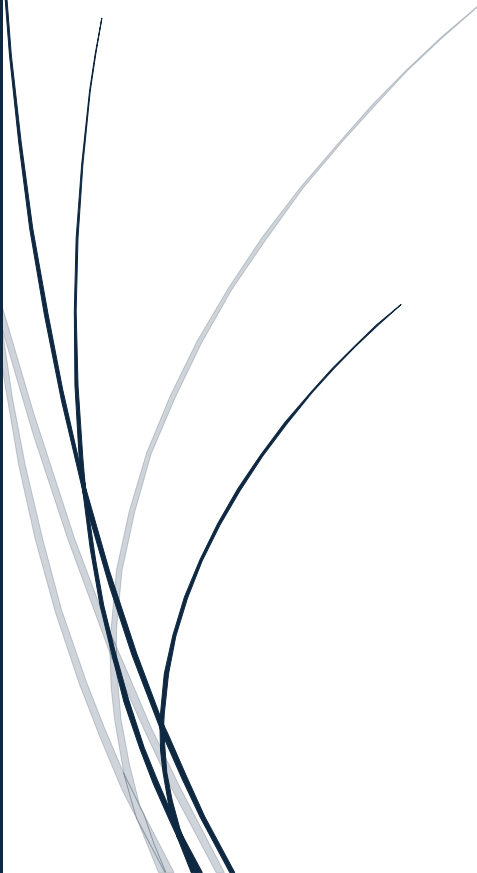


1/10/2025

WIM

Technical Assessment



Dewald Oosthuizen
0734297951

Contents

Optimal Cell Tower Frequency Assignment _____	3
Background _____	3
Initial Decision Making _____	4
Coordinate System _____	4
Distance Calculations _____	5
Programming Language _____	5
Application Development _____	5
Input Functionality _____	6
Graph Creation _____	8
Distance Calculation _____	9
Graph Illustration _____	10
Frequency Assignment _____	11
Frequency Assignment Correctness _____	13
Conclusion _____	15
References _____	16

Table of Figures

Figure 1. Input Data.....	6
Figure 2. Cell Tower Class	6
Figure 3. Frequency Range Input.....	7
Figure 4. N Closest Towers	7
Figure 5. Graph Creation Code	8
Figure 6. Haversine Formula (Dauni et al., 2019:3; MTL, 2024)	9
Figure 7. Distance Matrix.....	9
Figure 8. Full Graph Network	10
Figure 9. Frequency Calculation Code	11
Figure 10. Frequency Assignment	12
Figure 11. Closest 4 and 5 Towers.....	13
Figure 12. Manual Calculations	13
Figure 13. Frequency Assignment Consistency Check.....	14

Optimal Cell Tower Frequency Assignment

This project was initiated in response to a task provided by WIM Technologies to assess a candidate's potential in software development (Technologies, 2025). This project required the creation of a simple application to optimally assign frequencies to a range of Cell Tower, with the goal of assigning frequencies in such a manner that interference is minimised between Cell Towers (Technologies, 2025).

To accomplish this goal, a console application needed to be developed that can offered functionality ranging from dynamic input, allowing any number of Cell Towers to be evaluated, to the visualisation of distance calculations and the overall Cell Tower network. It is important to note the Cell Tower network will be illustrated using a graph, as per the project's requirements.

The rest of this paper will document the process of analysing the given Cell Tower information and the creation of the console application's functionality. Furthermore, visuals will be provided to illustrate the solution to the problem, as well as being used to provide additional context in some sections.

Following this overview of the paper's contents, the detailed structure includes the context of this paper, covering the background, initial research on provided information, and an evaluation of possible technologies to optimally accomplish the task. It then moves to the process of creating the project application, outlining thought processes, the creation of application functionality, and the testing of the created functionality to ensure correctness. Finally, this paper concludes with an evaluation of the project's success in addressing the initial problem, as well as providing future recommendations and improvements for any subsequent projects.

Background

The detailed problem revolves around the optimal assignment of frequencies (presuming radio frequencies) to several Cell Towers ensuring that those farthest apart can share the same frequency to minimise interference (Chou *et al.*, 1982:1321). This is important because when Cell Towers are close proximity and share the same frequency, they can interfere with each other's broadcasts, potentially leading to communication failures or distortion (Mawrey, 1998:2; Porko, 2011:12-15, 18; An *et al.*, 2017:2).

To develop a frequency allocation plan that minimises potential interference, necessary information is provided. This information includes a list of potential Cell Towers, each with a unique identifier, easting and northing coordinates, as well as longitude and latitude coordinates (Langley, 1998:46-47). Along with the Cell Tower information, an initial set of available frequencies is provided, which ranges from one hundred and ten ($N = 110$) to one hundred and fifteen ($N = 115$). All this information can be used to develop an algorithm that determines the most optimal frequency, of a given range, to allocate to a Cell Tower while minimising potential interference between Cell Towers.

Initial Decision Making

Given the information about the Cell Towers, certain design decisions needed to be made in order to effectively create a frequency assignment algorithm, such decisions include the programming language used to create the project and the manner in which the frequencies will be assigned.

