Base Station Selection in Mobile Networks

Adam Whitter-Jones
0924607
Supervisor: Prof. Steve Hurley

For the Qualification of:
BSc Computer Science with Distributed and Mobile Networks
At:
Cardiff University School of Computer Science and Informatics

28th April 2012

Acknowledgements

I would like to thank my personal tutor Steve Hurley for doing his utmost and helping me, whilst providing me with answers to all the questions that I have asked. I would also like to thank Matthew Morgan for helping me stay focused and showing me that there is support at the University in stressful situations. Finally, I would like to thank Richard Falconer for talking me through the logic of my coding problems and Rachel Sansom for tirelessly proof reading this document.

Abstract

This report has been written to document the design and implementation of a simulated annealing (SA) algorithm to aid the design of a mobile network. The algorithm is designed to choose the best sites to be on or off to maximise coverage, whilst reducing the cost of having many sites on. Default initial power is assigned when the application is first loaded, but these can be changed either to random power for random sites, uniform power for all sites or all sites can be turned off. The application presents a visualisation of the network and algorithm results so that coverage can be identified. The implementation was successful with the results of the algorithm detailed towards the end of this report.

Table of Contents

1.0 Introduction 1

  1.1 Assumptions Made 3

  1.2.0 Algorithm Research 3

    1.2.1 Hill Climbing 3

    1.2.2 Tabu Search 4

    1.2.3 Simulated Annealing 4

  1.3 Algorithm Choice 5

2.0 Background 6

  2.1 Overview 8

  2.2.0 Insights 8

    2.2.1 Picture Box 9

    2.2.2 PLM 9

    2.2.3 RTP/TTP 9

    2.2.4 Threads 10

3.0 Design 11

  3.1 GUI 11

  3.2 Data Flow 13

  3.3 Data Structures 13

  3.4 Algorithm 14

4.0 Implementation 16

  4.1 Reading Files 16

  4.2 Visualising Network 18

  4.3 Checking Coverage/Iterating list 20

  4.4.0 Algorithm 21

  4.4.1 Simulated Annealing Pseudo Code 22

  4.5 Threading 24

  4.6 Data Structures 25

5.0 Results 26

  5.1 Performance: Loading PLM 26

  5.2 Coverage Results: Uniform Power 25 27

  5.3 Coverage Results: Random Power 28

5.4 Algorithm Results: Fixed Sites Uniform Power 25 29
Attempt 1....................29
Attempt 2....................30
Attempt 3....................30
5.5.0 Algorithm Results: Random Sites & Powers 31
Attempt 1....................31
Attempt 2....................32
Attempt 3....................32
Attempt 4....................33
Attempt 5....................33
6.0 Future Work....................35
6.1 RTP/TTP 35
6.2 Efficiency 35
6.3 Interface 36
6.3.1 Customisation & Accessibility....................36
6.4 Exporting Data....................37
6.5 Programming Language 37
7.0 Conclusions 38
7.1 GUI....................38
7.2 Algorithm....................38
8.0 Reflection on Learning 40
Appendices....................I
Appendix A I
Code Attempt 1 I
Code Attempt 2 III
Appendix B V
Appendix C VII
Appendix D VIII
Glossary....................IX
Table of Abbreviations....................XI
Bibliography XII

1.0 Introduction

The introduction of this report aims to discuss the reason for which the project has been undertaken whilst discussing the beneficiaries of the work completed. In addition to this, the information held in the data files that are used is explained with its relevance to mobile networks explained diagrammatically. Important expected outcomes, assumptions and research are also detailed here.

This project has been undertaken in order to aid and, hopefully improve, the planning process to decide the most appropriate combination of active and inactive Base Station Transceivers (BSTs) when deploying a mobile phone network. BSTs are one of the key components required in a mobile phone network and permit end users to connect via their mobile telephone to other resources or people around the world. The issue that is encountered when planning these networks is both an issue of cost and also of signal propagation. Simply, the key question asked which will be solved by an algorithm is how can we place the minimum number of telephone masts (BSTs) and serve the maximum number of customers?

Through creating this application, the beneficiaries are both network providers and the customers that use their networks. Enabling a network service provider to plan and deploy a network that is as efficient as possible reduces time and cost. This would, hopefully, reduce corporate expenditure on some of the most costly operations that would filter down to the users of the network, potentially reducing the charges incurred for using the services provided. In addition to this reduction in cost, effective planning of a network is essential to serve as much of the population as possible and provide the fastest speeds (3). As we can see from several recent news articles, the digital age is creating fragmentation between urban and rural areas (6) (4). Communication and the internet are key factors in enabling businesses to grow and develop.

As well as these main beneficiaries, through taking the application away from a business perspective, it is possible to identify two additional beneficiaries:

1. The author – the act of creating this application and having to understand its nature has helped to develop an understanding for the way in which mobile networks operate and optimisation algorithms that can be used within network design and management. Additionally, the use of VB.Net for the project has helped to develop programming techniques and enhanced previous knowledge of the language.
2. Students – The author hopes that anyone researching mobile network planning will be able to see this report and the accompanying application so that they too can begin to understand the way in which mobile networks are planned and the methods that have been used to show this.

To assess and, hopefully, assist this issue, raw data relating to these BSTs has been obtained. We have several different plain text files from network operator Vodafone. These are as follows:

1. Reception Test Points (RTP) listed by their (x,y) coordinates.
2. Propagation Losses Matrix (PLM) which contains propagation losses (in dB) from each potential site to every RTP.
3. Service Test Points (STP) to which a known service is required (dBm) listed by their (x,y) coordinates.
4. Traffic Test Points (TTP) in which traffic demand is known (in Erlangs) and is listed by their (x,y) coordinates.
5. Mobile and Base Station Antenna Information (NET) in which the names and locations of sites are specified.

The data in the PLM is representative of Effective Isotropic Radiated Power (EIRP) associated with our masts and coordinates. EIRP is the amount of loss or gain a signal has taking in to account certain factors.

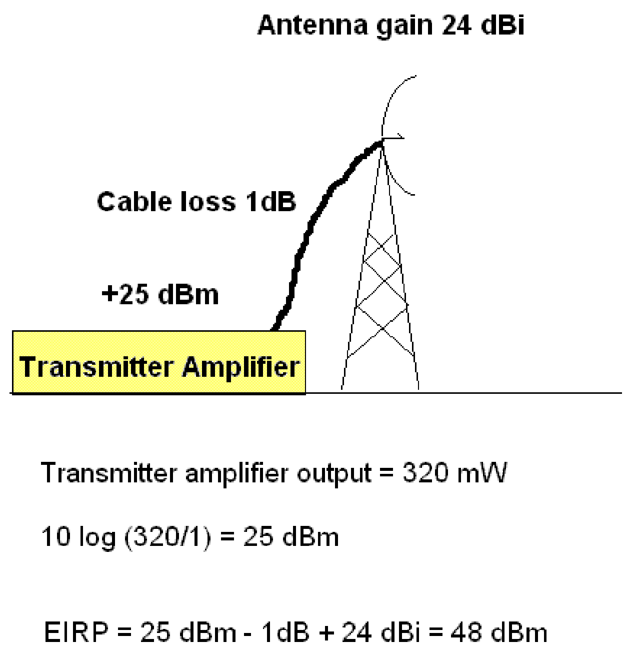


Figure 1 Example of EIRP (13)

The most important outcomes that have been completed are as follows:

- The need to carry out the simulated annealing algorithm efficiently.
- Hence, produce accurate results that can be viewed.
- Furthermore, allow user to interact with data and manipulate it as they wish, so that they can learn and understand the effects that different properties have on the network.

The report is structured in a way that includes insights discussed as a sub sections following this introductory information and interim report. The insights sub section details issues encountered and decisions that have since been made to rectify or resolve problems that have arisen since the interim report. Following from this is the design section that gives a high level overview of the application, its structure and the way data flows and is partitioned throughout the system. The section succeeding this is implementation; the aim of this section of the report is to expand and discusses the points raised in the previous design section from a more detailed low level perspective. After the sections discussing the theory and practical aspects of the application, the results of the algorithms are listed and evaluated as is information relating to performance enhancements achieved throughout the coding process.

Due to the limit on the amount of time available to write this application, there is also a section at the end of the document detailing future work that the author feels would be beneficial to the application. This is extended by a conclusions section and further text detailing the author's reflection on the project as a whole and what has been learnt.

1.1 Assumptions Made

- Hard coded data; i.e. files must be in root dir.
- User must be running Windows.

1.2.0 Algorithm Research

The aim of this project is to find an optimal solution of a pre-defined network whilst being able to provide this resolution within a reasonable amount of time. As a result, research on multiple different optimisation algorithms has been completed. A brief summary of each of these algorithms is detailed below.

1.2.1 Hill Climbing

Hill Climbing (HC) is an iterative algorithm that starts with a random solution to the problem in question. From this arbitrary solution, attempts are made to find better solution(s) by incrementing single elements within the solution. If this increment provides a better solution than the previous one, the incremental solution becomes the new solution and this process is repeated until no improvements can be made.

Hill climbing is effective for finding a local optimum but not necessarily the best possible solution, also known as a global optimum. A local optimum is a solution that cannot be improved by considering a neighbouring configuration. A characteristic of HC is that only local optima are guaranteed. This can be cured through using repeated local searches (or restarts) or alternatively more complex schemes based on iterations such as iterated local search, tabu search and on memory-less stochastic modifications such as simulated annealing.

The simplicity of the algorithm has made it a popular choice amongst the many different optimisation algorithms. HC is widely used in artificial intelligence as it can reach a goal state from any starting node although according to an article online (14), "more advanced algorithms such as simulated annealing or tabu search may give better results, in some situations hill climbing works just as well." HC can often produce better results than those found in other algorithms, for example when the time available to perform the calculations is limited, as in real time systems. As HC is an "anytime algorithm" it can provide a solution even if it is stopped before it was specified to end.

1.2.2 Tabu Search

Tabu Search (TS) utilises a local/neighbourhood search procedure to move from solution A, to a better solution, B, iteratively within the search space until a stopping criterion (such as maximum attempt limit or minimum score threshold) has been met.

The downfall of TS is that local search procedures often become stuck in poor scoring areas or an area within the search space where scores plateau. To avoid this and enable the algorithm to explore other areas of the search space that would otherwise be ignored, TS explores the neighbourhood of each solution throughout its progress. The solutions of the new neighbourhood are determined through memory structures that are referred to as the "tabu list". This list contains information on banned solutions such as solutions that have been visited in the past. The list is usually short term, for example only holding information relating to n-iterations ago. N is called the tabu tenure; this is the number of previous solutions to be stored.

"Current applications of TS span the realms of resource planning, telecommunications, VLSI design, financial analysis, scheduling, space planning, energy distribution, molecular engineering, logistics, pattern classification, flexible manufacturing, waste management, mineral exploration, biomedical analysis, environmental conservation and scores of others." (9)

1.2.3 Simulated Annealing

Simulated Annealing (SA) is a probabilistic meta-heuristic for the global optimisation problem of discovering an effective approximation to the global optimum of a given function in a large search space. "For certain problems, simulated annealing may be more efficient than exhaustive enumeration — provided that the goal is merely to find an acceptably good solution in a fixed amount of time, rather than the best possible solution."

Through each of the iterations in the SA heuristic, a neighbouring state of the current state is considered. These neighbour states are produced through altering the current state in a particular way, similar to HC; this action is called a move. The aim of the move is to produce minimal iterations to the current solution. As a result, different moves produce different neighbours. The searching of these neighbours is the critical part for optimisation as the final solution will only come from a tour of the successive neighbours generated from moves. Many simple heuristics move by finding the best neighbour from each successive move and stop when a solution is generated with no better neighbours. The issue with this approach is that the neighbours of a state have no guarantee that they contain better solutions. Hence failure to find a better solution does not guarantee that no better solution exists. As a result, the solution that the SA generates is called a local optimum whereas the best possible solution is referred to as the global optimum. “Meta-heuristics use the neighbours of a state as a way to explore the solutions space and can accept worse solutions in their search in order to accomplish that. This means that the search will not get stuck to a local optimum and if the algorithm is run for an infinite amount of time, the global optimum will be found.” (17)

If the new solution is better than the existing solution, it is accepted and becomes the current solution. If the new solution is worse than the current solution, it is probabilistically decided whether or not to move to the new state; this can lead the heuristic to move to solutions that are worse than the current solution. These steps are repeated until the system reaches a state that is good enough for the application or until the computation budget has been depleted.

1.3 Algorithm Choice

Many of the research papers listed indicate that SA produces the best results in a set amount of time comparative to HC and TS. This decision, combined with the fact that both SA and TS are extensions of HC have led to the decision for the application to incorporate a SA annealing algorithm both in terms of educational opportunity and the efficiency of generating results.

2.0 Background

The background section of this report aims to give a general overview about the way in which mobile networks work and the need for effective network planning. The data sets that have been used in this project will be discussed as will any insights that have been made since the interim report.

Recently we have seen a global increase in the technology and subscribers of wireless technology. Mobile network providers have realised the necessity of efficient design and planning of these networks and the importance of a stable and reliable service. Therefore, support offered to assist in this planning and development of said networks has increased. As the use of wireless technology grows year on year, the necessity for efficiency is something network providers continually strive for.

A key component of mobile networks is the Base Station Transceiver (BST). The purpose of the BST is to relay information from the network to a user's mobile device, or alternatively, from the user's mobile device back to the base station controller (BSC) to be sent across the home network to another user or it is passed from the BSC to the mobile switching centre (MSC) to be sent across the internet in the instance where the recipient is on a different mobile network.

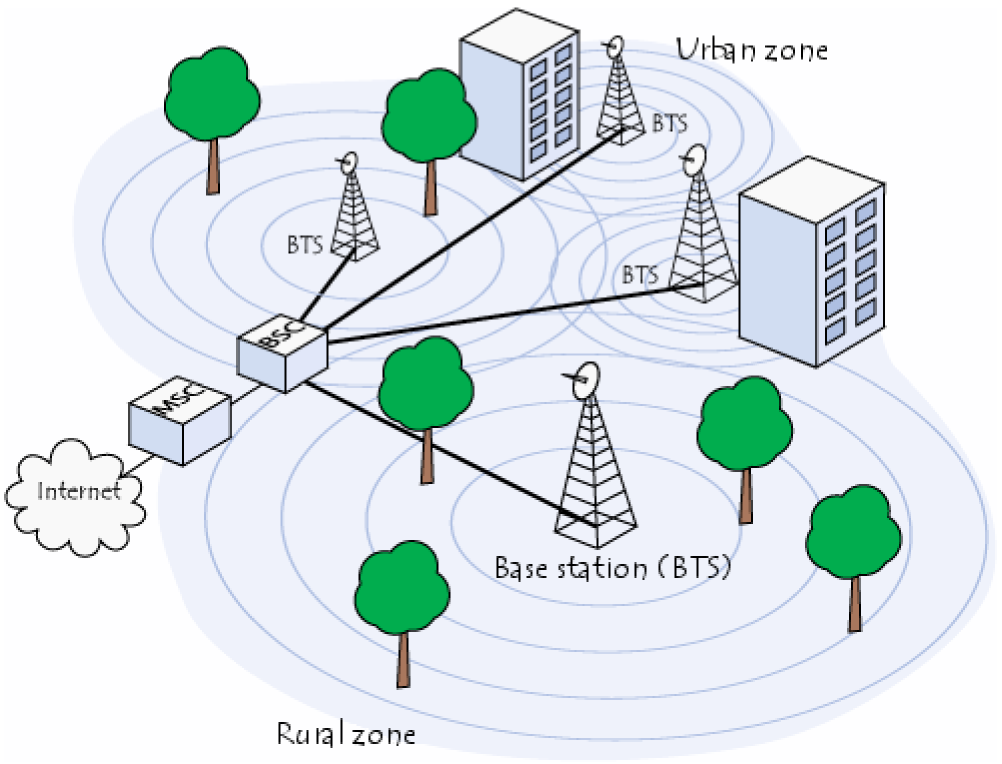


Figure 2 Graphic representation of a mobile network (15)

Every BST generates a cell of coverage that can be obtained from mobile devices within range. Depending on the type of cell and location, different amounts of coverage can be received as detailed below:

- Mega – tens of kilometres
- Macro – hundreds of meters to a few kilometres
- Micro – around two kilometres
- Pico – 200 metres or less
- Femto (Home-Cell) – around 10 metres

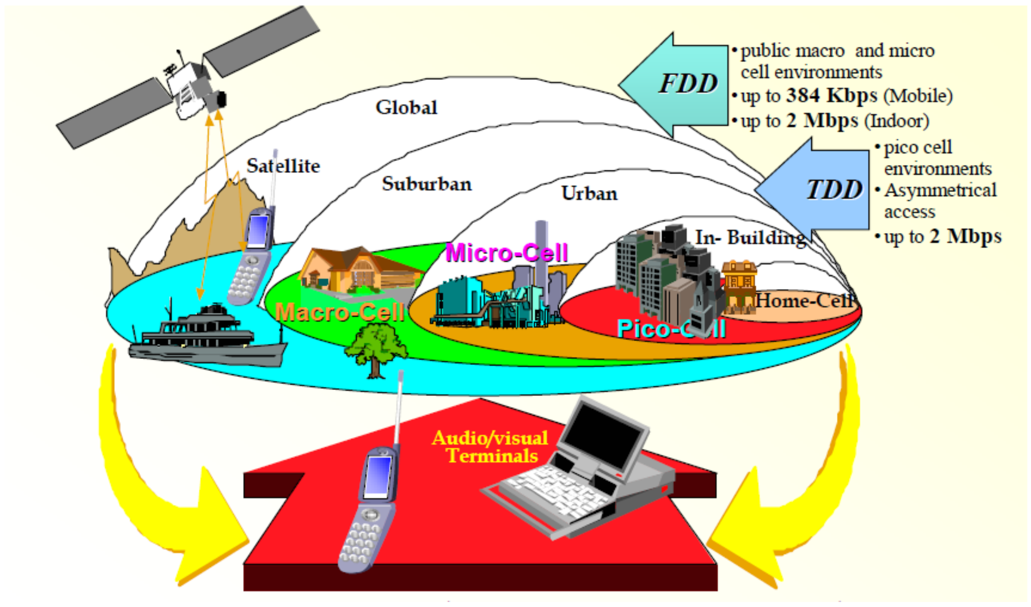


Figure 3 Graphic representation of mobile cell coverage (16)

To get optimum levels of coverage, the cells should overlap.

