inverse of the cost of the designs).

Technique	Max	MaxStd	Div	DivStd
DB	2.36181	0.31273	0.13477	0.02110
FB	2.38024	0.22771	0.08340	0.02420
MES	2.72688	0.09522	0.10254	0.01316

Figure 7.6 - Results of multiple-biased 10-port harness using the design-based, fragment-based, and mutually-exclusive selection. (Topology: 15 gen, 30 designs, Topological Routing: 15 gen, 8 designs).

Technique	Max	MaxStd	Div	DivStd
DB	1.63017	0.22570	0.12324	0.01370
FB	1.61796	0.26648	0.15273	0.02305
MES	2.21279	0.03927	0.07246	0.01264

Figure 7.7 - Results of multiple-biased 10-port harness using the design-based, fragment-based, and mutually-exclusive selection. (Topology: 15 gen, 30 designs, Topological Routing: 15 gen, 8 designs).

Technique	Max	MaxStd	Div	DivStd
DB	0.80433	0.12540	0.15078	0.01746
FB	0.85349	0.05924	0.15488	0.04016
MES	1.00672	0.13791	0.14174	0.04298

Figure 7.8 - Results of balanced 10-port harness using the design-based, fragment-based, and mutually-exclusive selection. (Topology: 15 gen, 30 designs, Topological Routing: 15 gen, 8 designs).

The results of the test show that the combination process exhibits suboptimal, and perhaps even deceptive properties, using blind selection techniques. That is, choosing both designs based solely on overall fitness without considering the interdependencies of the designs' fragments produces poorer results than otherwise. This is a major point as most existing genetic techniques applied to problems of this sort do not consider the interdependencies of the designs to be combined and thus are prone to produce inferior results in many of the combinations that take place.

7.4 Collaborative Analysis

There were three motivations in examining the performance of collections of routing agents. The first was to test if the relative performance of agents changes depending on the problem type (e.g., environment complexity of wire list bias). If found, it would imply that system robustness may be improved using a collection of routing agents. At issue is that as resources are fixed, it

may not be always advantageous to use many different agents since poorer performing agents consume computational resources which could have been used by more effective agents. The second motivation was to see if the relative performance of any of the agents also changes depending on the quality of the existing designs in the population (e.g., increases as the search proceeds). The third was to see if the automated search process could benefit from inclusion of the human designer. While including the designer in the search process was primarily motivated by the desire to address unmodeled objectives and not to optimize the prescribed objective function, it was hoped that designer would be able to make minor repairs (e.g., connecting a single termbundle to a different branch) to designs created by automated routers.

7.4.1 Effect of Problem Type on Agent Performance

A close analysis of agent performance requires isolating many problem specific variables such as the effect of the routing environment complexity, wirelist bias, and harness size on relative agent performance. To this end, tests were run with different agents on an assortment of design problems. Appendix D shows the output of six test cases of 10-port harnesses with various wirelist biases and environment complexities run for 15 generations (Appendix C shows the distribution of randomly generated solutions as a comparison of performance). The Copy-Best and Copy-Some agents were always allocated a fixed number of designs in the population for each of the runs. The remaining automated routers, Move-Branch, Combinor, Thick-Bundle and Wire Bundling, were used in various combinations for comparisons. To isolate which techniques were contributing the most to success of the team and which sets worked well together, each of these four agents was tested independently (in addition to the Copy-Best and Copy-Some agents) and then all of the agents, minus the one, were tested. Important contributors would produce relatively high results when used independently and relatively low results when missing from the team.

Table 7.9 summarizes the rankings of the automated operators. The numbers below each of the agent abbreviations, δ_c, is the percent decrease in the average lowest cost of designs found when only using the one agent (in addition to the copy-best and copy-some agents) compared to when all of the agents, minus that one, were used.

Wire List	Empty Environment	Maze Environment
Multiple Biased	**CM>MB>WB>TB** 2.54>-2.99>-8.88>-9.80	**CM>WB>MB>TB** 3.48>-3.09>-3.20>-9.97
Single Biased	**CM>MB>TB>WB** 3.66>-4.27>-5.35>-9.85	**CM>WB>TB>MB** 6.73 >-6.71>-8.79>-9.50
Balanced	**MB>CM>TB>WB** -0.59>-1.38>-3.80>-3.94	**WB>TB>CM>MB** 2.47>1.52>-4.42>-9.11

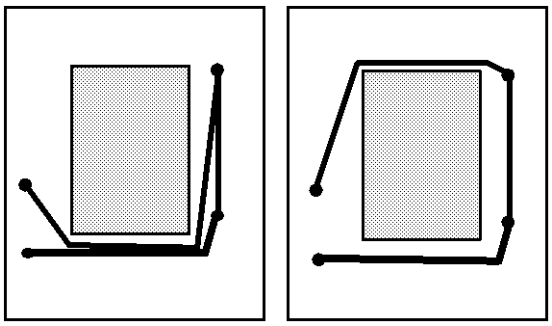
Figure 7.9 - Ranking of routing operators for the 10-port harness. The numbers represent the percent decrease in average lowest cost with and without the operator. (CM = combinor, MB=move-branch, WR=wire bundling, TB=thick-bundle).

Several interesting findings can be extracted from the ranking of the different routing operations.

First, it was found that the Combinor agent, by having a positive value of δ_c in most cases, actually worked better alone than when all the agents were used. The short analysis is that as the designs produced by the Combinor agent, on average, were sufficiently superior to those by the other agents, computer resources were used less efficiently whenever design slots were allocated to the other agents. To test if this was the case for other combinations, two additional tests were run. The first included just the Move-Branch and Combinor agents, again with the Copy-Best and Copy-Some agents, and the second contained all the automated routing agents. It was found that the Move-Branch and Combinor agents, when used together, outperformed any of the agents when working individually on four of the seven routing problems tested (see Appendix D). In the three cases where the Combinor still did better alone, its performance was considerably better than the Move-Branch operator when each was used individually. The collection of all the agents also failed to produce the best results in all of the cases. It is assumed that as the Move Branch and Combinor agents created a large number of new design fragments, adding additional agents (creators that output the same design) did little to increase the diversity of the population.

Second, there was found to be a change in the relative effectiveness of the operators depending on the environment – most notably the relative improvement of the Wire-Bundling agent (for the balanced wirelist case only) and a worsening of the Move-Branch agent as the environment

becomes more cluttered. This occurs as a result of the agents' particular solution strategies. For example, For example, as the Wire-Bundling agent starts with the shortest possible routing regardless of the routing environment's complexity, it performs relatively well as the environment complexity increases. The Move-Branch agent's performance drops considerably with increased environment complexity as its solution strategy attempts to produce routings with small spanning diameters (maximum length of connect path between any two ports) regardless of the routing environment. This is often counterproductive as optimal routings in cluttered environments usually require routings to have larger spanning diameters. As a simple example, the following figures show two routings for a cluttered environment. (The environment is "cluttered" because the ration of the free-space to total space is small, even though there is only one obstacle)



A. Typical MB Routing *B. Typical WB Routing*

Figure 7.4 - The Wire Bundling agent works well in cluttered environments.

The routing on the left (shown with wires individually routed for clarity) is a typical routing created by the Move-Branch agent while the lower cost one on the right is created using the Wire-Bundling agent.

Finally, it was found that the Combinor agent works relatively better when the wire list exhibits strong biases between a subset of ports. This is due, in part, to shape of an optimal routing for a harness with a biased wire list. For example, harnesses with multiple biased wire lists typically have optimal routings with criss-crossed bundles and therefore have multiple main backbones. As the Combinor combines two designs with separate main backbones it is well suited for harnesses with biased wire lists.

7.4.2 Effect of Average Population Fitness on Agent Performance

The degree of randomness of search techniques (e.g., the step size used in gradient methods) typically decreases over time as the solutions become more refined. While having high levels of diversity early in the search process is important, agents that have strong random tendencies become counterproductive as the search process converges on an optimal solution. It was expected that the relative effectiveness of the more aggressive Combinor agent compared with the more conservative Move-Branch agent would change as the search process progresses. This can be examined by comparing the designs created by the agents throughout the search process. To this end, 20 runs of the 10-port, multiple-biased wire list harness in the maze environment were made using all the agents simultaneously and the performance of each agent for each generation was recorded (See Appendix E for more details). By examining the results, it was found that Combinor agent tended to create better designs, on average, than the Move Branch agent during the first ten generations of the search process. However, by the fifteenth generation, this was no longer true. This would follow the assumption that the relative usefulness of the agents changes as the average fitness of the population increases.

7.4.3 Collaboration between the Designer and Routing Agents

As mentioned in chapter 4, both the DRI and ARA modules have shortcomings that can be mitigated by including both in the design process. Three different tests were made to examine the premise that the two modules work synergistically. The tests compared designs created solely by the designer directly with those of the automated routing agents as well as those when both were used together. The procedure was that the designer started the DRI and ARA modules and used the DRI to generate a routing while the ARA searched in the background. Next, the routing created by the designer was sent to the ARA as input. After the ARA converged on a design (at least 5 generations passed without an improved design found), the designers requested, edited, and updated the best design found by the ARA. The following lists the test cases examined and the best design found for each case.

	Best Random	DRI Alone	ARA Alone	Both ARA & DRI
Test Case 1 - multiple biased 10-port harness routed in the maze environment.	0.774 0.796 0.765	0.833 0.786 0.722	0.597 0.620 0.604	0.594 0.606 0.601
Test Case 2 - balance 10-port harness routed in the maze environment.	1.491	1.222	0.933	0.921
Test Case 3 - balanced 10-port harness routed in the empty environment.	1.179	0.953	0.861	0.854

Figure 7.10 - Comparison of cost designs found using various routing methods.

Three designers were used as test subjects to generate designs for Test Case 1. Two of them were given only a few minutes training regarding the operation of the system, the desired routing properties, and how their design decisions affected the overall fitness of the design. While both of the designers had no previous experience in harness design, they fully understood the objective. The third designer was the author of the paper. All three designers failed to produce initial designs that were substantially superior to the best randomly generated design of the first generation. Although the initial designs were submitted to the ARA system, none of them were used by ARA to create "best-of-generation" designs for subsequent generations. However, all three attempts to improve on the best designs found by the ARA were successful. Similar results were found for test cases 2 and 3.

These results clearly showed that the ARA was able to make considerable improvement on the best design that the designer was able to produce on his or her own. As the search time for each of the test cases was only about 15 minutes, the system was extremely effective at assisting the designer in finding near optimal solutions to the cable harness routing problem.

