10-1-2001

A heuristic algorithm to generate test program sequences for moving probe electronic test equipment

Bertha M. Arteta

Florida International University

DOI: 10.25148/etd.FI14032352

Follow this and additional works at: <https://digitalcommons.fiu.edu/etd>

Part of the [Industrial Engineering Commons](#)

Recommended Citation

Arteta, Bertha M., "A heuristic algorithm to generate test program sequences for moving probe electronic test equipment" (2001). *FIU Electronic Theses and Dissertations*. 1317.
<https://digitalcommons.fiu.edu/etd/1317>

This work is brought to you for free and open access by the University Graduate School at FIU Digital Commons. It has been accepted for inclusion in FIU Electronic Theses and Dissertations by an authorized administrator of FIU Digital Commons. For more information, please contact dcc@fiu.edu.

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

A HEURISTIC ALGORITHM TO GENERATE TEST PROGRAM
SEQUENCES FOR MOVING PROBE ELECTRONIC TEST EQUIPMENT

A thesis submitted in partial fulfillment of the

requirements for the degree of

MASTER OF SCIENCE

in

INDUSTRIAL ENGINEERING

by

Bertha M. Arteta

2001

To: Dean Vishwanath Prasad
College of Engineering

This thesis, written by Bertha M. Arteta, and entitled A Heuristic Algorithm To Generate Test program Sequences For Moving Probe Electronic Test Equipment, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this thesis and recommend that it be approved.

Dr. Venkat Panchanadam

Dr. Richard Linn

Dr. Ronald Giachetti, Major Professor

Date of Defense: October 1, 2001

The thesis of Bertha M. Arteta is approved.

Dean Vishwanath Prasad
College of Engineering

Dean Douglas Wartzok
University Graduate School

Florida International University, 2001

© Copyright 2000 by Bertha M. Arteta

All rights reserved.

DEDICATION

I dedicate this thesis to my family, especially my parents, whom without their love, patience, support, and encouragement, the completion of this work would not have been possible. I love you all.

ACKNOWLEDGMENTS

I would like to thank the members of my committee for their help and contributions especially my major professor Dr. Ronald Giachetti, who has guided me through this undertaking.

I would also like to thank my cousins Ximena and Henry and my friends Alba and Maneli for their help and support in good and bad times. Very special thanks to the Industrial Engineering Department at Florida International University, especially Nancy Urbina for her help and support during this new accomplish of my life.

Finally thanks to Gables Engineering, especially Tony Musiby, for his help in the data collection.

ABSTRACT OF THE THESIS

A HEURISTIC ALGORITHM TO GENERATE TEST PROGRAM
SEQUENCES FOR MOVING PROBE ELECTRONIC TEST EQUIPMENT

by

Bertha M. Arteta

Florida International University, 2001

Miami, Florida

Professor Ronald Giachetti, Major Professor

The electronics industry, is experiencing two trends one of which is the drive towards miniaturization of electronic products. The in-circuit testing predominantly used for continuity testing of printed circuit boards (PCB) can no longer meet the demands of smaller size circuits. This has lead to the development of moving probe testing equipment. Moving Probe Test opens up the opportunity to test PCBs where the test points are on a small pitch (distance between points). However, since the test uses probes that move sequentially to perform the test, the total test time is much greater than traditional in-circuit test. While significant effort has concentrated on the equipment design and development, little work has examined algorithms for efficient test sequencing. The test sequence has the greatest impact on total test time, which will determine the production cycle time of the product. Minimizing total test time is a NP-hard problem similar to the traveling salesman problem, except with two traveling salesmen that must coordinate their movements.

The main goal of this thesis was to develop a heuristic algorithm to minimize the Flying Probe test time and evaluate the algorithm against a “Nearest Neighbor” algorithm. The algorithm was implemented with Visual Basic and MS Access database. The algorithm was evaluated with actual PCB test data taken from Industry. A statistical analysis with 95% C.C. was performed to test the hypothesis that the proposed algorithm finds a sequence which has a total test time less than the total test time found by the “Nearest Neighbor” approach. Findings demonstrated that the proposed heuristic algorithm reduces the total test time of the test and, therefore, production cycle time can be reduced through proper sequencing.

TABLE OF CONTENTS

CHAPTER	PAGE
INTRODUCTION	1
1.1 Background.....	2
1.2 Problem Statement.....	6
1.2.1 <i>Flying Probe Test</i>	6
1.3 Goals and Specific Objectives	7
1.4 Significant Effort of Research	8
LITERATURE REVIEW	9
2.1 Travel Salesman Problem.....	9
2.2 Optimization Methods	11
2.2.1 <i>Improving and Constructive Search Heuristic</i>	12
2.2.2 <i>Tabu Search</i>	14
2.2.3 <i>Genetic Algorithm</i>	15
2.3 Moving Probe Test	16
METHODOLOGY	21
3.1 Data Collection Methodology	21
3.2 System Design Methodology.....	22
3.2.1 <i>Model Input Data</i>	24
3.2.2 <i>Model Output Data</i>	24
3.3 Outputs and Results	24

HEURISTIC ALGORITHM.....	26
4.1 Terminology definition.....	26
4.2 Assumptions	28
4.3 Algorithm Logic	30
4.4 Phase I : Network Sequence Determination	30
4.4.1 <i>Logic of Phase I</i>	31
4.4.2 <i>Mathematical Model Phase I</i>	32
4.4.3 <i>Phase I Example</i>	36
4.5 Phase II: Nodes Sequence.....	40
4.5.1 <i>Logic of Phase II</i>	41
4.5.2 <i>Phase II Example</i>	42
4.6 Phase III.....	45
4.6.1 <i>Logic of Phase III</i>	45
4.6.2 <i>Phase III Example</i>	50
SYSTEM DESIGN AND IMPLEMENTATION.....	54
5.1 Data Collection.....	54
5.1.1 <i>Netlist</i>	54
5.1.2 <i>Netlist Size Selection</i>	56
5.2 Model Input Data.....	59
5.3 Prototype System.....	60
5.4 Model Output Data	63
STATISTICAL ANALYSIS OF ALGORITHM	65
6.1 Outputs and Results	65

6.1.1	<i>Test-Steps</i>	65
6.1.2	<i>Distance</i>	66
6.1.3	<i>Computational Complexity</i>	67
6.1.4	<i>Run Time</i>	69
6.2	Statistical Analysis	71
CONCLUSION.....		78
REFERENCES		81
APPENDICES		83

LIST OF TABLES

TABLE	PAGE
Table 1: Netlist Data Example for Phase I.....	37
Table 2: ProbeBMove List Example for Phase II.....	43
Table 3: ProbeAMove List Example for Phase II.....	44
Table 4: Final Test Sequence for Netlist Example	53
Table 5: Estimated Total Test-Step for different Netlist Sizes	57
Table 6: Netlist Sample for Testing	58
Table 7: Netlist Example from PCB Industry converted as Input Data.....	60
Table 8: Output Example of Netlist from PCB Industry	64
Table 9: Number of Test-Steps by Netlist Sample	66
Table 10: Total Test Distance by Sample using Nearest Neighbor Algorithm and Proposed Algorithm.	67
Table 11: Computational Complexity by Sample using Nearest Neighbor Algorithm and Proposed Algorithm.	68
Table 12: Run Time by Sample using Nearest Neighbor Algorithm and Proposed Algorithm.	70
Table 13: Calculated Values for Normal Probability Plots	73
Table 14: Paired Distance Data.....	76

LIST OF FIGURES

FIGURE	PAGE
Figure 1: Flying Probe Test Machine	3
Figure 2: Cost Comparison between Bed-of-Nails and Flying Probe Test	5
Figure 3: Open and Short Example in An Electronic Circuits.....	6
Figure 4: System Architecture	23
Figure 5: Logic of Heuristic Algorithm (Phase I).....	31
Figure 6: Graphic Representation of Netlist Input Data Example.....	38
Figure 7: Graphic Representation of Netlist with First Network Selected	39
Figure 8: Netlist Examples with All Networks Selected in The Sequence.....	40
Figure 9: Logic of Heuristic Algorithm (Phase II)	41
Figure 10: Netlist Example with All Nodes Selected in ProbeAMove List	44
Figure 11: Logic of Heuristic Algorithm (Phase II)	49
Figure 12: Netlist Example with Nodes Selected for ProbeAMove List and ProbeBMove List.....	50
Figure 13: First Test-Step of Netlist Example	51
Figure 14: Test-Steps for Shorts from First Network of Netlist Example.....	52
Figure 15: Netlist from PCB Industry.....	55
Figure 16: Initialization Form for Prototype System.....	62
Figure 17: Complexity of Nearest Neighbor Algorithm Vs Heuristic Proposed.....	68
Figure 18: Run Time of Nearest Neighbor Algorithm Vs Heuristic Proposed.....	71
Figure 19: Normal Probability Plot for difference of the Pair of Observations.....	74

Figure 20: Normal Probability Plot for Difference of the Pair of Observations / Number of Test Points.....	74
Figure 21: Normal Probability Plot for Logarithm of Difference of the Pair of Observations.....	75
Figure 22: Total Distance of Nearest Neighbor Algorithm Versus Heuristic Proposed...	78

CHAPTER 1

INTRODUCTION

Electronics is one of the most competitive industries today and companies must be able to adapt to their customers' changing needs in order to survive. As technology rapidly changes, it is crucial for companies to quickly and successfully respond to the new technological challenges. The competition is globalizing at the same time as the market is moving to economically growing areas. A typical electronic product contains several components that are assembled on laminated boards, called printed circuit board (PCB). PCB manufacturing is capital intensive. The PCB is a component of many technological devices such as tachometers, cash registers, personal computers, and including the high growth areas of computer equipment, cellular telephones, and telecommunications equipment.

In order to move forward, the PCB industry must have an efficient and flexible test process. Cost-benefit analysis studies [Kovartovsky, 1999] have shown that a poorly designed test in complex assemblies, can increase the original production cost by as much as 50%. This shows how important it is to select an appropriate method to test the PCB. Test methods include electrical test, system test, automatic electrical test with automatic test equipment (ATE), including in-circuit test, moving probe test, automatic function test, scanning test and aging test methods, and automated optical inspection (AOI).

This effort focuses on the moving probe test sometimes called the flying probe test. Flying probe test is the latest development in electronic test. It tests circuits

sequentially and can be model as a variant of the Travel Salesman Problem. The manufacturing objective is to minimize test time so that cost of testing is minimized and more importantly manufacturing throughput is maximized. The main goal of this effort is to develop a new algorithm to minimize Flying Probe Test time. Efforts of this study demonstrate that grouping the data properly in combination with a good method of optimization offer the opportunity of improve sequence of the test.

This document is organized in chapters. The first chapter presents the problem statement, objectives of this proposal and its significance. Chapter 2 includes a literature review on the Traveling Salesman Problem and Flying Probe Test. Chapter 3 presents the methodology used in this project. Chapter 4 discusses the logic of the heuristic algorithm proposed. Chapter 5 presents the system design approach. Chapter 6 relates to the result and statistical analysis and finally Chapter 7 discuss conclusions and further work.

1.1 Background

In electronics, if there is a single fault then usually the entire system fails. A failure will prevent the system from functioning correctly; consequently testing the circuit for faults is important. Testing is necessary to evaluate functionality, electrical continuity and environmental stress screening. The method of test must be carefully chosen depending on cost of the method and the probably number of pieces of the assembly.

For testing electrical continuity, there are two known methods that are frequently used: Bed-of-Nails and Flying Probe Test. The Bed-of-Nails is a test fixture that has nails that contact the PCB and is connected to Automated Test Equipment and checks for

opens and shorts. A fixture kit costs between \$3,000 and \$15,000 depending on size and complexity of the board (Bouskela, E. 2000). The fixture associated with the Bed-of-Nails generates the most expensive cost in the production line for PCB electronic testing. However, the most important benefit of this test is the testing time where a full board can be tested in approximately 30 seconds for a medium sized board. Consequently, this method is very cost effective for high production quantity.

Flying Probe Test (See Figure 1) opens up the opportunity to test printed circuit boards that are not cost effective to test on fixture-based systems. Instead, two to four probes move over the board and perform the test.

Figure 1: Flying Probe Test Machine¹

For low-volume and high-mix production, Flying Probe is more effective since it eliminates the cost of fixturing. A second reason for using the Flying Probe Test is that the bed-of-nails cannot test boards where the test points are on a small pitch (distance

¹ Taken from Teradyne, 2000

between points). Only a flying probe can test these boards. As electronics gets smaller (beepers, cellular telephones, laptop computers) then it becomes more difficult to test those products. Flying probe is very effective in prototype development, high net count boards, and field returns.

Figure 2 shows a cost comparison between the Bed-of-Nails test and the Flying probe Test. The cost of the flying probe machine is greater than cost of the bed-of nails test machine. If it is assumed that the cost of the machine represents a portion of the variable cost of each board printed, then this variable cost will be higher for boards tested with flying probe. However, since the flying probe does not include the cost of a fixture, for low production it is reasonable the use of flying probe testers. A cost line in the figure shows how a certain amount of production (break-even point), bed-of-nails is more cost effective.

Figure 2: Cost Comparison between Bed-of-Nails and Flying Probe Test²

The major disadvantage of Flying Probe is the long test time. Since it tests the board sequentially, it takes more time than bed-of-nails which tests the PCB networks simultaneously. This difference in time makes flying probe uneconomical for high production runs since the manufacturer would not be able to attain the product rates necessary to stay competitive. Flying probe testing has been found appropriate for prototypes, which have low quantities and small products because of the small distance between components to be tested. An effort to improve Total Flying Probe Test time would make the test more profitable because reducing test time will reduce total manufacture time and hence reduce total cost.

² Giachetti, 2000

1.2 Problem Statement

The problem addressed by the research is the sequencing of the probes during test in order to minimize total test .

1.2.1 Flying Probe Test

PCBs are composed of networks. A network is a circuit that connects two or more points that have the same voltage. Flying probe test is used for testing the electrical continuity of the circuit. This tests if there are not any short or open circuits. A *short* is when two circuits are connected but they are not supposed to be connected and an *open* is when a circuit that should be connected is not connected (Kozich, 1992). See Figure 3 for an illustration of shorts and opens.

Figure 3: Open and Short Example in An Electronic Circuits³

³ Giachetii, 2000

To test for opens a probe can be put at the end of the circuits to be tested (For example U1 and U2) and a voltage will be applied on one side. If voltage is not sensed at the other side, there is an open. To test for shorts, voltage is applied to the circuit being tested and every other circuit is checked. If voltage is sensed on any of the circuits checked, it means there is a short.

A well-designed flying probe will deliver a rapid test development with minimum debug time (Bouskela, 2000). Test time can be reduced by finding an optimal sequence followed by the probes. This reduction in time would represent a competitive advantage of any Flying Probe Test Machine Manufacturer and a reduction in manufacturing cost.

1.3 Goals and Specific Objectives

The main goal of this research is to develop an algorithm to minimize the Flying Probe Test time. To validate the algorithm developed, it will be applied to different PCB sizes and statistically analyzed.

Run time and optimality of the heuristic algorithm proposed is compared with the results of a nearest neighbor approach.

The specific objectives of this proposal are as follows:

- Construct a heuristic algorithm to generate flying probe test sequences.
- Build a Visual Basic program that implements the s logic – Heuristic proposed and nearest neighbor.
- Generate Flying probe test sequences for different Printed Circuit Board Sizes using the two approaches: the proposed heuristic and the nearest neighbor.

- Compare results obtained between the proposed algorithm and the nearest neighbor based on optimality, complexity, and run time.
- Conduct statistical analysis and make conclusions.

1.4 Significant Effort of Research

In electronics, usually a small failure makes the entire system fails. A fail can be a solder-bridge, missing solder or, a missing or incorrect component. The cost of a failure in electronics depends on the phase of the manufacturing in which the failure is detected. If it is estimated that a failure detected in the fabrication phase cost one dollar, the same failure detected in the assembly phase will cost \$10. If the failure is detected in the testing phase, the cost will be increased to \$100, and finally in the customer the cost will be \$1000. As seen, the cost of a failure grows exponentially and testing represents a very important phase of electronics manufacturing.

The number of PCB manufactured is increasing over years. Developing new algorithms to reduce testing time can represent significant savings for the companies. Test and subsequent cost can raise production cost up to 50% of the original cost (Kovartovsky, 1999).

CHAPTER 2

LITERATURE REVIEW

The main purpose of this chapter is to analyze algorithms to generate flying probe test sequences. The flying probe is relatively new machine and few algorithms have been developed to generate test for them. Since the flying probe sequencing problem can be viewed as an adaptation variation of the traveling salesman problem the literature review focuses on algorithms for this type of problem.

2.1 Travel Salesman Problem

The simplest and most famous of routing problems is the Traveling Salesman Problem (TSP) (Rardin, 1998). Routing problems are those that organize a collection of customer locations, jobs, cities, points (as in PCB), and the objective is to find the shortest route that visits each point once and only once. The TSP problem is very easy to understand but it is very difficult to solve computationally. Small problems can be modeled and solved as a linear program or as quadratic assignment problem but larger problems can only be solved by heuristics algorithms that do not guarantee the optimality of the solution. The basic and simplest formulation of the Traveling Salesman Problem is the linear formulation. The TSP mathematical model is:

Objective Function

$$\text{Min} \sum_{i=1}^n \sum_{j=1}^n d_{ij} x_{ij} \quad \left\{ \begin{array}{l} x_{ij} = 1, \text{ if city } j \text{ follows city } i \text{ in the sequence} \\ x_{ij} = 0, \text{ otherwise} \end{array} \right. \quad (1)$$

Constraints:

$$\sum_{i=1}^n x_{ij} \quad j = 1 \dots n \quad \text{Each city } j \text{ is entered only once} \quad (2)$$

$$\sum_{j=1}^n x_{ij} \quad i = 1 \dots n \quad \text{Each city } i \text{ is exited once} \quad (3)$$

Sometimes, the above two constraints are insufficient because the solution includes subtours that satisfy the constraints, but not the problem. Then, it is necessary to break subtours with the addition of new constraints, one for each subtour. The constraints guarantee that every tour cross between points in the subtour and point outside it at least twice.

The Traveling Salesman Problem seeks a minimum-total-length route visiting every point in a given set exactly once. The Traveling Salesman Problem is a class of problems known as NP-Complete. This type of problem become very difficult to solve as the problem size grows, because the problem cannot be solved in polynomial time, (Nemhausen et al. 1988). This characteristics makes some problems impossible to solve for high size and the use of heuristic algorithms become very important.

Flying Probe Test can be model as an variation of the Traveling Salesman Problem since the test goal is to find the shortest sequence to visit each network once and only once. The difference between the Flying Probe Test and the traditional TSP is (1) there are two probes that move simultaneously and their movement must be coordinated

and (2) the probes must test each network once and only once for opens but will visit the networks more than once when testing for shorts.