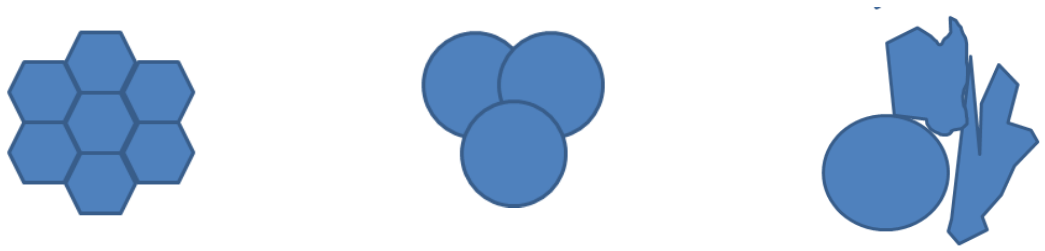


Figure 4 A graphic representations of theoretical, optimum and actual cell coverage.

The problem that we face is that if we turn on every BST, although this would provide us with maximum coverage, we would also encounter a large cost associated with having every BST on. Furthermore, providing a large power to any BST is said to results in a large emission of electromagnetic radiation affecting peoples' health (1), although these facts are disputed (2) (8).

To help provide an optimum solution in relation to which sites are on and off with haste, developing an algorithm to calculate and predict these results will help us to visualise the required information during the planning stage. This permits for a reduction in costs for mobile network providers whilst providing minimum disruption for service users.

2.1 Overview

The data set that I am using relates to 250 real sites, which is a fraction of the true amount (25k+) masts that exist in the United Kingdom. The site information I am working with is two dimensional and is plotted on a simple grid within Visual Basic .NET that spans to the maximum size of the dataset. The maximum size of the grid I am working with is determined by reading the RTP file and calculating the highest and lowest values from the data. Within the grid, BSTs, STPs and TTPs are plotted relative to the chosen grid size, currently 750 pixels. Taking the real life (x,y) coordinates of the BST, STP and TTP data, I have then manipulated these through the calculations:

```
x = ((x - BaseStation.minimumX) / (BaseStation.maximumX - BaseStation.minimumX))
y = ((y - BaseStation.minimumY) / (BaseStation.maximumY - BaseStation.minimumY))
```

Figure 5 Code to calculate the (x,y) values relative to the screen size

Where x and y relate to the location of the sites in pixels, x and y are the (x,y) real world coordinates of the sites read from the RTP file and minimum/maximum x/y are the real world (x,y) coordinates calculated from the RTP file.

It is important to realise at this stage that plotting the data in a two dimensional plane may be slightly unrealistic in that the lack of visualisation of terrain and physical objects in the x plane does not help us to determine any factors that may be reducing the strength of signal and coverage area of a specific BST.

2.2.0 Insights

This section provides an overview of issues that have been encountered during the coding stages of the application. As a result of these issues, executive decisions have been *made on how to rectify them; either through changing the code that has been written or through amending the files that contain the data (mainly the PLM) that were provided at the start of the project.*

2.2.1 Picture Box

During the first few weeks creating the application, one of the main performance issues which resulted in a reduced user experience was the time that it took to draw the network and any calculations such as the path loss for each site on the grid. As this performance issue severely impinged the operability of the application, an investigation commends to find the most effective way to improve the performance. The result was to draw the image in a picture box control; this control already has the performance adaptations required to draw efficiently as it is a Windows common control. To determine the amount of time saved, a `Date = Now` variable was included to check at the start and end of routines and calculated the difference to find out the time taken. This check has been included in many subsequent functions and subroutines to get an accurate reflection of the affect the changes made to the code have on the performance time. As a result, the image now renders almost immediately compared to the 12 seconds that were recorded previously.

2.2.2 PLM

It has been decided since the interim report that we would reduce the number of sites used in the path loss matrix. This decision was made due to the amount of time it took to read the initial PLM file, especially over multiple debugging sessions. To eliminate the fact that it might be Visual Basic .NET that was slowing the performance down, I decided to create a separate application to read the files in C++ (Appendix A), however, this still suffered similar performance issues. Further to this, I wrote three separate methods to read the file in VB.Net. Two methods used a StreamReader and one method used a ByteReader to read the files in to the application. Despite MSDN (Microsoft Developer Network) stating that the StreamReader was faster, I decided to do my own tests; as results varied on each occasion, I took an average of three and came to the conclusion that StreamReader was indeed faster than ByteReader, but this did not increase my performance. From these results, it was apparent that the multiple string operations in use to read the text could be a main factor in the performance deficiency. As a result, I decided to test the difference between using a SubString and StringArray functions. As the SubString function needs to create a StringArray for each SubString call, the most efficient way to carry out the comparisons was through using the StringArray function (Appendix B). Although this increased performance somewhat, the difference was not major; hence, reducing the sites to a tenth of their original size was the only option. Although this is less than desirable, the ability to have the complete list of sites is not far removed; through editing a few lines of code it could be included again in the application, however, it takes the loading time to over five minutes.

2.2.3 RTP/TTP

Due to limitations on the amount of time provided and the lack of progress that had been made towards the start of the project, detailed in the interim report, two pieces of data provided for use in the application (traffic test points and reception test points) are not used at all in any of the calculations within the application. The author feels that the most important aspect to be demonstrated through the application is the affect that turning sites on or off, or amending the site power, has on the overall network coverage. Although utilising all of the data read in the application was the most desirable outcome, prioritising the TTP or RTP data over the STP data would have meant that calculations on the capacity of the network would have been possible but this would have jeopardised the ability to implement the algorithms to calculate the best solution for network coverage.

Relating back to the interim report, the initial calculation for the Network Performance Measure (NPM) cannot be referred to as expected. The calculation specified is as follows:

NPM = % coverage (max) + %traffic/service (max) + cost (# of sites) [min]

Figure 6 Preferred calculation used to determine the NPM

Although the application currently maximises coverage and minimises the sites through the simulated annealing algorithm, the %traffic/service part of the calculation is currently omitted. Hence, the goal of providing an optimum solution given the above formula is not possible. Instead, the application aims to provide an optimum solution for minimising site cost and maximising coverage based on a reviewed formula of:

NPM = % coverage (max) + cost (# of sites) [min]

Figure 7 Actual calculation used to determine the NPM

This result is displayed on the main window of the application after the algorithms and calculations have been completed.

Coverage is determined by sites served/total sites and the cost of sites is determined by (total sites – active sites)/total sites. This produces a number between one and two.

2.2.4 Threads

Creating this application demonstrates the importance of using threads. Without using threads, the interface turns white when loading the large data files and does not provide any indication of what is happening. Additionally, when performing calculations on the main window, the window would often freeze and the task manager would ask the user if they wish to exit the application because it was not responding. The ability for the user to still interact with certain parts of the application whilst other important calculations are being completed in the background is paramount to ensuring the user's experience is pleasant. Building on these user centric issues, the use of threading also permits for the application to utilise parallel processing instead of sequential processing. Parallel processing assigns a single thread to each task in the loop, resulting in significant performance improvements when iterating through multi-dimensional lists.

3.0 Design

The design section of this report will give a general overview of how the application looks and is intended to work, whilst also discussing the way in which data flows through the system. Additionally, a high level explanation of the way in which the data is stored and the algorithms chosen to achieve the desired results will be given towards the end of this section.

As detailed in the abstract section of the interim report, the aim of this piece of software is to "help with planning and testing the positioning of telephone masts, referred to as BSTs within the telecommunications industry." After working on this project, it seems logical to deduce that the application is not efficient or powerful enough to manipulate the vast amounts of data in a live network and is therefore not suitable for planning a mobile network. Conversely, the application could instead be viewed as a teaching utility to understand the concepts of network design and performance algorithms associated with this industry.

From this specification, the current software that is in place meets these requirements in that the interface allows user interaction to assess the impact on signal propagation by activating different sites and assigning new powers. Further, the application also shows the user the most efficient sites to have on or off to provide maximum coverage by using the least sites.

Overall, this application is designed as more of a diagrammatic tool than a definitive network planning application. The ultimate aim was to demonstrate to the user the affect that different powers have on a site's signal and to show how an algorithm can be used effectively; to show the most efficient way to serve the most people whilst using the least sites. The application successfully visualises on the grid the effect of changing site powers or turning the site on or off.

3.1 GUI

The user interface as it stands is relatively simple, maintaining the generic look and feel of Windows form controls. It seemed an unnecessary waste of resources, given the time constraints of the project, to concentrate on the aesthetics of the program; however, there are enhancements to the initial plan that have been included in the application. One feature that was not detailed in the interim report was the incorporation of a splash screen. The author felt this was important for when the application was starting due to the amount of data that needs to be loaded. To provide some user feedback and reassurance, a timer and progress bar have been included on the splash screen to feed back to the user how much of the PLM has been read. The timer tickets every 500ms and checks a global variable to update the progress bar. Without this, the program appeared unresponsive on the initial loading of the required files.

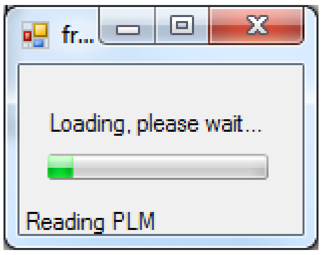
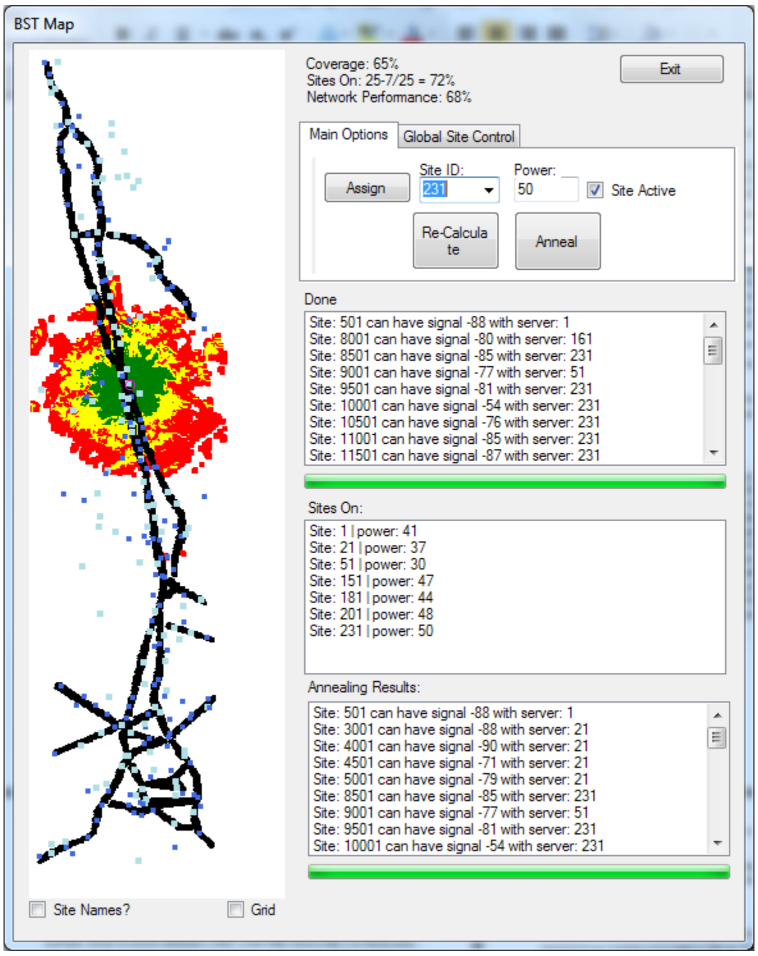


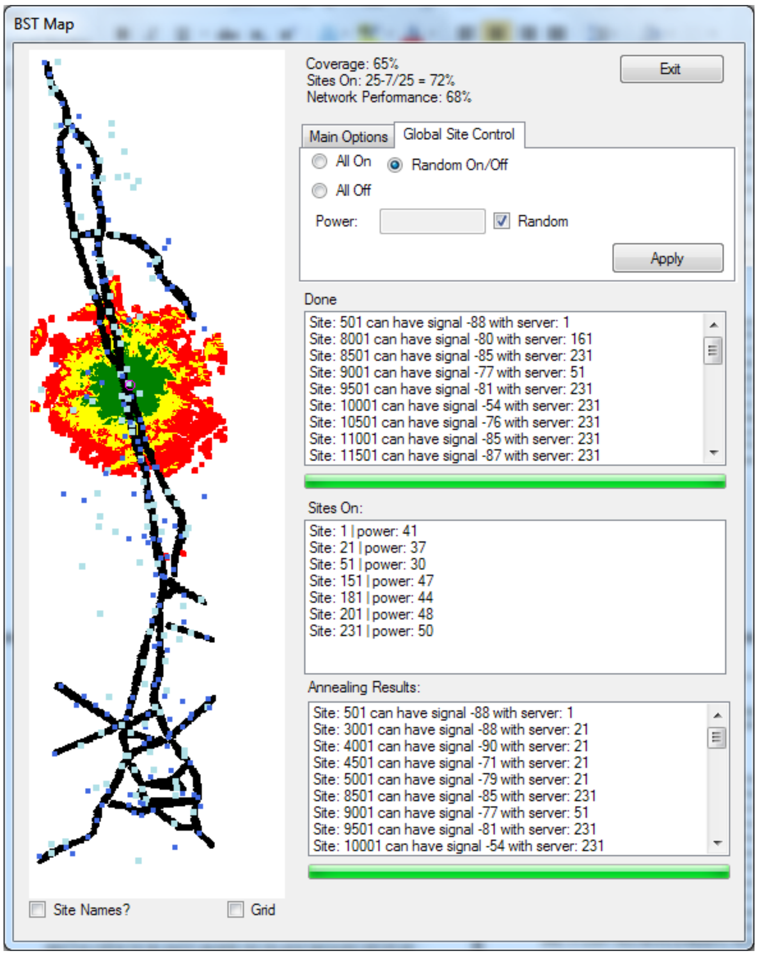
Figure 8 Initial Splash Screen

On the main screen there are several options included to try and help the user's experience. These are simple options to allow for more effective interaction with the application such as showing the active site names, turning the grid on or off and adding check boxes for active sites. Further, the incorporation of the combo box detailing the sites in the PLM allows the user to scroll through and see the resulting signal propagation on the grid. There is also the option on this screen to set all sites to on or off, to designate a uniform power to all sites and also to randomize the power at each site.



```
Coverage: 65%
Sites On: 25-7/25 = 72%
Network Performance: 68%

Site ID: 231    Power: 50    [✓] Site Active

Done
Site: 501 can have signal -88 with server: 1
Site: 8001 can have signal -80 with server: 161
Site: 8501 can have signal -85 with server: 231
Site: 9001 can have signal -77 with server: 51
Site: 9501 can have signal -81 with server: 231
Site: 10001 can have signal -54 with server: 231
Site: 10501 can have signal -76 with server: 231
Site: 11001 can have signal -85 with server: 231
Site: 11501 can have signal -87 with server: 231

Sites On:
Site: 1 | power: 41
Site: 21 | power: 37
Site: 51 | power: 30
Site: 151 | power: 47
Site: 181 | power: 44
Site: 201 | power: 48
Site: 231 | power: 50

Annealing Results:
Site: 501 can have signal -88 with server: 1
Site: 3001 can have signal -88 with server: 21
Site: 4001 can have signal -90 with server: 21
Site: 4501 can have signal -71 with server: 21
Site: 5001 can have signal -79 with server: 21
Site: 8501 can have signal -85 with server: 231
Site: 9001 can have signal -77 with server: 51
Site: 9501 can have signal -81 with server: 231
Site: 10001 can have signal -54 with server: 231
```

Figure 9 Main Screen



```
Coverage: 65%
Sites On: 25-7/25 = 72%
Network Performance: 68%

All On    [✓ Random On/Off]
All Off
Power:            [✓] Random

Done
Site: 501 can have signal -88 with server: 1
Site: 8001 can have signal -80 with server: 161
Site: 8501 can have signal -85 with server: 231
Site: 9001 can have signal -77 with server: 51
Site: 9501 can have signal -81 with server: 231
Site: 10001 can have signal -54 with server: 231
Site: 10501 can have signal -76 with server: 231
Site: 11001 can have signal -85 with server: 231
Site: 11501 can have signal -87 with server: 231

Sites On:
Site: 1 | power: 41
Site: 21 | power: 37
Site: 51 | power: 30
Site: 151 | power: 47
Site: 181 | power: 44
Site: 201 | power: 48
Site: 231 | power: 50

Annealing Results:
Site: 501 can have signal -88 with server: 1
Site: 3001 can have signal -88 with server: 21
Site: 4001 can have signal -90 with server: 21
Site: 4501 can have signal -71 with server: 21
Site: 5001 can have signal -79 with server: 21
Site: 8501 can have signal -85 with server: 231
Site: 9001 can have signal -77 with server: 51
Site: 9501 can have signal -81 with server: 231
Site: 10001 can have signal -54 with server: 231
```

Figure 10 Tabbed Control to Adjust all Sites

3.2 Data Flow

There are many different sub routines within the application. To begin, the application loads the splash screen and enters the thread to load the files via the modFiles module. These files are read, and the data is stored into the lists through the modData module. The structure of lists defined in the modData module is done so in relation to their classes. For example, the BST structure is detailed in clsBST. Once these files are read in to memory and the initial thread has been completed, the main form is loaded and the data is painted in to the picture box that contains the grid. To draw the data on the screen many of the relevant lists are iterated through the results that are drawn to the picture box. The options for interaction on this form mostly call the modCalc module to complete necessary algorithms and the values and results are either redrawn on the grid or detailed in the list boxes.

Within the modCalc module, specifically the simulated annealing algorithm, it is essential that the best solution found is recorded. Although the best solution when comparing old and new was being recorded, the overall best solution was not detailed or recorded correctly. Through adding a variable to record the best solution found so far, it was possible to determine whether the new solution was better than the overall best solution, and not just the previous solution. If this was the case, the current network structure needed to be recorded. At first, this seemed relatively simple through utilising a built in list function. However after further investigation it became apparent this was not the most appropriate route to follow. As a result, different methods had to be incorporated to the application to make copies of each list object.