As a note, tests of the routing package on a number of real-world harness routing projects at Lockheed suggested base-line requirements for a commercial system. The first is a mechanism, transparent to the user, to translate existing CAD environment representations (e.g., CSG) into the representation proposed in this work. Substantial effort was expending during the tests to acquire and convert the environment files. In fact, the routing of the harness comprised only a small fraction of the total design time. Another desired element for commercialization of the routing system is an interface between the routing package and automated part selection and layout (2-D construction) agents. Substantial work was developed during the First-Link and Next-Link projects [Park, Cutkosky, Conru, Lee 1992, 1993] in the area of agent-based engineering. It was found that by decomposing the larger problem of harness design into design features (e.g., part lists, routing specifications, and manufacturing cost estimates) and the tasks into to agents (e.g., part selection agent, harness routing agent, manufacturability agent) , these features could be used as information inputs and outputs of the various process agents. Once the design and process are decomposed, the designer, using the routing system, can use the output of downstream agents as feedback to the quality and feasibility of designs produced.

7.5 Summary of Experimental Results

A number of key observations were made during the experimentation:

1. Using domain knowledge (e.g., about how to separate and combine harness fragments) produced much better results than random methods. This implies that the benefits of favouring fragments with localized goodness outweigh the possible problems related to deception (e.g., a locally good fragment may cause wires to be poorly routed through the fragment's complement and result in higher overall costs).

2. None of the agents worked consistently best in all circumstances (i.e., with all kinds of design problems and in all stages of the search process). While this result would suggest that using a team of agents may be able to improve system robustness, it was found that the best router, when used independently, was often able to outperform all the other routers used together given identical resources. This implies that in order for an automated system to benefit from a team of agents, it would need to quickly identify the most effective agent(s) and allocate resources accordingly.

3. Agents that aggressively modified the designs (e.g., Combinor) worked best in the early phase when it was most useful to explore many different areas of the search space; more cautious modifications (e.g., by the Move Branch agent) worked best in the final stages when the average fitness was high.

4. A team of automated agents consistently produced designs of lower cost than humans working with the designer's routing interface alone. However, the designers were typically able to make minor improvements on the best designs produced by the automated routers.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

Cable harness design is a critical, but often overlooked, component to the design of many industrial and aerospace applications. It is often pushed back until the rest of the design is near completion because lengthy routing rework is usually required each time the environment and wiring specifications change. Not addressing the routing concerns early can have costly repercussions since routing problems discovered late in the design cycle are the most disruptive. An automated system that is able to give designers early warning as to potential harness routing problems would therefore be highly valuable. The creation of such a system, however, is non-trivial as the cable harness routing problem belongs to a class of search problems characterized by enormous search spaces, by constraints and objectives that are both difficult to define and compare, and by designs with interrelated, but decomposable, components.

This thesis presented an evolutionary approach to the CHRP that facilitated collaboration between a collection of automated routers and human designers, each using representations for the harness and routing environment that were specialized for their individual needs. The automated routers were able to focus on rapidly exploring the search space using a highly abstracted model of a harness. The designer was able to direct the search process by submitting harness designs created from scratch or by editing designs created by the automated routers. The set of automated routers presented a natural and flexible mechanism to assist the cable harness designer and routinely produced routings that were superior in quality to designs produced by the automated routers and designer working independently. The following subsections discuss some of the key points of the work.

8.1.1 Decomposition of the Routing Problem into Subtasks

The solution process began by decomposing the CHRP hierarchically in two sequential subtasks; the first was a high-level search for a good harness topology, the second was search for the best topological routing of a given topology to be used to define the goodness of the topology. The order of the subtasks is directly opposite to the way a designer typically thinks of the routing problem when addressing it manually – the designer starts by building a topological routing from scratch which, in turn, defines the topology. As the bundle costs are not known until the routing is fully defined, the designer is prone to create suboptimal designs. By using a hierarchical decomposition of the search problems, the bundle costs (derived from the topology) are known during the topological routing phase. With this information, the topological routing search process was able to be abstracted as a search for good transition locations in the free-space. Since the goodness of each of the topologies was measured by using its best topological routing, the topology search process could focus on using their relative qualities to direct the search.

8.1.2 Evolutionary Search Strategy

The success of the search processes (i.e., topology and topological routing) depended heavily on their ability to efficiently explore a large, multimodal search space that does not possess an underlying structure, such as a gradient, that can be used to guide the search. In the case of problems like the CHRP, there is no clear “direction” as to where to search next given a particular design. The solution to this problem required a balance between using overly

restrictive heuristics which may cause the search process to quickly settle on a suboptimal design and more random search processes that may be able to find better designs but at a cost of searching a larger volume of the search space. By formulating the search process as an evolutionary search problem, in which designs from a sampling of the search space are selectively reused to create new designs, the search was able to exploit the implied structure inherent in the design sampling. This was motivated by the observation that a random sampling of designs (i.e., topologies or topological routings) often contained designs in which only parts routed well in the environment. The evolutionary approach, by favouring reusing fragments of high quality to create new designs, was able to focus the search in promising areas of the search space without explicitly defining search heuristics.

The key difference between this formulation and standard genetic algorithms was in the way in which fragments were found and reused. Standard genetic algorithm formulations rely on "implicit" parallelism where it is assumed that by selecting designs with better-than-average fitnesses and randomly reusing the parts of these designs, good design fragments, assumed to more likely to occur in high quality designs, may be retained and reused. However, as standard genetic algorithms operate on encodings of a design, the chances that this occurs for a given operation are small. Obtaining good results often requires using large design populations. The formulation used in this work exploited "explicit" parallelism in that fragments with better-than-average local fitnesses were selected for reuse. This was motivated, in part, by the observation that good fragments were often hidden in designs with average or below-average overall fitnesses. By explicitly using the local fitness of fragments in the selection process, it was more likely that good fragments would be found. This enabled smaller design populations to be used with favourable results. By operating directly on the harness representation (i.e., both the topology and topological routing), instead of encoding as used in GA, the search process was also able to ensure that good fragments were retained. The CHRP was found to be well suited for an evolutionary approach as designs could be naturally decomposed into fragments.

By focusing on reusing fragments, designs are also more resilient to local issues. For example, if a change in the routing environment adversely affects the fitness or feasibility of a fragment of an existing design, the fragment's complement can still be combined with complementary fragments of other designs to quickly determine a new design. Another example is in the handling of constraints such as minimum bundle lengths. Constraint checking itself presented an interesting dilemma – what should be done with designs that violate constraints? While the first

instinct may be to discard these designs, by retaining them and applying local penalty functions to be violating parts (typically bundles), the fragments without violating component could be reused. This proved useful especially in the beginning of t he search process when approximately one third of the topological routings contained bundles that were shorter than the allowed minimum.

8.1.3 Development of Specialized Environment Representations

The evaluation process for the topological routing search needed an environment representation that was able to rapidly find short paths between points in the environment, keep track of the available capacity of regions of the free-space at low computational costs, and provide plenty of placement locations for transitions. The solution was a sparse graph of candidate transition locations that was derived from the medial axis transform representation of the free-space. In addition to meeting the above requirements, the representation had the property that the topological routing remained as far as possible away from the obstacles. The density of the graph representation also varied according to the size of free-space (i.e., areas with tight clearances were modelled with more points than large open areas).

In contrast, the designer's routing interface needed an environment representation that was able to quickly determine the shortest distance between a point and the nearest obstacle. The speed of these calculations was paramount as the simulation of the harness representation typically required hundreds of them to be performed during each time step. The distance values were therefore pre-computed and recorded in a voxel environment representation. In this way, the distance values could be quickly estimated by a lookup.

8.1.4 Integration of Designer Input

While the automated search process was able to routinely produce high quality designs when working alone, it had several shortcomings. First, it was only able to optimize a pre-defined routing objective function. Second, by focusing on looking for good fragments, the solution had problems discerning minor problems that could be readily recognized by a designer and repaired. It is for these reasons that it was imperative to include the designer during the search process.

This was accomplished by allowing the designer to directly interact with the designs created by the automated routers.

To effectively enable a natural man-machine interface, the designer interface needed a harness and environment representation that enabled the same level of interactivity that the designers were used to when working with a real cable harness. This was realized by representing the harness using a set of spheres connected by springs (i.e., models of axial and bending loading) and acted upon by a number of external loading. The representation took into account the topology of the harness, desired bundle properties, and collisions with obstacles and other routings in the routing environment. By allowing the lengths of the bundles to change via the insertion or deletion of spheres, the designer was able to interactively perform high-level modifications to the harness such as moving bundles and changing the harness topology. For the real-world routing problems tested, the time required to define a harness routing and manufacturing specifications was reduced from weeks to hours – even with minimal user training.

Experienced harness designers [Johnson 1994], when first presented with the prototype developed for this work, reported that they quickly felt comfortable using the interface even though their previous experience with CAD systems was limited. They were also able to easily identify the changes needed to address unmodeled design objectives since the representation of the harness in the routing environment gave visual cues needed to make subjective decisions.

The evolutionary formulation of the search also provided a natural collaboration metaphor for the designers when they interacted with an automated routers. By using a list containing alternative designs, the designers were able to select one, modify it as needed, and replace it to be reused by the automated routing agents. In this way, the designers (i.e., the designers involved in the experiments discussed in Section 7.5) felt as if they were working with a team of virtual designers. By being given a list of candidate designs, designers were also less likely to assume that the designs were optimal and therefore more comfortable interacting with the system.

8.2 Directions for future work

8.2.1 More aggressive use of designer input in the search process

In the current implementation, the ability of the designer to affect the automated search process as it progressed was limited to submitting complete designs in the hope that its design fragments would be reused in other designs. A more aggressive approach would be to allow the designer to specific high level constraints during the search process that would immediately affect the local fitness of the designs in the population. For example, the designer, after viewing a few designs generated by the automated routers may recognize that the routers are minimizing an objective function that results in designs with a bundle that passes over an important, but unmodeled, access channel. The designer may interactively increase a penalty to any bundle that crosses that location until the collection of automated routers adjust the design population accordingly. Another example is where a designer may decide, during the course of the search, that the routers should focus more effort on designs that include a particular fragment. By increasing the local fitness of the fragment, the routers should automatically start to favor the fragment in new designs.

8.2.2 Extensions to the flexible harness representation and its use

To improve response time, the interactive harness model described in this work is modeled using the simplest of primitives. It is therefore only able to generate a coarse estimate of the real paths of the harness. While the model is sufficient for conceptual design, a more sophisticated model is warranted if greater routing accuracy is required. Attractive candidates from the world of computer graphics include several approaches that use an interactive FEM to model motion [Terzopoulos 1998, Celniker 1991]. While these models provide definition of shape, they lack in their ability to address harness specific modelling needs, such as lateral stress due to sliding elements, torsion due to bending bundles in different directions at different locations, and environmental constraints. Additionally, the actual model of the innerworkings of cable harness bundles (e.g., wire slippage, insulation properties, and internal stresses) is not well understood [Trumer 1985].