Though some initial research and troubleshooting, it was decided that Cell Tower frequencies will be assigned to a target Cell Tower while considering the frequencies used by the tower's closest neighbours. In other words, the frequency given to a target Cell Tower will take into consideration the closest N neighbours to the Cell Tower (N is the number of neighbours to consider) and determine the frequencies that they are using. This list of used frequencies will be compared to the list of all frequencies to determine the frequencies that have yet to be assigned. From the unassigned frequencies, the smallest frequency, or the frequency closest to the lower bound of all available frequencies, will be assigned to the target Cell Tower. This process will then be repeated to assign frequencies to all Cell Towers. In the case where a Cell Tower cannot be assigned a frequency, it will be reported to the operator for further action. This manner of assigning frequencies to Cell Towers is the core idea of this project. The why correctness was ensured will be discussed in a subsequent section.

To implement this idea, additional research was necessary, including how distance between towers will be calculated and which coordinate system to use?

Coordinate System

The first consideration was the coordinate system. As previously mentioned, initial information provided easting and northing coordinates (Universal Transverse Mercator (UTM) system), as well as longitude and latitude coordinates (geographic coordinate system) (Langley, 1998:46-47). Determining the coordinate system to use is a core part of the proposed solution.

The main difference between the UTM (Universal Transverse Mercator) coordinate system and the geographical coordinate system is that the UTM coordinate system simplifies the spatial calculations and mapping by using a grid-based projection to transform earth's curved surface into a two-dimensional plane (Langley, 1998:47). This transformation abstracts complexities, such as angular measurements, making distance and area calculations simpler compared to the geographical coordinates system (Langley, 1998:46-47).

With this information, it would be logical to use the UTM coordinate system, as the task has towers stationed in close proximity to each other; however, to improve the project's applicability to more complex scenarios, the geographical coordinate system will be used (Langley, 1998:46-47). In simpler words, the longitude and latitude coordinates of each Cell Tower will be used in distance calculations, allowing these calculations to account for the earth's curvature and other factors (Langley, 1998:46-47).

Distance Calculations

With the coordinate system chosen a manner of calculating the distance between towers needs to be established. The distance metrics considered where, the Euclidean distance, the Manhattan distance, and the Great-Circle distance using the Haversine Formula (Young & Householder, 1938:19; Krislock & Wolkowicz, 2012:1-2; Malkauthekar, 2013:1-2; Dauni *et al.*, 2019:3).

Of these three metrics, it was determined that the Great-Circle distance (Haversine Formula), also known as “as-the-crow-flies” was the most suitable (Dauni *et al.*, 2019:3; MTL, 2024). This is due to its characteristics of accounting for the earth’s curvature and having acceptable accuracy for long-distance calculations (Dauni *et al.*, 2019:3; MTL, 2024). Consequently, this project will use the Great-Circle distance (Haversine Formula) for distance calculations between towers. Notably, using this distance metric the project's applicability to more complex scenarios.

Programming Language

With the proposed solution in mind, it was necessary to choose the appropriate programming language for its implementation. For this project, C# was chosen primarily due to the author’s familiarity with the language, enabling the use of existing expertise (Microsoft, 2024a).

Libraries

In addition to the main programming language, research was conducted on potential C# libraries to use for simplifying aspects of the project, mainly graph creation, to ensure that the project meets the proposed deadline. During this research, QuickGraph and Msagl were considered when researching options to effectively create the Cell Tower network graph visualisation (KeRNeLith, 2022; Microsoft & Lamb, 2023).

Both libraries offer a range of capabilities; however, initial coding using the QuickGraph library, revealed limitations, particularly in graph visualisation and creation, which led to the use of the Microsoft Msagl library for all graph creation and visualisation needs (KeRNeLith, 2022; Microsoft & Lamb, 2023). All of the mentioned aspects will be discussed in subsequent sections.

On a smaller note, the Ookii library was used when creating a dynamic file selector as the built in Windows library had consistency problems on the host machine (Ookii, 2021).

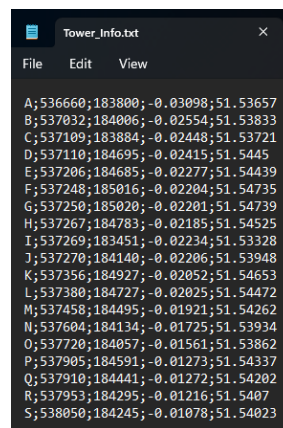
Application Development

This section will detail the creation of the console application to address the problem statement described in the Background section. This section covers the entire development process from the creation of the application’s input functionality to the assignment of Cell Tower frequencies and output validation.

Input Functionality

Before the application can accomplish its task, input is required. To gather the information on Cell Towers and the frequency range to use when performing allocation, dynamic functionality was implemented.