3.3 Data Structures

The most important data type used in this application is a list container. This container is easier to work with and offers performance enhancements when compared to a multi-dimensional array (11). In addition to this the majority of the data types used are long, this is due to efficiency (5). The most efficient data type to use in VB.NET is Integer, followed by Long, Short, Byte (in order of efficiency). However, as decimal numbers are also required, for example with (x,y) coordinates, the double data type has been utilised. This has been chosen "because the floating-point processors of current platforms perform all operations in double precision." Therefore, this data type is more efficient than data types Single and Decimal. Building on this efficiency, for variables where a Boolean could be used, a byte was often used instead. This is due to the fact that a byte only takes up one byte of storage whereas a Boolean takes up four bytes (10); this is a significant difference and where 0 and 1 could be used on a byte variable to determine where it was true/false or on/off.

3.4 Algorithm

As decided in the interim report, the algorithm chosen to determine the most effective solutions to our problem is simulated annealing. This algorithm initially started as a greedy algorithm, however, through adding the probability check and finding a reasonable start and end temperature, it was possible to use the simulated annealing method. Through the development lifecycle of the application, it became apparent that it should be more oriented around user interaction and experience, rather than solving the problem to the best of its ability. Overall, it seemed that the simulated annealing would be more suitable as it is able to find an acceptably good solution in a defined amount of time rather than completing an exhaustive search to find the best solution. As it stands, the algorithm is completed relatively quickly and the results are displayed. If the algorithm chosen tried to find the best solution possible, such as hill climbing, there would have been a detrimental affect on system resources and, consequently, the user experience would be impeded. An additional disadvantage of using an algorithm such as hill climbing is that there would be no definitive time relating to how long it would take to successfully complete the algorithm and whether or not a solution to the problem would be found. As a result, this uncertainty would make feeding back the results of the algorithm to the user a lot more difficult as there would be no easy method to decide how long the algorithm had left before completion. Conversely, the fixed iterations in relating to the starting temperature and cooling factor of the SA algorithm can give some logical education towards estimating the length of time taken to complete the calculation(s).

Initially, when the application contained significantly less lines of code, all the code was detailed in the main form. As the length of the code began to increase, the main form class became significantly more congested; it seemed logical at this time to make it more manageable through partitioning the code in-to separate modules and classes. The code itself is split in-to both separate classes and modules. The data structures for BST, RTP, STP, TTP and PLM has been categorised in-to separate class modules. Additionally, code has been split into separate modules depending on its functionality. For example, the code written to read the data from the text files is in the modFiles.vb module. Extending from this, there is a module created to assign the values read to the data structures within the application. These subroutines are detailed in the modData.vb module. The final module, modCalc.vb has the code written for the calculations carried out in the simulated annealing and path loss calculations.

Within the application, timers have been used in order to update interface objects, primarily progress bars, in real time. As these timers are invisible to the user and tick continuously, they provide a simple way to update the GUI as seamlessly as possible. As the information relating to the value of the progress bars is determined in different threads, a most appropriate way to deduce the progress bar value would be to use a

BackGroundWorker_ProgressChanged method instead. However, due to time constrains it was not possible to implement this method.

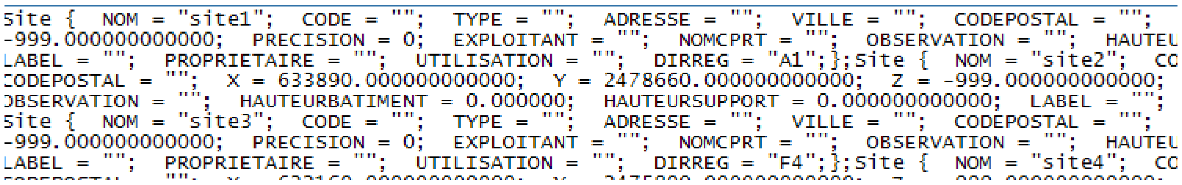
4.0 Implementation

This implementation section provides a lower level explanation of many of the details covered in the design section. Multiple data types and the structure in which data is stored will be discussed whilst comparing snippets of code that have been written. Further information relating to the use of threads, parallel processing and its resultant benefits are documented along with an explanation of the simulated annealing algorithm. Any technological terminology will be detailed towards the end of this document in the glossary.

As I felt that I was stronger at Visual Basic .NET than I was at Java, I have from the outset isolated myself in some ways through not being able to seek advice on my code or structure due to the lack of knowledge of VB.NET at University as it is not a language that is taught. Although I am not able to seek advice from faculty staff that I know of, the MSDN library provides a plethora of information that is sufficient for my needs. Further, the autocomplete IntelliSense feature that is incorporated in to the Visual Studio environment makes it easy to know the methods available when they are displayed to you as you type.

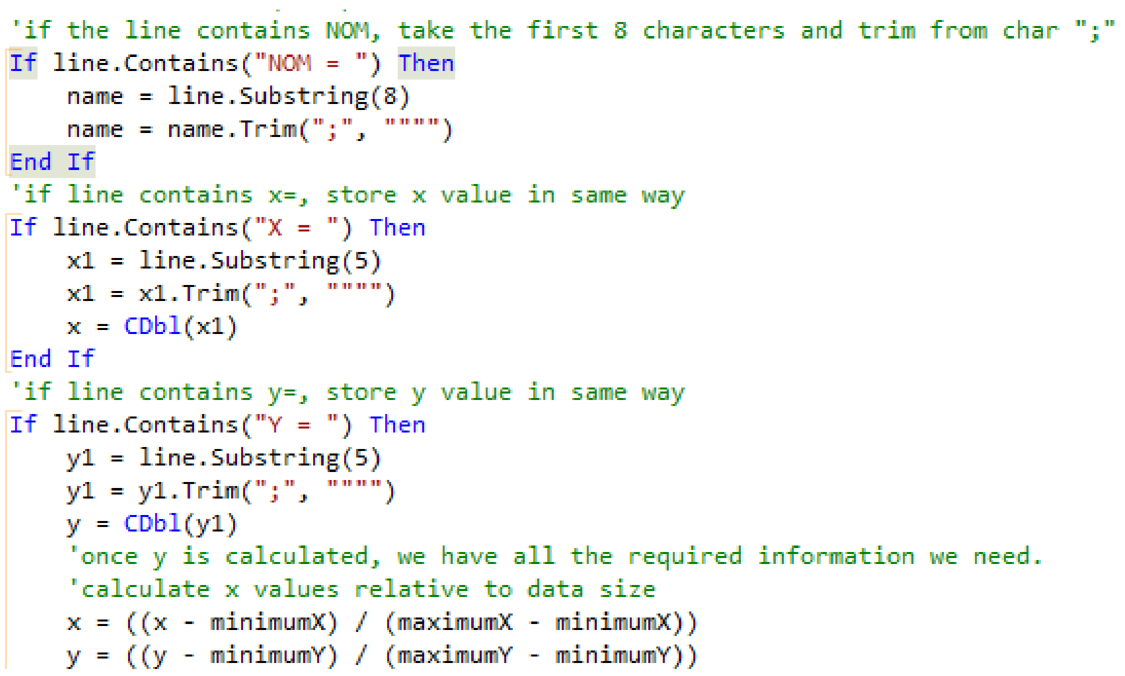
4.1 Reading Files

The structure of each file listed previously requires a different method of reading to be completed. For example, the structure of the NET file has information at the start of the document that is not necessary for the purpose of the document. As a result, lines have to be checked to ensure that they match the data that we are looking for.



```
Site {   NOM = "site1";   CODE = "";   TYPE = "";   ADRESSE = "";   VILLE = "";   CODEPOSTAL = "";
-999.000000000000;   PRECISION = 0;   EXPLOITANT = "";   NOMCPRT = "";   OBSERVATION = "";   HAUTEL
LABEL = "";   PROPRIETAIRE = "";   UTILISATION = "";   DIRREG = "A1";};Site {   NOM = "site2";   CO
CODEPOSTAL = "";   X = 633890.000000000000;   Y = 2478660.000000000000;   Z = -999.000000000000;
OBSERVATION = "";   HAUTEURBATIMENT = 0.000000;   HAUTEURSUPPORT = 0.000000000000;   LABEL = "";
Site {   NOM = "site3";   CODE = "";   TYPE = "";   ADRESSE = "";   VILLE = "";   CODEPOSTAL = "";
-999.000000000000;   PRECISION = 0;   EXPLOITANT = "";   NOMCPRT = "";   OBSERVATION = "";   HAUTEL
LABEL = "";   PROPRIETAIRE = "";   UTILISATION = "";   DIRREG = "F4";};Site {   NOM = "site4";   CO
```

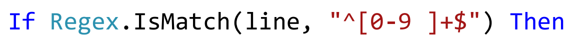
Figure 11 Sample Data from NET File



```
'if the line contains NOM, take the first 8 characters and trim from char ";"
If line.Contains("NOM = ") Then
    name = line.Substring(8)
    name = name.Trim(";", """")
End If
'if line contains x=, store x value in same way
If line.Contains("X = ") Then
    x1 = line.Substring(5)
    x1 = x1.Trim(";", """")
    x = CDbl(x1)
End If
'if line contains y=, store y value in same way
If line.Contains("Y = ") Then
    y1 = line.Substring(5)
    y1 = y1.Trim(";", """")
    y = CDbl(y1)
    'once y is calculated, we have all the required information we need.
    'calculate x values relative to data size
    x = ((x - minimumX) / (maximumX - minimumX))
    y = ((y - minimumY) / (maximumY - minimumY))
```

Figure 12 Sample Code for Reading NET File

A similar issue arises when we attempt to read the data held within the PLM file. For efficiency, a regular expression is used to filter out the lines that match the pattern for site information.



```
If Regex.IsMatch(line, "^[0-9 ]+$") Then
```

Figure 13 Sample Code for Filtering PLM File

The code written to read the data files utilises the StreamReader, as detailed in the design section. This stream reader returns each line read as a string. As a result, string operations need to be carried out to extract the information (SiteName, Xcoordinate, Ycoordinate, PathLoss) from the string. To do this, I tested two separate ways.

1. SubString – the first method chosen in an attempt to extract the data from the file was through using SubString(startIndex, endIndex) on each line. The SubString method is used to create a separate string given the start and end index specified. This method had to be carried out for each of the four variables that needed to be read.

2. StrArray – the second method tried was using StrArray. This method takes the initial string and turns it in to an array where each complete word is assigned an index in the array. By using this method, it is possible to skip directly to a position in the array where a known item is. I.e., it is known that position 0 would contain the site number, 1 would contain the Xcoordinate, 2 would contain the Ycoordinate and 3 would contain the PathLoss. As the size and use of the array is defined as soon as the line is read, the resulting cost is four times less than that of the SubString method.

To ensure that this way of reading the PLM was as efficient as possible, code was also written to test if a byteReader would provide more optimum results. The difference with the byteReader is that data is read in bytes, rather than line by line. Due to this, the size of each property needed to be defined in order for the reader to know how many bytes to read at a time. For example, SiteName is five bytes comprising of four digits and a space. Issues were encountered when the loss of a site, found at the end of each line, was only two digits compared to the majority of sites that had three digits. Hence, the data had to be changed to bypass this quickly before finalising the decision to use a byteReader and implementing it. After changing the structure slightly of the PLM file, the byteReader was used to read it. The resulting times taken to read this were of little difference to that of the StreamReader; so the idea was discarded.

4.2 Visualising Network

A key deliverable as defined in the interim report is the ability to visualise the data that is read from the relevant data files and display this information on screen. As detailed in the insights section, the visualisation is done in a picture box. To begin, the paint method for the picture box calculates the aspect ratio to draw the image based on the maximum (x,y) coordinates that have been read from the RTP file. Taking this information, a calculation is performed to determine what the coordinates specified would be in relation to the pixel location within the picture box. This is determined as follows:

```
ratioY = (maximumY - minimumY) / (maximumX - minimumX) * 150
```

Figure 14 Code to calculate the Y ratio

Without this calculation, the data was originally stretched much larger than was required, as can be seen on the following page.

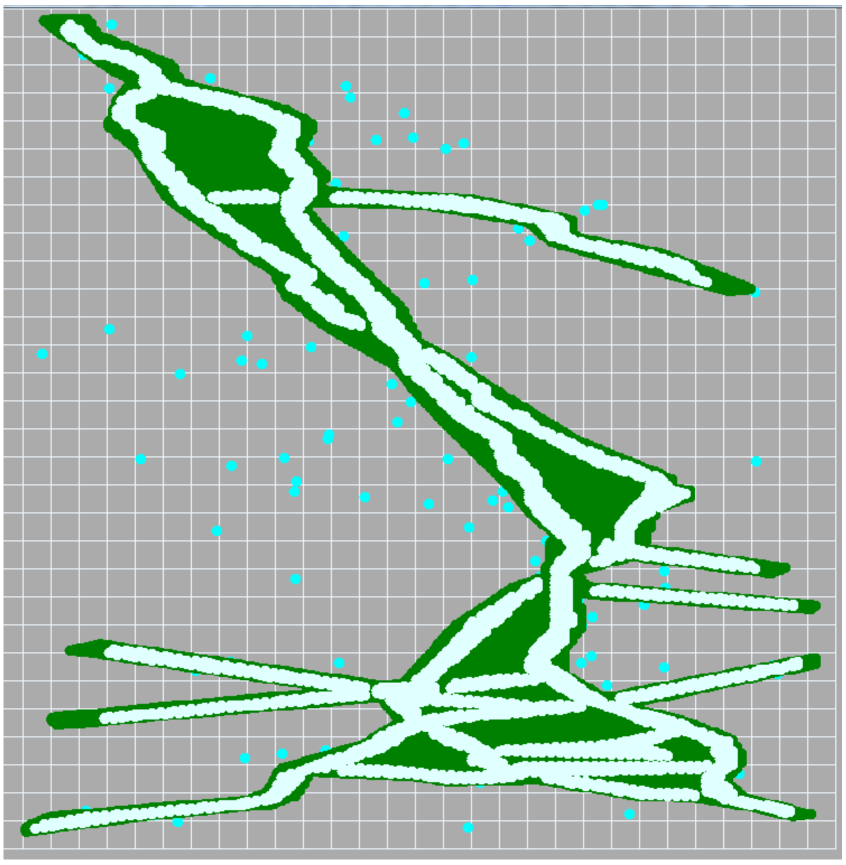


Figure 15 Previous Visualisation

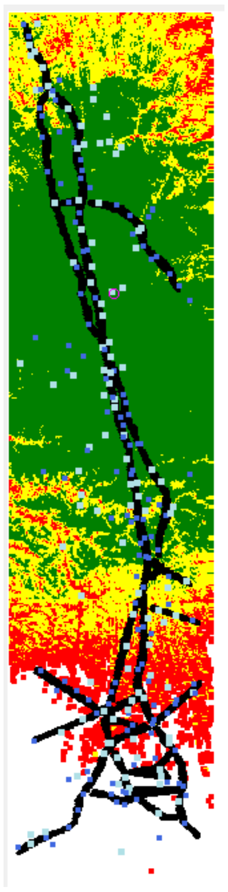
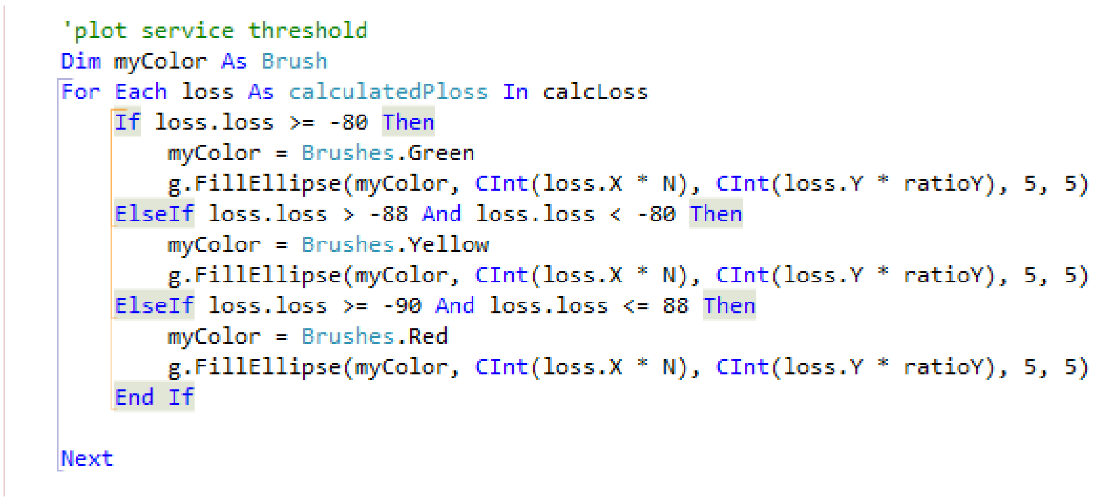


Figure 16 Current Visualisation

From using the simple equation to calculate the ratio, it was then possible to plot the data that had been read from the appropriate files inside the picture box. The lines of the grid within the picture box are spaced 25 pixels apart. To visualise the coverage at service test points from a specified site, certain colours are given which are dependent on the received signal strength. The closer the received strength is to zero, the stronger the signal. Therefore, the following colouring scheme has been used:

- If the signal is greater than -80, the colour used to depict the signal is green. This is to indicate that there is strong signal at these points.
- If the signal is greater than -88 but less than -80, the colour used to depict the signal is yellow. This is to indicate that there is medium signal received at these points.
- If the signal is greater than -90 but less than -88, the signal used to depict the signal is red. This is to indicate that there is poor signal received at these points.
- If the signal is less than -90, no colour is drawn. This is to remove congestion from the visualisation and indicates that there is no signal.



```
'plot service threshold
Dim myColor As Brush
For Each loss As calculatedPloss In calcLoss
    If loss.loss >= -80 Then
        myColor = Brushes.Green
        g.FillEllipse(myColor, CInt(loss.X * N), CInt(loss.Y * ratioY), 5, 5)
    ElseIf loss.loss > -88 And loss.loss < -80 Then
        myColor = Brushes.Yellow
        g.FillEllipse(myColor, CInt(loss.X * N), CInt(loss.Y * ratioY), 5, 5)
    ElseIf loss.loss >= -90 And loss.loss <= 88 Then
        myColor = Brushes.Red
        g.FillEllipse(myColor, CInt(loss.X * N), CInt(loss.Y * ratioY), 5, 5)
    End If

Next
```

Figure 17 Sample of Signal Strength Visualisation Code

These points are only drawn if the user assigns a power to a site or actives it using the “Assign Power” button.