While this work provides the framework for automating the routing of cable harnesses, it intentionally left many of the subjective objectives such as ease of installation, maintainability,

safety, vibration, and clamping to be addressed by the designer. This was not only because they are difficult to quantize, but also because of the resources required to calculate these issues would be prohibitive to rapid search. However, as the designer only examines a handful of the designs presented by the automated agents (i.e., not all the designs tested by the automated routers), it may be useful to create special-purpose automated tools to assist in the design evaluation. For example, this work assumed that the designer is responsible for ensuring that the harness can be easily installed and maintained. While this is a non-trivial task to automate, involving task planning (e.g., in the event that obstacles are to be removed), path planning, and satisfying many spatial constraints, it should be possible to build off the representation described in this work. The solution would involve modifying the harness representation to be more resistant to length changes as well as augmenting the environment representation to include installation and access channels. Next, a high-level planner could be created that suggests which ports need to be connected first. The actual installation of the harness could then be modeled as forces that pull on the ends of the harness towards the ports.

8.2.3 Reusing concepts in other routing domains

The CHRP is just one of a number of routing domains seen by cable harness designers. These domains all have many issues and constraints in common, such as requiring that the paths be continuous and collision-free and that the paths occupy a finite space. Many of the techniques and representations defined in this work can easily be applied to these other domains. For example, the routing of ribbon cable shares attributes of a bundle in space. In this domain, threads and spheres would be connected to one another to form a 2D surface of spheres and the associated bending spring stiffness in either direction could be altered to produce the desired anisometric properties.

The agent-based evolutionary approach can also be used for other routing problems where designs can be decomposed into fragments and the required capacity of the path segments (bundles) changes depending on topology. Logical possibilities include pipe and duct routing when many locations are to be connected by branches off a central trunk. While additional constraints would need to be address in the interactive routing representation (e.g., that bends can occur only at pre-defined angles), and other evolutionary operators created, the designer could still interact with the routing and those generated by automated routers in a similar fashion.

8.2.4 Adaptive division of computational resources

One issue not addressed in this work was how to best allocate computation resources between agents. Future work could consider a more flexible allocation of resources that uses the immediate effectiveness of the agents, measured, in part, on how well they improve the inputs they used or how their performance compares with other agents. In that way, the resources could be used more judiciously and the benefits of having a diverse collection of agents could be improved. As discussed in Chapter 7, the relative effectiveness of the agents changes depending on the routing problem as the search progresses. For example, in the beginning of a search process, agents that aggressively explored the search space, such as the Combinor agent, had a greater tendency of producing good designs. Later, more cautious techniques such as the move-branch agent which returns new routings that are significantly similar to its inputs, contributed more to the search process. If resource allocation is tied to performance in previous generations, then the collection of agents could shift resources to meet the changes in the nature of the problem.

Appendix A:

Evolutionary Strategy to Evaluate Topology Routing

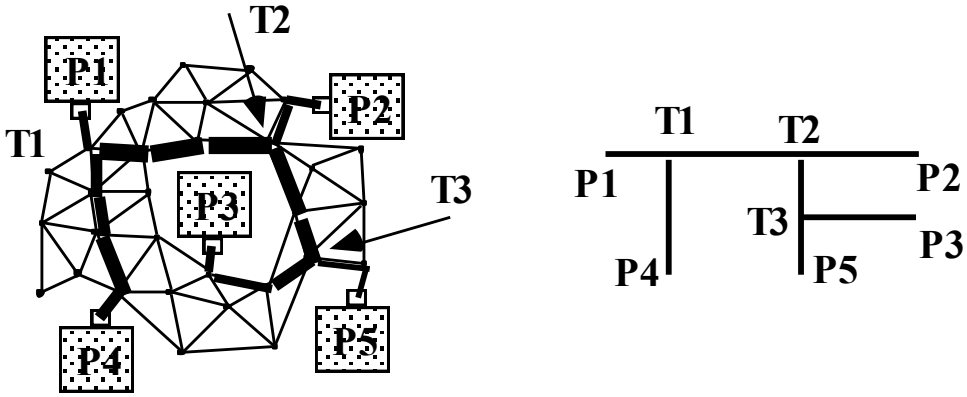
A.1 Transition Locator Introduction

As discussed in Chapter 4, each topology is routed in the AMA graph to determine its fitness value. While a number of heuristics could have been used to find a routing, it was quickly determined that using heuristics alone is not sufficiently robust to adequately solve a wide range of harness routing problems. This is due both to the wide range of possible routing environments and the enormous number of topological routings for a particular topology. The size of the search space grows exponentially as the number of ports in a harness increases. For example, consider the case of a 10-port cable harness topology in an AMA graph with 1000 nodes. The search problem is then to place the eight transitions in this graph such that the total "cost" of the harness is minimized. A brute force method would be to test the 10^{24} permutations for the global optimum. Clearly, this would be highly inefficient and impractical. Using heuristics to solve the

problem is further challenged in that the only structure in the search space is that above average designs often share similar fragments. However, this weak structure can be exploited using evolutionary (genetic) algorithms that reuse and recombine test designs in parallel.

The Transition Locator, developed for this work, uses a genetic formulation of the problem to quickly find locations for transitions that result in low cost routings. While genetic algorithms have been examined before in graph search problems, a formulation that isolates the routing of a topology and leverages heuristical methods has not been developed before. It includes a gradient search component that has the effect of reducing the search space well into a single routing. The genetic approach makes use of the fact that fragments of above average routings can often be combined to form still better designs.

The Transition Locator assumes that the topology has already been defined. For a simple example, consider the following 3-transitioned cable harness that has been routed in a simple environment graph. The topology just defines the connectivity information of the nodes of the cable harness as illustrated in Figure A.1B.



A. Routing of Harness *B. Configuration of Harness*

Figure A.1 - Routing and topology for a 3-transitioned cable harness

By viewing the harness as its topology, only the transitions need to be placed in the AMA graph to produce a routing. The first step of the process is to determine the cost per unit length for each of the bundles. The Transition Locator does this by routing each wire of the wiring list in the topology, using Dijkstra's shortest path algorithm [Dijkstra 1959] to determine the number of

wires going through each bundle. Once these costs are defined, the Transition Locator is ready to locate the transitions within the environment graph using a genetic procedure. The following sections give an overview of genetic algorithms, describe the genetic formulation and parameter settings, and compare the results to random search.

A.2 Overview of Genetic Algorithms

Genetic algorithms (GA) are a class of search methods that are patterned after the evolutionary processes in natural selection. Individuals in a population are analogous to points in the search space. Each individual (called a phenotype) is bi-directionally encoded in a machine-readable format, usually a string of numbers, such that each number corresponds to a particular characteristic of the design. This string of numbers is called the chromosome representation or genotype for the design. A general fitness value is also attached to the chromosome for comparison purposes.

Optimization using GA occurs though an evolutionary process in which populations of designs (chromosomes) are reused to form a new population. Each generation begins by selecting designs which are the most fit from the previous generation to serve as "parents' for the next generation. The probability that a design is selected is:

$$p_i = \frac{F_i}{\sum_{j=0}^{\#designs} F_j}$$

where F_i is the fitness of a design and

$$\sum_{j=0}^{\#designs} F_j$$

is the total fitness of the population. The selected designs are then combined or mutated via several genetic operators (discussed later) to form a population of new, and hopefully better, designs. As the population evolves, the overall quality of designs should increase as better designs are more likely to sire (i.e., pass on design information) for the next generation. A number of the best designs are often copied unaltered to ensure that the best designs are always retained.

A number of different genetic operators are employed to reuse parts of previous designs and typically fall into two classes: mutation operators and crossover operators. Mutation operators take a chromosome encoding of a design and alter it using a number of techniques. The new chromosome can then be decoded to produce the new design. Figure A.2 below shows a typical random mutation operation on a generic binary chromosome in which a bit is toggled (e.g., the fifth bit is toggled from 1 to 0). A mutation operation increases the population diversity by randomly moving to designs in the search space. However, as the new designs often share large elements of the mutated designs, the effect of mutation is limited.

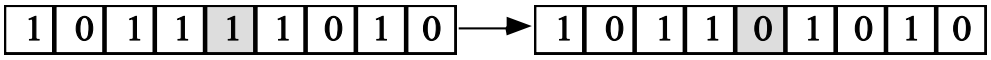
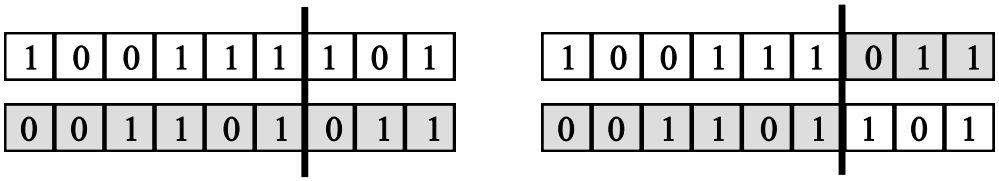


Figure .A.2 - Binary Genetic mutation changes only a single bit to create a "new" design.

The basic crossover routine takes two chromosomes from the current generation and exchanges some of the genetic information between them. It attempts to indirectly determine which sub-parts of the chromosomes are responsible for the designs' high quality. Figure A.3A below shows two generic binary chromosomes and a crossover site drawn as a heavy vertical line. During the crossover process, bits to the right of this crossover point are swapped between the chromosomes, resulting in the pair shown in Figure A.3B.



A. Initial Chromosomes *B. Chromosomes after Crossover*

Figure A.3 - Binary genetic crossover swaps parts of two genetic strings

The basic algorithm is outlined below.

1. Initialize a generation counter, t = 0
2. Define an initial population of λ designs, $\mathbf{P^0}=(D^0_1, D^0_2, .. D^0_\lambda)$ of test solutions
3. While (t < Maximum Generations) do
4. Evaluate the fitness of the members of $\mathbf{P^t}$
5. Select designs D_i^t and D_j^t from $\mathbf{P^t}$
6. Crossover D_i^t and D_j^t if random(1) < crossover probability
7. Mutate D_i^t and D_j^t if random(1) < probability
8. t = t + 1
9. Evaluate fitness of D_i^t and D_j^t and insert them into $\mathbf{P^t}$

A typical genetic algorithm performs crossover and mutation in series and has only a single crossover and mutator operator as shown below.