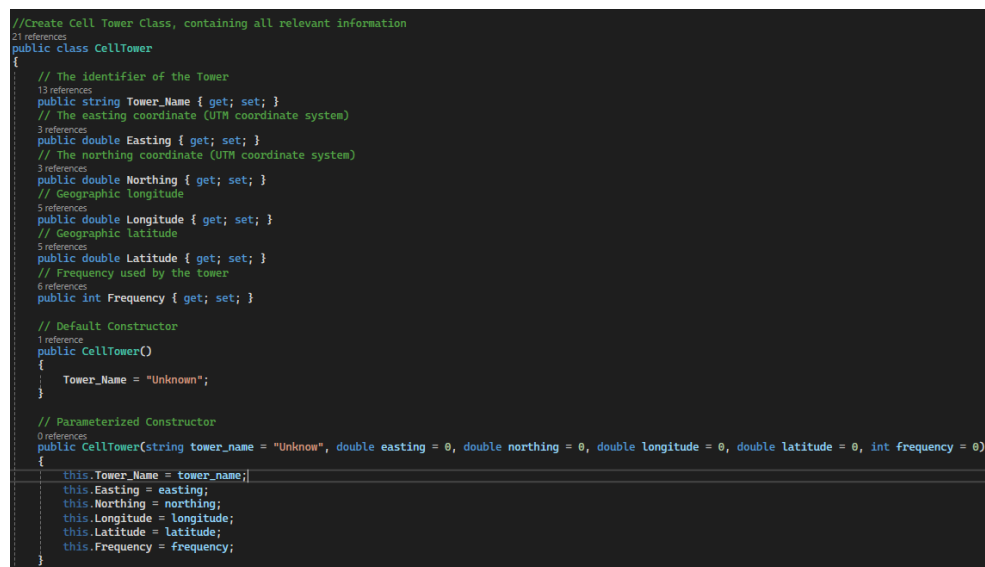
However, before the data can be loaded, it needs to be in a standardised format. To accomplish this, the Cell Tower information needs to be stored in a text file in the format of Cell Tower ID, Easting, Northing, Longitude, and Latitude, each separated with a semicolon. This approach was chosen due to the low setup time required. After this preprocessing step, users can select the text file through the application to load the data. The format is illustrated in Figure 1. Input Data.



```
A;536660;183800;-0.03098;51.53657
B;537032;184006;-0.02554;51.53833
C;537109;183884;-0.02448;51.53721
D;537110;184695;-0.02415;51.5445
E;537206;184685;-0.02277;51.54439
F;537248;185016;-0.02204;51.54735
G;537250;185020;-0.02201;51.54739
H;537267;184783;-0.02185;51.54525
I;537269;183451;-0.02234;51.53328
J;537270;184140;-0.02206;51.53948
K;537356;184927;-0.02052;51.54653
L;537380;184727;-0.02025;51.54472
M;537458;184495;-0.01921;51.54262
N;537604;184134;-0.01725;51.53934
O;537720;184057;-0.01561;51.53862
P;537905;184591;-0.01273;51.54337
Q;537910;184441;-0.01272;51.54202
R;537953;184295;-0.01216;51.5407
S;538050;184245;-0.01078;51.54023
```

Figure 1. Input Data

After the file selection the data is loaded into a list of type Cell Tower. The Cell Tower class is a custom class created to contain Cell Tower's information and simplify the future creation of graph nodes. The Cell Tower class is illustrated in Figure 2. Cell Tower Class.



```
//Create Cell Tower Class, containing all relevant information
21 references
public class CellTower
{
    // The identifier of the Tower
    13 references
    public string Tower_Name { get; set; }
    // The easting coordinate (UTM coordinate system)
    2 references
    public double Easting { get; set; }
    // The northing coordinate (UTM coordinate system)
    3 references
    public double Northing { get; set; }
    // Geographic longitude
    3 references
    public double Longitude { get; set; }
    // Geographic latitude
    5 references
    public double Latitude { get; set; }
    // Frequency used by the tower
    6 references
    public int Frequency { get; set; }

    // Default Constructor
    1 reference
    public CellTower()
    {
        Tower_Name = "Unknown";
    }

    // Parameterized Constructor
    0 references
    public CellTower(string tower_name = "Unknown", double easting = 0, double northing = 0, double longitude = 0, double latitude = 0, int frequency = 0)
    {
        this.Tower_Name = tower_name;
        this.Easting = easting;
        this.Northing = northing;
        this.Longitude = longitude;
        this.Latitude = latitude;
        this.Frequency = frequency;
    }
}
```

Figure 2. Cell Tower Class

Using a list along with objects enables extreme flexibility, as the list can continuously grow along with the input, and encapsulating Cell Tower information in an object allows standardisation, enabling simplistic data management and scalability (Liang, 2015:322, 366).