Code has also been written to show active sites as a light blue and inactive sites as dark blue. The site that is currently selected from the combo box is highlighted with a magenta circle so the user is able to see where the current selected site is located within the visualisation.

4.3 Checking Coverage/Iterating list

One of the essential functions of the application is the ability to check which sites within the STP list have sufficient coverage, given their path loss and input power compared against the STP permitted threshold. To complete this calculation originally, the function iterated through each item in the STP list, for each item in the STP list the function also iterated through every item in the PLM list. This results in a maximum of 4114500*29954=123245733000 iterations. If the coordinates of the item in the PLM list matched the item in the STP list, it proceeded to check and update the values for the loss for each of the 25 sites that the information is required for. After reviewing this code, the author decided to attempt to implement the List.FindAll method to reduce the lines of code written and, hopefully, optimise performance.

The findAll method is implemented so that for each of the items in the STP list, the findAll method creates a list of all the entries within the PLM list that have matching X coordinates. From this, the findAll method is called again to refine the list to all the entries that have matching Y coordinates. Of this, there should always be 25 results. For these 25 results, they are read and added to a class instance of stpServer, which, in turn, is added to the servers list within the STP list (Appendix D). If these servers already exist, for example from a previous calculation, the list is cleared, the values are calculated again and the items are added.

Once the 25 servers have been added to each item in the STP list, the number of sites that have coverage is checked by determining which STP items have a server that is not equal to zero. The total of these is then divided by the total number of sites. This returns a number between 0 and 1. As this is the result that we want to maximise, the closer the result is to 1, the better the coverage and resulting solution.

Coverage = stpCovered / stpCount

4.4.0 Algorithm

The choice to use simulated annealing was the result of research that shows that other optimisation algorithms, such as hill climbing, run the risk of getting stuck in local minima or maxima. The simulated annealing resolves this issue by allowing worse solutions to be accepted some of the time. This allows the algorithm to take some uphill steps so that it can escape local minima. Unlike hill climbing, simulated annealing selects a move at random from the neighbourhood whereas hill climbing chooses the best move from those available. If the newest solution is better than the current solution, it is always accepted. The difference lies in that with the simulated annealing algorithm if the next solution is worse, then it will be accepted based on some probability.

The algorithm is implemented in the application as follows. To begin, the current coverage is determined as detailed above. Once this is determined, the current number of sites that are on is calculated. The algorithm then calls another function that picks a random site and toggles it on or off. Once this is completed, the coverage calculation similar to the one detailed in the previous section is run to determine the coverage that the new solution generates. This coverage, combined with the siteCost is returned to the initial subroutine that called the toggle function.

Site Cost = (totalSites – sitesOn) / totalSites

The site cost calculation produces a result between 0 and 1. As this is a calculation that we want to minimise, the closer the solution is to zero, the more effective the resultant solution. This result, combined with the coverage result will produce a number anywhere between 0 and 2; the closer this number is to 1 the better the solution.

If the returned result is better than the existing result, this new solution is accepted. If the new solution produces a worse result than that already known, it is checked against some probability.

Probability = `If Rnd() * 1 < Math.E ^ ((sNew - sOld) / t) Then`

Figure 18 Calculation to determine probability of accepting solution

If the solution is better than the probability, it is accepted as the new solution. At this stage, it essential that the best solution found is recorded; the initial algorithm only made a comparison between the new and old solutions. Hence, an extra variable sBest was added. After the probability check has been completed, the new solution is compared to the best solution. If the new solution exceeds the best solution, sBest becomes the most recently calculated solution. This is demonstrated in the pseudo code below:

4.4.1 Simulated Annealing Pseudo Code

```
Sub Anneal()

set sOld to (25-sitesOn)/25
set sOld to sOld + Coverage
set tStart to 10
set sBest to sOld

Clear list tmpChangeSTP
Copy list minSTP to tmpChangeSTP

    While tStart > 0.01

Set sNew to function calculateChange()

        If sNew > sOld Then

Set sOld to the value of sNew

        Else

            Randomize()

Set g to random*1
Set f to Math.E^((sNew - sOld) / t)

            If g < f Then
Set sOld to the value of sNew
            End If
        End If

        If sOld > sBest Then

Set sBest to the value of sOld

Clear list tmpChangeSTP
Copy list changeSTP to tmpChangeSTP

Clear list plmTmpIndexList
Copy list plmIndexList to plmTmpIndexList

        End If

Next
        t = 0.8 * tStart

End Sub
```

Pseudo code of the simulated annealing algorithm

```
Function calculateChange() As Double

reCalculate: Set x to random * the size of plmIndexList

if x < 0 then set x to 0
if x > 24 then set x to 24

If x = tmpSite Then GoTo reCalculate

Set tmpSite to x; Set tmpSitesOn to 0

For each plm entry in the list plmIndexList
       if plm.serves = 0 then set plm.active to 0
       if plm.active = 1 then add 1 to tmpSitesOn
Next entry

if item X in plmIndexList is inactive (0) then
       set item X in plmIndexList to 0
       set userLossGain to minus plmIndexList(x).serves
If tmpSitesOn>0 then subtract 1 from tmpSitesOn
Else
if item X in plmIndexList is inactive (1) then
       set userLossGain to plus plmIndexList(x).serves
       If tmpSitesOn<25 then add 1 to tmpSitesOn
End If
For each stp entry in the list ChangeSTP
if the stp.SelectedServer = plmIndexList(x).site then
       If plmIndexList(x).active = inactive(0) Then
              Set stp.SelectedStrength to -999
              Set stp.SelectedServer to 0
       End If
End If

Loop as i from 0 to the size of stp.Servers
        If stp.Servers(i).serverName = plmIndeXList(x).Site Then
              Set stp.Servers(i).active to the value of plmIndexList(x).active
        End If
If stp.servers(i).active = active(1) Then
       if stp.servers(i).signalStrength is greater than stp.Threshold
       if stp.servers(i).signalStrength is greater than the current strength
              Set stp.selectedStrength to stp.servers(i).signalStrength
       if stp.SelectedServer is different to the stp.Servers(i).ServerName then
              set stp.SelectedServer to stp.servers(i).serverName

For each plmIndex entry in the list plmIndexList
       If plmIndex.site = stp.Servers(i).serverName Then
              add one to plm.Index.serves
       next item
End If
End If
End If
End If
next i

set SiteCost to (25-totalSitesOn) / 25
set Coverage to number of STP above threshold/size of changeSTP
        Return siteCost + Coverage
End Function
```

Figure 19 Pseudo Code of the calculateChange function

In addition to recording what the best result of the coverage and sites on is, the algorithm also needs to make an exact copy of the network structure that produced the best solution. As the network structure is stored in a series of list, the most obvious way to create a copy was to use the list.AddRange method. Although this created a copy of the best list of sites that were on or off, this copy was only a copy of the references to the values from the underlying parent class. Subsequently, as the actual values were never copied, if the values of any property in the underlying class changed so did the items within the wrapper. As a result, extra code had to be written to manually add the items one by one, iterating through the list until the end.

Another option to resolve this was to implement the ICloneable interface in the class definition. Through implement ICloneable, code can be written to complete shallow and deep copies of the list. A deep copy of the object means that the object itself would be copied as would all the objects that it points to. A shallow copy of the class would mean that the references to the objects are copied but the objects themselves are not, similar to the AddRange function. Thus, as the semantic of ICloneable is not inherently clear and it requires slightly more code, the simpler option of iterating and adding each item (as detailed previously) has been used.

4.5 Threading

Threading is an essential part of ensuring that the application stays responsive whilst carrying out the calculations on data structures than contain up to 4 million entries. One issue that I encountered when using threads was the difficulty between updating the user interface from a separate thread. To get around this, I initially added timers to the application that could run every 500ms to check if any of the variables in the application have changed and update the user interface accordingly. After further research, it was decided that this method is not the most effective way to do update the interface. Rather, the threading method that was used – BackgroundWorker - incorporates a method ProgressChanged which allows the thread to report its progress in order for the user interface to be updated. Additionally, the BackgroundWorker method also has the method CancelAsync. This method is important in that if the user accidentally clicks a button more than once, multiple threads may be created to carry out the exact same operation. Adding in a simple check, such as if thread.IsBusy then thread.CancelAsync, ensures that if a resource intensive operation is being carried out it can be cancelled before the next operation begins.

4.6 Data Structures

Once the data has been read from the appropriate files in to the application, it is stored in several different lists that contain properties relevant to the associated class. For example, the bstList has the properties bst.ID, bst.Site, bst.X, etc. The properties and variables of each class are defined in separate class files. The decision to use lists enables the items to be quickly sorted, searched and filtered, whilst offering an easy to work with list of methods to achieve the required outcomes. The list type is a dynamic and automatically resizing array; this removes the necessity to include ReDim Preserve which is "used at procedure level to reallocate storage space for an array variable of fixed size" (12). Furthermore, the use of lists was preferable over the ArrayList data type as ArrayLists suffer from boxing and are significantly larger on a 64bit machine because references essentially must reference more memory; in practise, the need to reference more memory results in more pointers (18).

It was previously stated in this report that the decision had been reached to use a PLM a tenth of its original size. Although this offered some performance improvements due to the reduced amount of data that needed to be read, issues still arose when having to compare the data held in the service test point list with that in the path loss matrix. For example, when the button is clicked to calculate the coverage of the service test points against the data held in the path loss matrix, the calculation has to first iterate through each item in the service test point list. For each item in the service test point list, it then must loop through the items in the path loss matrix list to determine the path loss of each point. Hence, checking the last value in the service test point list against the last value in the path loss matrix requires extra calculations to be completed. As discussed previously, this involves at most 123245733000 calculations to be completed on the records in the list. Consequently, the selection of data used for the service test point list has a step of 250 to increase the speed at which calculations can be completed for demonstrative purposes. This means that once the first item is taken from the list of service test points, the next item to be read would be the 251st as opposed to the 2nd.

In addition to this, the STP structure has a sub-list (Appendix C) relating to the loss associated to each site from the PLM. This is then used to decide which site in the sub-list provides the strongest signal.

5.0 Results

The results and evaluation section of this document includes results from the implemented algorithm along with results of attempted optimisations made to the code. The main drive for improvement was in relation to file loading, visualising the network and the efficiency of the algorithm. These results will then be critically evaluated.

5.1 Performance: Loading PLM

Attempt	Time (ms) in PLM method 1	Time (ms) in PLM method 2
1	742	921
2	689	948
3	692	897

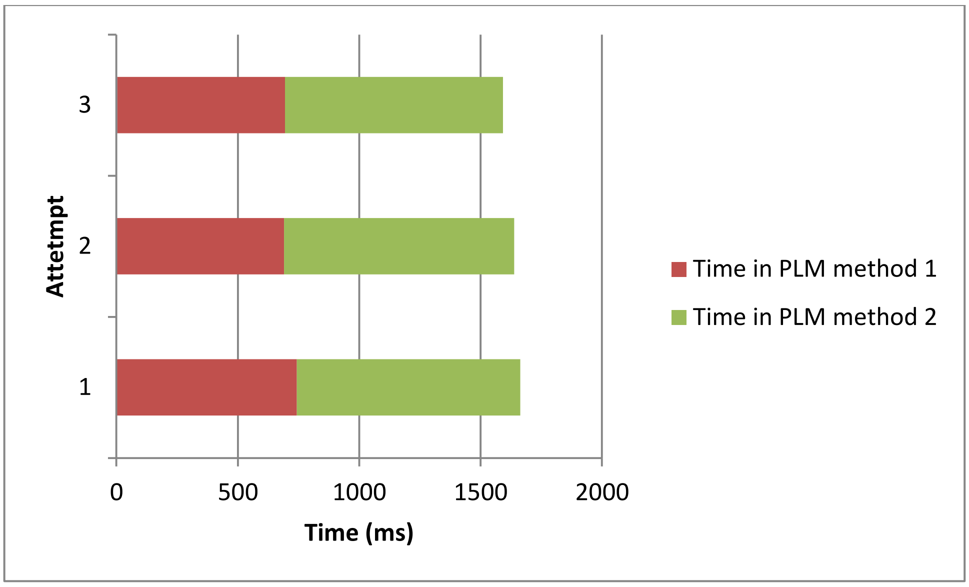
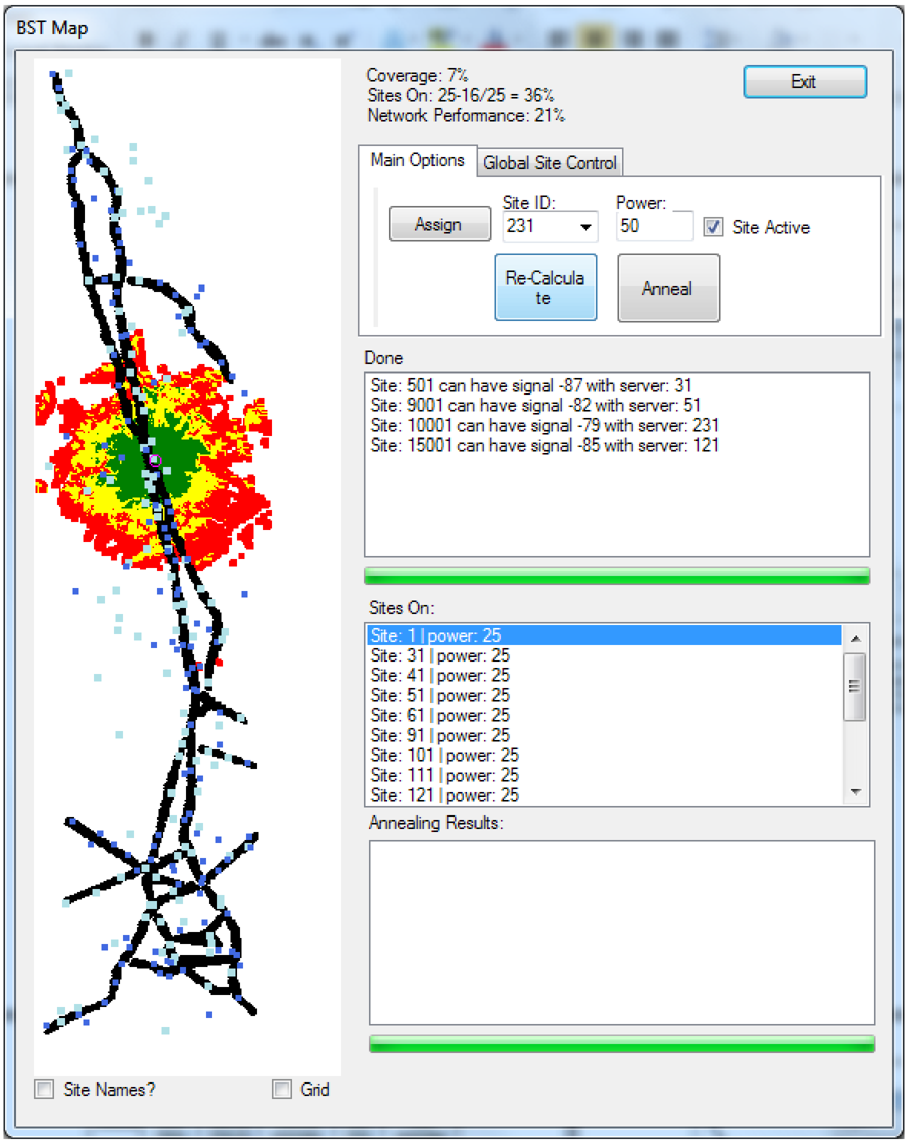


Figure 20 Graph Showing Time Difference when Reading PLM

These results indicate the time difference between the two methods that were written for reading the PLM file. Method one is used in the application. Method two can be found in appendix B. The results clearly show in each of the three attempts that method two takes significantly more time than method one. From these results it seemed logical to go with method one for reading the PLM file to save time.