Since the TSP is NP-hard the literature has focused on heuristic approaches to solve the problem. Usually they combine well-known algorithm and try to find solutions closer to the optimal without increase the complexity. Some authors combine different algorithms and apply them to specific models.

2.2 Optimization Methods

The simplest method to solve a problem is the most direct: analyze all the possibilities. Total enumeration solves a problem trying all possible combinations of variables. This approach finds a global optimal solution but it grows exponentially with the number of points to be visited. Exponential growth makes impossible and impractical to use Total Enumeration to solve even medium size problems. Current computers can solve 10^9 arithmetic operations in a second to solve a 80-variable model would require $2^{80} / 10^9 \approx 1,208 * 10^{12}$ seconds ≈ 388 centuries. Most PCBs will have more than 80 test points so the brute force method of total enumeration is an infeasible approach. Then it is natural to look for more complex methods to solve the problem in less time, even if they do not find a global optimal solution. However, good solutions close to the optimal can be, in many cases, an excellent option. The most important solution methods are reviewed below.

2.2.1 Improving and Constructive Search Heuristic

Heuristic algorithms are those that can be used to find a good feasible solution even though their optimality cannot be proved. Some of the most well known heuristic are the Nearest Neighbor algorithm, the use of space filling curves, convex hull approaches and many others. The least complex algorithm to use is the Nearest Neighbor (NN) but its optimality is very low (Rardin, 1998). Many authors have developed algorithms that improved and original solution found by NN or use NN at some point of their algorithm to solve small parts of it. In the following paragraphs we review algorithms that have been applied to the similar drill sequencing problems of PCB manufacturing.

Litke (1984) presented an algorithm developed for solving PCB drilling problems. It uses two sub-algorithms, one to find the cluster of points and the other an exhaustive search procedure to find the optimal path inside the cluster. First, it clusters all point on the board with a maximum of eight points per cluster. Then it creates new points with the center of gravity of the original points and they are clustered again. In the second stage, it plays with the different points and minimize distances inside the cluster. Since the size of each cluster is only eight, the exhaustive search inside the cluster is feasible. The algorithm generates very good results but it is very difficult to code since it must keep track of each cluster in each level of clustering.

Danusaputro, Chung-Lee, and Martin-Vega (1989) used Litke's heuristic and NN to capitalize on the structure of a PCB. The new algorithm use Litke's idea of divide the total number of points in cluster, but inside each cluster it uses NN to determine the sequence. This new algorithms does not break the clusters and they are preserved even if

there are more than eight points in the cluster. The algorithm starts in the origin and the first point in each cluster is the nearest to the last one in the sequence of the previous cluster. On the average, this new algorithm resulted in a distance (time) reduction of 28% per board. Comparison between both methods shows that Litke's algorithms requires more time, is more complex and does not give better results. This algorithm was implemented by the Northern Telecom Electronics in West palm Beach Florida.

A heuristic based on a continuous approximation developed by Castillo (1998) constructs a simple solution that is well suited for software tools. In principle, the points of the tour should be distributed over a region with a circular shape. The tour must pass through the depot that is located at the sector vertex. The tour then is constructed according to an optimal partition of the region into three zones: two inner symmetric sectors and an outer ring sector. The partition is defined uniquely by one parameter and this may be calculated so to make the expected tour length minimum. Important results from this study show that the construction procedure is virtually insensitive to the metric. In order to evaluate the performance of the algorithm, numerical tests were done and they concluded that distance (or time) were reduced in 5%. This improvement plus the attractive easy implementation make this algorithm a very good option for decision support system.

Zhang (2000) developed a new approach of the Nearest Neighbor algorithm call Nearest Neighbor's Neighbors (NNN). This algorithm is a modification of NN since not only consider the distance between a point and its direct neighbor, but also between the neighbor and the neighbor's neighbors. For large problems, the algorithm reduces the number of nearest neighbor's neighbor to be check at two because it consumes great

computation time. Zhang used Flying Probe Test problems to test the feasibility of the algorithm. Results show that with the new algorithm better solutions are found. Complexity of the algorithm increase greatly compared with the NN but computational time were still small for the new algorithm.

2.2.2 *Tabu Search*

Improving search can be effective on many models. Tabu search is a high-level heuristics method for solving optimization problems (Glover, 1990). The Tabu search method is designed to prevent other methods from becoming trapped at locally optimal solutions. Tabu search is still in an early stage of development, however, it has enjoyed a number of successes.

Tabu search starts with a current solution design as best solution and create a candidate list of moves. Each moves represents a new solution. The best admissible solution candidate from the moves is selected and records it as best solution if it improves on the previous best. Admissibility is based on the tabu restrictions and aspiration criteria. Those conditions are updated every time before a list of moves is generated. The search stops when it has completed a number of iterations. Each step may improve or degrade the objective function but the solution of the problem is the one recorded as best solution.

Tabu search has been applied in printed circuit board to drilling sequence problem. Results vary with models and details of tabu criteria (Glover, 1990). Quality of heuristic solutions can be enhanced by suitable implementation of tabu search.

2.2.3 *Genetic Algorithm*

A third popular method in improving search is known as genetic algorithm. It evolves good heuristic optima by following the process of biological evolution to find better and better solution (Lalena, 1998). It is inspired in Darwin's theory about evolution. The algorithm is started with a set of solutions called the population. Solutions from the population are used to form a new population. It is hoped that the new population will be better than the old one. The criteria selection is according to their fitness - the more suitable they are the more chances they have to reproduce. This is repeated until some condition (for example number of populations or improvement of the best solution) is satisfied.

Traveling salesman problem can be solved in less time using genetic algorithm and still very good solutions can be obtained. In this case, the population would be many random tours stored as sequence numbers. Then, select two of the better tours (parents) from the population and combine them to create two new solutions (children). It is expected that with the creation of the children, better solutions have been also created. To create the new sequence (children) a random point from the parent's sequences is picked and switched with every number in the sequence after the point. Good solutions become parents and reproduce again, while bad solutions are removed. Also, mutation is considered. Mutation is when randomly one of the solutions is changed looking for a better solution. Sometimes, the mutation can result in a better solution not found by the normal reproduction.

The major difficulty in the use of genetic algorithm in Traveling Salesman Problem is encoding the solution. Encoding cannot be only the list of cities in the order

they are visited. It should ensure that city adjacencies in the list are preserved from the parents to the children, and also that children do not contain repeated values (such as a sequence 123659857, where 4 is missing and 5 is twice). In the switching phase, some values can turn repeated while others can disappear. In this case, constraints of the problem are broken and the solution is not feasible. However, this problem can be fixed encoding the tours as 2-dimensional array (a $N \times N$ matrix) that store city adjacencies in both directions. This approach makes the problem very complex and difficult to compute.

Algorithms built by different authors have tried to minimize the objective function using different approaches. However, when the algorithm is so complex that cannot be easily code, the effort is not useful. Some others algorithms can be coded but need a large storage requirement in the computer due to exhaustive search. Depending on the final use of the algorithm, some of the approaches can be very benefited. For Flying probe test, only those algorithms that are easy to code and not very complex have value. The use of the nearest neighbor seems to be the best selection. However, an improvement of the nearest neighbor opens the possibility of a time and cost reduction in PCB manufacturing.

2.3 Moving Probe Test

The flying probe machine is relatively new development. Crowell and Keogh (1984) are among the first to show how the unlimited testing flexibility of the test makes it useful for electronic manufacturers. Tests that previously took weeks to develop, due to

the need to build a test fixture, are now available in hours (Hassig, 1998) with the flying probe which requires no fixtures.

The importance of the flying probe test in prototypes is showed by McAvinue (1999). Prototypes test must ensure that no manufacture defects exist and that those conditions are meet at short turnaround times. A justification of expected return on investment of the application is analyzed and compared with other approaches.

Most authors have focused their studies to show how good is the test in reducing testing cost. But not many have developed algorithm to generate test sequences. Other important issue to discuss is the optimal number of probes to use in the test. Current technology generally uses two (a minimum) or more probes to perform the test (Pendurkar, Tovey, Chatterjee, 1999). For multiple probes, Chou and Cheng (1993) developed an algorithm based on multidimensional traveling salesman problem (MDTSP). They used simulated annealing and the Lin-Kernighan algorithm to optimize the MDTSP. A tree is used to represent the routing structure of an interconnection. Leaves represent the terminals, non-leaf terminals represent vias and edges represent the wires of the interconnection.

Multiple probe testers are most used for manufactures. However, in 1999 a new technique that uses only one probe was developed. This technique detects attenuation of the test stimulus applied through a tuned load using a single probe. Cost of equipment of single-probe tester is cheaper than multiple-probes (Pendurkar, Tovey, Chatterjee, 1999) and that have made some authors developed algorithms to solve the sequence for the single probe test. This new technique has the potential to be faster, cheaper and then more cost effective.

Pendurkar, Tovey, Chatterjee (1999) assessed the advantage of single-probe test over double-probe test and intended to confirm that single-probe testing has an increasing advantage over double-probe testing as the number of nets increases. After that, they developed a heuristic algorithm to optimize the sequence of the probe. The algorithm divides the total number of test points in nets. A net represent a set of interconnected layer of substrate corresponding to a single electrical node. Each net has more that one terminal. A terminal is a pad on the topmost layer of substrate that is probed during substrate testing. Each terminal belongs to only one net. The heuristic procedure improve each net of the system and the practice an improved insertion which procedure consists of selecting the best terminal of the net with the best subtour cost and inserting that terminal and form a new subtour. The improvement is repeated until an optimized terminal tour is constructed. The experiment compared the results obtained by solving different netlist sizes with two basic algorithms, the insertion algorithm and a Shuffle algorithm. The factors discussed were cost and computational time. Cost depended of the distance followed by the probe. Results show that up to 40% reduction in probe traversal time can be obtained with the algorithm.

Satisfactory low-cost results are also obtained with an algorithm proposed by Kim, Pinshan and Se Hyun (1999). The algorithm is a combination of Simulated Annealing and Lin-Kernighan that intends to optimize the sequence of the probes of the flying test probe. These two approaches are well discuss by the authors. Simulated annealing derives from the physical process of heating and then slowly cooling a substance to obtain a strong crystalline structure. The simulated annealing process lowers the temperature by slow stages until the system “freezes” and no further changes occur.

At each temperature, the simulation must proceed long enough for the system to reach a steady state or equilibrium. The sequence of temperatures and the number of iterations applied to reach this state at each temperature comprise an annealing schedule. To apply simulated annealing, the system is initialized with a particular configuration. A new configuration is constructed by imposing a random displacement. If the energy of this new state is lower than that of the previous one, the change is accepted unconditionally and the system is updated. If the energy is greater, the new configuration is accepted probabilistically. Lin-K heuristic is an efficient realization of a k-Opt procedure with k variable in each step of improvement. Basically, it generates psuedo-random feasible solution T that satisfies a criterion "valid tour", attempt to find an improved feasible solution T_n by some transformation of T ; if an improved solution is found, then replace T by T_n , and start again looking for more transformation of T . If no improved solution can be found, then T is the locally optimum solution. Procedure is repeated from beginning until computation time runs out or the answers are satisfactory. The heart of the iterative procedure is to improve upon a given solution.

Multiple authors have discussed the need of reducing flying probe test time by finding an optimal sequence of the test. The sequence is limited by the fact that two or more probes need to coordinate their movements to optimize the sequence. Single probe testers are still in their initial phase of development. Currently there is a need of developing new algorithms that improve the flying probe test and reduce the disadvantage of the high-test time of the test. There are two important considerations in developing all of the discussed algorithms. First, the analysis of the data to determine a

method of grouping the test points of the netlist. Second, a method to optimize the sequence within the group and among groups. This approach was used to develop the heuristic algorithm proposed in this study.

CHAPTER 3

METHODOLOGY

This chapter presents the steps to meet the stated objectives of Section 1.3. It includes the data collection methodology, system design and solution approach.

3.1 Data Collection Methodology

The goal of this effort is to develop an algorithm that reduces total test time of Flying Probe Test. This test is performed on PCB. Consequently, the first step in this research is recognizing the PCB design and all data related to it.

The PCB design process generates a *Netlist* from the ECAD system, which contains the test points of the design. This list is generated to indicate the points that need to be contacted to perform a test. In this case, the test is to verify that connectivity exist between two points contacted. The *Netlist* also indicates the test points that need to be connected creating a net.

The use of actual industry data is a key factor in this study. Actual Netlists needed to be analyzed to determine the best approach used to create a system that takes the information in the Netlist and find an optimal sequence to perform the test. Gables Engineering provided the Netlists used in this study.

Gables Engineering is a company founded in 1946 that manufactures a wide range of avionics controls and systems. Their product line includes dedicated VHF and HF communication control panels, Radio Management Panels, Digital Audio Systems, and VHF Navigation control panels. This company provided the Netlists used in this study.

The criteria used to select the Netlists of this research were based on: a) size of the Netlist, and b) number of test point by net (test points interconnected). The Netlists selected are found in Appendix 1 to 10.

3.2 System Design Methodology

The design of the system (including input data, sequence determination of the test and outputs) had to address three major issues: a) inputting to the system the required test data, b) a solution approach to determine the sequence of the test-steps including probes movements and type of test, and c) an appropriate organization of the output and all the data associated to it.

The system architecture and its components are shown in Figure 4. The system architecture consists of four sections. The sections work as a sequence. Section 1 represents the data to be tested (Netlist). Section 2 is the transformation of the Netlist into input data. This phase takes the attributes of the test points recorded in the Netlist and creates a new list with all the attributes required by the system. The tool used for this transformation is MS Excel. Section 3 denotes the Database that is in charge of storage the input data and the results of running the algorithm (outputs). The total number of test-step by test increases exponentially with the number of test point in the Netlist. This characteristic makes necessary the use of a database to organize all the data (inputs and outputs). MS Access was used to organize and record the data. Finally, Section 4 represents the logic of the algorithm, which uses Visual basic as a tool to run the algorithm and determine the sequence of the test. The results (outputs) of the run are recorded in the database.

Figure 4: System Architecture

3.2.1 Model Input Data

The function of this section was the analysis of the input data required to test the algorithm. Netlist information was analyzed to decide what type of data transformation was required and how it could be accomplished. Also, to determine if new attributes needed to be added to the test point in order to meet all the requirements of the system. Section 4.2.1 shows the type of input data of the system and the transformation of it. MS Excel was the tool selected for organization and transformation of the input data. MS Access was used to record the data that would be used for testing.

3.2.2 Model Output Data

The function of this section is to determine the data required as output of the system design. The movements of the probes determine the distance of the sequence and that is why this movement is the output of the system. The output was concluded to be list of all the test-step of the test. Additionally, the movement of each probe, type of test performed and total distance by test-step is also included.

3.3 Outputs and Results

After the algorithm run is completed, the output data is analyzed. For each sequence of the test, three different variables were considered: a) Total Test Distance, b) Running Time, and c) Complexity. The two algorithm of the study (the algorithm proposed and the Nearest Neighbor) were analyzed based on these three variables and a Statistical Analysis was done.

The main goal of this research was develop an algorithm to minimize the Flying Probe Test time. Based on this it was needed to demonstrate, statistically, that the proposed algorithm could find better sequence for the test. Better sequences are represented by less total test time. A t-paired test was performed with the results of the algorithm (total test distance) and an analysis of the other two variables was completed. The samples used for the paired test were considered to be independent and the Confidence Interval selected was 95%. Statistical Tools embedded in MS Excel was used to perform this task. Chapter VI discusses the findings from the statistical analysis.

CHAPTER 4

HEURISTIC ALGORITHM

The objective function of an optimization model for the Flying Probe Test is to minimize the total time of the test. Calculate the time it takes a probe to move for one test-step to another would require including other factors such as speed of the probes, time it takes to initialize movement after testing a test-step, etc. Another method to reduce the time is optimizing the total distance followed by the probes. In this analysis, distance represents the most important factor to be analyzed.

A heuristic algorithm is presented to solve the flying probe test for the two-probe machine. The flying probe test requires testing for opens and shorts. In our heuristic approach, one probe is in charge of the opens and the other probe is in charge of the shorts.

The heuristic algorithm consists of three phases. Phase one has the objective function of determine the optimal sequence followed by certain set of nodes. It considers the geometric coordinates of all the nodes to determine the sequence to be followed in the test. Phase two organized all test point in the order determined in phase one. The third phase determines the movement of the probes and calculates the total distance of the test sequence.

4.1 Terminology definition

To understand the following discussion it is necessary to define a set of terminology used.

- *PermNetlist*: The list of networks and the test points (nodes) in each network. A reference designator and their geometric coordinates describe the test points.
- *Netlist*: A copy of PermNetlist used to record changes to the list.
- *Network*: A set of nodes that must be connected via a circuit (net).
- *Node*: A point that the probe makes contact with to test a circuit (test point). The node may correspond to a pad on which a component is connected, be a separate test point added by the designer solely for testing purposes, or be a via point. A via point is a plated thru-hole that connects one layer of the PCB with another layer.
- *Reference designator*: A unique identifier between 1 and the total number of nodes in a network. It is not repeated within a network and it is used to differentiate between nodes of a same network.
- *ProbeBmove*: The list of nodes (one per network) that determine the sequence in which networks are visited. Only one node per network needs to be contacted to test that there is no connectivity between any two networks (This assumes there are no opens in either network). Their geometric coordinates are associated. This list contains all the nodes to be visited by the Probe B.
- *ProbeAmove*: The list of nodes by network to be visited by probe A. This list contains all the nodes that are not included in ProbeBMove. The sequence of the networks is the same determined for ProbeBMove. Their geometric coordinates are associated.

- *Final Test*: The list of all test-steps required for the continuity test of the Netlist. The attributes of the list are sequence order, ProbeANet, ProbeAX, ProbeAY, ProbeBNet, ProbeBX, ProbeBY, Type of test (open or short), IncremDistance and TotalDistance
- *ProbeANet*: Network in which Probe A is located while performing a test-step.
- *ProbeAX*: X coordinate of test point touched by Probe A while performing a test-step
- *ProbeAY*: Y coordinate of test point touched by Probe A while performing a test-step
- *ProbeBNet*: Network in which Probe B is located while performing a test-step.
- *ProbeBX*: X coordinate of test point touched by Probe B while performing a test-step
- *ProbeBY*: Y coordinate of test point touched by Probe B while performing a test-step
- *Open*: Binary variable equals 1 if a test for open is performed.
- *Short*: Binary variable equals 1 if a test for short is performed.
- *IncremDistance*: Distance of the movement of the probes while performing a test-step.
- *TotalDistance*: Accumulated distance of the movement of the probes while performing a test-step.