Along with the input for Cell Tower information, the selection of possible frequency ranges to be used is also obtained. This is achieved through the application terminal by prompting users to specify the lower bound (the first frequency available for use) and the upper bound (the last frequency available for use). This input method is illustrated in Figure 3. Frequency Range Input.

```
Options available in the program:
1: read file and calculate tower frequencies
2: Change number of closest towers to consider when assigning frequencies
3: exit program

Enter a number:
1

Loading file contents.....

The File has Loaded

Please enter lower bound for possible frequencies (Inclusive):
110

Please enter upper bound for possible frequencies (Inclusive):
115
```

Figure 3. Frequency Range Input

This method is simplistic to set up correctly, which is the main reason for its selection.

Along with providing the Cell Tower information and frequency range, the user can also specify the number of closest towers to a target tower to be considered when assigning frequencies. This is illustrated in Figure 4. N Closest Towers.

```
C:\Users\User\OneDrive\Com x + v

Options available in the program:
1: read file and calculate tower frequencies
2: Change number of closest towers to consider when assigning frequencies
3: exit program

Enter a number:
2

The current number of closest towers to consider is 5.
Enter the new number of closest towers to consider:
10
```

Figure 4. N Closest Towers

This input is important for several reasons; however, the main reason is to facilitate hyperparameter optimisation, allowing the user to select the most optimal number of closest towers to a target tower for their given situation (Bischl *et al.*, 2023:1).

Graph Creation

With all the necessary information gathered it is now possible to implement the proposed solution to the Cell Tower Frequency assignment problem.

The initial step involved creating a graph to represent all the Cell Tower information, where towers are stored as vertices and the distances between them are represented as weighted edges connecting the towers. It may not be necessary to create a graph at this stage in the project, as it is possible to solve the problem without creating a graph and using lists instead; however, as the original task requires an illustration of the solution in a graph format, it was necessary to create a graph at this stage not only for the author to familiarise themselves with the graph creation library but also to evaluate the library's suitability for the project, which did prevent future problems as described in the Libraries section. The code for the graph creation is illustrated in Figure 5. Graph Creation Code.

```
// Generate Graph and assign frequencies
1 reference
public static Graph CreateGraph(List<CellTower> lCellTowers, List<int> lFrequencies)
{
    // Distance between towers
    double dDistance = 0;

    // User feedback
    Console.WriteLine("\n" + new string('-', 100));
    Console.WriteLine("\nCreating Graph.....");

    // Create an undirected graph that has vertices (cell towers) and edges (one double property for frequencies)
    Graph graph_Cell_Towers = new Graph("Cell_Tower_Network");

    // Edge and Nodes used for Customisation
    Node nNode = null;
    Edge eEdge = null;

    // Edge case (Only one tower)
    if (lCellTowers.Count < 2)
    {
        return graph_Cell_Towers;
    }

    // Add all cell tower objects to a Msagl graph as vertices
    for (int i = 0; i < lCellTowers.Count; i++)
    {
        nNode = graph_Cell_Towers.AddNode(lCellTowers[i].Tower_Name);
        nNode.UserData = lCellTowers[i];
        nNode.LabelText = lCellTowers[i].Tower_Name;
        nNode.Attr.Shape = Shape.Circle;
        nNode.Attr.FillColor = Color.LightCyan;
        nNode.Attr.Color = Color.Black;
    }

    // Add undirected edges between all towers, while avoiding duplicates
    for (int iTowerOne = 0; iTowerOne < lCellTowers.Count; iTowerOne++)
    {
        for (int iTowerTwo = iTowerOne + 1; iTowerTwo < lCellTowers.Count; iTowerTwo++) // Used to avoid duplicate edges (undirected graph, edges directional)
        {
            // Calculate the great-circle distance between the two towers
            dDistance = DistanceCalculation.CalDistance(
                lCellTowers[iTowerOne].Longitude,
                lCellTowers[iTowerOne].Latitude,
                lCellTowers[iTowerTwo].Longitude,
                lCellTowers[iTowerTwo].Latitude
            );

            // Create a weighted edge between the two towers with the distance between the two as an edge property
            eEdge = graph_Cell_Towers.AddEdge(lCellTowers[iTowerOne].Tower_Name, lCellTowers[iTowerTwo].Tower_Name);
            eEdge.LabelText = dDistance.ToString("F2") + " m";
            eEdge.Attr.Weight = (int)(dDistance * 1000.00); // Convert to millimeter as weight cannot accept a double
        }
    }

    // Return the created Cell Tower network
    Console.WriteLine("\nGraph Created!!\n");
    return graph_Cell_Towers;
}
```

Figure 5. Graph Creation Code

As indicated in Figure 5. Graph Creation Code: A custom distance calculation method was used when calculating the distance between two Cell Towers and subsequently stored as a weight on a graph edge. It is important to note that the graph library (Msagl) does not support any decimals as weights; therefore, the weight was stored as an integer but converted from meters to millimetres to retain an acceptable precision level for the task (Microsoft & Lamb, 2023).