5.2 Coverage Results: Uniform Power 25

Attempt	Time Taken (ms)	Sites On	Coverage (%)	Performance (%)
1	656	16	7	21
2	498	15	12	26
3	441	11	7	31



Coverage: 7%
Sites On: 25-16/25 = 36%
Network Performance: 21%

```
Site: 501 can have signal -87 with server: 31
Site: 9001 can have signal -82 with server: 51
Site: 10001 can have signal -79 with server: 231
Site: 15001 can have signal -85 with server: 121
```

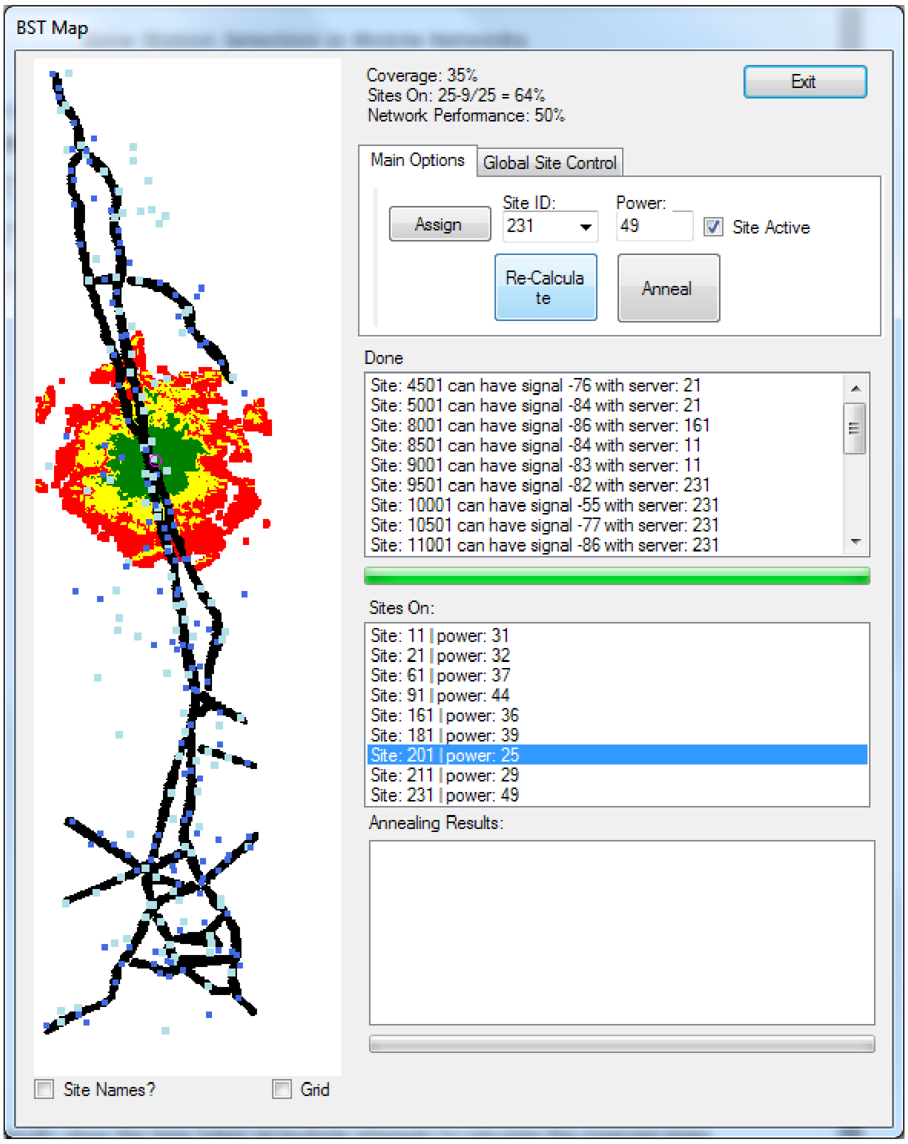
```
Site: 1 | power: 25
Site: 31 | power: 25
Site: 41 | power: 25
Site: 51 | power: 25
Site: 61 | power: 25
Site: 91 | power: 25
Site: 101 | power: 25
Site: 111 | power: 25
Site: 121 | power: 25
```

Figure 21 Visualisation of Coverage

The above results show the time taken on multiple attempts to calculate the coverage, given a set of sites with uniform power.

5.3 Coverage Results: Random Power

Attempt	Time Taken (ms)	Sites On	Coverage (%)	Performance (%)
1	409	9	35	50
2	428	9	28	42
3	542	11	25	40



Coverage: 35%
Sites On: 25-9/25 = 64%
Network Performance: 50%

Site ID: 231
Power: 49
Site Active

Done

```
Site: 4501 can have signal -76 with server: 21
Site: 5001 can have signal -84 with server: 21
Site: 8001 can have signal -86 with server: 161
Site: 8501 can have signal -84 with server: 11
Site: 9001 can have signal -83 with server: 11
Site: 9501 can have signal -82 with server: 231
Site: 10001 can have signal -55 with server: 231
Site: 10501 can have signal -77 with server: 231
Site: 11001 can have signal -86 with server: 231
```

Sites On:

```
Site: 11 | power: 31
Site: 21 | power: 32
Site: 61 | power: 37
Site: 91 | power: 44
Site: 161 | power: 36
Site: 181 | power: 39
Site: 201 | power: 25
Site: 211 | power: 29
Site: 231 | power: 49
```

Annealing Results:

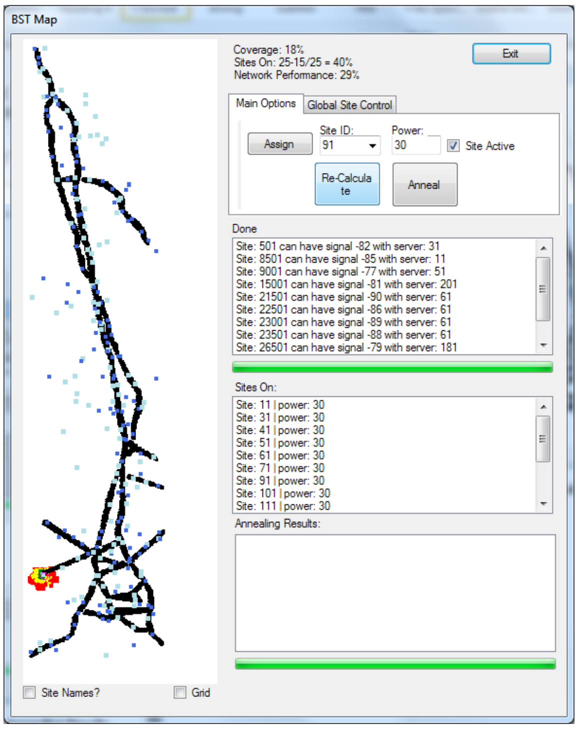
Figure 22 Visualisation of Coverage

The above results show the time taken on multiple attempts to calculate the coverage, given a set of sites with random power.

5.4 Algorithm Results: Fixed Sites Uniform Power 25

Attempt	Time Taken	Starting Sites On	Starting Coverage/ Performance (%)		Final Sites On	Final Coverage/ Performance (%)	
1	102	15	18	29	3	13	51
2	98	3	13	51	3	13	51
3	107	3	13	51	3	13	51

Attempt 1



```
Coverage: 18%
Sites On: 25-15/25 = 40%
Network Performance: 29%
```

Site ID: 91 | Power: 30 | Site Active ✓

Done

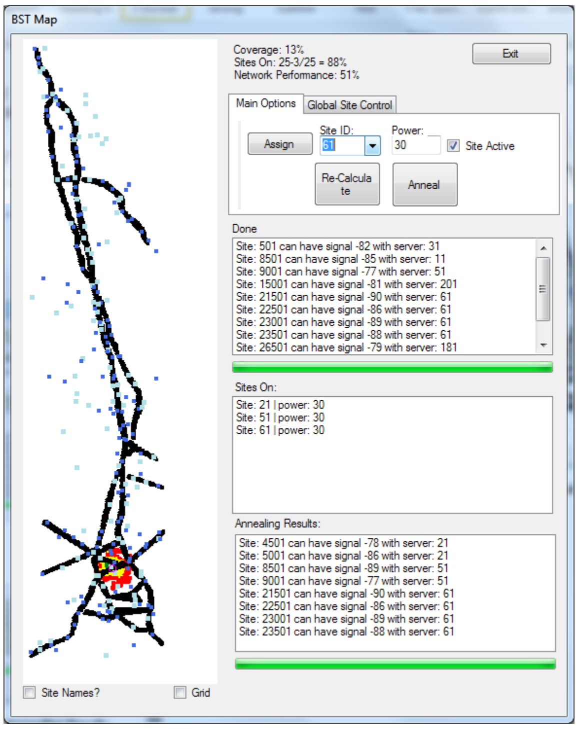
```
Site: 501 can have signal -82 with server: 31
Site: 8501 can have signal -85 with server: 11
Site: 9001 can have signal -77 with server: 51
Site: 15001 can have signal -81 with server: 201
Site: 21501 can have signal -90 with server: 61
Site: 22501 can have signal -86 with server: 61
Site: 23001 can have signal -89 with server: 61
Site: 23501 can have signal -88 with server: 61
Site: 26501 can have signal -79 with server: 181
```

Sites On:

```
Site: 11 | power: 30
Site: 31 | power: 30
Site: 41 | power: 30
Site: 51 | power: 30
Site: 61 | power: 30
Site: 71 | power: 30
Site: 91 | power: 30
Site: 101 | power: 30
Site: 111 | power: 30
```

Annealing Results:

Figure 23 Visualisation of Attempt 1 Coverage Results



```
Coverage: 13%
Sites On: 25-3/25 = 88%
Network Performance: 51%
```

Site ID: 61 | Power: 30 | Site Active ✓

Done

```
Site: 501 can have signal -82 with server: 31
Site: 8501 can have signal -85 with server: 11
Site: 9001 can have signal -77 with server: 51
Site: 15001 can have signal -81 with server: 201
Site: 21501 can have signal -90 with server: 61
Site: 22501 can have signal -86 with server: 61
Site: 23001 can have signal -89 with server: 61
Site: 23501 can have signal -88 with server: 61
Site: 26501 can have signal -79 with server: 181
```

Sites On:

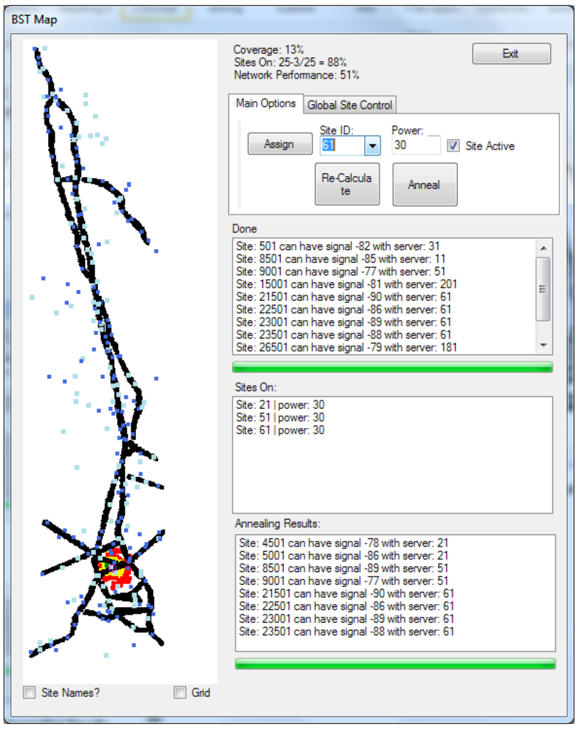
```
Site: 21 | power: 30
Site: 51 | power: 30
Site: 61 | power: 30
```

Annealing Results:

```
Site: 4501 can have signal -78 with server: 21
Site: 5001 can have signal -86 with server: 21
Site: 8501 can have signal -89 with server: 51
Site: 9001 can have signal -77 with server: 51
Site: 21501 can have signal -90 with server: 61
Site: 22501 can have signal -86 with server: 61
Site: 23001 can have signal -89 with server: 61
Site: 23501 can have signal -88 with server: 61
```

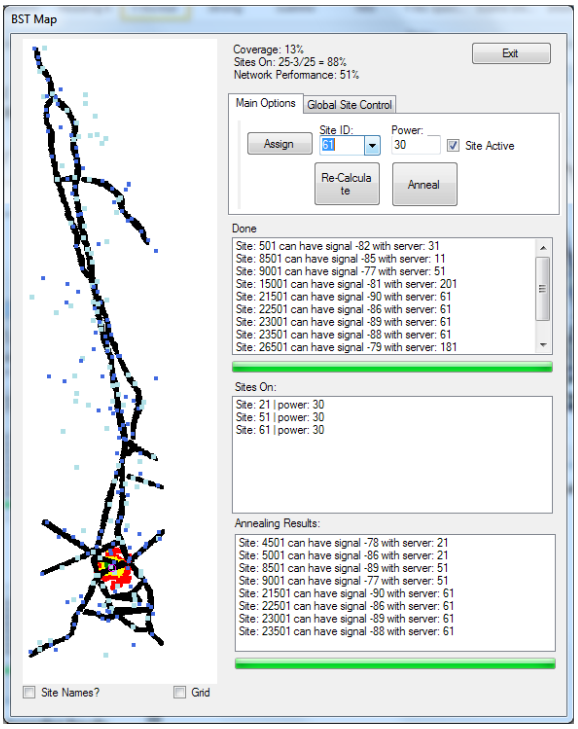
Figure 24 Visualisation of Attempt 1 Annealing Results

Attempt 2



```
Coverage: 13%
Sites On: 25-3/25 = 88%
Network Performance: 51%

Site ID: 61   Power: 30   [✓] Site Active

Done
Site: 501 can have signal -82 with server: 31
Site: 8501 can have signal -85 with server: 11
Site: 9001 can have signal -77 with server: 51
Site: 15001 can have signal -81 with server: 201
Site: 21501 can have signal -90 with server: 61
Site: 22501 can have signal -86 with server: 61
Site: 23001 can have signal -89 with server: 61
Site: 23501 can have signal -88 with server: 61
Site: 26501 can have signal -79 with server: 181

Sites On:
Site: 21 | power: 30
Site: 51 | power: 30
Site: 61 | power: 30

Annealing Results:
Site: 4501 can have signal -78 with server: 21
Site: 5001 can have signal -86 with server: 21
Site: 8501 can have signal -89 with server: 51
Site: 9001 can have signal -77 with server: 51
Site: 21501 can have signal -90 with server: 61
Site: 22501 can have signal -86 with server: 61
Site: 23001 can have signal -89 with server: 61
Site: 23501 can have signal -88 with server: 61
```

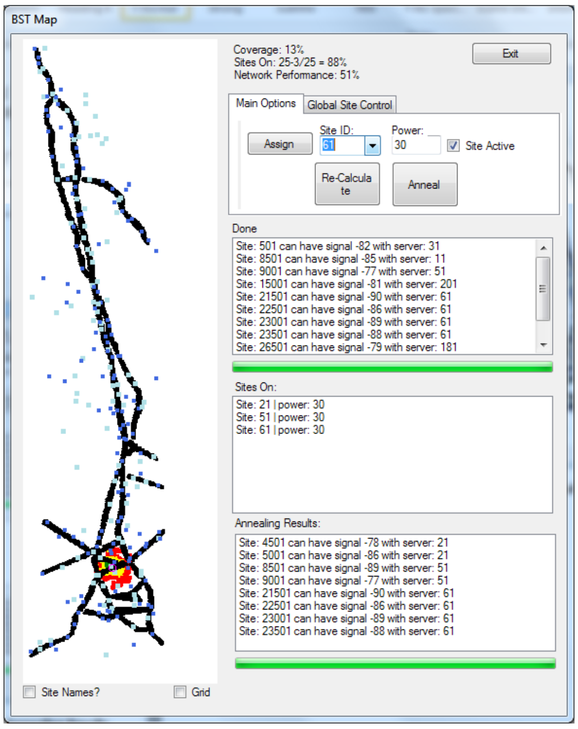
Figure 25 Visualisation of Attempt 2 Coverage Results



```
Coverage: 13%
Sites On: 25-3/25 = 88%
Network Performance: 51%

Site ID: 61   Power: 30   [✓] Site Active

Done
Site: 501 can have signal -82 with server: 31
Site: 8501 can have signal -85 with server: 11
Site: 9001 can have signal -77 with server: 51
Site: 15001 can have signal -81 with server: 201
Site: 21501 can have signal -90 with server: 61
Site: 22501 can have signal -86 with server: 61
Site: 23001 can have signal -89 with server: 61
Site: 23501 can have signal -88 with server: 61
Site: 26501 can have signal -79 with server: 181

Sites On:
Site: 21 | power: 30
Site: 51 | power: 30
Site: 61 | power: 30

Annealing Results:
Site: 4501 can have signal -78 with server: 21
Site: 5001 can have signal -86 with server: 21
Site: 8501 can have signal -89 with server: 51
Site: 9001 can have signal -77 with server: 51
Site: 21501 can have signal -90 with server: 61
Site: 22501 can have signal -86 with server: 61
Site: 23001 can have signal -89 with server: 61
Site: 23501 can have signal -88 with server: 61
```

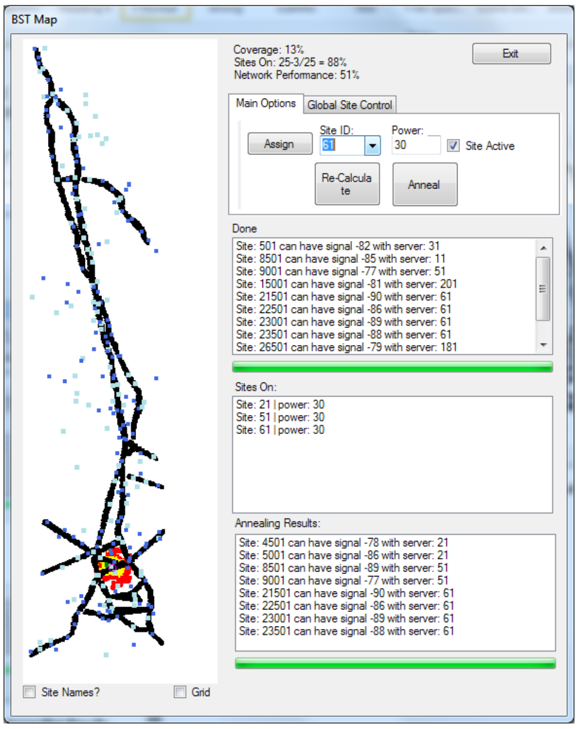
Figure 26 Visualisation of Attempt 2 Annealing Results

Attempt 3



```
Coverage: 13%
Sites On: 25-3/25 = 88%
Network Performance: 51%

Site ID: 61   Power: 30   [✓] Site Active

Done
Site: 501 can have signal -82 with server: 31
Site: 8501 can have signal -85 with server: 11
Site: 9001 can have signal -77 with server: 51
Site: 15001 can have signal -81 with server: 201
Site: 21501 can have signal -90 with server: 61
Site: 22501 can have signal -86 with server: 61
Site: 23001 can have signal -89 with server: 61
Site: 23501 can have signal -88 with server: 61
Site: 26501 can have signal -79 with server: 181

Sites On:
Site: 21 | power: 30
Site: 51 | power: 30
Site: 61 | power: 30

Annealing Results:
Site: 4501 can have signal -78 with server: 21
Site: 5001 can have signal -86 with server: 21
Site: 8501 can have signal -89 with server: 51
Site: 9001 can have signal -77 with server: 51
Site: 21501 can have signal -90 with server: 61
Site: 22501 can have signal -86 with server: 61
Site: 23001 can have signal -89 with server: 61
Site: 23501 can have signal -88 with server: 61
```

Figure 27 Visualisation of Attempt 3 Coverage Results



```
Coverage: 13%
Sites On: 25-3/25 = 88%
Network Performance: 51%

Site ID: 61   Power: 30   [✓] Site Active

Done
Site: 501 can have signal -82 with server: 31
Site: 8501 can have signal -85 with server: 11
Site: 9001 can have signal -77 with server: 51
Site: 15001 can have signal -81 with server: 201
Site: 21501 can have signal -90 with server: 61
Site: 22501 can have signal -86 with server: 61
Site: 23001 can have signal -89 with server: 61
Site: 23501 can have signal -88 with server: 61
Site: 26501 can have signal -79 with server: 181

Sites On:
Site: 21 | power: 30
Site: 51 | power: 30
Site: 61 | power: 30

Annealing Results:
Site: 4501 can have signal -78 with server: 21
Site: 5001 can have signal -86 with server: 21
Site: 8501 can have signal -89 with server: 51
Site: 9001 can have signal -77 with server: 51
Site: 21501 can have signal -90 with server: 61
Site: 22501 can have signal -86 with server: 61
Site: 23001 can have signal -89 with server: 61
Site: 23501 can have signal -88 with server: 61
```

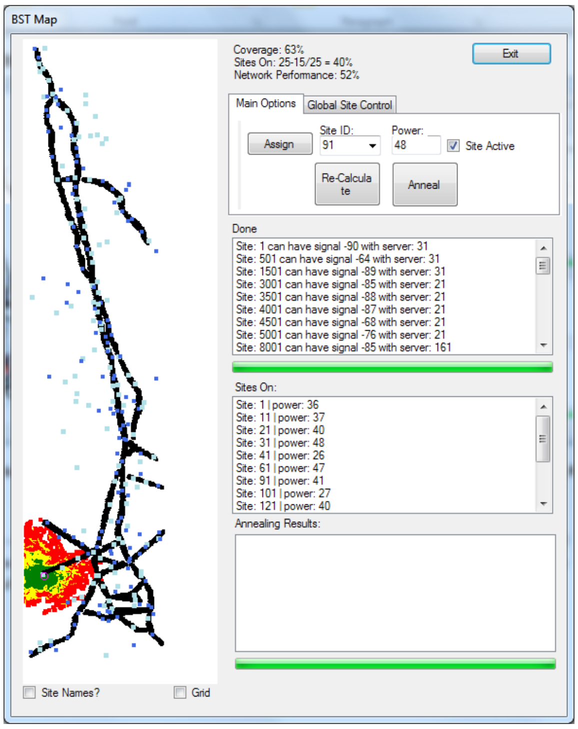
Figure 28 Visualisation of Attempt 3 Annealing Results

The above results show the time taken on multiple attempts to complete the simulated annealing algorithm, given a set of sites with uniform power of 25. After the first attempt, the results cannot be improved upon anymore. This is validated after the third attempt.