4.2 Assumptions

The heuristic algorithm makes the following set of assumptions:

- Each network contains two or three nodes. Since the great majority of nets have two or three nodes (Pendurkar et al, 1998), this study focused on these cases.
- Test uses sup norm distance (also called Tchebyshef). This distance is the maximum of the absolute differences in x or y coordinates. Probes move simultaneously in the x and y directions.
- Probes initiate movements in a common point designated as origin. The origin is a point, which geometric coordinators are one unit far from the first node to be visited. This assumption eliminates the possibility of affect the total test time by a long distance between the origin (0,0) and the set of test point, which usually are located close to each other. If the set of test-points are far away from the origin (0,0), the total test time could be highly incremented. In reality, the point of reference (origin (0,0)) to determine the geometric coordinates of the test point does not have to be the same point of reference for the probes to be initialized.
- Every node in the netlist contains known data such as Network number, reference designator and geometric coordinators that are the input of the algorithm. No values are missing in the list.
- Test points to be tested are located in the same layer. The test only considers boards with test points on a single side.
- Probability of probe collision is assuming to be the same for both algorithms to be tested (Proposed and Nearest Neighbor) since collision is out of scope. This assumption keeps collision apart of affect the statistical analysis of the results.

4.3 Algorithm Logic

The goal of this algorithm is reduce the total overall distance followed by the two probes (A and B). Given a certain number of networks and a certain number of nodes by network with its geometric coordinates, find the optimal sequence to test that there are no shorts or opens in the netlist. One of the probes is in charge of testing for opens (Probe A) while the other one is in charge of testing for shorts (Probe B). After determining the optimal sequences of networks, probe B moves among selected nodes while probe A stays in a network and tests for shorts. When both probes are in the same network, the test for opens is performed. In that case, probe B stays in the selected node and probe A moves to the other nodes and performs the test.

4.4 Phase I : Network Sequence Determination

To test for shorts it is necessary to contact two nodes, in which each node is from a different network. The test for shorts will be performed in a sequence. Probe A will contact one network (by touching one node in the network) while probe B contacts all other networks in the Netlist while performing the test for shorts. The objective of Phase I is to identify the test sequence for shorts through the networks. The sequence is followed by probe B. To complete this, it is necessary to select the node from each network that will be contacted. Phase I selects the node per network that will be contacted in the sequence. Also, it finds an optimal sequence of the nodes using a Nearest Neighbor algorithm. Input Data from Netlists to be tested are recorded on a list called PermNetlist.

4.4.1 Logic of Phase I

The logic of the algorithm-phase I algorithm is explained by Figure 5

Figure 5: Logic of Heuristic Algorithm (Phase I)

4.4.2 Mathematical Model Phase I

The mathematical model written for this phase is an effort to explain how the global optimal sequence for the networks could be found.

Variable definition

For the mathematical model, every network is assigned a number between 1 and n , where n is the total number of networks in the netlist. This number was used instead of the net name to track the networks. Here is a definition of the variables:

n : Total number of networks in the netlist

Network (i): Network with tracking number i

X_{ij} : x coordinate of node with reference designator j in network i

Y_{ij} : y coordinate of node with reference designator j in network i

Q_i : Number of nodes in network i

$$S_{i,k} = \begin{cases} 1 & \text{if network } k \text{ follows network } i \\ 0 & \text{otherwise} \end{cases}$$

$$P_{i,j} = \begin{cases} 1 & \text{if node } j \text{ in network } i \text{ will be visited during the sequence route} \\ 0 & \text{otherwise} \end{cases}$$

Also let's consider the set N defined as the group of networks in the Netlist.

Objective Function

The objective function of the mathematical model is to find the value of S and P that minimizes total distance. S is the variable that determines the sequence of the networks. P is the variable that denotes which node of the network is visited during the sequence. The equation is zero unless S and the two values of P are not zero. This condition is only truth when S relates two networks (i and k) and network k follows network i . Also, it is necessary that node j in network i , and node m in network k are the ones visited in the sequence. If these three values are truth (variables equal 1) then the distance calculated in the max function is added to the summation.

The four summation functions ensure all possible combinations are considered. First summation considers all networks to be in the exit position, the second summation consider all networks to be in the enter position. Third summation considers all nodes from exit network as the node visited in the sequence, and finally the fourth summation ensures all nodes from enter network are considered as the node visited in the sequence. The number of nodes per network is not a constant. Q determines the total number of nodes by network and that is why the last two summation function varies between one and Q .

$$\text{Min } Z = \sum_{i=1}^{n-1} \sum_{k=1, k \neq i}^n \sum_{j=1}^{Q_i} \sum_{m=1}^{Q_k} \max(|x_{ij} - x_{km}|, |y_{ij} - y_{km}|) s_{ik} p_{ij} p_{km} \quad (4)$$

Constraints:

The decision choices of this mathematical model are limited by the fact that every network must be visited only once and that only one node per network is selected in the sequence. Variable S controls the first limitation while variable P controls the second one.

$$\sum_{i=1}^n S_{i,k} = 1 \quad \forall k \in |N| \quad (5)$$

This constraint ensures that all networks are entered only once

$$\sum_{k=1}^n S_{i,k} = 1 \quad \forall i \in |N| \quad (6)$$

This constraint ensures that all networks are exited only once

$$\sum_{j=1}^{Q_i} P_{i,j} = 1 \quad \forall i \in |N| \quad (7)$$

This constraint assures that only one node by networks is visited during the sequence.

$$S_{ik}, P_{ij} = 0 \text{ or } 1 \quad (8)$$

Finally, the last constraint ensures that variables S and P are binaries.

The objective function of the model is non-linear because it involves products of decision variables. Consequently, the above optimization model is a non-linear program (NLP).

Computational Complexity

This model is an exhaustive search that considers all possible sequence of networks. Also, the model considers the selection of any node within a network. In each iteration, the model calculates the Tchebyshef distance from the current node to every other possible node to be selected.

Starting at the origin, the first iteration makes as much calculations as the total number of nodes in the netlist trying to find the closest node. The total number of calculations is:

$$\sum_{i=1}^n Qi = \text{Total number of nodes in the Netlist} \quad (9)$$

The second iteration calculates the distance from the selected node of the first network of the sequence to all other nodes of networks that has not been included in the sequence. The number of calculations of the second iteration is:

$$\sum_{i=2}^n Qi = \text{Total number of nodes in the} \quad (10)$$

The total number of calculations can be represented as:

$$\sum_{i=1}^n Qi + \sum_{i=2}^n Qi + \sum_{i=3}^n Qi + \dots + \sum_{i=n-1}^n Qi \quad (11)$$

$$\sum_{a=1}^{n-1} \left(\sum_{i=a}^n Qi \right) = \text{Total number of calculations} \quad (12)$$

The total number of calculations may not be high when the Netlist contains a small number of networks. However, it is a characteristic of the netlist to have a big number of networks. It makes impossible the exhaustive search to solve the mathematical model and find the global optimal. This model is considered NP-Hard.

4.4.3 Phase I Example

As an example of the approach of this phase, lets consider a Netlist with data recorded as in Table 1. This example uses integer values as geometric coordinates. This will help to better understand the graphic representation of the Netlist.

Table 1: Netlist Data Example for Phase I

Network	Ref. Designator	X Coord.	Y Coord.
152	2	10	6
152	1	5	2
232	3	12	8
232	2	15	3
232	1	8	3
321	2	6	1
321	1	1	1
401	3	7	8
401	2	3	8
401	1	3	5
515	2	18	2
515	1	14	7
648	2	22	10
648	1	16	9
723	3	22	2
723	2	23	7
723	1	20	7

A graphic representation of the netlist is shown in Figure 6. The lines represent the networks in the netlist and the larger numbers represents their names. The black points represent the nodes (test points) and the smaller numbers are the reference designators.

Figure 6: Graphic Representation of Netlist Input Data Example

Phase I starts by selecting the closest node to the origin. This node is the first one in the sequence and its network is the first network to be visited. Attributes associated to the node are recorded on list called ProbeBMove. Since the goal of this phase is to select one node per network in the sequence and one node from the network has been already selected, no other node from the network will be required to be contacted. Subsequently all other nodes in the same network are deleted from the netlist so they are not chosen during the subsequent selection. In the example, the closest node to the origin is node 1 from network 321. The node was found calculating the Tchebyshef distance from the origin to all other nodes in the Netlist. The selected node becomes the first node of the short test sequence. Network 321 is the first network visited and the node 2 of the same network (321) is deleted from the temporary netlist. Figure 7 illustrates the first step of this phase.

Figure 7: Graphic Representation of Netlist with First Network Selected

The next step in the algorithm is to find the closest node and insert it in the sequence. Tchebyshef distance is calculating from the last node visited to all other nodes. There are two candidates at equal distance from the last selected node. The first one is node 1 from network 152, which is located 4 units far. The second node located at the same distance is node 1 from network 401. Selecting one of the nodes over the other is arbitrary. The selected node is included in ProbeBMove list and all other nodes from the selected network are deleted from the netlist.

This process continues until all networks are visited (See Figure 8). The number of nodes visited in the sequence must be equal to the number of networks in the netlist. That is due to the fact that one node per network is visited in the sequence. The results of this phase are input for phase III.

Figure 8: Netlist Examples with All Networks Selected in The Sequence

4.5 Phase II: Nodes Sequence

Phase I determined the test sequence for shorts between networks. Phase II organizes all the nodes that were not selected in Phase I using the same sequence of networks of Phase I. The goal of this phase is build a list that includes all the nodes that will not be contacted by Probe B while testing for shorts. The new list called ProbeAMove includes all the nodes that will be contacted by probe A while testing for opens. The test for opens in the netlist is performed in the same sequence of networks determined in phase I. In the test, Probe B contacts the node of the network included in the ProbeBMove list and Probe A contacts all the other nodes of the network. Phase II does not optimize any sequence because the sequence was already found in phase I. Instead, it orders the nodes that were not selected in phase I in a list using the optimized sequence of networks found in phase I.

4.5.1 Logic of Phase II

The logic of the algorithm-phase I algorithm is explained by Figure 9

Figure 9: Logic of Heuristic Algorithm (Phase II)

4.5.2 Phase II Example

Every network has two or three nodes. To test for opens, it is required that both probes are located in the same network. The approach of the algorithm is that all the test for open in a network are performed sequentially. Once a pair of nodes from a network is tested for open, all other pairs in the network will be tested. When a network has two nodes every probe contacts one of the nodes and the test is performed. When a network has three nodes, one of the probes needs to contact two nodes while the other remains in the third node. For networks with a high number of nodes, an optimization of the sequence in which the test for open is performed would represent a significant reduction in total distance. However, the maximum number of nodes that one probe visit during the test is two (for those networks with three nodes) and based on the characteristic of the PCB design (nodes in a network are located close to each other) it is not necessary an optimization of the sequence of nodes not included in Phase I. If it is assumed that nodes in a network are located close to each other and an optimization of the sequence of the networks was found in phase I, it can be induced that by organizing the nodes that were not selected in phase I by network sequence, an optimal route is also found. All this values are saved on a list called ProbeAMove and include all nodes that were not selected in phase I. The total number of nodes included in this new list is equal to the total number of nodes in the netlist minus the nodes included in ProbeBMove (n).

As an example of the approach of this phase, lets consider the same example of Phase I. The results from phase I were recorded in a list called ProbeBMove. Table 2 shows the results and the sequence in which network are visited in the sequence.

Table 2: ProbeBMove List Example for Phase II

Sequence	Network	Ref. Designator	X Coord.	Y Coord.
1	321	1	1	1
2	401	1	3	5
3	152	1	5	2
4	232	1	8	3
5	515	1	14	7
6	648	1	16	9
7	723	1	20	7

Considering the sequence of the network, the first one visited is network 321. Node 1 is already selected for ProbeBMove so only node two has not been selected. Node 2 from network 321 becomes the first node in the list ProbeAMove. The second network in the sequence is 401. Nodes 2 and 3 from that network have not been selected so they become the second and third node in ProbeAMove. This process continues until all remaining nodes are in the ProbeAMove list. Figure 10 shows the Netlist example with all the nodes selected for ProbeAMove and the sequence.

Figure 10: Netlist Example with All Nodes Selected in ProbeAMove List

The blue points represent the nodes selected for ProbeBMove in phase I. The remained nodes are visited in this second list and the sequence is shown with red lines. These values are included in the ProbeAMove list.

Table 3: ProbeAMove List Example for Phase II

Sequence	Network	Ref. Designator	X Coord.	Y Coord.
1	321	2	6	1
2	401	2	3	8
3	401	3	7	8
4	152	2	10	6
5	232	3	12	8
6	232	2	15	3
7	515	2	18	2
8	648	2	22	10
9	723	2	23	7
10	723	3	22	2

Since this phase does not include any optimization, a mathematical model is not required. However, if the number of nodes in the network were larger, an optimal sequence of those nodes would reduce the total test distance. The mathematical model would be then a TSP model where every network would be treated as an independent sequence. The nodes would represent the cities and the node selected in Phase I would not be part of the sequence.

4.6 Phase III

Phase III uses the results from phase I and II to route the probes and make the test. The nodes included in the lists built in phase I and II are the nodes each probe will contact. In the case of Probe B, it moves among the nodes in its list testing for shorts. This movement is repeated forward and backward many times until all tests are performed. In the case of Probe A, it moves among the nodes in its list testing for opens. It only moves until the end of the list. When it gets to the end of the list, all the test has been performed.

4.6.1 Logic of Phase III

The logic of the algorithm-phase I algorithm is explained by Figure 11

Figure 11: Logic of Heuristic Algorithm (Phase II)

4.6.2 Phase III Example

Figure 12 shows the netlist example discussed in phase I and II. Blue nodes represent the nodes in the ProbeBMove list (just one per network) while the orange ones represent the nodes in ProbeAMove List (the remaining nodes).

Figure 12: Netlist Example with Nodes Selected for ProbeAMove List and ProbeBMove List

Basically, both nodes start at the origin and move to the first network to be tested (network 321). Probe B moves to the first node in ProbeBMove list while probe A moves to the first node in ProbeAMove list. Since both probes are in the same network, the test performed between the two nodes is for open. Incremental and total distance of the movement is calculated. If the first network has more than the two nodes already tested, probe A moves to third node while Probe B remains at its node. Again, a test for open would be performed. In the example, the first network of the sequence has only two nodes, then only one test-step for open is required. Figure 13 shows the first step. The distance of the step is the highest one of

the distance traveled by each probe. In this case, probe A travels the highest distance which is 6 units. That is the incremental distance of the step and the total distance is also 6 units since no other test-step has been performed.

Figure 13: First Test-Step of Netlist Example

After the first network is tested for opens, it is test that there is no connectivity between the first network and the other networks in the netlist (test for shorts). To do this task, probe A remains in the last node it visited in the current network while probe B starts moving in the sequence of ProbeBMove list. Every time a movement is executed by probe B a test for shorts is performed. Probe B moves until the last node in its list at which point all the test-steps that involved first network has finished. Figure 14 illustrates the test-step for shorts between the first network and all other networks.

Figure 14: Test-Steps for Shorts from First Network of Netlist Example

The next step is performing all the test-steps that involve the second network (network 401). Probe A moves to the next node in its list that is located in the second network, while probe B stays at the end of its list. At this point it is test that there is no connectivity between the second network and the last network where probe B is located. Probe B moves backward in its list, node by node, while perform test for short until it meets the network where probe A is located. When they meet, test for opens are performed in that network (network 401). At this point all the test-steps that involved second network have finished. Both probes move then to the third network and the process is repeated until all test-step are performed. At the end of the test both probes are located at the last network. Every test-step has an incremental distance and a total distance. See Table 4 for a list of all the test-steps of the Netlist example with the type of test performed and the incremental and total distance.

Table 4: Final Test Sequence for Netlist Example

	Probe A			Probe B			Type of Test		Distance	
Seq	Net	X	Y	Net	X	Y	Open	Short	Increm.	Total
1	321	6	1	321	1	1	1	0	6	6
2	321	6	1	152	5	2	0	1	4	10
3	321	6	1	232	8	3	0	1	3	13
5	321	6	1	401	3	5	0	1	5	18
6	321	6	1	515	14	7	0	1	11	29
7	321	6	1	648	16	9	0	1	2	31
8	321	6	1	723	20	7	0	1	4	35
9	152	10	6	723	20	7	0	1	5	40
10	152	10	6	648	16	9	0	1	4	44
11	152	10	6	515	14	7	0	1	2	46
12	152	10	6	401	3	5	0	1	11	57
13	152	10	6	232	8	3	0	1	5	62
14	152	10	6	152	5	2	1	0	3	65
15	232	15	3	232	8	3	1	0	5	70
16	232	12	8	232	8	3	1	0	5	75
17	232	12	8	401	3	5	0	1	5	80
18	232	12	8	515	14	7	0	1	11	91
19	232	12	8	648	16	9	0	1	2	93
20	232	12	8	723	20	7	0	1	4	97
21	401	3	8	723	20	7	0	1	9	106
22	401	3	8	648	16	9	0	1	4	110
23	401	3	8	515	14	7	0	1	2	112
24	401	3	8	401	3	5	1	0	11	123
25	401	7	8	401	3	5	1	0	4	127
26	515	18	2	515	14	7	1	0	11	138
27	515	18	2	648	16	9	0	1	2	140
28	515	18	2	723	20	7	0	1	4	144
29	648	22	10	723	20	7	0	1	8	152
30	648	22	10	648	16	9	1	0	4	156

CHAPTER 5

SYSTEM DESIGN AND IMPLEMENTATION

This chapter presents the system design and related issues encountered when implementing the algorithm in the prototype system. Described here is the data collection , system design , input and output model data.

5.1 Data Collection

In order to meet the stated purpose of this research it was necessary to become familiar with the type of data that needs to be processed. Moreover, to understand the data available, the transformation the data requires to become an input of the system and the information required as output.

The first step to address these issues was to become familiar with the list companies use to define the interconnections of test points in a PCB (Netlist). Gables Engineering provided the Netlists required for this research.

Two different decisions had to be made regard the Netlists to be selected for this study. First, an analysis of the data included in the Netlist to determine what information will be used and if any transformation of the data is required. Second, select the sizes of the Netlists to be tested. The number of networks and test points defines the size of the Netlist.

5.1.1 *Netlist*

A netlist is a list of networks and the test points (nodes) in each network. Each row of the Netlist defines a test point with fields for Net Name, Type, Layer, X

Coordinate and Y Coordinate. “Net Name” is a number used to define the network. “Type” indicates if the pad is a via or a thru hole pad for a component. “Layer” specifies in which layer the test point is located, if the layer is minimum it refers to the top of the board and maximum to the bottom. X is the x coordinate of the test point and Y is the y coordinate of the test point. Figure 15 shows a portion of a netlist⁴.