Distance Calculation

As mentioned previously, the haversine formula was used when calculating the distance between towers (Dauni *et al.*, 2019:3; MTL, 2024). The function in which this formula is applied accepts the longitude and latitude of two towers as input. These inputs are subsequently converted from degrees to radians, which is another angular measurement used in trigonometry and for defining trigonometric functions (Dauni *et al.*, 2019:3; MTL, 2024). The converted coordinates are then used to in the haversine formula as, illustrated in Figure 6. Haversine Formula , to calculate the distance between two towers while accounting for the Earth's curvature (Langley, 1998:46-47; Dauni *et al.*, 2019:3; MTL, 2024).

$$a = \sin^2\left(\frac{\Delta\text{lat}}{2}\right) + \cos(\text{lat}_1) \cdot \cos(\text{lat}_2) \cdot \sin^2\left(\frac{\Delta\text{lon}}{2}\right)$$

$$c = 2 \cdot \text{atan2}\left(\sqrt{a}, \sqrt{1-a}\right)$$

$$d = R \cdot c$$

Figure 6. Haversine Formula (Dauni *et al.*, 2019:3; MTL, 2024)

Using this formula all the distances, in meters, between towers were calculated, as illustrated in Figure 7. Distance Matrix.

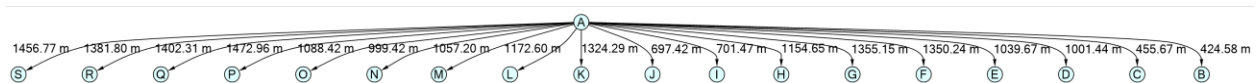
Distance Matrix of Cell Tower Network (meters):																			
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
A	0.00	424.57	455.67	1001.44	1039.66	1350.24	1355.15	1154.65	701.47	697.41	1324.29	1172.60	1057.20	999.42	1088.42	1472.96	1402.31	1381.80	1456.77
B	424.57	0.00	144.68	693.55	701.32	1032.92	1037.74	811.58	604.25	272.85	976.74	800.08	648.18	584.89	688.30	1049.46	978.05	963.23	1043.61
C	455.67	144.68	0.00	811.84	807.99	1141.34	1146.06	913.34	461.90	303.20	1073.11	885.82	704.15	553.91	633.90	1063.97	974.51	937.32	1006.38
D	1001.44	693.55	811.84	0.00	96.32	349.27	354.18	179.79	1255.27	577.25	337.96	271.10	400.95	747.10	882.06	800.56	838.09	931.67	1040.56
E	1039.66	701.32	807.99	96.32	0.00	333.36	338.07	114.98	1237.11	548.78	284.63	178.28	315.54	679.76	811.36	704.29	744.12	841.61	950.54
F	1350.24	1032.92	1141.34	349.27	333.36	0.00	4.91	234.14	1566.39	876.08	139.30	317.91	561.81	951.33	1068.92	782.11	876.55	1007.90	1111.70
G	1355.15	1037.74	1146.06	354.18	338.07	4.91	0.00	238.48	1570.88	880.54	140.73	321.23	565.27	954.79	1072.11	782.93	878.05	1009.77	1113.43
H	1154.65	811.58	913.34	179.79	114.98	234.14	238.48	0.00	1332.92	642.47	169.65	125.50	345.13	730.92	855.19	665.16	727.19	840.59	948.50
I	701.47	604.25	461.90	1255.27	1237.11	1566.39	1570.88	1332.92	0.00	690.45	1480.35	1281.68	1062.06	761.11	755.33	1305.50	1179.10	1085.87	1113.22
J	697.41	272.85	303.20	577.25	548.78	876.08	880.54	642.47	690.45	0.00	792.01	596.62	401.39	333.40	456.73	777.67	705.77	698.77	785.44
K	1324.29	976.74	1073.11	337.96	284.63	139.30	140.73	169.65	1480.35	792.01	0.00	202.35	444.60	831.78	943.87	643.88	737.32	869.58	972.90
L	1172.60	800.08	885.82	271.10	178.28	317.91	321.23	125.50	1281.68	596.62	202.35	0.00	244.61	633.89	751.20	541.87	601.75	716.91	824.43
M	1057.20	648.18	704.15	400.95	315.54	561.81	565.27	345.13	1062.06	401.39	444.60	244.61	0.00	389.53	510.29	456.33	454.26	532.85	641.43
N	999.42	584.89	553.91	747.10	679.76	951.33	954.79	730.92	761.11	333.40	831.78	633.89	389.53	0.00	138.99	546.98	432.87	383.56	458.79
O	1088.42	688.30	633.90	882.06	811.36	1068.92	1072.11	855.19	755.33	456.73	943.87	751.20	510.29	138.99	0.00	565.11	428.12	332.67	379.42
P	1472.96	1049.46	1063.97	800.56	704.29	782.11	782.93	665.16	1305.50	777.67	643.88	541.87	456.33	546.98	565.11	0.00	150.28	299.83	374.71
Q	1402.31	978.05	974.51	838.09	744.12	876.55	878.05	727.19	1179.10	705.77	737.32	601.75	454.26	432.87	428.12	150.28	0.00	151.97	240.30
R	1381.80	963.23	937.32	931.67	841.61	1007.90	1009.77	840.59	1085.87	698.77	869.58	716.91	532.85	383.56	332.67	299.83	151.97	0.00	108.93
S	1456.77	1043.61	1006.38	1040.56	950.54	1111.70	1113.43	948.50	1113.22	785.44	972.90	824.43	641.43	458.79	379.42	374.71	240.30	108.93	0.00

Figure 7. Distance Matrix

To validate the accuracy of these calculations, a random selection of towers was tested by manually calculating their distances to other towers and cross-referencing the results with an online calculator, which confirmed their correctness (MTL, 2024).