5.5.0 Algorithm Results: Random Sites & Powers

Attempt	**Time Taken**	**Starting Sites On**	**Starting Coverage/ Performance (%)**		**Final Sites On**	**Final Coverage/ Performance (%)**	
1	117	15	63	52	4	23	54
2	98	4	23	54	6	53	65
3	119	6	53	65	8	63	66
4	143	8	63	66	8	65	66
5	108	8	65	66	7	63	68

Attempt 1



Coverage: 63%
Sites On: 25-15/25 = 40%
Network Perfomance: 52%

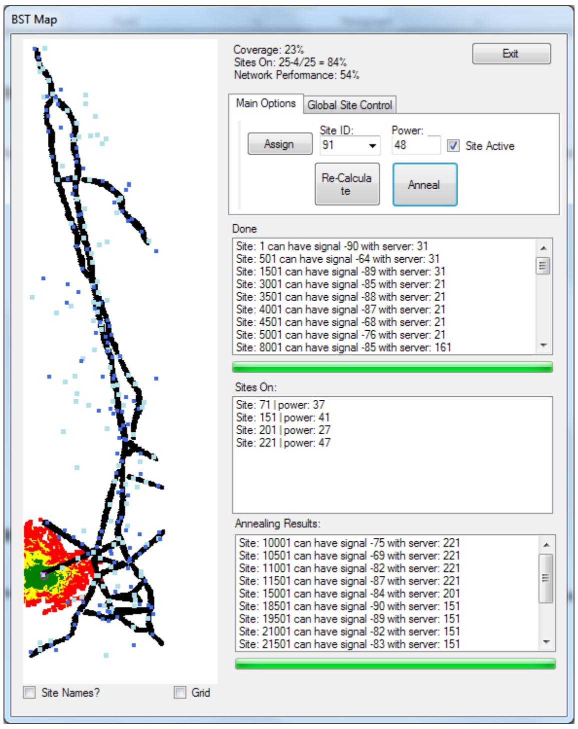
Site ID: 91
Power: 48
☑ Site Active

Done
```
Site: 1 can have signal -90 with server: 31
Site: 501 can have signal -64 with server: 31
Site: 1501 can have signal -89 with server: 31
Site: 3001 can have signal -85 with server: 21
Site: 3501 can have signal -88 with server: 21
Site: 4001 can have signal -87 with server: 21
Site: 4501 can have signal -68 with server: 21
Site: 5001 can have signal -76 with server: 21
Site: 8001 can have signal -85 with server: 161
```

Sites On:
```
Site: 1 | power: 36
Site: 11 | power: 37
Site: 21 | power: 40
Site: 31 | power: 48
Site: 41 | power: 26
Site: 61 | power: 47
Site: 91 | power: 41
Site: 101 | power: 27
Site: 121 | power: 40
```

Annealing Results:

Figure 29 Visualisation of Attempt 1 Coverage Results



Coverage: 23%
Sites On: 25-4/25 = 84%
Network Perfomance: 54%

Site ID: 91
Power: 48
☑ Site Active

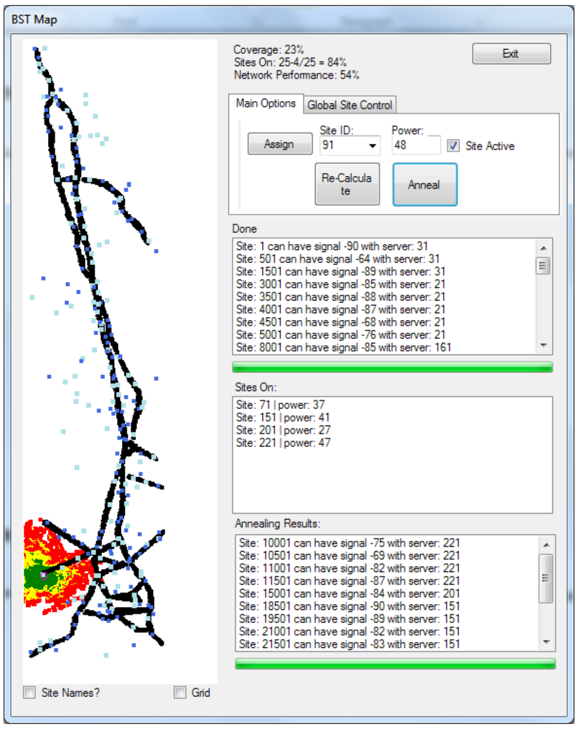
Done
```
Site: 1 can have signal -90 with server: 31
Site: 501 can have signal -64 with server: 31
Site: 1501 can have signal -89 with server: 31
Site: 3001 can have signal -85 with server: 21
Site: 3501 can have signal -88 with server: 21
Site: 4001 can have signal -87 with server: 21
Site: 4501 can have signal -68 with server: 21
Site: 5001 can have signal -76 with server: 21
Site: 8001 can have signal -85 with server: 161
```

Sites On:
```
Site: 71 | power: 37
Site: 151 | power: 41
Site: 201 | power: 27
Site: 221 | power: 47
```

Annealing Results:
```
Site: 10001 can have signal -75 with server: 221
Site: 10501 can have signal -69 with server: 221
Site: 11001 can have signal -82 with server: 221
Site: 11501 can have signal -87 with server: 221
Site: 15001 can have signal -84 with server: 201
Site: 18501 can have signal -90 with server: 151
Site: 19501 can have signal -89 with server: 151
Site: 21001 can have signal -82 with server: 151
Site: 21501 can have signal -83 with server: 151
```

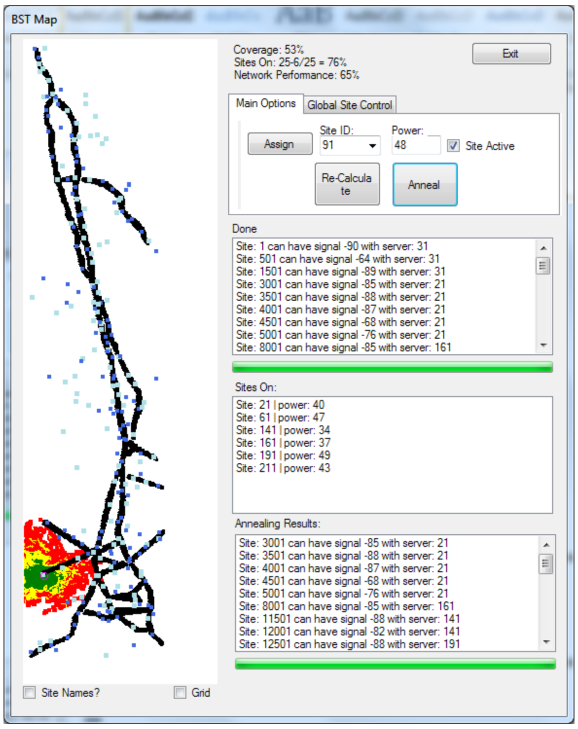
Figure 30 Visualisation of Attempt 1 Annealing Results

Attempt 2



```
Coverage: 23%
Sites On: 25-4/25 = 84%
Network Performance: 54%

Site ID: 91   Power: 48   [✓] Site Active

Done
Site: 1 can have signal -90 with server: 31
Site: 501 can have signal -64 with server: 31
Site: 1501 can have signal -89 with server: 31
Site: 3001 can have signal -85 with server: 21
Site: 3501 can have signal -88 with server: 21
Site: 4001 can have signal -87 with server: 21
Site: 4501 can have signal -68 with server: 21
Site: 5001 can have signal -76 with server: 21
Site: 8001 can have signal -85 with server: 161

Sites On:
Site: 71 | power: 37
Site: 151 | power: 41
Site: 201 | power: 27
Site: 221 | power: 47

Annealing Results:
Site: 10001 can have signal -75 with server: 221
Site: 10501 can have signal -69 with server: 221
Site: 11001 can have signal -82 with server: 221
Site: 11501 can have signal -87 with server: 221
Site: 15001 can have signal -84 with server: 201
Site: 18501 can have signal -90 with server: 151
Site: 19501 can have signal -89 with server: 151
Site: 21001 can have signal -82 with server: 151
Site: 21501 can have signal -83 with server: 151
```

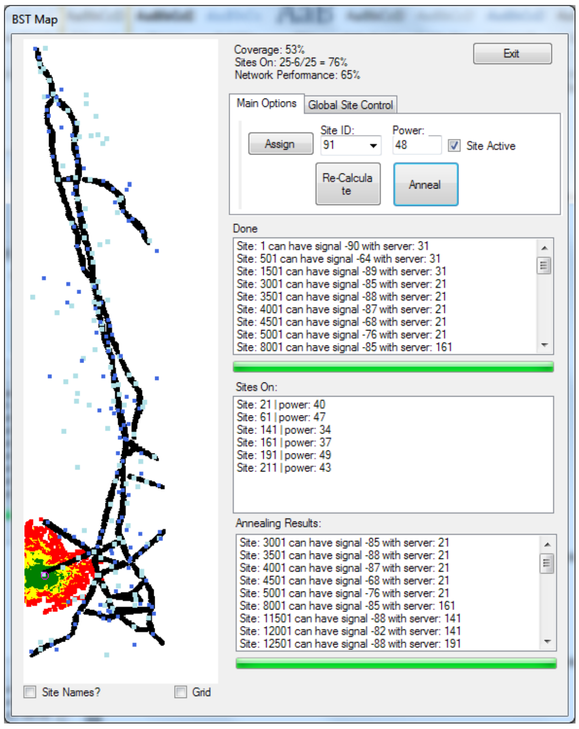
Figure 31 Visualisation of Attempt 2 Coverage Results



```
Coverage: 53%
Sites On: 25-6/25 = 76%
Network Performance: 65%

Site ID: 91   Power: 48   [✓] Site Active

Done
Site: 1 can have signal -90 with server: 31
Site: 501 can have signal -64 with server: 31
Site: 1501 can have signal -89 with server: 31
Site: 3001 can have signal -85 with server: 21
Site: 3501 can have signal -88 with server: 21
Site: 4001 can have signal -87 with server: 21
Site: 4501 can have signal -68 with server: 21
Site: 5001 can have signal -76 with server: 21
Site: 8001 can have signal -85 with server: 161

Sites On:
Site: 21 | power: 40
Site: 61 | power: 47
Site: 141 | power: 34
Site: 161 | power: 37
Site: 191 | power: 49
Site: 211 | power: 43

Annealing Results:
Site: 3001 can have signal -85 with server: 21
Site: 3501 can have signal -88 with server: 21
Site: 4001 can have signal -87 with server: 21
Site: 4501 can have signal -68 with server: 21
Site: 5001 can have signal -76 with server: 21
Site: 8001 can have signal -85 with server: 161
Site: 11501 can have signal -88 with server: 141
Site: 12001 can have signal -82 with server: 141
Site: 12501 can have signal -88 with server: 191
```

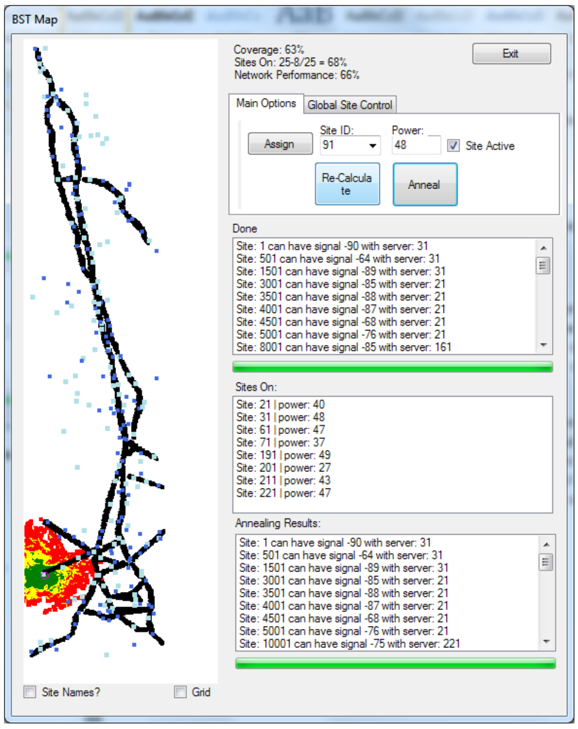
Figure 32 Visualisation of Attempt 2 Annealing Results

Attempt 3



```
Coverage: 53%
Sites On: 25-6/25 = 76%
Network Performance: 65%

Site ID: 91   Power: 48   [✓] Site Active

Done
Site: 1 can have signal -90 with server: 31
Site: 501 can have signal -64 with server: 31
Site: 1501 can have signal -89 with server: 31
Site: 3001 can have signal -85 with server: 21
Site: 3501 can have signal -88 with server: 21
Site: 4001 can have signal -87 with server: 21
Site: 4501 can have signal -68 with server: 21
Site: 5001 can have signal -76 with server: 21
Site: 8001 can have signal -85 with server: 161

Sites On:
Site: 21 | power: 40
Site: 61 | power: 47
Site: 141 | power: 34
Site: 161 | power: 37
Site: 191 | power: 49
Site: 211 | power: 43

Annealing Results:
Site: 3001 can have signal -85 with server: 21
Site: 3501 can have signal -88 with server: 21
Site: 4001 can have signal -87 with server: 21
Site: 4501 can have signal -68 with server: 21
Site: 5001 can have signal -76 with server: 21
Site: 8001 can have signal -85 with server: 161
Site: 11501 can have signal -88 with server: 141
Site: 12001 can have signal -82 with server: 141
Site: 12501 can have signal -88 with server: 191
```

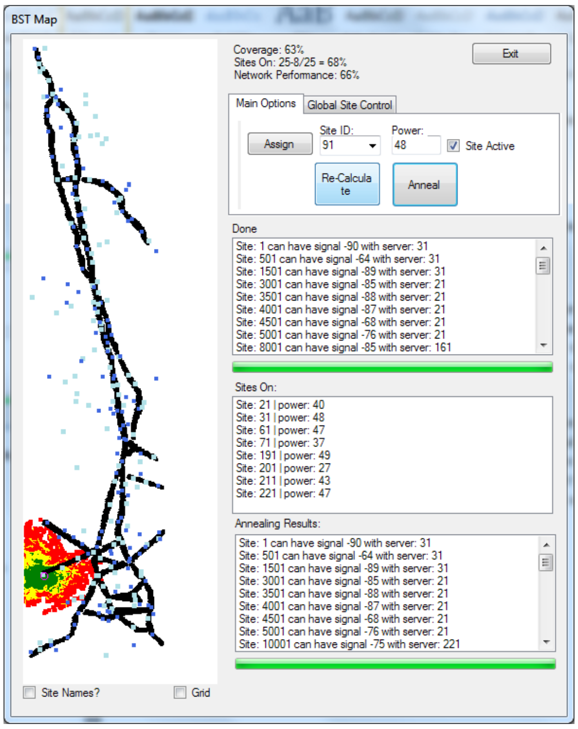
Figure 33 Visualisation of Attempt 3 Coverage Results



```
Coverage: 63%
Sites On: 25-8/25 = 68%
Network Performance: 66%

Site ID: 91   Power: 48   [✓] Site Active

Done
Site: 1 can have signal -90 with server: 31
Site: 501 can have signal -64 with server: 31
Site: 1501 can have signal -89 with server: 31
Site: 3001 can have signal -85 with server: 21
Site: 3501 can have signal -88 with server: 21
Site: 4001 can have signal -87 with server: 21
Site: 4501 can have signal -68 with server: 21
Site: 5001 can have signal -76 with server: 21
Site: 8001 can have signal -85 with server: 161

Sites On:
Site: 21 | power: 40
Site: 31 | power: 48
Site: 61 | power: 47
Site: 71 | power: 37
Site: 191 | power: 49
Site: 201 | power: 27
Site: 211 | power: 43
Site: 221 | power: 47

Annealing Results:
Site: 1 can have signal -90 with server: 31
Site: 501 can have signal -64 with server: 31
Site: 1501 can have signal -89 with server: 31
Site: 3001 can have signal -85 with server: 21
Site: 3501 can have signal -88 with server: 21
Site: 4001 can have signal -87 with server: 21
Site: 4501 can have signal -68 with server: 21
Site: 5001 can have signal -76 with server: 21
Site: 10001 can have signal -75 with server: 221
```