Net Name	Type	Layer	X	Y
-----	----	-----	-	-
\$8008	Via	(Maximum)	13.635	13.135
\$8008	Via	(Maximum)	13.375	12.885
\$7713	Via	(Maximum)	13.580	13.135
\$7713	Via	(Maximum)	13.350	12.835
\$7717	Via	(Maximum)	13.690	13.135
\$7717	Via	(Maximum)	13.400	12.835
\$7722	Via	(Maximum)	13.505	12.885
\$7722	Via	(Maximum)	13.910	13.135
\$7907	Via	(Maximum)	10.704	10.740
\$7907	Via	(Maximum)	10.975	11.650
\$7911	Via	(Maximum)	10.975	11.450
\$7911	Via	(Maximum)	10.861	10.740
\$7908	Via	(Maximum)	10.782	11.110
\$7908	Via	(Maximum)	10.975	11.600
\$7909	Via	(Maximum)	10.783	10.740

Figure 15: Netlist from PCB Industry

For the purposes of this research, it is necessary to know all the test points and their geometric coordinates. Type and Layer information are not used. . All the test point needs to be contacted no matter the type of test point. That is why type information would not be used. At the same time, the scope of this study only includes single-sided board; subsequently, the layer will always be the same within a design to be tested.

⁴ Netlist from Gables Engineering, 2001

5.1.2 Netlist Size Selection

The selection of Netlist sizes for the experiment needed an analysis of two factors. First, an evaluation of the specifications of the flying probe machine⁵. This evaluation would give an estimation of the total test time expected based on the number of nets in the netlist. Second, consider the average sizes used by the company that provided the Netlist sample (Gables Engineering).

The general specifications of the machine show that up to 20 test-steps can be performed in one second. Considering this information and assuming that all nets in the Netlist have only two test points, the total test time for different Netlist sizes was estimated as shown in Table 5.

⁵ Javelin™ 1004 Flying prober. Teradyne, 2000

telephones and beepers; this new production environment made this research to included netlist sizes up to 200 nets.

To determine the range of netlist sizes of this study, it was also necessary to consider the averages sizes used by the industry. Gables Engineering creates many different sizes of panels. Panels usually have a Rear Connector PCB, Power Supply PCB, Digital PCB, Front Connector PCB and Lighting PCB. They may also require boards for switches which boards would be as tiny as less than an inch square. Most rear connectors, power supply and front connectors have an average of 100 to 150 nets while digital have an average 300 to 350 nets. These two factors (test time based on number of test-points and averages of industry) determined the selection of the range for the study. It was selected ranges between 0 to 350 networks, which represent in average, Netlists with up to 700 test points. Table 6 shows the netlist sample selected for testing.

Table 6: Netlist Sample for Testing

Design Number	Number of Networks	Number of Test-Points
1	27	70
2	43	115
3	75	168
4	83	211
5	129	330
6	149	306
7	203	408
8	251	492
9	258	532
10	283	624

5.2 Model Input Data

The function of this section is to gather the data required as input by the system (solution algorithms). The analysis of the Netlists concluded that only Net Name, X and Y coordinates of every test point are required for our algorithm. However, it was necessary to have a reference number for test point in a same net that would help for a better handling of the data.

Every test point in the netlist has multiple values associated to it (Net Name, X and Y coordinates). Manual input of the data is time consuming. Microsoft Excel was used as an intermediary between the ASCII output file containing the Netlist and the implementation of the heuristic algorithm. Every data from a test point (including Type and Layer) is located in a cell. Columns would reference the attributes of the test point while rows reference single test points. Since Type and Layer are not input for the system, this values are deleted from the table and the fourth new attribute is added using Microsoft Excel functions (Counts). The fourth value is a count of test point in a same net. It is called reference designator. The reference designator is a number between 1 and the total number of test point in a net (Q_i). The function of this new value is to reference the test point within a network. Table 7 shows the transformation of the Netlist from PCB Industry to the input data of the algorithm.

Table 7: Netlist Example from PCB Industry converted as Input Data

Network	Ref. Designator	X Coord.	Y Coord.
8008	2	13.635	13.135
8008	1	13.375	12.885
7713	2	13.58	13.135
7713	1	13.35	12.835
7717	2	13.69	13.135
7717	1	13.4	12.835
7722	2	13.505	12.885
7722	1	13.91	13.135
7907	2	10.704	10.74
7907	1	10.975	11.65
7911	2	10.975	11.45
7911	1	10.861	10.74
7908	2	10.782	11.11
7908	1	10.975	11.6

Another important consideration in this data analysis is the units of the geometric coordinates. Frequently Netlists use inches as units of reference. In some other cases, hundreds or thousands of inches are used as units. The accuracy of the system is very important since the optimality of the solution is based on the total distance. For this reason, thousands of inches was selected as the units to be used for the system. Microsoft Excel was also used to make the appropriate transformation of the units when Netlists does not meet this requirement.

5.3 Prototype System

The prototype system used MS Access and Visual Basic as tools to implement the algorithm. As discussed in Section 3.3, MS Access was used to store the input and output

data and all the list that interact while solving the algorithm. The system was built using ActiveX Data Objects (ADO). It was designed to allow programmers to access the Jet database engine, as supplied with Microsoft Access (Wright's, 1998). It is one of the most popular methods of accessing database. Visual Basic shipped with the Jet database engine, which allowed creating the database.

The database development can be split in two sections:

- a) Creating the database itself: this section included creating the tables. Tables are used to records data related to specific list (e.g. Netlist table, ProbeAMove table, etc). Specific tables also record the solution of the sequence of the test (e.g. FinalTest table) or the value of total distance, complexity and run time (Output table).
- b) Creating the front-end: This is the visible part of the application, what the users see. The front end has six functions controlled by six different buttons (See Figure 16).

Figure 16: Initialization Form for Prototype System

Clean tables: The function of this button is to clean all the tables in the database. Tables may include data from the last netlist tested. Records in all tables of the database are deleted.

Data Input: The function of this button is to record the input data of the system. It opens the Netlist table and the values are inserted. Since the range of values is usually big, they are copied from the Excel spreadsheet.

Run Algorithm: This button runs the algorithm. It creates the values of the tables based on the results of the logic of the algorithm. Appendix 11 shows the Visual Basic code for the Heuristic Algorithm proposed. Appendix 12 shows the Visual Basic code for the Nearest neighbor approach.

Final Test Sequence: This button opens the FinalTest table. This table records the sequence of the test by test-step. It includes the position of probe A and B, type of test and distance.

Output: The function of this button is to open the Output table, which has recorded the Total distance of the test, complexity, and Run time. These values are used for the statistical analysis.

Exit: This button closes the database.

5.4 Model Output Data

The function of this section is to gather the data required as output by the system (solution algorithms). The goal of this research is to determine the sequence to be followed by the probes. Therefore, the output of the system must contain a detail list of the sequence of the probes test-step by test-step and the type of test performed (open or short). Every test-step must show the geometric coordinates for each probe. Also, for a better understanding of the sequence and a validation of it, it was included the name of the net contacted by probe. Table 8 illustrates an example of a few rows of an output of the system. The first column denotes the sequence number of the test-step. The next set of columns is the output related to Probe A. The values show the net name, and the geometric coordinate to which probe A must be routed. The same set of outputs is related to probe B. The logic of the test must ensure that the probes will never touch the same set of test points. The following two columns define the type of test to be performed. Only one of the options can be truth and it is represented with a number 1. By definition of

opens and shorts (Prasad, 1997) when the probes are located in a same net, the test perform is for opens, and when they are in different nets, the test perform is for shorts. Finally, the last two columns show the incremental distance and the total distance by test-step. Appendix 13 shows the Final Test list for a sample netlist of 27 networks and 70 test-points using the Heuristic algorithm proposed. Appendix IV shows the Final Test list for the same sample using the nearest neighbor.

Table 8: Output Example of Netlist from PCB Industry

Seq	Probe A			Probe B			Type of Test		Distance	
	Net	X	Y	Net	X	Y	Open	Short	Increm.	Total
1	7911			7911			1	0	1	1
2	7911			7907			0	1	3	4
3	7911			7722			0	1	3	7
5	7911			7717			0	1	2	9
6	7911			7908			0	1	4	13
7	7911			8008			0	1	3	16
8	7911			7713			0	1	2	18
9	7907			7713			0	1	3	21
10	7907			8008			0	1	2	23
11	7907			7908			0	1	5	28

In addition to the list of the sequence of the test, three variables are recorded by netlist tested. These values depend on the algorithm used to solve the sequence and relate to the optimality and complexity of the algorithm.

CHAPTER 6

STATISTICAL ANALYSIS OF ALGORITHM

This chapter presents the results of the experiment performed with the prototype system presented in Chapter 5. The sample PCBs used were actual designs obtained courtesy of Gables Engineering in Coral Gables, Florida.

6.1 Outputs and Results

The factors analyzed in this experiment were total test distance, complexity and total run time. The ten Netlist samples (Appendix A) of this study were run using the two different approaches: The proposed algorithm and the Nearest Neighbor. The results of each run were recorded in the database.

6.1.1 Test-Steps

The number of test-steps for a Netlist depends on the number of nets and the total number of test points in the Netlist. Table 9 shows the total number of test-steps for each Netlist. Note that the number of test-steps for a Netlist is independent of the approach used to generate the test sequence.

Table 9: Number of Test-Steps by Netlist Sample

Sample	Number of Test Points	Number of Nets	Number of Test-Steps
1	70	27	394
2	115	43	975
3	168	75	2868
4	211	83	3531
5	330	129	8457
6	306	149	11183
7	408	203	20708
8	492	251	31649
9	532	258	33169
10	624	283	40244

6.1.2 Distance

As discussed in chapter 4, the goal of the proposed algorithm is to find a sequence followed by the probes that reduces overall test distance. The final distance followed by the probe for every tested netlist represent the most important factor of analysis. The total test distance calculated using the two algorithms of the study is found in Table 10.

Table 10: Total Test Distance by Sample using Nearest Neighbor Algorithm and Proposed Algorithm.

	Distance		
Sample	Nearest Neighbor (A)	Proposed (B)	Difference (A-B)
1	408811	210867	197944
2	1779966	881791	898175
3	2741691	1047983	1693708
4	4709626	1805521	2904105
5	12742671	3891371	8851300
6	20928932	6514547	14414385
7	29267699	6611081	22656618
8	19137551	5625483	13512068
9	33924481	9730541	24193940
10	68325809	14271123	54054686

Table 10 clearly shows that the total test distance found by the proposed algorithm is less than the distance found by the nearest neighbor algorithm. However, a statistical analysis needed to confirm this hypothesis. The results of this analysis are shown in Section 6.2.

6.1.3 Computational Complexity

The second factor to be analyzed in this study is the complexity of the algorithms. The complexity represents the number of calculations performed by the algorithms while optimizing the sequence. This value was calculated using a count in the function that calculates any distance in the algorithms. The complexity is proportional to how exhaustive the search is. The results of running the algorithms are found in Table 11.

Table 11: Computational Complexity by Sample using Nearest Neighbor Algorithm and Proposed Algorithm.

Sample	Complexity	
	Nearest Neighbor	Proposed
1	9328	940
2	37459	2506
3	160037	6287
4	246664	8748
5	905023	20847
6	1157254	23124
7	2834889	41562
8	5604495	66381
9	5866698	68147
10	8374428	87849

Figure 17: Complexity of Nearest Neighbor Algorithm Vs Heuristic Proposed

Evidently the complexity of the proposed algorithm is, in average, 95% less than the Nearest Neighbor. The high number of calculations of the Nearest Neighbor approach is due to the fact that the algorithm optimize the sequence of all the test points to be visited by the probes while testing for shorts and opens. Every time a probe needs to make a movement, the algorithms search for the nearest neighbor. To find this nearest neighbor, the algorithm needs to calculate the distance between the current test point (where the probe is located) and all others possible next test point. Instead, the proposed algorithm finds a sequence that reduces the overall distance of only one of the probes (probe B). The other probe (probe A) only follows the same sequence of nets found for probe B.

6.1.4 Run Time

The last factor to be analyzed in this study is the run time of each algorithm. Run time of any program is very important because it can reduce the efficiency of any optimization method. If run time of a program was too high, it would not be used no matter how good it was. Companies are looking for a good and “*fast*” answer instead that a very good but slow answer. The results of run time for the algorithms are found in Table 11.

Table 12: Run Time by Sample using Nearest Neighbor Algorithm and Proposed Algorithm.

Sample	Run Time (Sec)	
	Nearest Neighbor	Proposed
1	0.1250	0.0833
2	0.2917	0.0833
3	0.7917	0.7500
4	1.2083	0.2083
5	3.8333	0.5417
6	4.7917	0.5000
7	11.2083	0.7083
8	21.1250	1.1250
9	21.7500	1.0417
10	30.9167	1.3750

As seen in Table 12 run time for both approaches is very low. None represents a disadvantage for their algorithms. The maximum run time obtained was 30 seconds for a Netlist with 624 test points.

Figure 18: Run Time of Nearest Neighbor Algorithm Vs Heuristic Proposed

Computational complexity and run time grow exponentially as problem size grow (See Figure 17 and Figure 18). The results of these two factors for the proposed algorithm are very low compared with the Nearest Neighbor approach. However, the difference does not show a representative advantage for any of the algorithms presented. The third factor, distance, does show a significant difference. The following section makes a statistical analysis to discuss the difference of distance between the algorithms.

6.2 Statistical Analysis

The two algorithms, heuristic proposed and nearest neighbor, were used to measure the total test distance of the flying probe test. The results of the test were analyzed to test the hypothesis that the mean of the difference μ_d is zero. The test procedure consists of obtaining the difference of the pair of observation on each of the n specimens (Ayyub, 1997). Each of the n specimens corresponds to one sample Netlist. A

t-paired test was used to compare the difference of the pair of observations. For this comparison, it was assumed that the two algorithms were independent and that difference of pair observations followed a normal distribution. The null hypothesis is:

$$\mathbf{H_0: \mu_d = 0} \qquad \text{null hypothesis} \qquad (13)$$

The null hypothesis states that there is no statistical difference between the two algorithms.

$$\mathbf{H_A: \mu_d < 0} \qquad \text{alternative hypothesis} \qquad (14)$$

The rejection of the hypothesis would conclude that the total distance of the proposed algorithm is less than the total distance of the nearest neighbor.

The first step in this analysis is to verify the assumption of the normality of the difference of pair observations. The t-paired test requires an assumption of normality for the sampled population. Normal probability plots were used to check the normality assumption. Probability plotting is a graphical method for determining whether sample data conform to a hypothesized distribution based on a subjective visual examination of the data. A poor fit of the data to the line of the plot would indicate that either the assumed distribution is not the population or that the sample statistics are poor estimators of the population parameters. If the plot does not reflect that data is normally distributed, the values needs to be transformed

In order to select the best set of data for the statistical analysis, three different plots were analyzed to select the one the better follows a normal distribution. The three plots are : a) difference of the pair of observations, b) difference of the pair of observations / number of test points, c) logarithm of difference of the pair of observations. Values used for the plot are found in Table 13

Table 13: Calculated Values for Normal Probability Plots

Sample	Distance by Nearest Neighbor (A)	Distance by Proposed (B)	Difference (A-B)	Difference/ number of test points	Log (Difference)
1	408811	210867	197944	2828	5.297
2	1779966	881791	898175	7810	5.953
3	2741691	1047983	1693708	10082	6.229
4	4709626	1805521	2904105	13764	6.463
5	12742671	3891371	8851300	26822	6.947
6	20928932	6514547	14414385	47106	7.159
7	29267699	6611081	22656618	55531	7.355
8	19137551	5625483	13512068	27464	7.131
9	33924481	9730541	24193940	45477	7.384
10	68325809	14271123	54054686	86626	7.733

Figure 19: Normal Probability Plot for difference of the Pair of Observations

**Figure 20: Normal Probability Plot for Difference of the Pair of Observations /
Number of Test Points**

Figure 21: Normal Probability Plot for Logarithm of Difference of the Pair of Observations

The first plot does not represent a good option for the test because two points (first and last one) are located far from the line. Also, the points tend to situate up to the line. This set of values is rejected. The second plot groups the point closer to the line but still the first point is located far from the line. In this case, there is also an accumulation of point up to the line. The third plot clearly satisfies the assumption of normality. A logarithmic transformation best meets the requirement of normality for conducting the paired t-test.

Now it is possible to test the null hypothesis (equation 13). The confidence coefficient selected was 95%. The test statistic is calculated as,

$$t_o = \frac{d}{S_d \sqrt{n}} \quad (15)$$

where d is the average of the difference of the pair of observations. The standard deviation is calculated as,

$$S_d^2 = \frac{\sum D^2 - \frac{(\sum D)^2}{n}}{n-1} \quad (16)$$

where D represents the difference for any pair of observations. The null hypothesis, H_0 :

$\mu_d = 0$ is rejected if $|t_0| > t_{\alpha, n-1}$.

Table 14: Paired Distance Data

Sample	Distance by Nearest Neighbor	Distance by Proposed	Difference ¹⁰	Difference ²
1	408811	210867	5.297	28.053
2	1779966	881791	5.953	35.443
3	2741691	1047983	6.229	38.798
4	4709626	1805521	6.463	41.771
5	12742671	3891371	6.947	48.261
6	20928932	6514547	7.159	51.248
7	29267699	6611081	7.355	54.099
8	19137551	5625483	7.131	50.847
9	33924481	9730541	7.384	54.519
10	68325809	14271123	7.733	59.797
Sum			67.650	462.836
Average			6.765	46.284

¹⁰ The value used in the difference is the Logarithm of the distance due to the required transformation

$$S_d^2 = \frac{\sum D^2 - \frac{(\sum D)^2}{n}}{n-1} = \frac{462.836 - \frac{67.650^2}{10}}{10-1} = 0.576$$

$$S_d = \sqrt{0.576} = 0.758$$

$$t_o = \frac{6.765}{0.758\sqrt{10}} = 2.81$$

Choosing $\alpha = 0.05$, it is found $t_{0.025,9} = 2.262$. Since $t_o = 2.81$ is greater than $t_{0.025} = 2.262$ we reject the null hypothesis. Rejection of the null hypothesis provides strong evidence the two algorithms differ in their mean distance. Since the average distance of the proposed heuristic algorithm is less than the average distance of the nearest neighbor we conclude the heuristic algorithm will generally find a shorter test sequence.

CHAPTER 7

CONCLUSION

The three factors discussed (total distance, computational complexity, and run time) and the statistical analyses demonstrate that the proposed heuristic algorithm finds a better test sequence than nearest neighbor does for the flying probe test. The outputs of each sample)were compared to find that there is a significant reduction of the total distance by using the proposed heuristic. Figure 22 shows the reduction of the total test distance by sample. This chart clearly illustrates that the reduction of the distance by the proposed algorithm is on average 66% when compared to a nearest neighbor algorithm.