Graph Illustration

With all the Cell Towers and edges created, Msagl's graph visualisation capabilities were utilised to create a full graph visualisation. The following image illustrates one of the Cell Towers with weighted edges, containing the distance from the target tower to the other towers, connected to all other towers (Microsoft & Lamb, 2023).



Unfortunately, due to the many edges that are created the full illustration of the graph is complex, as seen in Figure 8. Full Graph Network.



Figure 8. Full Graph Network

This graph illustration can be improved in future projects by leveraging geometry nodes, a feature of the Msagl library that enables the plotting of nodes based on their coordinates. However, due to technical difficulties and time constraints, this feature could not be implemented in the current project (Microsoft & Lamb, 2023).

Frequency Assignment

With all the necessary information calculated and gathered, the proposed method of assigning frequencies to Cell Towers, as mentioned in the Initial Decision Making Section, was implemented as an algorithm. The code is illustrated in Figure 9. Frequency Calculation Code.

```
// Calculate the frequencies of each tower
1 reference
public static Graph CalFrequencies(Graph graph_Cell_Towers, List<int> lFrequencies, int iFrequency_Consideration)
{
    // Used as a copy of lFrequencies to determine available frequencies for the target tower
    List<int> lAvailable_Frequencies = new List<int>();
    // Used frequency of a tower
    int iUsed_Frequency = 0;

    //Used for Information display
    bool bErrors = false;

    // For each CellTower node in the graph (in Alphabetical Order)
    foreach (var Tower in graph_Cell_Towers.Nodes.OrderBy(node => (node.UserData as CellTower).Tower_Name))
    {
        // Retrive all edges of the given target tower
        var All_Tower_Edges = graph_Cell_Towers.Edges.Where(edge => edge.SourceNode == Tower || edge.TargetNode == Tower);

        // Sort all edges in desending order according to their distance from the target tower and take the closest n towers
        All_Tower_Edges = All_Tower_Edges.OrderBy(edge => edge.Attr.Weight).Take(iFrequency_Consideration);

        // Make a copy of all available frequencies
        lAvailable_Frequencies = new List<int>(lFrequencies);

        // Check all frequencies used by closest n towers to target tower
        foreach (var Edge in All_Tower_Edges)
        {
            if (Tower == Edge.SourceNode)
            {
                iUsed_Frequency = ((Edge.TargetNode.UserData) as CellTower).Frequency;
                lAvailable_Frequencies.Remove(iUsed_Frequency);
            }
            else
            {
                iUsed_Frequency = ((Edge.SourceNode.UserData) as CellTower).Frequency;
                lAvailable_Frequencies.Remove(iUsed_Frequency);
            }
        }

        // If not all frequencies are used assign the first available frequency to the target tower
        if (lAvailable_Frequencies.Count != 0)
        {
            // Assign a tower the first available frequency
            (Tower.UserData as CellTower).Frequency = lAvailable_Frequencies.First();
        }
        else
        {
            if (bErrors == false)
            {
                bErrors = true;
                Console.WriteLine("\n" + new string('-', 100));
                Console.WriteLine("\nFrequency Assignment Warnings:\n");
            }

            // If no frequencies are available indecate the tower that has the problem
            Console.WriteLine($"No available frequency for {Tower.UserData as CellTower}");
        }
    }

    // Return the graph with each tower now owning a frequency
    return graph_Cell_Towers;
}
```

Figure 9. Frequency Calculation Code

In this algorithm, the previously created graph is used as input, along with a list of valid frequencies and a parameter specifying the N number of closest towers to a target tower to consider before assigning a frequency.

For each node (Cell Tower) in the graph, all its connected edges are added to a list, effectively creating a list containing the distances of each tower to the target tower. This list is then sorted, and the closest N towers are selected based on the previously specified number.

Using this list of the closest towers, the frequencies already used by these towers are compared to the list of all valid frequencies to identify frequencies that have yet to be assigned. From the unassigned frequencies, the smallest frequency, or the frequency closest to the lower bound of all available frequencies, will be assigned to the target Cell Tower.

This process will then be repeated to assign frequencies to all Cell Towers. In cases where a Cell Tower cannot be assigned a frequency, it will be reported to the operator for further action. The completed frequency assignment is illustrated in Figure 10. Frequency Assignment on the list of initial Cell Towers provided, while considering the closest five (N = 5) towers for each target tower.