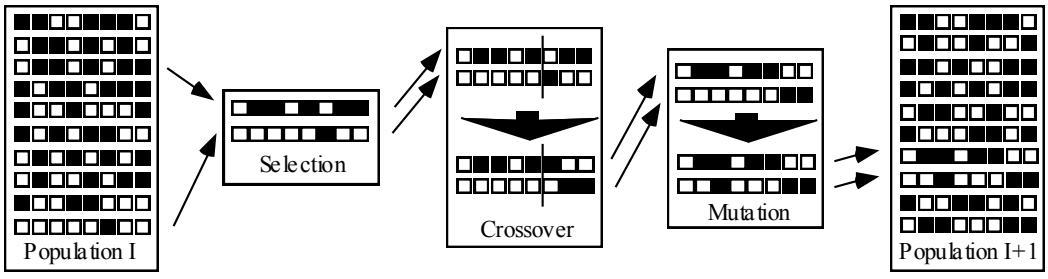


Figure A.4 - Standard genetic algorithm performs crossover and mutation in series.

One of the key discriminators used to see if a genetic approach may be applicable for a given problem is the level of epistasis in a problem. Epistasis is associated with the degree to which a change in one part of a design affects the goodness of the rest of the design. Large problems with a high degree of epistasis have highly coupled parameters and are usually solved using a Monte Carlo technique. Problems with low epistasis have uncoupled parameters which can be optimized independently. Problems with low epistasis have uncoupled parameters which can be

optimized independently. As shown in Section 4.3.2, the qualities of the transition locations in the topological routing problem are only interrelated with their adjacent transitions and therefore the problem has only moderate levels of epistasis.

A.3 Transition Locator Genetic Representation and Operators

The chromosome representation for the transition location problem is simply a list of AMA graph nodes for each of the transitions as shown below. The alphabet for the chromosome is the number of nodes in the AMA graph. The fitness equation for the routing is defined in Chapter 2.

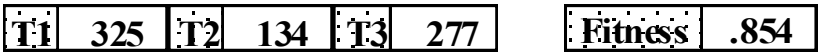


Figure A.5. - Routing Chromosome and Fitness

Two mutation operators are used to move the transitions in the environment graph. The first simply chooses one or more transitions and moves them randomly to other AMA Graph nodes The second moves the transition to adjacent graph nodes which results in a lower cost. The chance that either operation will occur during a mutation is equal. While having limited effect when used alone, mutation adds a considerable amount of new information when used along with a crossover operator. The crossover operator transforms two individuals by randomly swapping any number of their transition locations.

A.4 Genetic Parameter Settings

Since the Transition Locator is called many times by the automated routing agents described in Chapter 5, speed is of high concern. The size of the population, total number of generations, and mutation and crossover probabilities all have a profound effect on the effectiveness of the Transition Locator. In general, the fitness improves as the population size and number of generations increase. However, it is subject to diminishing returns.

On the other hand, the effects of the crossover and mutation probabilities on fitness is a bit more subtle. It appears that a crossover probability of 85% with around 50% allele (transition) swapping produces the best results regardless of the other parameters. This means that 85% of

the designs in a generation will have been created from crossover and 15% will have survived intact from the last generation. Of the 85% created by crossover, about half of the transitions are swapped. As the size of the problem (i.e., number of transitions) increases, the percentage of swapped alleles decreases as there is a greater incentive to retain larger fragments of the parent.

The benefits from mutation tend to decrease as the problem size increases. For example, for the 8-transition case, an 4% mutation rate (meaning 4% of all the individuals will have at least one transition moved) gives the best results while a 1% rate works best for a 20 port harness. The table below lists suggested parameters to produce good routings for a given topology depending on its number of transitions. When the Transition Locator was used in the topology search, the suggested parameters for both the population size and number of generations were halved to increase speed. Using fewer resources for the Transition Locator is possible because only a comparison of the "expected" fitness is needed during the topology search.

Number of Transitions	Pop. Size	# Gen	% Mutation	% Crossover	% Allele to Swap
8	30	10	4	85	50
10	40	12	3	85	50
12	50	15	3	85	45
15	70	18	2	85	40
20	90	22	1	85	40

Table A.1: Suggested GA parameters for Transition Locator

A.5 Seeding the Population Using the Rubberband Heuristic

The initial designs in GA problems are usually generated randomly. However, considerable search time can be saved if the initial population is seeded with good designs. A simple heuristic router, called the rubberband router agent [Conru 1993], is used to produce a good routing

quickly given a specific topology as input. Routing is performed in three stages. First, the transitions are placed randomly in the environment using their real-value coordinates instead of AMA graph node locations. They are then moved locally until the following cost function is minimized:

$$Cost = \sum_{i=1}^{\#bundles} K_i L_i$$

where K_i is the cost per unit length and L_i is the Euclidean distance between the end nodes of the i^{th} bundle, respectively. This process can be seen in the figure below.

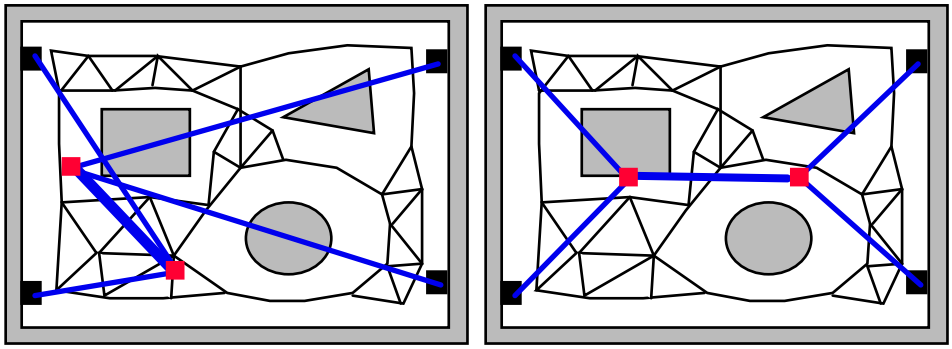


Figure A.6 - The rubber band process starts by choosing transition Locations randomly then moving them iteratively to minimize cost..

Once the transitions settle, each is moved to the nearest AMA graph node. Next, the nodes are moved along the AMA graph to nodes that minimize the graph-based CHRP objective function as shown in the figure below.

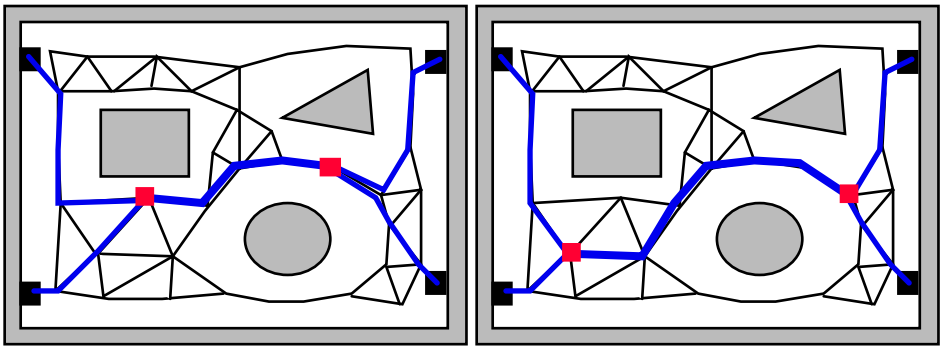


Figure A.7 - The transitions are then moved to the closest AMA nodes and them moved locally along the graph to a local minimum.

Seeding the initial population has both desired and undesired affects on the search process. While it does create an instant leading contender design, it potentially biases the design population towards designs that share a similar routing. This may reduce the chances that better routings, unrelated to the seeded design, is found. As a compromise, the GA process is allowed to run for a quarter of the total generations before a design found by the rubberband heuristic is inserted in the design population. This gives the evolutionary process time to generate a few good candidates that can compete with seeded design.

A.6 Comparison with Random Search

Random search, in which the transition locations are randomly placed in the AMA graph, was used as a simple comparison of the effectiveness of the Transition Locator. Table 2 lists the results of averaging ten runs of the Transition Locator on ten randomly generated harness topologies of three to seven transitions. For comparison, two tests using the random search method were run. Column RS1 gives the average of the best cost found after testing the same number of routings as the Transition Locator. Column RS20 gives the average of the best cost found using the same amount of CPU time as the Transition Locator (approximately 20 times the number of tests).

#Transitions	GA	RS1	RS20
3	0.514	0.576 (112%)	0.538 (105%)
4	0.783	0.969 (124%)	0.897 (115%)
5	1.062	1.481 (139%)	1.344 (127%)
6	1.600	2.333 (146%)	2.169 (136%)
7	2.315	3.233 (140%)	3.087 (133%)

Table A.2 - Comparison with Random Search.

While random search appears to give similar results as the Transition Locator for a 3-transitioned harness, it quickly falls behind as the number of transitions increases. This is expected since the size of the search space increases factorially with the number of transitions.

Appendix B

Overview of Shortest Path Planning Techniques

Shortest path, or point-to-point, planning is one of the more common routing problems. It is typically defined as follows:

> Given two points in space, find a connected path between them that minimizes a cost function subject to an assortment of kinematic and spatial constraints.

Cost functions typically include penalties for length and for passing through hazardous regions as well as bonuses for using prescribed routing channels. Kinematic constraints, such as limitations on the minimum bend radius, are also often included. While the CHRP, as a whole, is more complex than the shortest-path problem, the routing techniques used can be applied to the routing of the individual bundles of a harness.

The techniques used in shortest path planning fall into two categories: enumeration and abstraction. Enumeration attempts to create a complete path by combining feasible path segments. In the 2D case, the possibilities are usually "go straight," "curve to the right," and "curve to the left." In 3D, the options are "go straight" and "curve (in any number of discrete directions) from the tangent." While the shortest path of bounded curvature consists of only straight line segments and curves of maximum curvature, the resulting path would fail to ensure the C^4 continuity condition required for a physical cable path.

Examples of enumeration methods include one developed by Latombe [1991] and Barraquand [1992]. Barraquand showed that it is possible to find a path between two end positions and orientations of a car-like robot, subject to nonholonomic constraints, so long as there are at least two different positions in the car's steering wheel and it can move in reverse. Using this result, a fast path planner [Latombe 1993] was built that could recursively define a path that was compromised entirely of curved and straight segments.

Enumeration methods are most often used when the routing domain is uncertain or global information is unknown. However, enumeration methods routinely require evaluating many potential routings. For this reason, abstraction-based routing methods (discussed below) are more widely used than enumeration techniques.