Figure 22: Total Distance of Nearest Neighbor Algorithm Versus Heuristic Proposed

The reduction of the distance is based on the movement of the probes. The movement of the probe B through nets is optimized while probe A always moves trying to reach probe B. The route was found by optimization of the sequence of nets and then the movement of the probes is also optimized. The sequence found by the proposed algorithms is not a global optimal. Finding a global optimal would require an exhaustive search that is impossible due to the high number of test points in the Netlist. However, the test points within nets are located so close that it allows to determine the sequence of the test based on a sequence of nets.

The reduction of distance, computational complexity and run time by sample increases with the number of test points in the netlist. However, as discussed in section 5.1.2, to test a netlist with more than 250 nets would take so much time that it would make the manufacturing time very high. This type of test (high number of nets) would only be good for prototyping.

This study reiterates the importance of the use of operation research in many areas including manufacturing. Optimization methods open up the opportunity to reduce cost, time and increase production by finding optimal ways of performing a task. In this case, operation research was used to improve the test time of an electronic test. The test time of the flying probe test is limiting this new technique of become very useful and highly productive.

This study analyzed the flying probe test and found improved test sequences. However, a complete study would include many other aspect not analyzed in this study such as speed of the probes, time it takes to initialize movement after testing a test-step, and collision of the probes. Also, other methods of optimization can be used to optimize

the sequence of the nets. The algorithms used in the study did not have high computational time. A more exhaustive search of the sequence could find a better solution for the sequence and still not represent a disadvantage for its computational time.

REFERENCES

- Ayyub, B., McCuen, R. 1997. probability, Statistics, & Reliability for Engineers. CRC Press LLC, Boca Raton, Florida.
- Bouskela, E. 1999. Low-Cost ATE in a High-Mix Environment. *Evaluation Engineering*. August 2000.
- Bouskela, E. 2000. Bringing flying probers into the production line. *Electronics Manufacture and Test*. March 2000.
- Castillo, J. 1999. A heuristic for the traveling salesman problem based on a continuous approximation. *Transportation Research*. Part B, p. 123-152.
- Chou, N.,C. Cheng. 1993. Optimal test size and efficient probe routing for substrate verification using two-probe testers. *Proceeding of International Symposium of Hybrid Microelectronics*, p. 276-281.
- Chung, H., J. Norback. 1991. A clustering and Insertion heuristic applied to a large routing problem in food distribution “. *Journal of the Operational Research Society*, 42, p. 555-564.
- Crowell, J.C., R.J. Keogh, J.A. Conti, 1984. Moving probe Bare Board Tester Offers Unlimited Testing Flexibility. *Industrial Electronic Equipment Design*. September, p. 473,484.
- Danusaputro, S., C. Lee, L. Martin-Vega, 1990. An efficient algorithm for drilling Printed circuit board. *Computers and Industrial Engineering*, 18, p.145-151.
- Glover, Fred. Tabu Search: A Tutorial, 1990. *Interfaces*. Vol 4, p. 74-94.
- Hassig, R.A. 1998. A look at flying probe test (PCB test). *Circuit Assembly*, Vol. 9,no.10 p. 40-43.
- Jünger, M., G. Reinelt and G. Rinaldi 1996, "The Traveling Salesman Problem", in: M. Ball, T. Magnanti, C. L. Monma and G.L. Nemhauser (eds.), *Handbooks in Operations Research and Management Sciences: Networks*, North-Holland
- Kim, B.C., P. Jiang, S. Park, 1999. A Probe Scheduling algorithm for MCM Substrates. *Proceedings of IEEE Computer Society International Test Conference (ICSM'99)*. Atlantic City, NJ, USA, p. 31-37.
- Kovartovsky, R. 1999. Vertical and horizontal integration in electrical testing. *CircuiTree* Vol.12, no.9, p. 116-122.

- Kozicki, M. 1992. Modern Electronic handbook: an overview. Van Nostrand Reinhold, New York.
- Litke, J.D. 1984. An Improved solution to the traveling salesman problem with thousands of nodes. *Communs ass. Comput. Mach.* Vol 27, p. 1227-1236.
- McAvinue, B. 1999. Test strategies for complex electronic assembly: evaluating and implementing flying probe technology. Proceedings of Technical Program, SMTA International, p.509-519.
- Nemhausen, G.,W. Laurence, A. Wolsey. 1988. Integer and combinatorial optimization. Wiley, New York.
- Pendurkar, R., C. Tovey, A. Chatterjee. 1999. Single-probe traversal optimization for testing of MCM substrate interconnections. *IEEE Transactions on Computer-Aided Design of integrated Circuits and Systems*, vol.18, no. 8 p. 1178-1191.
- Rardin, R. 1998. Optimization in Operations Research. Prentice-Hall, New Jersey.
- Steel, R., J. Torrie, and D. Dickey. Principles and Procedures of Statistics: A Biometrical Approach McGraw Hill, 1997.
- Zhang, H. 2000. Individual Project. Operation Research, Applications in Manufacturing. Spring.

www.lalena.com/ai/tsp, Obtained 13/12/01

www.Teradyne.com, Obtained 13/12/01

APPENDICES

Appendix I – Netlist Sample

Cadstar Design Editor Version 4.5.1.2
 Design: F:\PCD\Arc\09 19-02.arc
 Design Title: Bae 148 Lighting Board
 Units: Thousandths of an inch
 Decimal Places: 0
 Date: Tuesday, June 10, 2001
 Time: 9:29 AM

Net Name	Type	Layer	X	Y
N12V	Via	(Maximum)	14840	12083
N12V	Via	(Maximum)	15100	11275
V REF TP7.1		(Maximum)	16395	11700
V REF Via		(Maximum)	16345	11750
V REF J7.3		(Maximum)	16090	11750
\$1,201	Via	(Maximum)	10429	10494
\$1,201	Via	(Maximum)	10800	10805
\$1,201	T1.2	(Maximum)	10900	10905
\$1,202	Via	(Maximum)	12725	10460
\$1,157	TP1.1	(Maximum)	16280	11850
\$1,157	Via	(Maximum)	16445	11975
\$1,098	Via	(Maximum)	14760	11950
\$1,098	Via	(Maximum)	16040	10875
\$1,098	Via	(Maximum)	16175	11825
\$1,099	Via	(Maximum)	15100	11125
\$1,099	Via	(Maximum)	15175	10385
\$1,099	Via	(Maximum)	15175	10500
\$1,143	J7.11	(Maximum)	16350	10050
\$1,143	J7.12	(Maximum)	16429	10050
\$1,174	Via	(Maximum)	13700	12300
\$1,174	TP10.1	(Maximum)	13235	12130
P12V	Via	(Maximum)	12650	10590
P12V	J7.4	(Maximum)	12650	10190
P12V	Via	(Maximum)	14175	10350
\$1,193	TP8.1	(Maximum)	14850	12350
\$1,193	Via	(Maximum)	14045	12400
\$1,107	Via	(Maximum)	13575	10425
\$1,107	Via	(Maximum)	10350	10100
\$1,107	Via	(Maximum)	11190	10110
\$1,137	Via	(Maximum)	15905	12825
\$1,137	Via	(Maximum)	15905	12525
\$1,137	Via	(Maximum)	15905	12675

\$1,196 T1.10	(Maximum)	12465	10785
\$1,196 Via	(Maximum)	12525	10725
\$1,196 T1.6	(Maximum)	11210	10740
\$1,113 C3.1	(Maximum)	14660	12665
\$1,113 C4.1	(Maximum)	14800	12245
\$1,113 Via	(Maximum)	16160	12275
\$1,369 T1.1	(Maximum)	11975	12200
\$1,369 Via	(Maximum)	12225	11655
\$1,369 TP2.1	(Maximum)	10429	10257
\$1,172 Via	(Maximum)	13410	12255
\$1,172 J4.1	(Maximum)	13700	12400
\$1,202 J4.2	(Maximum)	16240	11925
\$1,202 Via	(Maximum)	16440	12190
P15 Via	(Maximum)	11680	10950
P15 Via	(Maximum)	11685	10740
P15 Via	(Maximum)	12135	10700
\$1,190 Via	(Maximum)	12135	11200
\$1,190 Via	(Maximum)	11680	11160
\$1,213 Via	(Maximum)	14250	10460
\$1,213 Via	(Maximum)	15855	10255
\$1,130 Via	(Maximum)	13260	11825
\$1,130 Via	(Maximum)	13260	11970
\$1,130 Via	(Maximum)	13260	11675
P16 J7.1	(Maximum)	15490	10255
P16 Via	(Maximum)	16105	11465
\$1,124 TP9.1	(Maximum)	11885	11325
\$1,124 Via	(Maximum)	11430	11785
\$1,124 Via	(Maximum)	11885	11825
\$1,110 T1.5	(Maximum)	14675	12250
\$1,110 Via	(Maximum)	14505	12380
\$1,110 Via	(Maximum)	14450	12770
\$1,104 Via	(Maximum)	15050	11950
\$1,104 Via	(Maximum)	15025	10625
\$1,104 Via	(Maximum)	14275	12370
\$1,103 Via	(Maximum)	15385	12210
\$1,103 J7.10	(Maximum)	15385	12280
5VD Via	(Maximum)	14660	12565
5VD Via	(Maximum)	15935	12235

End of report

Appendix II – Visual Basic Code of Proposed Algorithm

Option Compare Database

Option Explicit

Public dbcurrent As Database

Public Complex1 As Long

Public Complex As Long

Public InitTime As Date

Public EndTime As Date

Function CopyNetlist() As Double

Dim rsdPermNetList As Recordset

Dim rsdNetList As Recordset

Set dbcurrent = CurrentDb()

Set rsdPermNetList = dbcurrent.OpenRecordset("Permnetlist", dbOpenDynaset)

Set rsdNetList = dbcurrent.OpenRecordset("Netlist", dbOpenDynaset)

Do Until rsdPermNetList.EOF

 With rsdNetList

 .AddNew

 .Fields(0) = rsdPermNetList.Fields(0)

 .Fields(1) = rsdPermNetList.Fields(1)

 .Fields(2) = rsdPermNetList.Fields(2)

 .Fields(3) = rsdPermNetList.Fields(3)

 .Update

 End With

 rsdPermNetList.MoveNext

Loop

End Function

Function CreateNetworksTable() As Double

Dim QuanofNet As Integer

Dim SeqN As Double

SeqN = 1

Set dbcurrent = CurrentDb()

Dim rsdPermNetList As Recordset

```

Set rsdPermNetList = dbcurrent.OpenRecordset("PermNetlist", dbOpenDynaset)
Dim rsdProbeBMove As Recordset
Set rsdProbeBMove = dbcurrent.OpenRecordset("ProbeBMove", dbOpenDynaset)
Dim rsdNetworks As Recordset
Set rsdNetworks = dbcurrent.OpenRecordset("Networks", dbOpenDynaset)

rsdProbeBMove.MoveFirst

Do Until rsdProbeBMove.EOF
    QuanofNet = 0
    rsdPermNetList.MoveFirst

    Do Until rsdPermNetList.EOF
        If rsdPermNetList!Network = rsdProbeBMove!Network Then
            QuanofNet = QuanofNet + 1
        End If

        rsdPermNetList.MoveNext
    Loop

    With rsdNetworks
        .AddNew
        !Sequence = SeqN
        !Network = rsdProbeBMove!Network
        !NumofNet = QuanofNet
        .Update
        SeqN = SeqN + 1
    End With

    rsdProbeBMove.MoveNext
Loop
End Function

Function Sup(X1 As Double, X2 As Double, Y1 As Double, Y2 As Double) As Double

Complex1 = Complex1 + 1

If (Abs(X1 - X2) > Abs(Y1 - Y2)) Then
    Sup = Abs(X1 - X2)
Else
    Sup = Abs(Y1 - Y2)
End If

```


End Function

Function CreateProbeBTable()

Dim Selnet As String

Dim Selnode As Double

Dim XSelnode As Double

Dim YSelnode As Double

Dim dist1 As Double

Dim dist2 As Double

Dim tempnet As String

Dim tempnode As Double

Dim Xtempnode As Double

Dim Ytempnode As Double

Dim tempdist As Double

Dim SeqB As Double

Complex = 0

Complex1 = 0

Selnet = 0

Selnode = 0

XSelnode = 0

YSelnode = 0

SeqB = 1

InitTime = Time

Set dbcurrent = CurrentDb()

Dim rsdNetList As Recordset

Set rsdNetList = dbcurrent.OpenRecordset("Netlist", dbOpenDynaset)

Dim rsdProbeBMove As Recordset

Set rsdProbeBMove = dbcurrent.OpenRecordset("ProbeBMove", dbOpenDynaset)

Do Until rsdNetList.RecordCount = 0

dist1 = 9999999

rsdNetList.MoveFirst

Do Until rsdNetList.EOF

dist2 = Sup(rsdNetList!x, XSelnode, rsdNetList!y, YSelnode)

If dist2 < dist1 Then

tempnet = rsdNetList!Network

tempnode = rsdNetList!ref_desig

Xtempnode = rsdNetList!x

Ytempnode = rsdNetList!y

```
tempdist = dist2  
dist1 = dist2
```

```
End If  
rsdNetList.MoveNext  
Loop
```

```
With rsdProbeBMove  
    .AddNew  
    !Sequence = SeqB  
    !Network = tempnet  
    !ref_desig = tempnode  
    !x = Xtempnode  
    !y = Ytempnode  
    .Update  
    SeqB = SeqB + 1  
End With
```

```
Selnet = tempnet  
Selnode = tempnode  
XSelnode = Xtempnode  
YSelnode = Ytempnode
```

```
rsdNetList.MoveFirst
```

```
Do Until rsdNetList.EOF  
    If rsdNetList!Network = Selnet Then  
        rsdNetList.Delete  
    End If  
    rsdNetList.MoveNext  
Loop
```

```
Loop  
Complex = Complex1  
End Function
```

```
Function CreateProbeATable()
```

```
Set dbcurrent = CurrentDb()
```

```
Dim XVal As Double  
Dim YVal As Double  
Dim Selnet2 As String  
Dim Selnode2 As Double  
Dim Incredist2 As Double
```

Dim SeqA As Double

SeqA = 1

Dim rsdProbeBMove As Recordset

Set rsdProbeBMove = dbcurrent.OpenRecordset("ProbeBMove", dbOpenDynaset)

Dim rsdPermNetList As Recordset

Set rsdPermNetList = dbcurrent.OpenRecordset("PermNetlist", dbOpenDynaset)

Dim rsdProbeAMove As Recordset

Set rsdProbeAMove = dbcurrent.OpenRecordset("ProbeAMove", dbOpenDynaset)

rsdProbeBMove.MoveFirst

Do Until rsdProbeBMove.EOF

 Selnet2 = rsdProbeBMove!Network

 Selnode2 = rsdProbeBMove!ref_desig

 rsdPermNetList.MoveFirst

 Do Until rsdPermNetList.EOF

 If (rsdPermNetList!Network = Selnet2) And (rsdPermNetList!ref_desig <>
Selnode2) Then

 Incredist2 = Sup(rsdPermNetList!x, XVal, rsdPermNetList!y, YVal)

 XVal = rsdPermNetList!x

 YVal = rsdPermNetList!y

 With rsdProbeAMove

 .AddNew

 !Sequence = SeqA

 !Network = rsdPermNetList!Network

 !ref_desig = rsdPermNetList!ref_desig

 !x = rsdPermNetList!x

 !y = rsdPermNetList!y

 .Update

 SeqA = SeqA + 1

 End With

 End If

 rsdPermNetList.MoveNext

 Loop

 rsdProbeBMove.MoveNext

Loop

End Function

Function TestSequence() As Double

Dim total As Double

```

Dim i As Integer
Dim m As Integer
Dim n As Integer
Dim NumofNet As Integer
Dim ProbeAX As Double
Dim ProbeAY As Double
Dim ProbeBX As Double
Dim ProbeBY As Double
Dim Dist3 As Double
Dim Dist4 As Double
Dim dist5 As Double
Dim TTD As Double
Dim SeqT As Double

Set dbcurrent = CurrentDb()

Dim rsdProbeAMove As Recordset
Set rsdProbeAMove = dbcurrent.OpenRecordset("ProbeAMove", dbOpenDynaset)
Dim rsdProbeBMove As Recordset
Set rsdProbeBMove = dbcurrent.OpenRecordset("ProbeBMove", dbOpenDynaset)
Dim rsdFinalTest As Recordset
Set rsdFinalTest = dbcurrent.OpenRecordset("FinalTest", dbOpenDynaset)
Dim rsdNetworks As Recordset
Set rsdNetworks = dbcurrent.OpenRecordset("Networks", dbOpenDynaset)

rsdNetworks.MoveLast
n = rsdNetworks.RecordCount
rsdProbeAMove.MoveFirst
i = 0
rsdProbeBMove.MoveFirst
m = 1
rsdNetworks.MoveFirst

ProbeAX = rsdProbeBMove!x - 1
ProbeAY = rsdProbeBMove!y - 1
ProbeBX = rsdProbeBMove!x - 1
ProbeBY = rsdProbeBMove!y - 1
TTD = 0
SeqT = 1

Do While i < n

    Dist3 = Sup(rsdProbeAMove!x, ProbeAX, rsdProbeAMove!y, ProbeAY)
    Dist4 = Sup(rsdProbeBMove!x, ProbeBX, rsdProbeBMove!y, ProbeBY)
    If Dist3 >= Dist4 Then

```

```

    dist5 = Dist3
Else
    dist5 = Dist4
End If
TTD = TTD + dist5
ProbeAX = rsdProbeAMove!x
ProbeAY = rsdProbeAMove!y
ProbeBX = rsdProbeBMove!x
ProbeBY = rsdProbeBMove!y

With rsdFinalTest
    .AddNew
    !Sequence = SeqT
    !probeanet = rsdProbeAMove!Network
    !probea_x = rsdProbeAMove!x
    !probea_y = rsdProbeAMove!y
    !probebnet = rsdProbeBMove!Network
    !probeb_x = rsdProbeBMove!x
    !probeb_y = rsdProbeBMove!y
    !Open = 1
    !short = 0
    !IncremDistance = dist5
    !totaldistance = TTD
    .Update
    SeqT = SeqT + 1

End With
i = i + 1
NumofNet = rsdNetworks!NumofNet
rsdNetworks.MoveNext
If NumofNet = 3 Then
    rsdProbeAMove.MoveNext
    Dist3 = Sup(rsdProbeAMove!x, ProbeAX, rsdProbeAMove!y, ProbeAY)
    Dist4 = Sup(rsdProbeBMove!x, ProbeBX, rsdProbeBMove!y, ProbeBY)
    If Dist3 >= Dist4 Then
        dist5 = Dist3
    Else
        dist5 = Dist4
    End If
    TTD = TTD + dist5
    ProbeAX = rsdProbeAMove!x
    ProbeAY = rsdProbeAMove!y
    With rsdFinalTest
        .AddNew
        !Sequence = SeqT