Tower information:					
Tower Name: A	Easting: 536660	Northing: 183800	Longitude: -0.03098	Latitude: 51.53657	Frequency: 110
Tower Name: B	Easting: 537032	Northing: 184006	Longitude: -0.02554	Latitude: 51.53833	Frequency: 111
Tower Name: C	Easting: 537109	Northing: 183884	Longitude: -0.02448	Latitude: 51.53721	Frequency: 112
Tower Name: D	Easting: 537110	Northing: 184695	Longitude: -0.02415	Latitude: 51.5445	Frequency: 110
Tower Name: E	Easting: 537206	Northing: 184685	Longitude: -0.02277	Latitude: 51.54439	Frequency: 111
Tower Name: F	Easting: 537248	Northing: 185016	Longitude: -0.02204	Latitude: 51.54735	Frequency: 110
Tower Name: G	Easting: 537250	Northing: 185020	Longitude: -0.02201	Latitude: 51.54739	Frequency: 112
Tower Name: H	Easting: 537267	Northing: 184783	Longitude: -0.02185	Latitude: 51.54525	Frequency: 112
Tower Name: I	Easting: 537269	Northing: 183451	Longitude: -0.02234	Latitude: 51.53328	Frequency: 113
Tower Name: J	Easting: 537270	Northing: 184140	Longitude: -0.02206	Latitude: 51.53948	Frequency: 110
Tower Name: K	Easting: 537356	Northing: 184927	Longitude: -0.02052	Latitude: 51.54653	Frequency: 113
Tower Name: L	Easting: 537380	Northing: 184727	Longitude: -0.02025	Latitude: 51.54472	Frequency: 114
Tower Name: M	Easting: 537458	Northing: 184495	Longitude: -0.01921	Latitude: 51.54262	Frequency: 113
Tower Name: N	Easting: 537604	Northing: 184134	Longitude: -0.01725	Latitude: 51.53934	Frequency: 111
Tower Name: O	Easting: 537720	Northing: 184057	Longitude: -0.01561	Latitude: 51.53862	Frequency: 112
Tower Name: P	Easting: 537905	Northing: 184591	Longitude: -0.01273	Latitude: 51.54337	Frequency: 110
Tower Name: Q	Easting: 537910	Northing: 184441	Longitude: -0.01272	Latitude: 51.54202	Frequency: 113
Tower Name: R	Easting: 537953	Northing: 184295	Longitude: -0.01216	Latitude: 51.5407	Frequency: 114
Tower Name: S	Easting: 538050	Northing: 184245	Longitude: -0.01078	Latitude: 51.54023	Frequency: 115

Figure 10. Frequency Assignment

This algorithm was run considering the closest four (N = 4) and five (N = 5) towers and resulted in the frequency assignments, illustrated by Figure 11. Closest 4 and 5 Towers.



Figure 11. Closest 4 and 5 Towers

Frequency Assignment Correctness

The most difficult part of the problem space was the evaluation of the frequency assignment's correctness. This was challenging due to the absence of labelled data to compare the results to and other factors (Nasteski, 2017:4; Shyam & Chakraborty, 2021:3).

To validate the correctness of the algorithm's application, as well as confirm the most optimal frequency assignments, the entire algorithm was applied manually and the results compared. This included the manual calculation of distances between towers, and the assignment of frequencies. This comparison is illustrated in the following images.