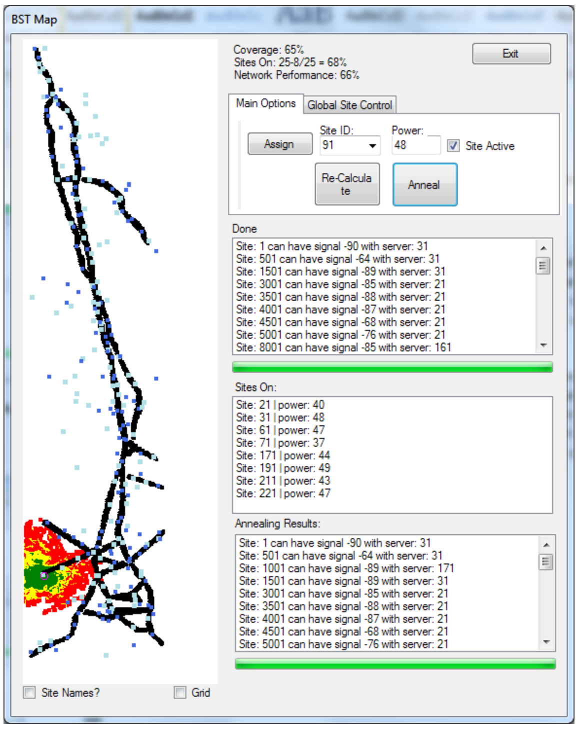
Figure 34 Visualisation of Attempt 3 Annealing Results

Attempt 4



```
Coverage: 63%
Sites On: 25-8/25 = 68%
Network Performance: 66%

Site ID: 91   Power: 48   [✓] Site Active

Done
Site: 1 can have signal -90 with server: 31
Site: 501 can have signal -64 with server: 31
Site: 1501 can have signal -89 with server: 31
Site: 3001 can have signal -85 with server: 21
Site: 3501 can have signal -88 with server: 21
Site: 4001 can have signal -87 with server: 21
Site: 4501 can have signal -68 with server: 21
Site: 5001 can have signal -76 with server: 21
Site: 8001 can have signal -85 with server: 161

Sites On:
Site: 21 | power: 40
Site: 31 | power: 48
Site: 61 | power: 47
Site: 71 | power: 37
Site: 191 | power: 49
Site: 201 | power: 27
Site: 211 | power: 43
Site: 221 | power: 47

Annealing Results:
Site: 1 can have signal -90 with server: 31
Site: 501 can have signal -64 with server: 31
Site: 1501 can have signal -89 with server: 31
Site: 3001 can have signal -85 with server: 21
Site: 3501 can have signal -88 with server: 21
Site: 4001 can have signal -87 with server: 21
Site: 4501 can have signal -68 with server: 21
Site: 5001 can have signal -76 with server: 21
Site: 10001 can have signal -75 with server: 221
```

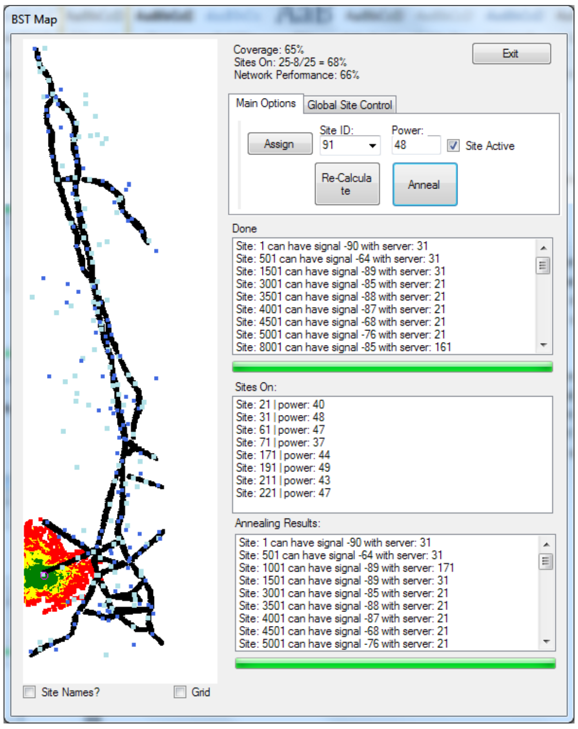
Figure 35 Visualisation of Attempt 4 Coverage Results



```
Coverage: 65%
Sites On: 25-8/25 = 68%
Network Performance: 66%

Site ID: 91   Power: 48   [✓] Site Active

Done
Site: 1 can have signal -90 with server: 31
Site: 501 can have signal -64 with server: 31
Site: 1501 can have signal -89 with server: 31
Site: 3001 can have signal -85 with server: 21
Site: 3501 can have signal -88 with server: 21
Site: 4001 can have signal -87 with server: 21
Site: 4501 can have signal -68 with server: 21
Site: 5001 can have signal -76 with server: 21
Site: 8001 can have signal -85 with server: 161

Sites On:
Site: 21 | power: 40
Site: 31 | power: 48
Site: 61 | power: 47
Site: 71 | power: 37
Site: 171 | power: 44
Site: 191 | power: 49
Site: 211 | power: 43
Site: 221 | power: 47

Annealing Results:
Site: 1 can have signal -90 with server: 31
Site: 501 can have signal -64 with server: 31
Site: 1001 can have signal -89 with server: 171
Site: 1501 can have signal -89 with server: 31
Site: 3001 can have signal -85 with server: 21
Site: 3501 can have signal -88 with server: 21
Site: 4001 can have signal -87 with server: 21
Site: 4501 can have signal -68 with server: 21
Site: 5001 can have signal -76 with server: 21
```

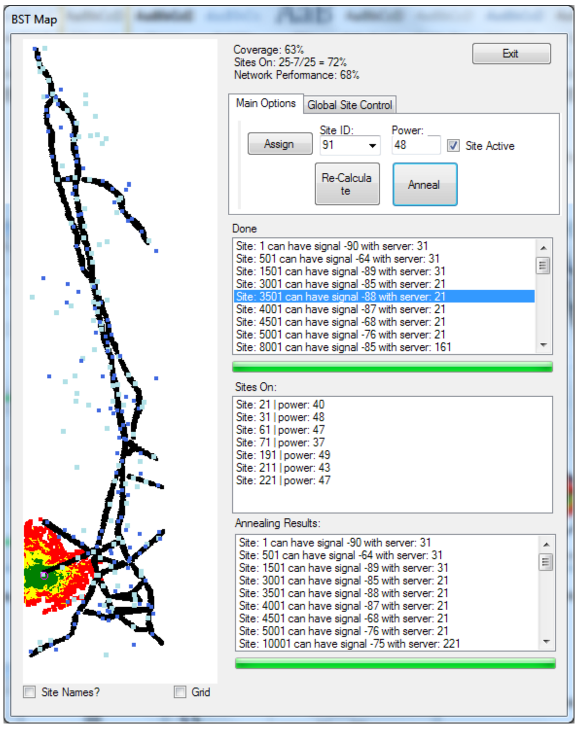
Figure 36 Visualisation of Attempt 4 Annealing Results

Attempt 5



```
Coverage: 65%
Sites On: 25-8/25 = 68%
Network Performance: 66%

Site ID: 91   Power: 48   [✓] Site Active

Done
Site: 1 can have signal -90 with server: 31
Site: 501 can have signal -64 with server: 31
Site: 1501 can have signal -89 with server: 31
Site: 3001 can have signal -85 with server: 21
Site: 3501 can have signal -88 with server: 21
Site: 4001 can have signal -87 with server: 21
Site: 4501 can have signal -68 with server: 21
Site: 5001 can have signal -76 with server: 21
Site: 8001 can have signal -85 with server: 161

Sites On:
Site: 21 | power: 40
Site: 31 | power: 48
Site: 61 | power: 47
Site: 71 | power: 37
Site: 171 | power: 44
Site: 191 | power: 49
Site: 211 | power: 43
Site: 221 | power: 47

Annealing Results:
Site: 1 can have signal -90 with server: 31
Site: 501 can have signal -64 with server: 31
Site: 1001 can have signal -89 with server: 171
Site: 1501 can have signal -89 with server: 31
Site: 3001 can have signal -85 with server: 21
Site: 3501 can have signal -88 with server: 21
Site: 4001 can have signal -87 with server: 21
Site: 4501 can have signal -68 with server: 21
Site: 5001 can have signal -76 with server: 21
```

Figure 37 Visualisation of Attempt 5 Coverage Results



```
Coverage: 63%
Sites On: 25-7/25 = 72%
Network Performance: 68%

Site ID: 91   Power: 48   [✓] Site Active

Done
Site: 1 can have signal -90 with server: 31
Site: 501 can have signal -64 with server: 31
Site: 1501 can have signal -89 with server: 31
Site: 3001 can have signal -85 with server: 21
Site: 3501 can have signal -88 with server: 21
Site: 4001 can have signal -87 with server: 21
Site: 4501 can have signal -68 with server: 21
Site: 5001 can have signal -76 with server: 21
Site: 8001 can have signal -85 with server: 161

Sites On:
Site: 21 | power: 40
Site: 31 | power: 48
Site: 61 | power: 47
Site: 71 | power: 37
Site: 191 | power: 49
Site: 211 | power: 43
Site: 221 | power: 47

Annealing Results:
Site: 1 can have signal -90 with server: 31
Site: 501 can have signal -64 with server: 31
Site: 1501 can have signal -89 with server: 31
Site: 3001 can have signal -85 with server: 21
Site: 3501 can have signal -88 with server: 21
Site: 4001 can have signal -87 with server: 21
Site: 4501 can have signal -68 with server: 21
Site: 5001 can have signal -76 with server: 21
Site: 10001 can have signal -75 with server: 221
```

Figure 38 Visualisation of Attempt 5 Annealing Results

The above results show the time taken on multiple attempts to complete the simulated annealing algorithm given a set of sites with random power. The results are indicative that the algorithm runs successfully. After the fifth attempt on the above results, the performance cannot be increased further. This can be seen when attempting to run the algorithm several times more.

6.0 Future Work

This section of the report details aspects that the author wanted to include in the application and insights that have been researched nearing the end of development. The research done produced important results in relation to increasing performance and enhancing the efficiency of the code, however there was not enough time left in the project to implement these results. Additionally, non-performance related aspects, such as providing a more interactive user interface and more user customisation are discussed here. All future work detailed below is done so in order of importance, from the author's viewpoint.

6.1 RTP/TTP

The initial aim of the application was to use all of the data provided to find optimum network solutions. As detailed previously, this was not possible due to the time constraints and initial lack of progress. To ensure that the application meets the goals originally laid out, including the data held in the RTP and TTP structures, it is essential to ensure that the algorithm can return the best possible solution. As the TTP structure contains critical information in relation to the capacity of each site, omitting this is detrimental to the end result in that the current solution provided may minimise the total number of sites on but it may also turn off sites that serve significantly more users than others.

6.2 Efficiency

An underlying issue preventing the effectiveness of the application is its performance issues in relation to loading files and carrying out calculations on the large data sets. In some aspects these performance impingements are determined by the hardware of the computer; a more powerful processor, for example, may allow for much quicker operations. Conversely, it may be possible to make the application much more efficient using the same architecture that it was developed on. One way that this may be achievable, primarily in relation to the PLM file, would be to have multiple threads reading defined segments of the file. This could either be done using a byteReader, to read a set amount of bytes per thread, or, alternatively, using the more efficient StreamReader to read a defined amount of lines. If these threads could run concurrently, it could reduce the loading time.

Furthermore, the efficiency of the application in relation to its data structures is something that could definitely be improved upon, primarily in the PLM which is the most inefficiency file and data structure. The application could be coded in a way that, instead of entering each line of the matrix as a new item in the list, each (x,y) coordinate would have one entry so there were no duplicates. Expanding from this, the coordinates would have a sub list which defines which sites cover the coordinates and the resulting loss. This would make the initial iterative search through the list more efficient as the list would be shorter.

Expanding on this, the List (Of T) data structure used has a function FindAll(Predicate(of T)). This function scans the list and can return the found values to a sub list, including the indexes. If this method had been initially utilised, it would have vastly reduced the amount of code written and may have led to performance enhancements.

6.3 Interface

The interface of the application could be enhanced so that the grid in which the network data is drawn can be clicked on. For example, the picture box could be divided in to clickable regions (Appendix C). Clicking on a region would bring up the test points and base stations in that area. Clicking on the test points within each region would then allow the user to see the information relating to the test point such as:

- Which BST serves it
- Received signal strength

Clicking on a BST within the region would display information such as:

- BST Name/ID
- Input Power
- Whether the site is active or not
- Number of test points served by the BST
    - ID of test point(s) served.
    - Received strength at test point(s)

The author feels that making this aspect of the interface more interactive and illustrative would heighten interest in the overall purpose of the application, make the information more readily accessible and easier to understand and, finally, make it look and feel more professional.

6.3.1 Customisation & Accessibility

As the application currently stands, many aspects that should and could be defined by the user are hard coded. The most important aspects that should be defined by the user is the location of the data files; at present, the location of the data files is hard coded so that they must be in the root directory of the executable. Giving the user a simple file dialog to specify where the files are located could provide a better user experience.

Additionally, one aspect that should be included is the option to specify the colours of the controls and grid drawing on the screen and its accessibility relating to font sizes and language. Colours are an important factor that should be considered on the application as using strong contrast, such as the black font on a white background proves difficult for dyslexic people to process (7). Conversely, colour blind people prefer a strong contrast and find the use of pastel colours difficult to process. Although many of the controls' colours on the application are defined by the operating system, the grid that displays the mobile user data has colours that are not customisable and means that the visualisation of the network performance may be completely unnoticeable for some users.

6.4 Exporting Data

To build on the results generated by the algorithm, one aspect that could be incorporated in to the application for evaluating the algorithm is the option to export the results to a text file. Once exported, these results could be read in to a different application that could assess the effectiveness of the SA algorithm. This facility could also benefit the user in that they would be more readily able to assess the effect that assigning different powers has to different sites and the overall network performance.

Additionally, the author feels an essential aspect that has been omitted from the application is the ability to export errors from try/catch blocks to an external error log. As the application stands, any errors that occur are in some ways invisible to the user with no clear feedback of what has gone wrong or where. Therefore, having the option to log these errors outside the application and further detailing the events that caused them would be invaluable to the developer to create a stable application. Building on this, a further option to "contact developer" and attach the error log would make error reporting quick and easy for the user whilst providing real time issues that can be rectified throughout the software life cycle. This could be added through importing System.Net.Mail and declaring a mail class as SMTPClient in to the application.

6.5 Programming Language

Although utilising VB.NET to create the program initially seemed a good idea due to the author's familiarity with the environment, the platform dependence and lack of knowledge from those who the author went to for support was detrimental to the overall development. As a result, with the knowledge of what the application is meant to do and the flexibility and power of other languages, it may be more effective to attempt rewriting the application in a different language.

7.0 Conclusions

The conclusions section of this report draws on the implementation and results section to deduce any positive and negative aspects of the project, whilst also raising points which could change the results obtained.

The outset of the project presented two main goals, to visualise the network within a GUI and to further implement a simulated annealing algorithm to optimise the network, both of which have been completed successfully.

7.1 GUI

The GUI provides a simple way to interact with the data and allows the user to interact and amend certain properties of the network whilst showing the results of such actions. The main downfall of the application is the reduced size of the area in which the network is visualised; this could be resolved by simply changing code. This would allow for a more detailed look at the network through reducing the congestion that results from producing a small image. In comparison to the start of the project, the visualisation of the network is completed much quicker and shows no noticeable lag.

Apart from the visualisation, the controls of the application also respond intelligently. For example, the user cannot press the coverage button multiple times. If no data has been changed, certain methods will not run as they have no need to.

7.2 Algorithm

The development of the algorithm has been successful in that results are optimised between each attempt at running the SA algorithm. After each attempt in the algorithm, the network performance measure always increases, even if it is to the detriment of the coverage. This is due to the fact that it may be cheaper to loose service to an area comparative to the cost of supplying power to the site. In addition to this, the algorithm is quite simple as it only takes in to account the loss of users covered as opposed to the initial goal of also allowing for the loss of capacity on the network from the TTP file. It would also be essential in a real scenario to include a contingency if sites were to fail.

In order to optimise the network further, other algorithms could be used to determine the location of sites as opposed to which ones were on or off. This could be successfully incorporated through a genetic algorithm. In addition to this, there are two types of antenna that can be used in BSTs. It is assumed from the data set that the masts in use are omnidirectional. Through using directional antenna, radiation could be focussed in a certain direction to optimise the network and improve coverage.

An important note to be made is that if the user is looking to design a network with a small number of base stations, higher powers yield better results but the cost and associated health risks are often a deterrent. Conversely, using low power for sites will permit for more base stations which will introduce a contingency for base station failure.

The data shows that in terms of cost to the network operator, it may be more effective to distribute fewer sites but distribute higher power. Using a uniform power of 25 produced a performance measure of 51. Using random powers produced a performance measure of 68.

As a simple data set has been used, it is not possible to critically evaluate the results - given the initial large data set provided - but it is fair to deduce that the algorithm works as expected. To incorporate the larger data set, creating the application in a more powerful language, such as Java, may have been beneficial.

8.0 Reflection on Learning

Throughout my education at Cardiff University I have had the opportunity to enhance my understanding and further my interest in mobile telecommunications that was first encountered under my employment with Vodafone. I have developed an understanding of the mathematics that is inherent in many important aspects of computer science, particularly in artificial intelligence and optimisation algorithms.

I have had the opportunity to learn the most widely used programming language, Java and develop a firm understanding of its underlying principles whilst having the responsibility and option to build on my previous experience within VB.NET. I have learned from this that, whilst I find the IDE of Visual Studio (and hence VB.NET) easy to work with, the performance of the background code that is automatically written from the drag and drop environment is not as powerful and potentially not as efficient as could be found in other programming languages. This has led me to the conclusion that whilst VB.NET makes it easy to work with controls and containers, the rationale for using VB.NET for this project was flawed. The amount of data that is read in to the application combined with the amount of calculations required has a detrimental effect on the performance of the application. This could be due to poor coding practise but could also be due to the inefficiency of the language having a depredation on performance. In addition to this, the advantage of Java's WORA (Write One Run Anywhere) does not apply to VB.NET and, hence, constrains the application to only be run in a Windows environment. It is now apparent that writing the application in Java would have been preferable.