Abstraction-based routing methods speed the search process by pre-calculating an abstraction of the routing environment. There are hundreds of different path planning algorithms that rely on an abstracted environment, and the applicability of each depends on factors such as environment certainty, desired routing accuracy, kinematic constants, and routing speed.

A common abstraction is a graph of the free-space such as the AMA graph discussed in Chapter 3. The following figure shows a typical point-to-point routing.

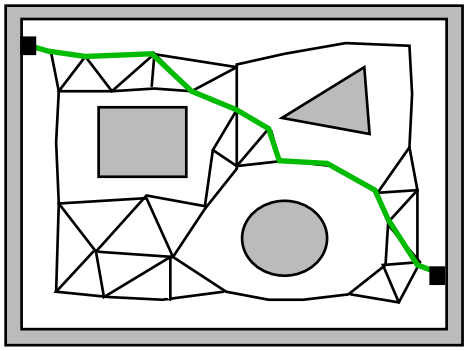


Figure B.1 - Single point-to-point routing

Graph-based, minimal-cost point-to-point routings can be found in no worse than $O(nG^2)$ time where nG is the number of graph nodes using any brute-force technique. One of the best techniques, the A* search algorithm [Nilsson 1980], has a computational complexity of O(nG) in the best case and $O(nG^2)$ in the worst.

Other techniques were examined for use in the point-to-point path planning aspect of cable harness routing. The first was the 2D path planner developed by Wilfong [1988]**.** His planner starts by creating a network of straight paths (analogous to city roads) that subdivide the space between obstacles and then plan arcs which smoothly connect two intersecting paths. Given m "lanes", the algorithm can find a path with the minimum number of bends in $O(m^2)$ after a pre-processing step of $O(m^2(n^2+\log m)$, where n is associated with the environment complexity. This sort of routing technique, while producing smooth paths that can be of specified maximum curvature, does not readily extend to the 3D environments needed for the cable harness routing problem.

Another type of shortest path planner is that of Jo & Didier [1990], who used a fluid flow metaphor to define paths in 2D. Their motivation was to find a method of accommodating changes in the routing environment. In their work, the environment is first represented by an approximate cell decomposition. Next a fluid source is placed at the starting point and a sink is placed at the destination. A FEA model of the system is then used to determine the fluid flow at each cell of the environment. As water dynamics are not modeled, the fastest decent path becomes the shortest path. The main problem that limits this planner's usefulness for the cable harness (bundle) routing problem is that it requires all the cells in the environment to be processed just to find a single path.

Jain, et. al [1992] created a routing system to route pipes (limited to 2 ports). They start by discretizing the environment using large parallelepipeds, generating a graph connecting them, and then assigning travelling costs to each link. The shortest path is found on this graph by using the A* algorithm. This method is able to rapidly find very coarse routings that must be refined using post-processing. This coarse abstraction works well for the case of routings between two locations. However, the coarse discretization becomes problematic for the cable harness routing problem because of its limitations in maintaining cell capacity information and in defining the locations of harness transitions.

Papadimitrou [1985] created a polynomial approximation method for the 3D minimal path problem by defining points on the edges of the obstacles in the environment and then connecting them to form an approximate 3D visibility graph. The shortest path found on this graph is no longer than 1+e times the length of the global shortest path, where e is related to the spacing of the points on the obstacles. The biggest drawback of this method is that its computational complexity quickly overcomes all but the smallest problems.

A number of techniques for routing paths in 2D using genetic algorithms have also been formulated in the past few years. While most rely on a grid representation of the routing environment [Davidor 1991; Shibata 1993; Shing 1993], others such as Lin's [1994] are able to route any path in the free space. These latter ones are of interest in that the environment does not need to be pre-processed in order to determine a routing. A straight-line path is first assumed then various GA operators are used to make the path feasible. The primary difficulty with these genetic path planning techniques is the inherit inefficiencies related to their probabilistic search method.

Appendix C

Randomly Generated Topological Routings

In most optimization problems, it is often computationally infeasible to determine the globally optimal solution. This is true for the cable harness routing problem. However, the fitness distribution of randomly generated designs offer an idea as to the percent of designs above a given fitness value. To that end, two thousand randomly chosen topologies for each of the test cases in Chapter 7 were run with transition locator parameters that were twice that as used by the test runs in chapter 7 (equivalent to four times the resources). The following graphs show the fitness distribution for each of the test cases. Since the best designs found in all of the test cases are far better than any randomly determined, the number of topologies with high fitness is assumed extremely small. As the search technique developed in this work is able to routinely return designs that fall into this small section of the search space after testing only a few hundred designs, it appears to use computational resources efficiently.

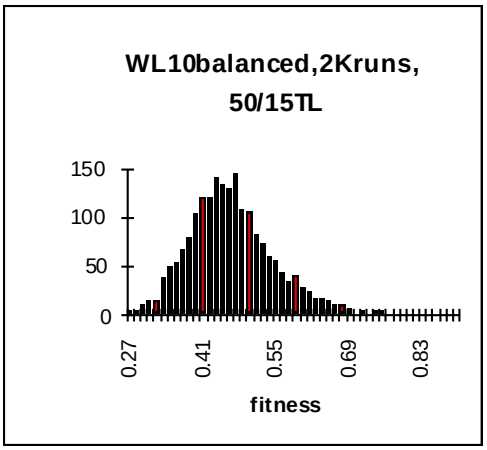


Figure C.1 - 2000 randomly generated routings of a balanced 10 port harness in the maze environment using the transition location with 4x more resources than used in Chapter 7.

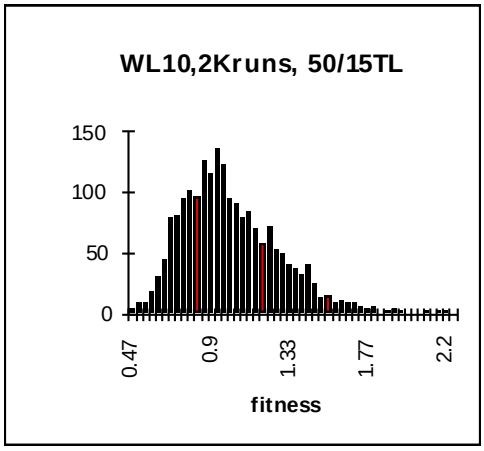


Figure C.2 - 2000 randomly generated routings of a multiple biased 10 port harness in the maze environment using the transition location with 4x more resources than used in Chapter 7.

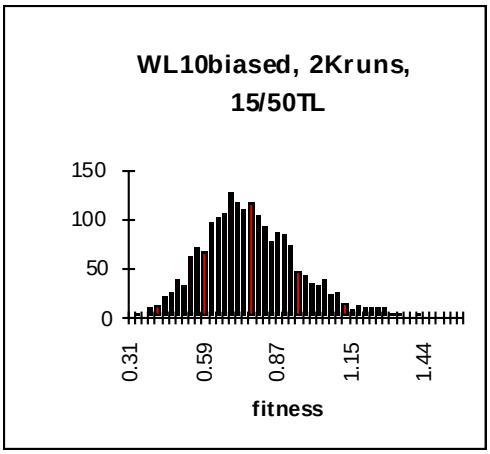
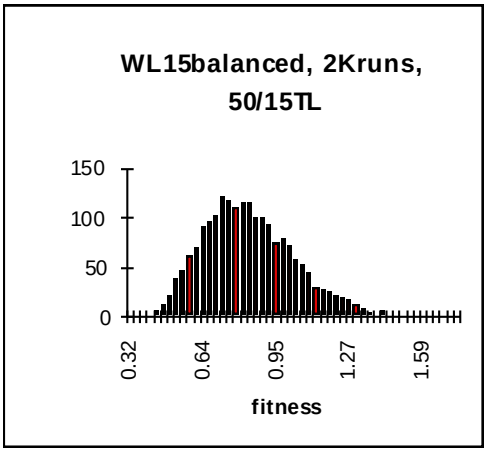


Figure C.3 - 2000 randomly generated routings of a single biased 10 port harness in the maze environment using the transition location with 4x more resources than used in Chapter 7.



FigureC.4 - 2000 randomly generated routings of a balanced 15 port harness in the maze environment using the transition location with 4x more resources than used in Chapter 7.

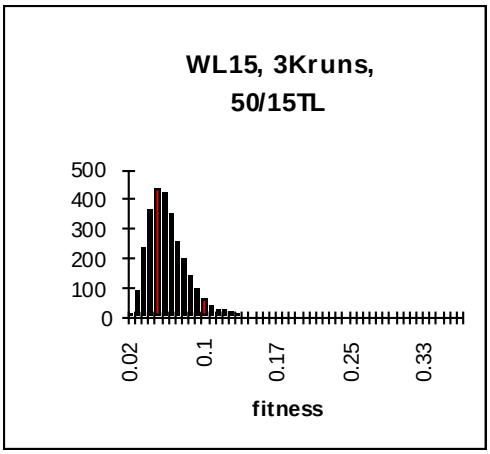


Figure C.5 - 2000 randomly generated routings of a multiple-biased 15 port harness in the maze environment using the transition location with4 more resources than used in Chapter 7.

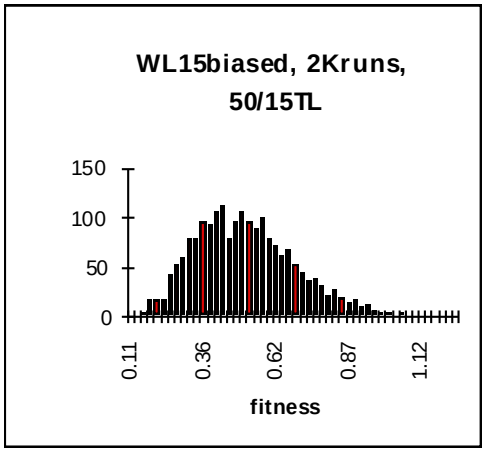


Figure C.6- 2000 randomly generated routings of a single-biased 15 port harness in the maze environment using the transition location with 4x more resources than used in Chapter 7.