```

```

!probeanet = rsdProbeAMove!Network
!probea_x = rsdProbeAMove!x
!probea_y = rsdProbeAMove!y
!probebnet = rsdProbeBMove!Network
!probeb_x = rsdProbeBMove!x
!probeb_y = rsdProbeBMove!y
!Open = 1
!short = 0
!IncremDistance = dist5
!totaldistance = TTD
.Update
SeqT = SeqT + 1

```

```

End With
End If

```

```

rsdProbeBMove.MoveNext
m = m + 1

```

```

Do Until rsdProbeBMove.EOF
    Dist3 = Sup(rsdProbeAMove!x, ProbeAX, rsdProbeAMove!y, ProbeAY)
    Dist4 = Sup(rsdProbeBMove!x, ProbeBX, rsdProbeBMove!y, ProbeBY)
    If Dist3 >= Dist4 Then
        dist5 = Dist3
    Else
        dist5 = Dist4
    End If

```

```

TTD = TTD + dist5
ProbeBX = rsdProbeBMove!x
ProbeBY = rsdProbeBMove!y
With rsdFinalTest
    .AddNew
    !Sequence = SeqT
    !probeanet = rsdProbeAMove!Network
    !probea_x = rsdProbeAMove!x
    !probea_y = rsdProbeAMove!y
    !probebnet = rsdProbeBMove!Network
    !probeb_x = rsdProbeBMove!x
    !probeb_y = rsdProbeBMove!y
    !Open = 0
    !short = 1
    !IncremDistance = dist5
    !totaldistance = TTD
    .Update

```

SeqT = SeqT + 1

End With
rsdProbeBMove.MoveNext
m = m + 1

Loop

m = m - 1
rsdProbeBMove.MovePrevious
If i < n Then
 rsdProbeAMove.MoveNext
 Dist3 = Sup(rsdProbeAMove!x, ProbeAX, rsdProbeAMove!y, ProbeAY)
 Dist4 = Sup(rsdProbeBMove!x, ProbeBX, rsdProbeBMove!y, ProbeBY)
 If Dist3 >= Dist4 Then
 dist5 = Dist3
 Else
 dist5 = Dist4
 End If
 TTD = TTD + dist5
 ProbeAX = rsdProbeAMove!x
 ProbeAY = rsdProbeAMove!y
 With rsdFinalTest
 .AddNew
 !Sequence = SeqT
 !probeanet = rsdProbeAMove!Network
 !probea_x = rsdProbeAMove!x
 !probea_y = rsdProbeAMove!y
 !probenet = rsdProbeBMove!Network
 !probeb_x = rsdProbeBMove!x
 !probeb_y = rsdProbeBMove!y
 !Open = 0
 !short = 1
 !IncremDistance = dist5
 !totaldistance = TTD
 .Update
 SeqT = SeqT + 1
 End With

 i = i + 1
 NumofNet = rsdNetworks!NumofNet
 rsdNetworks.MoveNext

Do Until m = i + 1

rsdProbeBMove.MovePrevious

m = m - 1

Dist3 = Sup(rsdProbeAMove!x, ProbeAX, rsdProbeAMove!y, ProbeAY)

Dist4 = Sup(rsdProbeBMove!x, ProbeBX, rsdProbeBMove!y, ProbeBY)

If Dist3 >= Dist4 Then

dist5 = Dist3

Else

dist5 = Dist4

End If

TTD = TTD + dist5

ProbeBX = rsdProbeBMove!x

ProbeBY = rsdProbeBMove!y

With rsdFinalTest

.AddNew

!Sequence = SeqT

!probeanet = rsdProbeAMove!Network

!probea_x = rsdProbeAMove!x

!probea_y = rsdProbeAMove!y

!probenet = rsdProbeBMove!Network

!probeb_x = rsdProbeBMove!x

!probeb_y = rsdProbeBMove!y

!Open = 0

!short = 1

!IncremDistance = dist5

!totaldistance = TTD

.Update

SeqT = SeqT + 1

End With

Loop

rsdProbeBMove.MovePrevious

m = m - 1

Dist3 = Sup(rsdProbeAMove!x, ProbeAX, rsdProbeAMove!y, ProbeAY)

Dist4 = Sup(rsdProbeBMove!x, ProbeBX, rsdProbeBMove!y, ProbeBY)

If Dist3 >= Dist4 Then

dist5 = Dist3

Else

dist5 = Dist4

End If

TTD = TTD + dist5


```

ProbeBX = rsdProbeBMove!x
ProbeBY = rsdProbeBMove!y
With rsdFinalTest
    .AddNew
    !Sequence = SeqT
    !probeanet = rsdProbeAMove!Network
    !probea_x = rsdProbeAMove!x
    !probea_y = rsdProbeAMove!y
    !probenet = rsdProbeBMove!Network
    !probeb_x = rsdProbeBMove!x
    !probeb_y = rsdProbeBMove!y
    !Open = 1
    !short = 0
    !IncremDistance = dist5
    !totaldistance = TTD
    .Update
    SeqT = SeqT + 1

End With

If NumofNet = 3 Then
    rsdProbeAMove.MoveNext
    Dist3 = Sup(rsdProbeAMove!x, ProbeAX, rsdProbeAMove!y, ProbeAY)
    Dist4 = Sup(rsdProbeBMove!x, ProbeBX, rsdProbeBMove!y, ProbeBY)
    If Dist3 >= Dist4 Then
        dist5 = Dist3
    Else
        dist5 = Dist4
    End If

    TTD = TTD + dist5
    ProbeAX = rsdProbeAMove!x
    ProbeAY = rsdProbeAMove!y
    With rsdFinalTest
        .AddNew
        !Sequence = SeqT
        !probeanet = rsdProbeAMove!Network
        !probea_x = rsdProbeAMove!x
        !probea_y = rsdProbeAMove!y
        !probenet = rsdProbeBMove!Network
        !probeb_x = rsdProbeBMove!x
        !probeb_y = rsdProbeBMove!y
        !Open = 1
        !short = 0
        !IncremDistance = dist5

```

```

        !totaldistance = TTD
        .Update
        SeqT = SeqT + 1

    End With

    End If
    rsdProbeAMove.MoveNext
    rsdProbeBMove.MoveNext
    m = m + 1

End If
Loop

Dim rsdOutPut As Recordset
Set rsdOutPut = dbcurrent.OpenRecordset("Output", dbOpenDynaset)

EndTime = Time

EndTime = EndTime - InitTime
With rsdOutPut
    .AddNew
    !Total_Distance_Test = TTD
    !complexity = Complex
    !time_execution = CDbl(EndTime)

    .Update
End With

End Function

```

Appendix III – Visual Basic Code of Nearest Neighbor Algorithm

```
Option Compare Database
Option Explicit
Public dbcurrent As Database
Public Complex As Long
Public Complex1 As Long
Public InitTime As Date
Public EndTime As Date
Function nilRunAlgorithm()
End Function
Function CopyNetlist() As Double

' this loop copies the contents of the permanent netlist
' to the temporary netlist record by record
Dim rsdPermNetList As Recordset
Dim rsdNetlist As Recordset

Set dbcurrent = CurrentDb()
Set rsdPermNetList = dbcurrent.OpenRecordset("Permnetlist", dbOpenDynaset)
Set rsdNetlist = dbcurrent.OpenRecordset("Netlist", dbOpenDynaset)

Do Until rsdPermNetList.EOF

    With rsdNetlist
        .AddNew
        .Fields(0) = rsdPermNetList.Fields(0)
        .Fields(1) = rsdPermNetList.Fields(1)
        .Fields(2) = rsdPermNetList.Fields(2)
        .Fields(3) = rsdPermNetList.Fields(3)
        .Update
    End With

    rsdPermNetList.MoveNext

Loop
End Function

Function CreateNetworksTable() As Double

Dim QuanofNet As Integer
Dim SeqN As Double

SeqN = 1
```

```

Set dbcurrent = CurrentDb()

Dim rsdPermNetList As Recordset
Set rsdPermNetList = dbcurrent.OpenRecordset("PermNetlist", dbOpenDynaset)
Dim rsdProbeBMove As Recordset
Set rsdProbeBMove = dbcurrent.OpenRecordset("ProbeBMove", dbOpenDynaset)
Dim rsdNetworks As Recordset
Set rsdNetworks = dbcurrent.OpenRecordset("Networks", dbOpenDynaset)

rsdProbeBMove.MoveFirst

Do Until rsdProbeBMove.EOF
    QuanofNet = 0
    rsdPermNetList.MoveFirst

    Do Until rsdPermNetList.EOF
        If rsdPermNetList!Network = rsdProbeBMove!Network Then
            QuanofNet = QuanofNet + 1
        End If

        rsdPermNetList.MoveNext
    Loop

    With rsdNetworks
        .AddNew
        !Sequence = SeqN
        !Network = rsdProbeBMove!Network
        !NumofNet = QuanofNet
        .Update
        SeqN = SeqN + 1
    End With

    rsdProbeBMove.MoveNext

Loop
End Function

Function Sup(X1 As Double, X2 As Double, Y1 As Double, Y2 As Double) As Double

Complex = Complex + 1

If (Abs(X1 - X2) > Abs(Y1 - Y2)) Then
    Sup = Abs(X1 - X2)
Else

```

```
Sup = Abs(Y1 - Y2)
End If
```

```
End Function
```

```
Function CreateProbeBTable()
```

```
Dim Complex As Integer
```

```
Dim Selnet As String
```

```
Dim Selnode As Double
```

```
Dim XSelnode As Double
```

```
Dim YSelnode As Double
```

```
Dim dist1 As Double
```

```
Dim dist2 As Double
```

```
Dim tempnet As String
```

```
Dim tempnode As Double
```

```
Dim Xtempnode As Double
```

```
Dim Ytempnode As Double
```

```
Dim tempdist As Double
```

```
Dim SeqB As Double
```

```
Complex = 0
```

```
Complex1 = 0
```

```
Selnet = 0
```

```
Selnode = 0
```

```
XSelnode = 0
```

```
YSelnode = 0
```

```
SeqB = 1
```

```
InitTime = Time
```

```
Set dbcurrent = CurrentDb()
```

```
Dim rsdNetlist As Recordset
```

```
Set rsdNetlist = dbcurrent.OpenRecordset("Netlist", dbOpenDynaset)
```

```
Dim rsdProbeBMove As Recordset
```

```
Set rsdProbeBMove = dbcurrent.OpenRecordset("ProbeBMove", dbOpenDynaset)
```

```
Do Until rsdNetlist.RecordCount = 0
```

```
dist1 = 9999999
```

```
rsdNetlist.MoveFirst
```

```
Do Until rsdNetlist.EOF
```

```
dist2 = Sup(rsdNetlist!x, XSelnode, rsdNetlist!y, YSelnode)
```

```
If dist2 < dist1 Then
```

```

    tempnet = rsdNetlist!Network
    tempnode = rsdNetlist!ref_desig
    Xtempnode = rsdNetlist!x
    Ytempnode = rsdNetlist!y
    tempdist = dist2
    dist1 = dist2

End If
    rsdNetlist.MoveNext
Loop

With rsdProbeBMove
    .AddNew
    !Sequence = SeqB
    !Network = tempnet
    !ref_desig = tempnode
    !x = Xtempnode
    !y = Ytempnode
    .Update
    SeqB = SeqB + 1
End With

Selnet = tempnet
Selnode = tempnode
XSelnode = Xtempnode
YSelnode = Ytempnode

rsdNetlist.MoveFirst

Do Until rsdNetlist.EOF
    If rsdNetlist!Network = Selnet Then
        rsdNetlist.Delete
    End If
    rsdNetlist.MoveNext
Loop

Loop
End Function

Function CreateProbeATable()

Set dbcurrent = CurrentDb()

Dim rsdPermNetList As Recordset
Set rsdPermNetList = dbcurrent.OpenRecordset("Permnetlist", dbOpenDynaset)

```

```

Dim rsdNetlist As Recordset
Set rsdNetlist = dbcurrent.OpenRecordset("Netlist", dbOpenDynaset)

Do Until rsdPermNetList.EOF

    With rsdNetlist
        .AddNew
        .Fields(0) = rsdPermNetList.Fields(0)
        .Fields(1) = rsdPermNetList.Fields(1)
        .Fields(2) = rsdPermNetList.Fields(2)
        .Fields(3) = rsdPermNetList.Fields(3)
        .Update
    End With

    rsdPermNetList.MoveNext

Loop
End Function

Function TestSequence() As Double

Dim SelnetA As String
Dim SelnodeA As Double
Dim SelnetAX As Double
Dim SelnetAY As Double
Dim SelnetB As String
Dim SelnodeB As Double
Dim SelnetBX As Double
Dim SelnetBY As Double
Dim ProbeAX As Double
Dim ProbeAY As Double
Dim ProbeBX As Double
Dim ProbeBY As Double
Dim Dist3 As Double
Dim Dist4 As Double
Dim dist5 As Double
Dim Dist6 As Double
Dim dist7 As Double
Dim TTD As Double
Dim SeqT As Double
Dim i As Long

Set dbcurrent = CurrentDb()

Dim rsdProbeBMove As Recordset

```

```

Set rsdProbeBMove = dbcurrent.OpenRecordset("ProbeBMove", dbOpenDynaset)
Dim rsdNetlist As Recordset
Set rsdNetlist = dbcurrent.OpenRecordset("Netlist", dbOpenDynaset)
Dim rsdFinalTest As Recordset
Set rsdFinalTest = dbcurrent.OpenRecordset("FinalTest", dbOpenDynaset)
Dim rsdNetlist2 As Recordset
Set rsdNetlist2 = dbcurrent.OpenRecordset("Netlist2", dbOpenDynaset)

```

```

rsdProbeBMove.MoveFirst

```

```

ProbeAX = rsdProbeBMove!x - 1
ProbeAY = rsdProbeBMove!y - 1
ProbeBX = rsdProbeBMove!x - 1
ProbeBY = rsdProbeBMove!y - 1
TTD = 0
SeqT = 1

```

```

Do Until rsdProbeBMove.EOF
    SelnetB = rsdProbeBMove!Network
    SelnodeB = rsdProbeBMove!ref_desig
    SelnetBX = rsdProbeBMove!x
    SelnetBY = rsdProbeBMove!y

```

```

    rsdNetlist.MoveFirst
    Do Until rsdNetlist.EOF
        With rsdNetlist2
            .AddNew
            !Network = rsdNetlist!Network
            !ref_desig = rsdNetlist!ref_desig
            !x = rsdNetlist!x
            !y = rsdNetlist!y
            .Update
        End With

```

```

        rsdNetlist.MoveNext
    Loop

```

```

    rsdNetlist2.MoveLast
    i = rsdNetlist2.RecordCount

```

```

    Do Until i = 0
        Dist3 = 999999
        rsdNetlist2.MoveFirst
        Do Until rsdNetlist2.EOF

```



```

If (rsdNetlist2!Network = SelnetB) And (rsdNetlist2!ref_desig = SelnodeB) Then
    rsdNetlist2.Delete
    i = i - 1
Else
    Dist4 = Sup(rsdNetlist2!x, ProbeBX, rsdNetlist2!y, ProbeBY)
    Complex1 = Complex1 + 1
    If Dist4 < Dist3 Then
        SelnetA = rsdNetlist2!Network
        SelnodeA = rsdNetlist2!ref_desig
        SelnetAX = rsdNetlist2!x
        SelnetAY = rsdNetlist2!y
        Dist3 = Dist4
    End If
End If
rsdNetlist2.MoveNext
Loop

If (SelnetA = SelnetB) Then
    dist5 = Sup(SelnetAX, ProbeAX, SelnetAY, ProbeAY)
    Dist6 = Sup(SelnetBX, ProbeBX, SelnetBY, ProbeBY)
    If dist5 >= Dist6 Then
        dist7 = dist5
    Else
        dist7 = Dist6
    End If
    TTD = TTD + dist7
    ProbeAX = SelnetAX
    ProbeAY = SelnetAY
    ProbeBX = SelnetBX
    ProbeBY = SelnetBY
    With rsdFinalTest
        .AddNew
        !Sequence = SeqT
        !probeanet = SelnetA
        !probea_x = SelnetAX
        !probea_y = SelnetAY
        !probebnet = rsdProbeBMove!Network
        !probeb_x = rsdProbeBMove!x
        !probeb_y = rsdProbeBMove!y
        !Open = 1
        !short = 0
        !IncremDistance = dist7
        !totaldistance = TTD
        .Update
        SeqT = SeqT + 1
    End With
End If

```

```

End With

rsdNetlist2.MoveFirst
Do Until rsdNetlist2.EOF
    If (rsdNetlist2!Network = SelnetA) And (rsdNetlist2!ref_desig = SelnodeA)
Then
        rsdNetlist2.Delete
        i = i - 1
        End If
        rsdNetlist2.MoveNext
    Loop

Else
    dist5 = Sup(SelnetAX, ProbeAX, SelnetAY, ProbeAY)
    Dist6 = Sup(SelnetBX, ProbeBX, SelnetBY, ProbeBY)
    If dist5 >= Dist6 Then
        dist7 = dist5
    Else
        dist7 = Dist6
    End If
    TTD = TTD + dist7
    ProbeAX = SelnetAX
    ProbeAY = SelnetAY
    ProbeBX = SelnetBX
    ProbeBY = SelnetBY
    With rsdFinalTest
        .AddNew
        !Sequence = SeqT
        !probeanet = SelnetA
        !probea_x = SelnetAX
        !probea_y = SelnetAY
        !probebnet = rsdProbeBMove!Network
        !probeb_x = rsdProbeBMove!x
        !probeb_y = rsdProbeBMove!y
        !Open = 0
        !short = 1
        !IncremDistance = dist7
        !totaldistance = TTD
        .Update
        SeqT = SeqT + 1
    End With

    rsdNetlist2.MoveFirst
    Do Until rsdNetlist2.EOF
        If rsdNetlist2!Network = SelnetA Then

```

```

        rsdNetlist2.Delete
        i = i - 1
    End If
    rsdNetlist2.MoveNext
Loop

End If
Loop

ProbeBX = rsdProbeBMove!x
ProbeBY = rsdProbeBMove!y

rsdNetlist.MoveFirst
Do Until rsdNetlist.EOF
    If rsdNetlist!Network = SelnetB Then
        rsdNetlist.Delete
    End If
    rsdNetlist.MoveNext
Loop

rsdProbeBMove.MoveNext
Loop

Dim rsdOutPut As Recordset
Set rsdOutPut = dbcurrent.OpenRecordset("Output", dbOpenDynaset)

EndTime = Time

EndTime = EndTime - InitTime
With rsdOutPut
    .AddNew
    !Total_Distance_Test = TTD
    !complexity = Complex1
    !time_execution = CDBl(EndTime)
    .Update
End With

End Function