One of the strengths of VB.NET is the accessibility of parallel processing and threads. Although taught the concepts and benefits of threads in previous modules, this project has been demonstrative of the necessity of threads in any application to keep the interface responsive to user interaction. Furthermore, the power of parallel processing is something that I had not encountered in previous language modules and I feel this provides me with invaluable experience that I can take forward to future employment.

Away from the coding aspect of this application, the project has required me to research and understand particular algorithms that have been in question, such as Tabu, Hill Search and Simulated Annealing and also helped me discover the use of a Genetic Algorithm that could be used if the aim of the project was to calculate the optimum locations of base stations within the network rather than the best combination of on/off given the fixed locations of these sites.

This project has also been a huge test of my time management. Working whilst at University poses a real strain on the amount of time that one is able to assign to university work. Far from having a negative impact, working and studying has forced me to prioritise requirements whilst working more efficiently to accomplish the goals that I set.

Appendices

Appendix A

The two pieces of code below are methods that have been written in C++ to attempt to read the PLM file in order to compare performance differences. The difference in time taken to read this large file was negligible.

Code Attempt 1

```
#include iostream
#include fstream
#include vector
#include cstdlib

int main(int argc, char* argv[]) {
        if (argc!=3) {
                std::cout"Usage: parser.exe source_path
output_path"std::endl;
                return 1;
        }

        //Parse arguments
        std::string path = argv[1];
        std::string output = argv[2];

        //Open input file
        std::vectorchar data;
        {
                std::fstream infile(path.c_str(), std::fstream::in);
                std::istreambuf_iteratorchar start(infile);
                std::istreambuf_iteratorchar end;
                if (!infile.is_open()) {
                        std::cout  path.c_str()  " failed to open. ";
                        return 1;
                }
                if (!infile.good()) {
                        std::cout  path.c_str()  " did not become
ready for I/O operations. ";
                        return 1;
                }

                //Populate vector from stream buffers
                data = std::vectorchar(start, end);
                infile.close();
        }

        std::cout  "File reading finished, parsing..."  std::endl;

        //Erase very first space
        data.erase(data.begin());

        for (std::vectorchar::iterator c = data.begin();
c!=data.end(); ++c) {
                //Strip leading spaces
                if ((*c == '\n')) {
                        int count=0;
                        while (*(c+1+count)==' ') count++;
                        data.erase(c+1,c+1+count);
                }

                //Strip silly lines
                //

                //Replace spaces with commas
                if (*c == ' ') *c=',';
        }

        //Write out to new file
        std::ofstream ofile(output.c_str(), std::ofstream::binary);
        ofile.write(&data[0],data.size());
        ofile.close();
        return 0;
}
```

Code Attempt 2

```
#include iostream
#include fstream
#include vector
#include cstdlib

int main(int argc, char* argv[]) {
        if (argc!=3) {
                std::cout"Usage: parser.exe source_path
output_path"std::endl;
                return 1;
        }

        //Parse arguments
        std::string path = argv[1];
        std::string output = argv[2];

        //Open file I/O

        std::fstream infile(path.c_str(), std::fstream::in);
        //std::istreambuf_iteratorchar start(infile);
        //std::istreambuf_iteratorchar end;
        if (!infile.is_open()) {
                std::cout  path.c_str()  " failed to open. ";
                return 1;
        }
        if (!infile.good()) {
                std::cout  path.c_str()  " did not become ready for I/O
operations. ";
                return 1;
        }
        std::ofstream ofile(output.c_str());
        if (!ofstream.is_open()) {
                std::cout  output.c_str()  " failed to open. ";
                return 1;
        }
        if (!ofstream.good()) {
                std::cout  output.c_str()  " did not become ready for I/O
operations. ";
                return 1;
        }



        infile.getline(&data[0],3);
        std::coutdata[1]std::endl;



        infile.close();
        ofile.close();

        return 0;
}
```

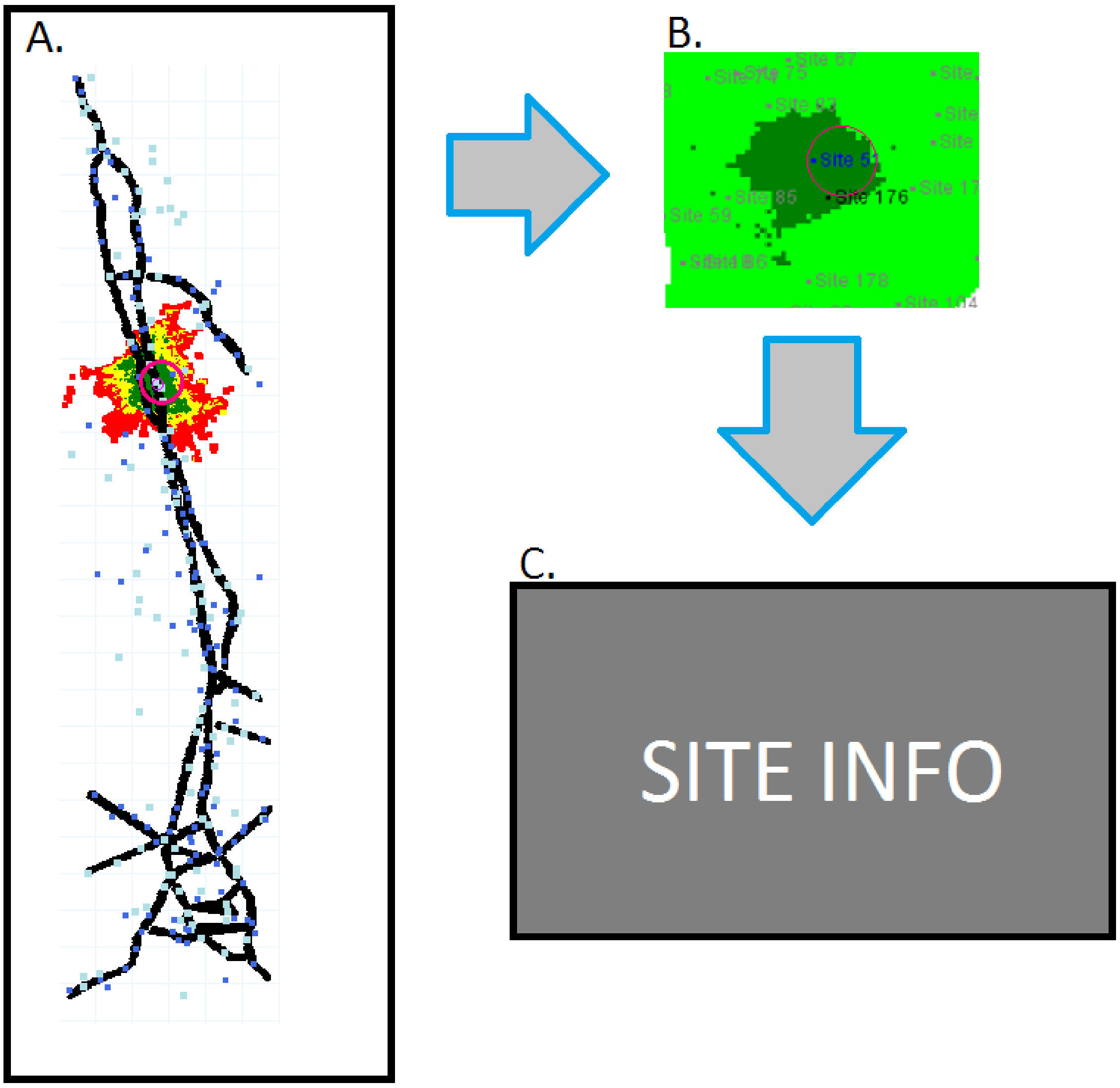
Appendix B

The code below was written in VB.NET in attempt to get the PLM file to read quicker. The method below was discarded in favour of the one that is present in the source package.

```
Public Sub readPLM()
        Dim i As Integer = 0
        Dim startTime As Date = Now
        Dim site, loss As Long
        Dim x, y As Long
        Try
            Dim reader As New BinaryReader(New
FileStream(System.AppDomain.CurrentDomain.BaseDirectory & "net1_0_25sites.txt",
FileMode.Open))
            Dim buffer() As Byte
            Dim lineSize As Byte = 0
            Dim bytesRead As Long
            Const SITE_BLOCK As Byte = 5   ' Read blocks of 1,024 bytes.
            Const X_BLOCK As Byte = 7
            Const Y_BLOCK As Byte = 8
            Const LOSS_BLOCK As Byte = 2


            Do While bytesRead  reader.BaseStream.Length
                buffer = reader.ReadBytes(SITE_BLOCK)

                'Console.WriteLine(Encoding.Default.GetString(buffer))
                site = CLng(Encoding.Default.GetString(buffer))
                bytesRead += SITE_BLOCK
                buffer = reader.ReadBytes(X_BLOCK)
                'Console.WriteLine(Encoding.Default.GetString(buffer))
                x = CLng(Encoding.Default.GetString(buffer))
                bytesRead += X_BLOCK
                buffer = reader.ReadBytes(Y_BLOCK)
                'Console.WriteLine(Encoding.Default.GetString(buffer))
                y = CLng(Encoding.Default.GetString(buffer))
                bytesRead += Y_BLOCK
                buffer = reader.ReadBytes(LOSS_BLOCK)
                'Console.WriteLine(Encoding.Default.GetString(buffer))
                loss = CLng(Encoding.Default.GetString(buffer))
                bytesRead += LOSS_BLOCK
                Console.WriteLine(reader.PeekChar.ToString)
                If Not reader.PeekChar = 13 Then
                    buffer = reader.ReadBytes(1)
                    loss = loss & CLng(Encoding.Default.GetString(buffer))
                    bytesRead += 1
                Else
                    reader.ReadByte()
                End If
                reader.ReadByte()


                'buffer = reader.ReadBytes(2)
                'bytesRead = bytesRead + buffer.Length
                i += 1
                addPLM(i, site, loss, x, y)
            Loop
            reader.Close()
        Catch E As Exception
            ' Let the user know what went wrong.
            Console.WriteLine("The PLM file could not be read at line " & i & ":")
            Console.WriteLine(E.Message)
        End Try

        Dim runLength As Global.System.TimeSpan = Now.Subtract(startTime)
        Dim millisecs As Integer = runLength.Milliseconds
        Console.WriteLine("Time taken in new PLM: " & millisecs)

    End Sub
```

Appendix C

The below illustration is an example of how the interactive map would appear. Clicking on a grid reference in the main map (A) would then load up a zoomed image with more detailed coverage colouring (B). Clicking on a site within this zoomed image would then present the site information (C)



Appendix D

The below code details the structure of the STP class and also its sub-list; servers.

```
Imports Microsoft.VisualBasic

Public Class stpServer
    Public Property serverName As Long
    Public Property signalStrength As Long
    Public Property active As Byte
End Class

Public Class serviceThreshold
    Public Property ID As Long
    Public Property X As Double
    Public Property Y As Double
    Public Property threshold As Double
    Public Property servers As New List(Of stpServer)
    Public Property selectedServer As Long
    Public Property selectedStrength As Long
End Class
```

Glossary

Algorithm - a process or set of rules to be followed in calculations or other problem-solving operations.

Base Station Controller - the part of a mobile telephone network that handles allocation of radio channels, receives measurements from mobile phones, controls handovers from one base station transceiver (BST) to another, and stores database information for all sites.

Base Station Transceiver - equipment to assist wireless communication between a mobile network and user equipment, such as a mobile telephone.

ByteReader – A class within VB.NET that implements a file reader that reads data from a file a byte at a time.

dBm - power ratio in decibels (dB) of the measured power referenced to one milliwatt (mW).

Decibel - the unit used to measure the intensity of sound or the power level of electrical signals through comparing it with a given level on a logarithmic scale.

EIRP - equivalent isotropic radiated power. This is the power that would have to be emitted in all directions to produce a particular intensity and so takes account of the transmitter power plus the characteristics of the antenna.

Erlangs - a unit of traffic intensity in a telephone system.

Mobile Switching Centre - The mobile switching centre is the primary service delivery node in the network, responsible for routing voice calls and text messaging, as well as other services (such as conference calls, FAX and circuit switched data). The MSC sets up and releases the end-to-end connection, handles mobility and hand-over requirements during the call and takes care of charging and real time pre-paid account monitoring.

Microsoft Developer Network – this is the portion of Microsoft responsible for managing the firm's relationship with developers and testers.

Propagation - the travel of an electrical signal through a medium such as air or free space.

Regular Expressions - a method to describe how to match a text string to a pattern. Some regular expressions can look rather complex (and some are) but this gives them great abilities.

Simulated Annealing - an optimization algorithm that makes random changes to data, in order to improve a specific criterion.

StreamReader – A class within VB.NET that implements a text reader that reads characters from a byte stream in a particular encoding.

Visual Basic 6 – Visual Basic 6 is an object-oriented computer programming language that creates applications designed to run within an Windows environment.

Visual Basic .NET - Visual Basic .NET (VB.NET) is an object-oriented computer programming language that can be viewed as an evolution of Microsoft's Visual Basic (VB) which is generally implemented on the Microsoft .NET Framework.

Visual Studio - an integrated development environment (IDE) from Microsoft; it is used to develop console and graphical user interface applications along with Windows Forms applications, web sites, web applications, and web services in both native code together with managed code for all platforms supported by Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework and Microsoft Silverlight.

Vodafone – a mobile network provider in the United Kingdom.

Table of Abbreviations

BSC – base station controller

BST – base station transceiver

dB – decibels.

EIRP – effective isotropic radiated power.

GUI – graphical user interface

HC – hill climbing

IDE – integrated development environment

MS – millisecond(s)

MSC – mobile switching centre

MSDN – Microsoft developer network

NET – mobile and base station antenna information

NPM – network performance measure

PLM – propagation loss matrix

RTP – reception test point.

SA – simulated annealing

STP – service test point

TS – tabu search

TTP – traffic test point

VB – Visual Basic

WORDA – Write Once Run Anywhere

Bibliography

1. BBC NEWS. 2004. *Phone Masts - A Health Risk?* [online]. [Accessed 25 April 2012]. Available from World Wide Web: http://www.bbc.co.uk/insideout/westmidlands/series6/phone_masts.shtml

2. BBC NEWS. 2006. *'No evidence' of mast health risk*. [online]. [Accessed 24 March 2012]. Available from World Wide Web: http://news.bbc.co.uk/1/hi/health/4771080.stm

3. BBC NEWS. 2011. *O2 begins '4G' LTE mobile data trial in London*. [online]. [Accessed 8 April 2012]. Available from World Wide Web: http://www.bbc.co.uk/news/technology-15717913

4. BBC NEWS. 2011. *UK faces superfast digital divide say network providers*. [online]. [Accessed 4 April 2012]. Available from World Wide Web: http://www.bbc.co.uk/news/technology-15679101

5. BROWN, Gordon. 2002. *Performance Optimization in Visual Basic.NET*. [online]. [Accessed 31 January 2012]. Available from World Wide Web: http://msdn.microsoft.com/en-us/library/aa289513(v=vs.71).aspx

6. CELLAN-JONES, Rory. 2011. *Our 4G future*. [online]. [Accessed 4 April 2012]. Available from World Wide Web: http://www.bbc.co.uk/news/technology-15854582

7. COLOUR BLIND AWARENESS. *Teachers*. [online]. [Accessed 20 April 2012]. Available from World Wide Web: http://www.colourblindawareness.org/teachers/

8. FLEMING, Nic. 2007. *Mobile phone masts 'do not damage health'*. [online]. [Accessed 25 March 2012]. Available from World Wide Web: http://www.telegraph.co.uk/news/uknews/1558441/Mobile-phone-masts-do-not-damage-health.html

9. FRED GLOVER, Manuel Laguna. *Tabu Search*. [online]. [Accessed 25 April 2012]. Available from World Wide Web: http://www.dei.unipd.it/~fisch/ricop/tabu_search_glover_laguna.pdf

10. KURNIAWAN, Budi. 2001. *VB.NET Data Types*. [online]. [Accessed 1 February 2012]. Available from World Wide Web: http://ondotnet.com/pub/a/dotnet/2001/07/30/vb7.html

11. MICROSOFT. *List(Of T) Class*. [online]. [Accessed 1 February 2012]. Available from World Wide Web: http://msdn.microsoft.com/en-us/library/6sh2ey19.aspx

12. MICROSOFT. *ReDim Statement*. [online]. [Accessed 13 March 2012]. Available from World Wide Web: http://msdn.microsoft.com/en-us/library/w8k3cys2(v=vs.71).aspx

13. SCHOOL OF ENGINEERING, UNIVERSITY OF GREENWICH AT MEDWAY. *EIRP*. [online]. [Accessed 12 March 2012]. Available from World Wide Web: http://engweb.info/courses/wdt/lecture06/eirp.GIF

14. SCRIBD. *Hill Climbing Methods*. [online]. [Accessed 5 April 2012]. Available from World Wide Web: http://www.scribd.com/doc/7290255/Hill-Climbing-Methods

15. TECHVIRAL.COM. *BST*. [online]. [Accessed 5 March 2012]. Available from World Wide Web: http://www.techviral.com/wp-content/uploads/2012/01/telephonie-mobile-images-reseau-cellulaire.png

16. THE ITU ASSOCIATION OF JAPAN. *Chapter 7. W-CDMA Technology*. [online]. [Accessed 14 April 2012]. Available from World Wide Web: http://www.ituaj.jp/06_ic/training/cellular/04/chap07_01.pdf

17. WIKIPEDIA. *Simulated Annealing*. [online]. [Accessed 18 March 2012]. Available from World Wide Web: http://en.wikipedia.org/wiki/Simulated_annealing

18. WIL, Josh. 2004. *ArrayList's vs. generic List for primitive types and 64-bits*. [online]. [Accessed 14 April 2012]. Available from World Wide Web: http://blogs.msdn.com/b/joshwil/archive/2004/04/13/112598.aspx