Appendix D

Agent-level Collaboration Run Data

Copy Best	Copy Some	Move Bran.	Combine	TBO	WBO	Max Fit	Max Std	Div	Div Std.
X	X	X	.	.	.	2.28846	0.16930	0.10898	0.01091
X	X	.	X	.	.	2.69340	0.13859	0.10547	0.00602
X	X	.	.	X	.	1.88245	0.18425	0.03555	0.01019
X	X	.	.	.	X	2.27929	0.03867	0.02383	0.00576
X	X	X	X	.	.	2.71255	0.11941	0.12227	0.01804
X	X	.	X	X	X	2.60098	0.14966	0.10781	0.00833
X	X	X	.	X	X	2.34322	0.15679	0.11133	0.01725
X	X	X	X	.	X	2.80837	0.04717	0.12773	0.01278
X	X	X	X	X	.	2.57886	0.12532	0.14062	0.02580
X	X	X	X	X	X	2.59460	0.09169	0.15078	0.02229

Table D.1 - Multiple-biased 10-port harness in maze environment

(Cgen=15, Csize=30, Tlgen=8, Tlsize=15)

Copy Best	Copy Some	Move Bran.	Combine	TBO	WBO	Max Fit	Max Std	Div	Div Std.
X	X	X	.	.	.	1.49959	0.20087	0.10898	0.03900
X	X	.	X	.	.	2.23773	0.04353	0.08789	0.00817
X	X	.	.	X	.	1.54315	0.00030	0.01367	0.00000
X	X	.	.	.	X	1.49494	0.00804	0.01914	0.00749
X	X	X	X	.	.	2.21048	0.07109	0.10547	0.01481
X	X	.	X	X	X	1.63504	0.02571	0.12695	0.00517
X	X	X	.	X	X	2.19525	0.02423	0.09961	0.01105
X	X	X	X	.	X	1.70913	0.27395	0.11211	0.02014
X	X	X	X	X	.	2.19516	0.06609	0.12305	0.00000
X	X	X	X	X	X	1.95604	0.12547	0.13867	0.00970

Table D.2 - Single-biased 10-port harness in maze environment

(Cgen=15, Csize=30, Tlgen=8, Tlsize=15)

Copy Best	Copy Some	Move Bran.	Combine	TBO	WBO	Max Fit	Max Std	Div	Div Std.
X	X	X	.	.	.	0.90754	0.10487	0.09531	0.01458
X	X	.	X	.	.	0.81464	0.07753	0.17031	0.02633
X	X	.	.	X	.	1.11667	0.02158	0.01602	0.00524
X	X	.	.	.	X	1.10684	0.04204	0.02031	0.00627
X	X	X	X	.	.	0.96471	0.16417	0.15703	0.02970
X	X	.	X	X	X	1.08325	0.01036	0.08906	0.01661
X	X	X	.	X	X	1.17406	0.06421	0.09453	0.01285
X	X	X	X	.	X	1.05063	0.04454	0.12656	0.02275
X	X	X	X	X	.	1.00273	0.08163	0.12988	0.01795
X	X	X	X	X	X	1.05222	0.03723	0.12656	0.01159

Table D.3 - Balanced 10-port harness in maze environment

(Cgen=15, Csize=30, Tlgen=8, Tlsize=15)

Copy Best	Copy Some	Move Bran.	Combine	TBO	WBO	Max Fit	Max Std	Div	Div Std.
X	X	X	.	.	.	3.22812	0.10737	0.10117	0.01484
X	X	.	X	.	.	3.66994	0.17275	0.07344	0.02221
X	X	.	.	X	.	2.57255	0.22002	0.02656	0.00982
X	X	.	.	.	X	2.66627	0.02834	0.01563	0.00437
X	X	X	X	.	.	3.83008	0.12636	0.11406	0.00812
X	X	.	X	X	X	3.63896	0.19232	0.09844	0.01144
X	X	X	.	X	X	3.31566	0.08703	0.10430	0.02106
X	X	X	X	.	X	3.81261	0.05368	0.12852	0.01798
X	X	X	X	X	.	3.80676	0.15049	0.12227	0.01285
X	X	X	X	X	X	3.81138	0.06185	0.14531	0.01110

Table D.4 - Multiple-biased 10-port harness in empty environment

(Cgen=15, Csize=30, Tlgen=8, Tlsize=15)

Copy Best	Copy Some	Move Bran.	Com bine	TBO	WBO	Max Fit	Max Std	Div	Div Std.
X	X	X	.	.	.	2.09075	0.06537	0.11133	0.01968
X	X	.	X	.	.	2.48499	0.04960	0.08984	0.02248
X	X	.	.	X	.	1.63552	0.15419	0.03008	0.00611
X	X	.	.	.	X	1.96898	0.00000	0.01367	0.00000
X	X	X	X	.	.	2.45773	0.09630	0.12070	0.02604
X	X	.	X	X	X	2.48072	0.04023	0.09492	0.00563
X	X	X	.	X	X	2.14653	0.13950	0.10469	0.01734
X	X	X	X	.	X	2.42842	0.06845	0.10703	0.01335
X	X	X	X	X	.	2.43923	0.01897	0.11406	0.01766
X	X	X	X	X	X	2.41622	0.05732	0.12695	0.00744

Table D.5 - Single-biased 10-port harness in empty environment

(Cgen=15, Csize=30, Tlgen=8, Tlsize=15)

Copy Best	Copy Some	Move Bran.	Com bine	TBO	WBO	Max Fit	Max Std	Div	Div Std.
X	X	X	.	.	.	1.36420	0.10923	0.07930	0.01465
X	X	.	X	.	.	1.36619	0.09669	0.09883	0.00963
X	X	.	.	X	.	1.18867	0.00098	0.01367	0.00000
X	X	.	.	.	X	1.26537	0.00000	0.01992	0.00639
X	X	X	X	.	.	1.37610	0.03896	0.11406	0.02747
X	X	.	X	X	X	1.39660	0.09356	0.07773	0.01276
X	X	X	.	X	X	1.44365	0.01684	0.09648	0.00812
X	X	X	X	.	X	1.38394	0.11035	0.11094	0.00910
X	X	X	X	X	.	1.48135	0.08976	0.11328	0.01381
X	X	X	X	X	X	1.41487	0.08356	0.11953	0.01931

Table D.6 - Balanced 10-port harness in empty environment

(Cgen=15, Csize=30, Tlgen=8, Tlsize=15)

Copy Best	Copy Some	Move Bran.	Com bine	TBO	WBO	Max Fit	Max Std	Div	Div Std.
X	X	X	.	.	.	0.60226	0.14532	0.00590	0.00115
X	X	.	X	.	.	0.74583	0.12486	0.00789	0.00063
X	X	.	.	X	.	0.44718	0.03310	0.00088	0.00033
X	X	.	.	.	X	0.76648	0.00980	0.00073	0.00000
X	X	X	X	.	.	0.71305	0.06223	0.00875	0.00148
X	X	.	X	X	X	0.72871	0.03211	0.00485	0.00065
X	X	X	.	X	X	0.75516	0.01671	0.00436	0.00056
X	X	X	X	.	X	0.80959	0.04897	0.00822	0.00105
X	X	X	X	X	.				
X	X	X	X	X	X	0.74143	0.05810	0.00757	0.00082

Table D.7 - Multiple-biased 15-port harness in maze environment

(Cgen=15, Csize=30, Tlgen=8, Tlsize=15)

Appendix E

Examination of Agent Performance

A careful examination of the genealogy of a design gives a greater insight as to how the effectiveness of an agent changes over time. The following figure shows two generations of a genealogical tree for a design as well as the fitness change of the design as it evolves.

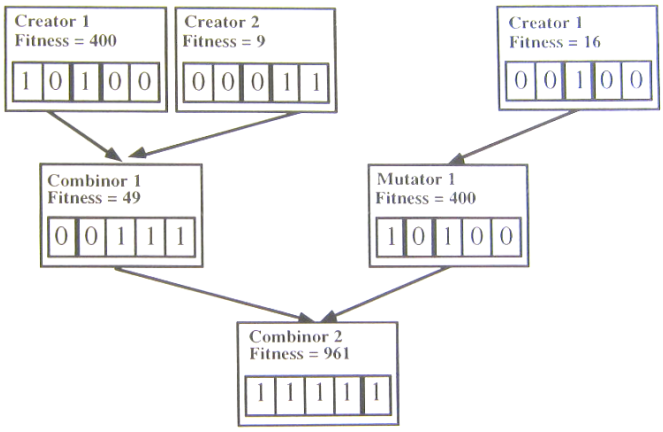


Figure E.1 – The relationship between agents can be examined via design genealogies.

The reasons why designs improve or do not can be estimated at each step of the process. For example, the following list possible explanation for the effects of the lower three agents shown in Figure E.1.

Agent	Possible Explanations
Combinor 1	• Combinor 1 and designs from Creator 1 are incompatible (the design from Creator 1 become worse)
	• Combinor 1 and designs from Creator 2 are compatible (the design from Creator 2 improved)
	• Designs from Creator 1 and Creator 2 are neutrally compatible when acted on by agent Combinor 1(one design improved while the other did not).
Mutator 1	• Designs from Mutator 1 and Creator 1 are compatible
Combinor 2	• Combinor 2 and designs from Combinor 1 are compatible.
	• Combinor2 and designs from Mutator 1 are compatible.
	• Designs from Combinor 1 and Mutator 1 are compatible when acted on by Combinor 2.

Table E.1 – Comparison of designs found using various routing methods.

This "compatibility" between agents can be given a numerical measure thereby making it possible to estimate compatibility trends. As the tree in Figure E.1 is very small, it is obviously impossible to know which of the above reasons are more valid. However if many such tests are performed throughout the entire search process and aggregated, then general trends may be discussed. The average fitness of designs created by one agent, **a**, using a design that was created by another (possibly the same) agent, **b**, as input was used to quantize the effectiveness of an agent and its compatibility with others. The following equation defines this measure more precisely.

$$F_{ab}^{t} = \frac{1}{n}\sum^{n} F(D_{a}^{t}(D_{b}^{t-1}))$$

F^t_{ab} is the average fitness of designs created by agent *a* when it uses designs created by agent *b* as input during generation *t*. The variable, *n*, is the number of instances that this situation occurred.

The following table displays the agent matrix for a single generation (the entire run is located on the following two pages). The agents listed in the first column are the ones that create the output designs whereas the first row lists the agents that created the input design. The first number of each cell is the average fitness of the output design, F^t_{ab}, and the second number is the number of occurrences of the input/output situation, *n*..