```

Appendix IV – Proposed Algorithm Final Test Sequence for Netlist Sample 1

Sequence	probeANet	ProbeA_X	ProbeA_Y	probeBNet	ProbeB_X	ProbeB_Y	Open	Short	IncremDistance	TotalDistance
1	\$1,107	11190	10110	\$1,107	10350	10100	1	0	841	841
2	\$1,107	13575	10425	\$1,107	10350	10100	1	0	2385	3226
3	\$1,107	13575	10425	\$1,369	10429	10257	0	1	157	3383
4	\$1,107	13575	10425	\$1,201	10429	10494	0	1	237	3620
5	\$1,107	13575	10425	\$1,196	11210	10740	0	1	781	4401
6	\$1,107	13575	10425	\$1,190	11680	11160	0	1	470	4871
7	\$1,107	13575	10425	\$1,124	11885	11325	0	1	205	5076
8	\$1,107	13575	10425	P15	11680	10950	0	1	375	5451
9	\$1,107	13575	10425	P12V	12650	10190	0	1	970	6421
10	\$1,107	13575	10425	\$1,202	12725	10460	0	1	270	6691
11	\$1,107	13575	10425	\$1,130	13260	11675	0	1	1215	7906
12	\$1,107	13575	10425	\$1,174	13235	12130	0	1	455	8361
13	\$1,107	13575	10425	\$1,172	13410	12255	0	1	175	8536

Sequence	probeANet	ProbeA_X	ProbeA_Y	probeBNet	ProbeB_X	ProbeB_Y	Open	Short	IncremDistance	TotalDistance
14	\$1,107	13575	10425	\$1,193	14045	12400	0	1	635	9171
15	\$1,107	13575	10425	\$1,104	14275	12370	0	1	230	9401
16	\$1,107	13575	10425	\$1,110	14505	12380	0	1	230	9631
17	\$1,107	13575	10425	5VD	14660	12565	0	1	185	9816
18	\$1,107	13575	10425	\$1,113	14660	12665	0	1	100	9916
19	\$1,107	13575	10425	N12V	14840	12083	0	1	582	10498
20	\$1,107	13575	10425	\$1,098	14760	11950	0	1	133	10631
21	\$1,107	13575	10425	\$1,103	15385	12280	0	1	625	11256
22	\$1,107	13575	10425	\$1,137	15905	12675	0	1	520	11776
23	\$1,107	13575	10425	\$1,157	16445	11975	0	1	700	12476
24	\$1,107	13575	10425	V REF	16345	11750	0	1	225	12701
25	\$1,107	13575	10425	P16	16105	11465	0	1	285	12986
26	\$1,107	13575	10425	\$1,099	15175	10500	0	1	965	13951
27	\$1,107	13575	10425	\$1,213	15855	10255	0	1	680	14631

Sequence	probeANet	ProbeA_X	ProbeA_Y	probeBNet	ProbeB_X	ProbeB_Y	Open	Short	IncramDistance	TotalDistance
28	\$1,107	13575	10425	\$1,143	16350	10050	0	1	495	15126
29	\$1,369	12225	11655	\$1,143	16350	10050	0	1	1350	16476
30	\$1,369	12225	11655	\$1,213	15855	10255	0	1	495	16971
31	\$1,369	12225	11655	\$1,099	15175	10500	0	1	680	17651
32	\$1,369	12225	11655	P16	16105	11465	0	1	965	18616
33	\$1,369	12225	11655	V REF	16345	11750	0	1	285	18901
34	\$1,369	12225	11655	\$1,157	16445	11975	0	1	225	19126
35	\$1,369	12225	11655	\$1,137	15905	12675	0	1	700	19826
36	\$1,369	12225	11655	\$1,103	15385	12280	0	1	520	20346
37	\$1,369	12225	11655	\$1,098	14760	11950	0	1	625	20971
38	\$1,369	12225	11655	N12V	14840	12083	0	1	133	21104
39	\$1,369	12225	11655	\$1,113	14660	12665	0	1	582	21686
40	\$1,369	12225	11655	5VD	14660	12565	0	1	100	21786
41	\$1,369	12225	11655	\$1,110	14505	12380	0	1	185	21971

Sequence	probeANet	ProbeA_X	ProbeA_Y	probeBNet	ProbeB_X	ProbeB_Y	Open	Short	IncremDistance	TotalDistance
42	\$1,369	12225	11655	\$1,104	14275	12370	0	1	230	22201
43	\$1,369	12225	11655	\$1,193	14045	12400	0	1	230	22431
44	\$1,369	12225	11655	\$1,172	13410	12255	0	1	635	23066
45	\$1,369	12225	11655	\$1,174	13235	12130	0	1	175	23241
46	\$1,369	12225	11655	\$1,130	13260	11675	0	1	455	23696
47	\$1,369	12225	11655	\$1,202	12725	10460	0	1	1215	24911
48	\$1,369	12225	11655	P12V	12650	10190	0	1	270	25181
49	\$1,369	12225	11655	P15	11680	10950	0	1	970	26151
50	\$1,369	12225	11655	\$1,124	11885	11325	0	1	375	26526
51	\$1,369	12225	11655	\$1,190	11680	11160	0	1	205	26731
52	\$1,369	12225	11655	\$1,196	11210	10740	0	1	470	27201
53	\$1,369	12225	11655	\$1,201	10429	10494	0	1	781	27982
54	\$1,369	12225	11655	\$1,369	10429	10257	1	0	237	28219
55	\$1,369	11975	12200	\$1,369	10429	10257	1	0	545	28764

Sequence	probeANet	ProbeA_X	ProbeA_Y	probeBNet	ProbeB_X	ProbeB_Y	Open	Short	IncremDistance	TotalDistance
56	\$1,201	10900	10905	\$1,201	10429	10494	1	0	1295	30059
57	\$1,201	10800	10805	\$1,201	10429	10494	1	0	100	30159
58	\$1,201	10800	10805	\$1,196	11210	10740	0	1	781	30940
59	\$1,201	10800	10805	\$1,190	11680	11160	0	1	470	31410
60	\$1,201	10800	10805	\$1,124	11885	11325	0	1	205	31615
61	\$1,201	10800	10805	P15	11680	10950	0	1	375	31990
62	\$1,201	10800	10805	P12V	12650	10190	0	1	970	32960
63	\$1,201	10800	10805	\$1,202	12725	10460	0	1	270	33230
64	\$1,201	10800	10805	\$1,130	13260	11675	0	1	1215	34445
65	\$1,201	10800	10805	\$1,174	13235	12130	0	1	455	34900
66	\$1,201	10800	10805	\$1,172	13410	12255	0	1	175	35075
67	\$1,201	10800	10805	\$1,193	14045	12400	0	1	635	35710
68	\$1,201	10800	10805	\$1,104	14275	12370	0	1	230	35940
69	\$1,201	10800	10805	\$1,110	14505	12380	0	1	230	36170

Sequence	probeANet	ProbeA_X	ProbeA_Y	probeBNet	ProbeB_X	ProbeB_Y	Open	Short	IncramDistance	TotalDistance
70	\$1,201	10800	10805	5VD	14660	12565	0	1	185	36355
71	\$1,201	10800	10805	\$1,113	14660	12665	0	1	100	36455
72	\$1,201	10800	10805	N12V	14840	12083	0	1	582	37037
73	\$1,201	10800	10805	\$1,098	14760	11950	0	1	133	37170
74	\$1,201	10800	10805	\$1,103	15385	12280	0	1	625	37795
75	\$1,201	10800	10805	\$1,137	15905	12675	0	1	520	38315
76	\$1,201	10800	10805	\$1,157	16445	11975	0	1	700	39015
77	\$1,201	10800	10805	V REF	16345	11750	0	1	225	39240
78	\$1,201	10800	10805	P16	16105	11465	0	1	285	39525
79	\$1,201	10800	10805	\$1,099	15175	10500	0	1	965	40490
80	\$1,201	10800	10805	\$1,213	15855	10255	0	1	680	41170
81	\$1,201	10800	10805	\$1,143	16350	10050	0	1	495	41665
82	\$1,196	12525	10725	\$1,143	16350	10050	0	1	1725	43390
83	\$1,196	12525	10725	\$1,213	15855	10255	0	1	495	43885

Sequence	probeANet	ProbeA_X	ProbeA_Y	probeBNet	ProbeB_X	ProbeB_Y	Open	Short	IncremDistance	TotalDistance
84	\$1,196	12525	10725	\$1,099	15175	10500	0	1	680	44565
85	\$1,196	12525	10725	P16	16105	11465	0	1	965	45530
86	\$1,196	12525	10725	V REF	16345	11750	0	1	285	45815
87	\$1,196	12525	10725	\$1,157	16445	11975	0	1	225	46040
88	\$1,196	12525	10725	\$1,137	15905	12675	0	1	700	46740
89	\$1,196	12525	10725	\$1,103	15385	12280	0	1	520	47260
90	\$1,196	12525	10725	\$1,098	14760	11950	0	1	625	47885
91	\$1,196	12525	10725	N12V	14840	12083	0	1	133	48018
92	\$1,196	12525	10725	\$1,113	14660	12665	0	1	582	48600
93	\$1,196	12525	10725	5VD	14660	12565	0	1	100	48700
94	\$1,196	12525	10725	\$1,110	14505	12380	0	1	185	48885
95	\$1,196	12525	10725	\$1,104	14275	12370	0	1	230	49115
96	\$1,196	12525	10725	\$1,193	14045	12400	0	1	230	49345
97	\$1,196	12525	10725	\$1,172	13410	12255	0	1	635	49980

Sequence	probeANet	ProbeA_X	ProbeA_Y	probeBNet	ProbeB_X	ProbeB_Y	Open	Short	IncremDistance	TotalDistance
98	\$1,196	12525	10725	\$1,174	13235	12130	0	1	175	50155
99	\$1,196	12525	10725	\$1,130	13260	11675	0	1	455	50610
100	\$1,196	12525	10725	\$1,202	12725	10460	0	1	1215	51825
101	\$1,196	12525	10725	P12V	12650	10190	0	1	270	52095
102	\$1,196	12525	10725	P15	11680	10950	0	1	970	53065
103	\$1,196	12525	10725	\$1,124	11885	11325	0	1	375	53440
104	\$1,196	12525	10725	\$1,190	11680	11160	0	1	205	53645
105	\$1,196	12525	10725	\$1,196	11210	10740	1	0	470	54115
106	\$1,196	12465	10785	\$1,196	11210	10740	1	0	60	54175
107	\$1,190	12135	11200	\$1,190	11680	11160	1	0	470	54645
108	\$1,190	12135	11200	\$1,124	11885	11325	0	1	205	54850
109	\$1,190	12135	11200	P15	11680	10950	0	1	375	55225
110	\$1,190	12135	11200	P12V	12650	10190	0	1	970	56195
111	\$1,190	12135	11200	\$1,202	12725	10460	0	1	270	56465

Sequence	probeANet	ProbeA_X	ProbeA_Y	probeBNet	ProbeB_X	ProbeB_Y	Open	Short	IncramDistance	TotalDistance
112	\$1,190	12135	11200	\$1,130	13260	11675	0	1	1215	57680
113	\$1,190	12135	11200	\$1,174	13235	12130	0	1	455	58135
114	\$1,190	12135	11200	\$1,172	13410	12255	0	1	175	58310
115	\$1,190	12135	11200	\$1,193	14045	12400	0	1	635	58945
116	\$1,190	12135	11200	\$1,104	14275	12370	0	1	230	59175
117	\$1,190	12135	11200	\$1,110	14505	12380	0	1	230	59405
118	\$1,190	12135	11200	5VD	14660	12565	0	1	185	59590
119	\$1,190	12135	11200	\$1,113	14660	12665	0	1	100	59690
120	\$1,190	12135	11200	N12V	14840	12083	0	1	582	60272
121	\$1,190	12135	11200	\$1,098	14760	11950	0	1	133	60405
122	\$1,190	12135	11200	\$1,103	15385	12280	0	1	625	61030
123	\$1,190	12135	11200	\$1,137	15905	12675	0	1	520	61550
124	\$1,190	12135	11200	\$1,157	16445	11975	0	1	700	62250
125	\$1,190	12135	11200	V REF	16345	11750	0	1	225	62475

Sequence	probeANet	ProbeA_X	ProbeA_Y	probeBNet	ProbeB_X	ProbeB_Y	Open	Short	IncramDistance	TotalDistance
126	\$1,190	12135	11200	P16	16105	11465	0	1	285	62760
127	\$1,190	12135	11200	\$1,099	15175	10500	0	1	965	63725
128	\$1,190	12135	11200	\$1,213	15855	10255	0	1	680	64405
129	\$1,190	12135	11200	\$1,143	16350	10050	0	1	495	64900
130	\$1,124	11885	11825	\$1,143	16350	10050	0	1	625	65525
131	\$1,124	11885	11825	\$1,213	15855	10255	0	1	495	66020
132	\$1,124	11885	11825	\$1,099	15175	10500	0	1	680	66700
133	\$1,124	11885	11825	P16	16105	11465	0	1	965	67665
134	\$1,124	11885	11825	V REF	16345	11750	0	1	285	67950
135	\$1,124	11885	11825	\$1,157	16445	11975	0	1	225	68175
136	\$1,124	11885	11825	\$1,137	15905	12675	0	1	700	68875
137	\$1,124	11885	11825	\$1,103	15385	12280	0	1	520	69395
138	\$1,124	11885	11825	\$1,098	14760	11950	0	1	625	70020
139	\$1,124	11885	11825	N12V	14840	12083	0	1	133	70153

Sequence	probeANet	ProbeA_X	ProbeA_Y	probeBNet	ProbeB_X	ProbeB_Y	Open	Short	IncremDistance	TotalDistance
140	\$1,124	11885	11825	\$1,113	14660	12665	0	1	582	70735
141	\$1,124	11885	11825	5VD	14660	12565	0	1	100	70835
142	\$1,124	11885	11825	\$1,110	14505	12380	0	1	185	71020
143	\$1,124	11885	11825	\$1,104	14275	12370	0	1	230	71250
144	\$1,124	11885	11825	\$1,193	14045	12400	0	1	230	71480
145	\$1,124	11885	11825	\$1,172	13410	12255	0	1	635	72115
146	\$1,124	11885	11825	\$1,174	13235	12130	0	1	175	72290
147	\$1,124	11885	11825	\$1,130	13260	11675	0	1	455	72745
148	\$1,124	11885	11825	\$1,202	12725	10460	0	1	1215	73960
149	\$1,124	11885	11825	P12V	12650	10190	0	1	270	74230
150	\$1,124	11885	11825	P15	11680	10950	0	1	970	75200
151	\$1,124	11885	11825	\$1,124	11885	11325	1	0	375	75575
152	\$1,124	11430	11785	\$1,124	11885	11325	1	0	455	76030
153	P15	12135	10700	P15	11680	10950	1	0	1085	77115

Sequence	probeANet	ProbeA_X	ProbeA_Y	probeBNet	ProbeB_X	ProbeB_Y	Open	Short	IncremDistance	TotalDistance
154	P15	11685	10740	P15	11680	10950	1	0	450	77565
155	P15	11685	10740	P12V	12650	10190	0	1	970	78535
156	P15	11685	10740	\$1,202	12725	10460	0	1	270	78805
157	P15	11685	10740	\$1,130	13260	11675	0	1	1215	80020
158	P15	11685	10740	\$1,174	13235	12130	0	1	455	80475
159	P15	11685	10740	\$1,172	13410	12255	0	1	175	80650
160	P15	11685	10740	\$1,193	14045	12400	0	1	635	81285
161	P15	11685	10740	\$1,104	14275	12370	0	1	230	81515
162	P15	11685	10740	\$1,110	14505	12380	0	1	230	81745
163	P15	11685	10740	5VD	14660	12565	0	1	185	81930
164	P15	11685	10740	\$1,113	14660	12665	0	1	100	82030
165	P15	11685	10740	N12V	14840	12083	0	1	582	82612
166	P15	11685	10740	\$1,098	14760	11950	0	1	133	82745
167	P15	11685	10740	\$1,103	15385	12280	0	1	625	83370

Sequence	probeANet	ProbeA_X	ProbeA_Y	probeBNet	ProbeB_X	ProbeB_Y	Open	Short	IncrementDistance	TotalDistance
168	P15	11685	10740	\$1,137	15905	12675	0	1	520	83890
169	P15	11685	10740	\$1,157	16445	11975	0	1	700	84590
170	P15	11685	10740	V REF	16345	11750	0	1	225	84815
171	P15	11685	10740	P16	16105	11465	0	1	285	85100
172	P15	11685	10740	\$1,099	15175	10500	0	1	965	86065
173	P15	11685	10740	\$1,213	15855	10255	0	1	680	86745
174	P15	11685	10740	\$1,143	16350	10050	0	1	495	87240
175	P12V	14175	10350	\$1,143	16350	10050	0	1	2490	89730
176	P12V	14175	10350	\$1,213	15855	10255	0	1	495	90225
177	P12V	14175	10350	\$1,099	15175	10500	0	1	680	90905
178	P12V	14175	10350	P16	16105	11465	0	1	965	91870
179	P12V	14175	10350	V REF	16345	11750	0	1	285	92155
180	P12V	14175	10350	\$1,157	16445	11975	0	1	225	92380
181	P12V	14175	10350	\$1,137	15905	12675	0	1	700	93080

Sequence	probeANet	ProbeA_X	ProbeA_Y	probeBNet	ProbeB_X	ProbeB_Y	Open	Short	IncremDistance	TotalDistance
182	P12V	14175	10350	\$1,103	15385	12280	0	1	520	93600
183	P12V	14175	10350	\$1,098	14760	11950	0	1	625	94225
184	P12V	14175	10350	N12V	14840	12083	0	1	133	94358
185	P12V	14175	10350	\$1,113	14660	12665	0	1	582	94940
186	P12V	14175	10350	5VD	14660	12565	0	1	100	95040
187	P12V	14175	10350	\$1,110	14505	12380	0	1	185	95225
188	P12V	14175	10350	\$1,104	14275	12370	0	1	230	95455
189	P12V	14175	10350	\$1,193	14045	12400	0	1	230	95685
190	P12V	14175	10350	\$1,172	13410	12255	0	1	635	96320
191	P12V	14175	10350	\$1,174	13235	12130	0	1	175	96495
192	P12V	14175	10350	\$1,130	13260	11675	0	1	455	96950
193	P12V	14175	10350	\$1,202	12725	10460	0	1	1215	98165
194	P12V	14175	10350	P12V	12650	10190	1	0	270	98435
195	P12V	12650	10590	P12V	12650	10190	1	0	1525	99960