Agent	Copy Best	Copy Some	Move Branch	Combinor	Total
Copy Best	2.570| 16	****| 0	****| 0	****| 0	2.557| 20
Copy Some	2.666| 5	1.973| 65	1.889| 53	1.807| 52	1.903| 180
Move Branch	2.348| 9	1.717| 58	1.622| 47	1.559| 61	1.667| 180
Combinor	1.863| 5	1.677| 60	1.564| 51	1.691| 57	1.646| 179

Table E.2 – Sample listing of design inputs for each agent for a single generation

The data for this generation, shows, for example, that the Copy Best agent used 8 designs that were generated previously by the copy best agent and 2 designs that were from the copy some agent. This ratio makes sense as the design produced by the Copy Best agent are typically the best of the next generation. The relative effectiveness of agents using input from a particular agent can be estimated by comparing the values of a single column. For example, when designs from the Copy Some agent, assumed to be a relatively high-quality sample of the design pool, were used as inputs, the Move Branch agent outperformed the Combinor agent 1.717 to 1.677. If this was found to be true for several generations, then it would add credence to the assumption that that agent is more effective during that stage of the development. The far right column of the data on the following two pages shows that average fitness of all the designs created by each of the agents for a given generation. The following lists the entire run.

References

1. Akman, Varol. “Geometry and Graphics Applied to Robotics.” Theoretical Foundations of Computer Graphics and CAD, Springer-Verlag Berlin (1988):

2. Aneja, Y.P., “An integer linear programming approach to the Steiner problem in graphs”, Networks v10, (1980) p 167-178.

3. Back, Thomas; Hoffmeister, F. “Extended Selection Mechanisms in Genetic graphs”,. Proceedings of the 4th International Conference on GA (July 1991), p92-99.

4. Baker, K. “Adaptive Selection Methods for Genetic Algorithms.” Proc. of the 1st Intl. Conference on GA and their Applications (1985), p101-111.

5. Baraff, D.; Witkin, A. “Dynamic Simulation of Non-penetrating Flexible Bodies.” Computer Graphics 26 (July 1992): 303-308.

6. Barraquand, J., Langlois, B., and Latombe,.J.C. “Numerical Potential Field Techniques for Robot Path Planning.” Tech. Report No STAN-CD-89-1285, Department of Computer Science, Stanford University (1989).

7. Barraquand, Jerome; Latombe, Jean-Claude. “NonholonomicMultibody Mobile Robots: Controllability and Motion Planning in the Presence of Obstacles.” Algorithmica, Springer-Verlag (1992):

8. Basu, A.; Aloimonos, J. “Approximate Constrained Motion Planning.” Proc. of the IEEE Intl. Conf. on Robotics and Automation (1990)/

9. Beasley,J.E., “An algorithm for the Steiner problem in graphs”, Networks 14 (1984) p147-159

10. Bedau, M.; Ronneburg, F.; Zwick, M. “Dynamics of Diversity in an Evolving Population.” Parallel Problem Solving from Nature 2 (1992), p95-104.

11. Belmonte, M.; Mandow, L.; Perez de la Cruz, J.; Conejo, R.; Morales, R. “A collaborative architecture for design evaluation and refinement.” AI in collaborative Design 1 (1993), p3-13.

12. Bennett, K.; Ferris, M.; Ioannidis, Y. “A Genetic Algorithm for Database Query Optimization.” Proceedings of the 4th International Conference on GA (July 1991), p400-404.

13. Birmingham, W.; Darr, T.; Durfee, E.; Ward, A.; Wellman, M. “Supporting mechantronic design via a distributed network of intelligent agents.” AI in Collaborative Design 11 (1993), pg 15-34.

14. Blum, H. “Transformation for extracting new descriptors of shape.” Models for Perceptions of Speech and Visual Form (1967), p362-380.

15. Branki, N.E. “An AI based framework for collaborative design.” AI in Collaborative Design 1 (1993), p35-46.

16. Brooks, R.A. and Lozano-Perez, T. “A Subdivision Algorithm in Configuration Space for Findpath with Rotation.” Proceedings of the 8th International Conference on Artificial Intelligence Karlsruhe, FRG (1983), p799-806.

17. Buttitta, B.; Orlando P.; Sorbello F.; Vassallo G. “Montreale: A New Genetic Algorithm for the Solution of the Channel Routing Problem.” Proceedings of Advanced Computer Technology, Reliable Systems, and Applications 5 (1991), p462-466.

18. Cayley, A. “A theorem on trees.” Quarterly Journal of Mathematics (1889), p26-28.

19. Celniker, George; Gossard, Dave. “Deformable Curve and Surface Finite-Elements for Free-Form Shape Design.” Computer Graphics 25 n 4 (July 1991), p257-266.

20. Chapman, C.; Saitou, K.; Jakiela, M. “Genetic Algorithms as an Approach to Configuration and Topology Design.” ASME Design Automation Conference, September 1993, Albuquerque, NM (1993).

21. Chen, Pang C.; Hwang, Yong K. “ Practical Path Planning among Movable Obstacles.” Proceedings of the 1991 IEEE Intl Conf on Robotics and Automation, Sacramento, Ca., (April 1991).

22. Cohoon, James P.; Hegde, Shailesh U.; Martin, Worthy N. “Distributed Genetic Algorithms for the Floorplan Design Problem.” IEEE Transactions on Computer-Aided Design of Integrated Circuis and Systems 10 (April 1991): 483-492.

23. Colorni, A.; Dorigo, M.; Maniezzo. “An investigation of some properties of an “Ant algorithm”.” Parallel Problem Solving from Nature 2 (1992), p509-520.

24. Connolly, C.I.; Burns, J.B.; Weiss, R. “Path Planning Using Laplace's Equation.”IEEE ? (1990): 2102-2106.

25. Conru, A. “Computational support for interactive cable harness routing and design.” ASME Design Automation Conference, September 1993, Albuquerque, NM (1993)

26. Conru, A. “A Genetic Approach to the Cable Harness Routing Problem.” IEEE World Congress on Computational Intelligence (1994) p200-206.

27. Crocker, M., Personal communications with Mike Crocker, project manager of cable routing toolkit at Lockheed, (1996).

28. Davidor, Yuval. Genetic Algorithms and Robotics: a heuristic strategy for optimization. 1 ed., Vol. 1. New Jersey: World Scientific, 1990.

29. Davidor, Y. Genetic Algorithms and Robotics. World Scientific, 1991.

30. Davidor, Y.; Ben-Kiki, O. “The Interplay among the Genetic Algorithm operators: Information theory tools used in a holistic way.” Parallel Problem Solving from Nature 2 (1992), p75-84.

31. De Souza, Pedro S.; Talukdar, “Sarosh N. “Genetic Algorithms in Asynchronous Teams.” Fourth Conference on Genetic Algorithms July (1991).

32. Dijkstra, E.W. “A Note on Two Problems in Connexion with Graphs.” Numerische Mathematic 1 (1959), p269-271.

33. Dreyfus, S.E. and Wagner, R.A., “The Steiner problem in graphs”, Networks 1 (1971) p195-207.

34. Dutta, D.; Srinivas, Y.L. “A Cyclide Based Approach for Computing Freespaces in Three Dimensions.” Proceedings of Computer Graphics International, Tokyo (1992):

35. Evan, S. Algorithmic Combinatorics. New York: The MacMillan Company, 1973.

36. Farin, Gerald. “From Conics to NURBS: A Tutorial and Survey.” IEEE Computer Graphics and Applications, 1992, 78-86.

37. Foulds, L.R. and Rayward-Smith, V.J, “Steiner problems in graphs: algorithms and applications”, Eng. Opt. 7 (1983) [7-16.

38. Fraichard, Th. “Smooth Trajectory Planning for a Car in a Structured World.” Proc.of the IEEE Intl. Conf. on Robotics and Automation, Sacramento, CA (1991).

39. Franklin, W.R.; Akman, Varol.“Partitioning the Space to Calculate Shortest Paths to any Goal around Polyhedral Obstacles.” ISA (1985): 45-52.

40. Franklin, W.Randolph; Akman, Varol; Verrilli, Colin. “Voronoi diagrams with barriers and on polyhedra for minimal path planning.” The Visual Computer 1 (1985): 133-150.

41. Garey, M.R.; Johnson, D.S. “The Rectilinear Steiner Tree Problem is NP-Complete.” SIAM Journal Applied Mathematics (1977), p826-834.

42. Gerritis, M.; Hogeweg, P. “Redundant coding of an NP-complete problem allows effective Genetic Algorithm search.” Parallel Problem Solving from Nature 1 (1990). P70-79.

43. Gilbert, E.N., Pollak, H. O. “Steiner Minimal Trees.” SIAM J. Appllied Mathematics 16 (January 1968), p1-29.

44. Goldberg, D.E. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Publ. Co.,Inc., 1989.

45. Grefenstette, John. “Cenetic algorithms for changing environments.” Parallel Problem Solving from Nature 2 (1992), p137-144.

46. Group, KQML Advisory. “An Overview of KQML: A Knowledge Query and Manipulation Lanquage.” Handout (1992).

47. Hakimi, S.L., Steiner’s problem in graphis and its implications. Networks, 1 (1971) p113-133.

48. Hesser, J.; Manner R.; Stucky O. “Optimization of Steiner Trees using Genetic Algorithms.” Proceedings of the 3rd International conference on Genetic Algorithms (June 1989), p231-236.

49. Hesser, J.; Manner, R.; Stucky, O. “On Steiner Trees and Genetic Algorithms.” Lecture Notes in Artificial Intelligence: Parallelism, Learning, Evolution (1989), p509-525.

50. Ho, Jan-Ming; Vijayan, G.; Wong, C. K. “New Algorithms for the Rectilinear Steiner Tree Problem.” IEEE Transactions on Computer-Aided Design, Vol 9, n2, February 1990 9 (2 1990): 185-193.

51. Hoffmann, C. “How to Construct the Skeleton of CSG Objects.” Technical Report CSD-TR-1014, C.S. Department, Purdue University (1990).

52. Hogg, Tad; Williams, Colin P. “Solving the Really Hard Problems with Cooperative Search.” AAAI (1993): 231-236.

53. Hyafil, L.R.; Rivest, L. “Graph Partitioning and Constructing Optimal Decision Trees are Polynomial Complete Problems.” IRIA laboria, Rapport de Recherche 33, Le Chesnay, (1973).

54. Jacobs, P.; Canny, J. “”Planning Smooth Paths for Mobile Robots.” IEEE Intl. Conf. on Robotics and Automation (1989).

55. Jain, D; Chatterjee, M.; Unemori, A.; Thangam, N. “A Knowledge Based Automatic Pipe Routing System.” Computers in Engineering 1 (1992):

56. Jo, D.; Didier, K. “A Reactive Robot Navigation System Based on a Fluid Dynamics Metaphor.” Parallel Problem Solving from Nature 1 (October 1990):, p356-362.

57. Johnson, J.; Jenson, M. “Personal communications with two aerospace cable harness designers at Lockheed.” (1994).

58. Kass, Michael; Witkin, Andrew; Terzopoulos, Demetri. “Snakes: Active Contour Models.”,Proceedings of the 1st Intl. Conference on Computer Vision, (1987), p259-268.

59. Khatib, Oussama. “Real-Time Obstacle Avoidance for Manipulators and Mobile Robots.” The International Journal of Robotics Research 5 (Spring 1986): 90-98.

60. Khedro, T.; Genesereth, M.R.; Teicholz, P.M. “GCDA: A framework for collaborative distributed multidisciplinary design.” AI in Collaborative Design 1 (1993), p67-81.

61. Korngold, Jacob; Gabriele, Gary. “Computer Aided Design of Cables for Large Computer Systems: A Case Study in Quick Prototyping.” Computers in Engineering 1 (1992): 311-318.

62. Koza, J.R. Genetic Programming: on the Programming of Computers my Means of Natural Selection. Cambridge, MA: The MIT Press, 1992.

63. Kruskal, J.B. “On the Shortest Spanning Subtree of A Graph and the Traveling Salesman Problem.” Proceedings of the American Mathematical Society 7 (January 1956), p48-50.

64. Lander, S.E.; Lesser, V.R. “Organizing cooperative search among heterogeneous expert agents.” AI in Collaborative Design 1 (1993), p89-101.

65. Laszewski, G. “Intelligent Structural Operators for the K-way Graph Partitioning Problem.” Proceedings of the 4th International Conference on GA (1991), p45-52.

66. Latombe, J.C. Robot Motion Planning. Boston, Mass., Kluwer Academic Publishers, 1990.

67. Latombe, J.C. “A Fast Path Planner for a Car-Like Indoor Mobile Robot.” Ninth National Conference on Artif. Intel., AAAI-91, Anaheim, CA (July 14-19 1991).

68. Lee, D.T.; Smith, J.M.; Liebman, J.S. “An O(nlogn) Heuristic Algorithm Rectilienear Steiner Tree Problem.” Engineering Optimization 4 (April 1980), p179-192.

69. Lee, D.T. “Medial Axis Transformation of a Planar Shape.” IEEE Transaction of Pattern Analysis and Machine Intelligence PAMI-4 (1982).

70. Lozano-Perez, Tomas, Wesley, Michael A. “ An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles.” Communications of the ACT v22 (no 10 1979): 560-570.

71. Mahfoud, S. “|Genetic Drift in Sharing Methods.” IEEE World Congress on Evolutionary Computation 1 (1994), p67-72.

72. Mason, C. “MATE: An experiment while you sleep test bed. “ AI in Collaborative Design, 11th National Conference on AI (1993), p103-106.

73. Melzak, Z. A., “On the problem of Steiner”, Canadian Math Bulletin, Vol 4 (1961), p143-148.

74. Miriyala, S.; Hashmi, J.; Sherwani, N. “Switchbox Steiner Tree Problem in Presence of Obstacles.” 1991 IEEE Conference on Computer-Aided Design (1991):

75. Mitchell, J.S.B. “Planning Shortest Paths.” Stanford Thesis (1986):

76. Narikawa, N., S. Fujimoto, N. Sasaki, and S. Azuma. “An Automated layout design system for Industrial Plant Piping.” Computers in Engineering 1 (1991): 1-6.

77. Ngo, J.T.; Marks, J. “Spacetime Constraints Revisited.” Computer Graphics Proceedings (1993), p343-350.

78. Nilsson, Nils. “Principles of Artificial Intelligence, Morgan Kaufmann Publishers.” (1980).

79. Ostermeier, Andreas. “An Evolution Strategy with momentum Adaptation of the Random Number Distribution.” Parallel Problem Solving from Nature 2 (1992), p197-206.

80. Papadimitriou, Christos H. “An Algorithm for Shortest-Path Motion in Three Dimensions.” Information Processing Letters 20 (1985): 259-263.

81. Paredis, J. “The emergence of Data Structures from Local Interactions.” Lecture notes in Computer Graphics, Parallel Problem Solving from Nature 1 (1990), p411-414.

82. Park, H.; Cutkosky, M.; Conru, A.; Lee S-H. “Computational Support for Concurrent Engineering of Cable Harnesses.” Computers in Engineering Conference San Francisco, CA (1992).

83. Park, H.; Cutkosky, M.; Conru, A.; Lee S-H. “Agent-Based Architecture for Concurrent Cable Harness Design.” (1993).

84. Patrikalakis, N; Gursoy, H. “Shape Interrogation by Medial Axis Transform.” Design Lab. Memo., Sea Grant College Program, MIT (1989).

85. Pentland, Alex; Sclaroff, Stan.“Closed-Form Solutions for Physically Based Shape Modeling and Recognition.” IEEE Transactions on Pattern Analysis and Machine Intelligence 13 (7 1991): 715-729.

86. Prim, R.C. “Shortest Connection Networks and Some Generalizations.” Bell System Technical Journal (1957).

87. Quinlan, S., Khatib O. “Elastic Bands: Connecing Path Planning and Control.” 1991.

88. Quinlan, Sean; Khatib, Oussama.“Towards Real-Time Execution of Motion Tasks.”, working document, 1992.

89. Schwefel, Hans-Paul. Numeric Optimization of Computational Models. Wiley, 1981.

90. SDRC. “SDRC WWW Homepage (http://www.sdrc.com).” (1996).

91. Shahookar, K; Maxumder, P. “A Genetic Approach to Standard Cell Placement Using Meta-Genetic Parameter Optimization.” IEEE Transactions on Computer-Aided Design 9 (May 1990). P500-511.

92. Shibata, T.; Fukuda, T. “Robot Motion Planning by Genetic Algorithm with Fuzzy Critic.” 8th IEEE Int. Symp. On Intelligent Control August 25-27 (1993).

93. Shimada, Kenji; gossard, David. “Automated Shape Generation of Components in Mechanical Assemblies.” ASME Advamces om Design Automation 1 (1992), p51-58.

94. Shing, M.T.; Parker G.B. “Genetic Algorithms for the Development of Real-Time Multi-Heuristic Search Strategies.” Proc. 5th ICGA (1993), p565-570.

95. Shore, M.L., Foulds, L.R., Gibbons, P.B., “An algorithm for the Steiner problem in graphs”, Networks 12 (1982), p323-333.

96. Sims, K. “Artificial Evolution for Computer Graphics.” Computer Graphics 35 (4 1991), P319-328.

97. Srinivasan, V.; Nackman, L. “Voronoi Diagram for Multiply-connected Polygonal Damains I: Algorithm.” IBM Journal of Reseach and Development 31 (March 1987).

98. Sriram, D. “Computer supported collaborative engineering.” AI in Collaborative Design 1 (1993), p117-127.

99. Sudhalkar, Atul; Gursoz, Levent; Prinz, Fritz. “Continous Sketletons of Discrete Objects.” 2nd ACM/IEEE Symposium on Solid Modeling and Applications, Montreal, Canada, May 1993.

100. Szeliski, R.; Tonnesen, D. “Surface Modeling with Oriented Particle Systems.” Computer Graphics 26 (1980), p185-194.

101. Takahashi, H.; Matsuyama, A. “An approximate solution for the Steiner problem in graphs.” Math. Japonica 24 (1980), p573-577.

102. Talukdar, S.N., deSouza P.S., Q., & C., R.V. “A-Teams: Multi-Agent Organizations for Distributed Iteration.” EPRI/NSF Workshop on Application of Advanced Mathematics to Power Systems (1991).

103. Talukdar, Sarosh. “Asynchronous Teams.” Fourth International Symposium on Expert Systems Application to Power Systems, La Trobe University, Melbourne, Australia, January 4-8, 1993 (1993).

104. Terzopoulos, Demetri; Fleischer, Kurt. “Deformable Models.” The Visual Computer 4 (1988), p306-331.

105. Terzopoulos, D.; Metaxas, D. “Dynamic 3D Models with Local and Global Deformations: Deformable Suuperquadrics.” IEEE Transactions on Pattern Analysis and Machine Intelligence 13 (July 1991), 703-714.

106. Thangiah, S.R.; Nygard, K.E. “School bus routing using genetic algorithms.” SPIE Applications of Artificial Intelligence X: Knowledge-Based Systems 1707 (1992), 387-98.

107. Thangiah, S.R.; Nygard, K. E. “MICAH: A Genetic Algorithm System for Multi-Commodity Transshipment Problems.” 8th Conference on Artif. Intel. For Applications (March 26, 1992, Monterey, CA), p240-246.

108. Thingvold, J. A.; Cohen, E. “Physical Modeling with B-spline Surfaces for Interactive Design and Animation.” Proceedings of the 1990 Symosium on Interactive 3D Graphics, v 24 n3, p127-139.

109. Trummer, Richard; Gishpert, John; Hoeft, Lothar; Freyer, Eric. “Investigation of Stresses Produced in Electrical Wiring during Bending.” Annual Connectors and Interconnection Technology Symposium Proceedings 18th. Publ by Electronic Connector Study Group Inc, For Washington, PA, USA (1985), 14-28.

110. Vasseur, H.A.; Pin, F.G.; Taylor, J.R. “Navigation of a Car-Like Mobile Robot using a Decomposition of the Environment in Convex Cells.” Proc. of the 1991 IEEE Intl. Conf. on Robotics and Automation, Sacramento, CA (April 1991).

111. Wang, L.; Lai, Y.; Liu, B. “Simultaneous Pin Assignment and Global Wiring for Custom VLSI Design.” IEEE International Symposium on Circuits and Systems Part 4 (of 5) (June 11-14, 1991) Singapore 1991), p2128-2131.

112. Welch, R.V.; Dixon, J.R. “Extending the iterative redesign model to configuration design; sheet metal brackets as an example.” Design theory and Methodology - DTM, Montreal, Quebec CANADA: ASME (1989), p81-88.

113. Whitley, D. “The GENITOR Algorithm and Selection Pressure: Why rank-based allocation of reproduction trials is best.” Proc. of the 3rd Intl Conf on GA and their Applications (1989), p116-121.

114. Wilfong, G.T.;. “Motion Planning for an Autonomous Vehicle.” 1988 IEEE Intl Conf. on Robotics and Automation (1988), p529-533

115. Winston, P.H. Artificial Intelligence. Reading, MA: Addison-Wesley Pub. Co, 1992.

116. Winter, Pawel. “Steiner Problem in Networks: A Survey.” Networks 17 (1987), p129-167.

117. Yu, Xinhua, Goldak, J.A., Dong, Lingxian. “ Constructing 3-D Discrete Medial Axis.” Proceeding of Symposium on Solid Modeling Foundations and CAD/CAM Applications (1991).