Sequence	probeANet	ProbeA_X	ProbeA_Y	probeBNet	ProbeB_X	ProbeB_Y	Open	Short	IncremDistance	TotalDistance
196	\$1,202	16440	12190	\$1,202	12725	10460	1	0	3790	103750
197	\$1,202	16240	11925	\$1,202	12725	10460	1	0	265	104015
198	\$1,202	16240	11925	\$1,130	13260	11675	0	1	1215	105230
199	\$1,202	16240	11925	\$1,174	13235	12130	0	1	455	105685
200	\$1,202	16240	11925	\$1,172	13410	12255	0	1	175	105860
201	\$1,202	16240	11925	\$1,193	14045	12400	0	1	635	106495
202	\$1,202	16240	11925	\$1,104	14275	12370	0	1	230	106725
203	\$1,202	16240	11925	\$1,110	14505	12380	0	1	230	106955
204	\$1,202	16240	11925	5VD	14660	12565	0	1	185	107140
205	\$1,202	16240	11925	\$1,113	14660	12665	0	1	100	107240
206	\$1,202	16240	11925	N12V	14840	12083	0	1	582	107822
207	\$1,202	16240	11925	\$1,098	14760	11950	0	1	133	107955
208	\$1,202	16240	11925	\$1,103	15385	12280	0	1	625	108580
209	\$1,202	16240	11925	\$1,137	15905	12675	0	1	520	109100

Sequence	probeANet	ProbeA_X	ProbeA_Y	probeBNet	ProbeB_X	ProbeB_Y	Open	Short	IncramDistance	TotalDistance
210	\$1,202	16240	11925	\$1,157	16445	11975	0	1	700	109800
211	\$1,202	16240	11925	V REF	16345	11750	0	1	225	110025
212	\$1,202	16240	11925	P16	16105	11465	0	1	285	110310
213	\$1,202	16240	11925	\$1,099	15175	10500	0	1	965	111275
214	\$1,202	16240	11925	\$1,213	15855	10255	0	1	680	111955
215	\$1,202	16240	11925	\$1,143	16350	10050	0	1	495	112450
216	\$1,130	13260	11970	\$1,143	16350	10050	0	1	2980	115430
217	\$1,130	13260	11970	\$1,213	15855	10255	0	1	495	115925
218	\$1,130	13260	11970	\$1,099	15175	10500	0	1	680	116605
219	\$1,130	13260	11970	P16	16105	11465	0	1	965	117570
220	\$1,130	13260	11970	V REF	16345	11750	0	1	285	117855
221	\$1,130	13260	11970	\$1,157	16445	11975	0	1	225	118080
222	\$1,130	13260	11970	\$1,137	15905	12675	0	1	700	118780
223	\$1,130	13260	11970	\$1,103	15385	12280	0	1	520	119300

Sequence	probeANet	ProbeA_X	ProbeA_Y	probeBNet	ProbeB_X	ProbeB_Y	Open	Short	IncremDistance	TotalDistance
224	\$1,130	13260	11970	\$1,098	14760	11950	0	1	625	119925
225	\$1,130	13260	11970	N12V	14840	12083	0	1	133	120058
226	\$1,130	13260	11970	\$1,113	14660	12665	0	1	582	120640
227	\$1,130	13260	11970	5VD	14660	12565	0	1	100	120740
228	\$1,130	13260	11970	\$1,110	14505	12380	0	1	185	120925
229	\$1,130	13260	11970	\$1,104	14275	12370	0	1	230	121155
230	\$1,130	13260	11970	\$1,193	14045	12400	0	1	230	121385
231	\$1,130	13260	11970	\$1,172	13410	12255	0	1	635	122020
232	\$1,130	13260	11970	\$1,174	13235	12130	0	1	175	122195
233	\$1,130	13260	11970	\$1,130	13260	11675	1	0	455	122650
234	\$1,130	13260	11825	\$1,130	13260	11675	1	0	145	122795
235	\$1,174	13700	12300	\$1,174	13235	12130	1	0	475	123270
236	\$1,174	13700	12300	\$1,172	13410	12255	0	1	175	123445
237	\$1,174	13700	12300	\$1,193	14045	12400	0	1	635	124080

Sequence	probeANet	ProbeA_X	ProbeA_Y	probeBNet	ProbeB_X	ProbeB_Y	Open	Short	IncremDistance	TotalDistance
238	\$1,174	13700	12300	\$1,104	14275	12370	0	1	230	124310
239	\$1,174	13700	12300	\$1,110	14505	12380	0	1	230	124540
240	\$1,174	13700	12300	5VD	14660	12565	0	1	185	124725
241	\$1,174	13700	12300	\$1,113	14660	12665	0	1	100	124825
242	\$1,174	13700	12300	N12V	14840	12083	0	1	582	125407
243	\$1,174	13700	12300	\$1,098	14760	11950	0	1	133	125540
244	\$1,174	13700	12300	\$1,103	15385	12280	0	1	625	126165
245	\$1,174	13700	12300	\$1,137	15905	12675	0	1	520	126685
246	\$1,174	13700	12300	\$1,157	16445	11975	0	1	700	127385
247	\$1,174	13700	12300	V REF	16345	11750	0	1	225	127610
248	\$1,174	13700	12300	P16	16105	11465	0	1	285	127895
249	\$1,174	13700	12300	\$1,099	15175	10500	0	1	965	128860
250	\$1,174	13700	12300	\$1,213	15855	10255	0	1	680	129540
251	\$1,174	13700	12300	\$1,143	16350	10050	0	1	495	130035

Sequence	probeANet	ProbeA_X	ProbeA_Y	probeBNet	ProbeB_X	ProbeB_Y	Open	Short	IncremDistance	TotalDistance
252	\$1,172	13700	12400	\$1,143	16350	10050	0	1	100	130135
253	\$1,172	13700	12400	\$1,213	15855	10255	0	1	495	130630
254	\$1,172	13700	12400	\$1,099	15175	10500	0	1	680	131310
255	\$1,172	13700	12400	P16	16105	11465	0	1	965	132275
256	\$1,172	13700	12400	V REF	16345	11750	0	1	285	132560
257	\$1,172	13700	12400	\$1,157	16445	11975	0	1	225	132785
258	\$1,172	13700	12400	\$1,137	15905	12675	0	1	700	133485
259	\$1,172	13700	12400	\$1,103	15385	12280	0	1	520	134005
260	\$1,172	13700	12400	\$1,098	14760	11950	0	1	625	134630
261	\$1,172	13700	12400	N12V	14840	12083	0	1	133	134763
262	\$1,172	13700	12400	\$1,113	14660	12665	0	1	582	135345
263	\$1,172	13700	12400	5VD	14660	12565	0	1	100	135445
264	\$1,172	13700	12400	\$1,110	14505	12380	0	1	185	135630
265	\$1,172	13700	12400	\$1,104	14275	12370	0	1	230	135860

Sequence	probeANet	ProbeA_X	ProbeA_Y	probeBNet	ProbeB_X	ProbeB_Y	Open	Short	IncremDistance	TotalDistance
266	\$1,172	13700	12400	\$1,193	14045	12400	0	1	230	136090
267	\$1,172	13700	12400	\$1,172	13410	12255	1	0	635	136725
268	\$1,193	14850	12350	\$1,193	14045	12400	1	0	1150	137875
269	\$1,193	14850	12350	\$1,104	14275	12370	0	1	230	138105
270	\$1,193	14850	12350	\$1,110	14505	12380	0	1	230	138335
271	\$1,193	14850	12350	5VD	14660	12565	0	1	185	138520
272	\$1,193	14850	12350	\$1,113	14660	12665	0	1	100	138620
273	\$1,193	14850	12350	N12V	14840	12083	0	1	582	139202
274	\$1,193	14850	12350	\$1,098	14760	11950	0	1	133	139335
275	\$1,193	14850	12350	\$1,103	15385	12280	0	1	625	139960
276	\$1,193	14850	12350	\$1,137	15905	12675	0	1	520	140480
277	\$1,193	14850	12350	\$1,157	16445	11975	0	1	700	141180
278	\$1,193	14850	12350	V REF	16345	11750	0	1	225	141405
279	\$1,193	14850	12350	P16	16105	11465	0	1	285	141690

Sequence	probeANet	ProbeA_X	ProbeA_Y	probeBNet	ProbeB_X	ProbeB_Y	Open	Short	IncramDistance	TotalDistance
280	\$1,193	14850	12350	\$1,099	15175	10500	0	1	965	142655
281	\$1,193	14850	12350	\$1,213	15855	10255	0	1	680	143335
282	\$1,193	14850	12350	\$1,143	16350	10050	0	1	495	143830
283	\$1,104	15025	10625	\$1,143	16350	10050	0	1	1725	145555
284	\$1,104	15025	10625	\$1,213	15855	10255	0	1	495	146050
285	\$1,104	15025	10625	\$1,099	15175	10500	0	1	680	146730
286	\$1,104	15025	10625	P16	16105	11465	0	1	965	147695
287	\$1,104	15025	10625	V REF	16345	11750	0	1	285	147980
288	\$1,104	15025	10625	\$1,157	16445	11975	0	1	225	148205
289	\$1,104	15025	10625	\$1,137	15905	12675	0	1	700	148905
290	\$1,104	15025	10625	\$1,103	15385	12280	0	1	520	149425
291	\$1,104	15025	10625	\$1,098	14760	11950	0	1	625	150050
292	\$1,104	15025	10625	N12V	14840	12083	0	1	133	150183
293	\$1,104	15025	10625	\$1,113	14660	12665	0	1	582	150765

Sequence	probeANet	ProbeA_X	ProbeA_Y	probeBNet	ProbeB_X	ProbeB_Y	Open	Short	IncremDistance	TotalDistance
294	\$1,104	15025	10625	5VD	14660	12565	0	1	100	150865
295	\$1,104	15025	10625	\$1,110	14505	12380	0	1	185	151050
296	\$1,104	15025	10625	\$1,104	14275	12370	1	0	230	151280
297	\$1,104	15050	11950	\$1,104	14275	12370	1	0	1325	152605
298	\$1,110	14450	12770	\$1,110	14505	12380	1	0	820	153425
299	\$1,110	14675	12250	\$1,110	14505	12380	1	0	520	153945
300	\$1,110	14675	12250	5VD	14660	12565	0	1	185	154130
301	\$1,110	14675	12250	\$1,113	14660	12665	0	1	100	154230
302	\$1,110	14675	12250	N12V	14840	12083	0	1	582	154812
303	\$1,110	14675	12250	\$1,098	14760	11950	0	1	133	154945
304	\$1,110	14675	12250	\$1,103	15385	12280	0	1	625	155570
305	\$1,110	14675	12250	\$1,137	15905	12675	0	1	520	156090
306	\$1,110	14675	12250	\$1,157	16445	11975	0	1	700	156790
307	\$1,110	14675	12250	V REF	16345	11750	0	1	225	157015

Sequence	probeANet	ProbeA_X	ProbeA_Y	probeBNet	ProbeB_X	ProbeB_Y	Open	Short	IncremDistance	TotalDistance
308	\$1,110	14675	12250	P16	16105	11465	0	1	285	157300
309	\$1,110	14675	12250	\$1,099	15175	10500	0	1	965	158265
310	\$1,110	14675	12250	\$1,213	15855	10255	0	1	680	158945
311	\$1,110	14675	12250	\$1,143	16350	10050	0	1	495	159440
312	5VD	15935	12235	\$1,143	16350	10050	0	1	1260	160700
313	5VD	15935	12235	\$1,213	15855	10255	0	1	495	161195
314	5VD	15935	12235	\$1,099	15175	10500	0	1	680	161875
315	5VD	15935	12235	P16	16105	11465	0	1	965	162840
316	5VD	15935	12235	V REF	16345	11750	0	1	285	163125
317	5VD	15935	12235	\$1,157	16445	11975	0	1	225	163350
318	5VD	15935	12235	\$1,137	15905	12675	0	1	700	164050
319	5VD	15935	12235	\$1,103	15385	12280	0	1	520	164570
320	5VD	15935	12235	\$1,098	14760	11950	0	1	625	165195
321	5VD	15935	12235	N12V	14840	12083	0	1	133	165328

Sequence	probeANet	ProbeA_X	ProbeA_Y	probeBNet	ProbeB_X	ProbeB_Y	Open	Short	IncremDistance	TotalDistance
322	5VD	15935	12235	\$1,113	14660	12665	0	1	582	165910
323	5VD	15935	12235	5VD	14660	12565	1	0	100	166010
324	\$1,113	16160	12275	\$1,113	14660	12665	1	0	225	166235
325	\$1,113	14800	12245	\$1,113	14660	12665	1	0	1360	167595
326	\$1,113	14800	12245	N12V	14840	12083	0	1	582	168177
327	\$1,113	14800	12245	\$1,098	14760	11950	0	1	133	168310
328	\$1,113	14800	12245	\$1,103	15385	12280	0	1	625	168935
329	\$1,113	14800	12245	\$1,137	15905	12675	0	1	520	169455
330	\$1,113	14800	12245	\$1,157	16445	11975	0	1	700	170155
331	\$1,113	14800	12245	V REF	16345	11750	0	1	225	170380
332	\$1,113	14800	12245	P16	16105	11465	0	1	285	170665
333	\$1,113	14800	12245	\$1,099	15175	10500	0	1	965	171630
334	\$1,113	14800	12245	\$1,213	15855	10255	0	1	680	172310
335	\$1,113	14800	12245	\$1,143	16350	10050	0	1	495	172805

Sequence	probeANet	ProbeA_X	ProbeA_Y	probeBNet	ProbeB_X	ProbeB_Y	Open	Short	IncremDistance	TotalDistance
336	N12V	15100	11275	\$1,143	16350	10050	0	1	970	173775
337	N12V	15100	11275	\$1,213	15855	10255	0	1	495	174270
338	N12V	15100	11275	\$1,099	15175	10500	0	1	680	174950
339	N12V	15100	11275	P16	16105	11465	0	1	965	175915
340	N12V	15100	11275	V REF	16345	11750	0	1	285	176200
341	N12V	15100	11275	\$1,157	16445	11975	0	1	225	176425
342	N12V	15100	11275	\$1,137	15905	12675	0	1	700	177125
343	N12V	15100	11275	\$1,103	15385	12280	0	1	520	177645
344	N12V	15100	11275	\$1,098	14760	11950	0	1	625	178270
345	N12V	15100	11275	N12V	14840	12083	1	0	133	178403
346	\$1,098	16175	11825	\$1,098	14760	11950	1	0	1075	179478
347	\$1,098	16040	10875	\$1,098	14760	11950	1	0	950	180428
348	\$1,098	16040	10875	\$1,103	15385	12280	0	1	625	181053
349	\$1,098	16040	10875	\$1,137	15905	12675	0	1	520	181573

Sequence	probeANet	ProbeA_X	ProbeA_Y	probeBNet	ProbeB_X	ProbeB_Y	Open	Short	IncremDistance	TotalDistance
350	\$1,098	16040	10875	\$1,157	16445	11975	0	1	700	182273
351	\$1,098	16040	10875	V REF	16345	11750	0	1	225	182498
352	\$1,098	16040	10875	P16	16105	11465	0	1	285	182783
353	\$1,098	16040	10875	\$1,099	15175	10500	0	1	965	183748
354	\$1,098	16040	10875	\$1,213	15855	10255	0	1	680	184428
355	\$1,098	16040	10875	\$1,143	16350	10050	0	1	495	184923
356	\$1,103	15385	12210	\$1,143	16350	10050	0	1	1335	186258
357	\$1,103	15385	12210	\$1,213	15855	10255	0	1	495	186753
358	\$1,103	15385	12210	\$1,099	15175	10500	0	1	680	187433
359	\$1,103	15385	12210	P16	16105	11465	0	1	965	188398
360	\$1,103	15385	12210	V REF	16345	11750	0	1	285	188683
361	\$1,103	15385	12210	\$1,157	16445	11975	0	1	225	188908
362	\$1,103	15385	12210	\$1,137	15905	12675	0	1	700	189608
363	\$1,103	15385	12210	\$1,103	15385	12280	1	0	520	190128

Sequence	probeANet	ProbeA_X	ProbeA_Y	probeBNet	ProbeB_X	ProbeB_Y	Open	Short	IncremDistance	TotalDistance
364	\$1,137	15905	12525	\$1,137	15905	12675	1	0	520	190648
365	\$1,137	15905	12825	\$1,137	15905	12675	1	0	300	190948
366	\$1,137	15905	12825	\$1,157	16445	11975	0	1	700	191648
367	\$1,137	15905	12825	V REF	16345	11750	0	1	225	191873
368	\$1,137	15905	12825	P16	16105	11465	0	1	285	192158
369	\$1,137	15905	12825	\$1,099	15175	10500	0	1	965	193123
370	\$1,137	15905	12825	\$1,213	15855	10255	0	1	680	193803
371	\$1,137	15905	12825	\$1,143	16350	10050	0	1	495	194298
372	\$1,157	16280	11850	\$1,143	16350	10050	0	1	975	195273
373	\$1,157	16280	11850	\$1,213	15855	10255	0	1	495	195768
374	\$1,157	16280	11850	\$1,099	15175	10500	0	1	680	196448
375	\$1,157	16280	11850	P16	16105	11465	0	1	965	197413
376	\$1,157	16280	11850	V REF	16345	11750	0	1	285	197698
377	\$1,157	16280	11850	\$1,157	16445	11975	1	0	225	197923

Sequence	probeANet	ProbeA_X	ProbeA_Y	probeBNet	ProbeB_X	ProbeB_Y	Open	Short	IncramDistance	TotalDistance
378	V REF	16090	11750	V REF	16345	11750	1	0	225	198148
379	V REF	16395	11700	V REF	16345	11750	1	0	305	198453
380	V REF	16395	11700	P16	16105	11465	0	1	285	198738
381	V REF	16395	11700	\$1,099	15175	10500	0	1	965	199703
382	V REF	16395	11700	\$1,213	15855	10255	0	1	680	200383
383	V REF	16395	11700	\$1,143	16350	10050	0	1	495	200878
384	P16	15490	10255	\$1,143	16350	10050	0	1	1445	202323
385	P16	15490	10255	\$1,213	15855	10255	0	1	495	202818
386	P16	15490	10255	\$1,099	15175	10500	0	1	680	203498
387	P16	15490	10255	P16	16105	11465	1	0	965	204463
388	\$1,099	15175	10385	\$1,099	15175	10500	1	0	965	205428
389	\$1,099	15100	11125	\$1,099	15175	10500	1	0	740	206168
390	\$1,099	15100	11125	\$1,213	15855	10255	0	1	680	206848
391	\$1,099	15100	11125	\$1,143	16350	10050	0	1	495	207343

Sequence	probeANet	ProbeA_X	ProbeA_Y	probeBNet	ProbeB_X	ProbeB_Y	Open	Short	IncremDistance	TotalDistance
392	\$1,213	14250	10460	\$1,143	16350	10050	0	1	850	208193
393	\$1,213	14250	10460	\$1,213	15855	10255	1	0	495	208688
394	\$1,143	16429	10050	\$1,143	16350	10050	1	0	2179	210867