Cell Tower A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	1	2	3	4	5 Letters					
A		424.576	455.6737	1001.439	1039.665	1350.237	1355.147	1154.647	701.4705	697.4157	1324.292	1172.604	1057.202	999.4213	1088.418	1472.961	1402.313	1381.8	1456.766	424.576	455.6737	697.4157	701.4705	999.4213	C	J	I	N
B	424.576		144.676	693.5466	701.3232	1032.92	1037.737	811.5846	604.2524	272.8462	976.7402	800.0787	648.1798	584.8954	688.3017	1049.465	978.0512	963.2326	1043.608	144.676	272.8462	424.576	584.8954	604.2524	C	J	A	N
C	455.6737	144.676		811.8359	807.9903	1141.344	1146.056	913.3385	461.8976	303.1999	1073.106	885.8181	704.1472	553.9151	633.8981	1063.975	974.5114	937.3216	1006.382	144.676	303.1999	455.6737	461.8976	553.9151	B	J	A	I
D	1001.439	693.5466	811.8359		96.31895	349.2702	354.1834	179.7883	1255.269	577.2505	337.9599	271.1045	400.9537	747.0955	882.0615	800.5599	838.0941	931.6691	1040.559	96.31895	179.7883	271.1045	337.9599	349.2702	E	H	L	K
E	1039.665	701.3232	807.9903	96.31895		333.3567	338.0756	114.9852	1237.111	548.7815	284.6266	178.2849	315.5424	679.7639	811.3563	704.291	744.12	841.611	950.54	96.31895	114.9852	178.2849	284.6266	315.5424	D	H	L	M
F	1350.237	1032.92	1141.344	349.2702	333.3567		4.913245	234.1393	1566.394	876.0804	139.2995	317.9134	561.8065	951.3339	1068.921	782.1102	876.5552	1007.896	1111.703	4.913245	139.2995	234.1393	317.9134	333.3567	G	K	H	L
G	1355.147	1037.737	1146.056	354.1834	338.0756	4.913245		238.4797	1570.875	880.5389	140.7279	321.2252	565.2662	954.7898	1072.111	782.9359	878.0527	1009.767	1113.431	4.913245	140.7279	238.4797	321.2252	338.0756	F	K	H	E
H	1154.647	811.5846	913.3385	179.7883	114.9852	234.1393	238.4797		1332.918	642.4743	169.6476	125.4996	345.1348	730.9242	855.1897	665.1602	727.1932	840.5931	948.5025	114.9852	125.4996	169.6476	179.7883	234.1393	E	L	K	D
I	701.4705	604.2524	461.8976	1255.269	1237.111	1566.394	1570.875	1332.918		690.4491	1480.348	1281.683	1062.065	761.1118	755.3295	1305.503	1179.095	1085.866	1113.218	461.8976	604.2524	690.4491	701.4705	755.3295	C	B	J	A
J	697.4157	272.8462	303.1999	577.2505	548.7815	876.0804	880.5389	642.4743	690.4491		792.0069	596.6194	401.3905	333.3976	456.7325	777.6748	705.7745	698.7673	785.4432	272.8462	303.1999	333.3976	401.3905	456.7325	B	C	N	M
K	1324.292	976.7402	1073.106	337.9599	284.6266	139.2995	140.7279	169.6476	1480.348	792.0069		202.3522	444.6046	831.7839	943.8704	643.8792	737.3249	869.5776	972.9009	139.2995	140.7279	169.6476	202.3522	284.6266	F	G	H	L
L	1172.604	800.0787	885.8181	271.1045	178.2849	317.9134	321.2252	125.4996	1281.683	596.6194	202.3522		244.6064	633.8891	751.2001	541.8703	601.752	716.9108	824.4277	125.4996	178.2849	202.3522	244.6064	271.1045	H	E	K	M
M	1057.202	648.1798	704.1472	400.9537	315.5424	561.8065	565.2662	345.1348	1062.065	401.3905	444.6046	244.6064		389.5275	510.2894	456.3273	454.2608	532.8456	641.4268	244.6064	315.5424	345.1348	389.5275	400.9537	L	E	H	N
N	999.4213	584.8954	553.9151	747.0955	679.7639	951.3339	954.7898	730.9242	761.1118	333.3976	831.7839	633.8891	389.5275		138.9883	546.9802	432.8651	383.5575	458.789	138.9883	333.3976	383.5575	389.5275	432.8651	O	J	R	M
O	1088.418	688.3017	633.8981	882.0615	811.3563	1068.921	1072.111	855.1897	755.3295	456.7325	943.8704	751.2001	510.2894	138.9883		565.1118	428.1207	332.6717	379.4157	138.9883	332.6717	379.4157	428.1207	456.7325	N	R	S	Q
P	1472.961	1049.465	1063.975	800.5599	704.291	782.1102	782.9359	665.1602	1305.503	777.6748	643.8792	541.8703	456.3273	546.9802	565.1118		150.282	299.8297	374.7076	150.282	299.8297	374.7076	456.3273	541.8703	Q	R	S	M
Q	1402.313	978.0512	974.5114	838.0941	744.12	876.5552	878.0527	727.1932	1179.095	705.7745	737.3249	601.752	454.2608	432.8651	428.1207	150.282		151.9699	240.3032	150.282	151.9699	240.3032	428.1207	432.8651	P	R	S	O
R	1381.8	963.2326	937.3216	931.6691	841.611	1007.896	1009.767	840.5931	1085.866	698.7673	869.5776	716.9108	532.8456	383.5575	332.6717	299.8297	151.9699		108.9328	108.9328	151.9699	299.8297	332.6717	383.5575	S	Q	P	O
S	1456.766	1043.608	1006.382	1040.559	950.54	1111.703	1113.431	948.5025	1113.218	785.4432	972.9009	824.4277	641.4268	458.789	379.4157	374.7076	240.3032	108.9328		108.9328	240.3032	374.7076	379.4157	458.789	R	Q	P	O

Figure 12. Manual Calculations

Figure 12. Manual Calculations illustrates the excel version of the manual calculations conducted (Microsoft, 2024b).

4																			
A	110	111	112	113	114	115	B	C	J	I									
B	110	111	112	113	114	115	C	J	A	N									
C	110	111	112	113	114	115	B	J	A	I									
D	110	111	112	113	114	115	E	H	L	K									
E	110	111	112	113	114	115	D	H	L	K									
F	110	111	112	113	114	115	G	K	H	L									
G	110	111	112	113	114	115	F	K	H	L									
H	110	111	112	113	114	115	E	L	K	D									
I	110	111	112	113	114	115	C	B	J	A									
J	110	111	112	113	114	115	B	C	N	M									
K	110	111	112	113	114	115	F	G	H	L									
L	110	111	112	113	114	115	H	E	K	M									
M	110	111	112	113	114	115	L	E	H	N									
N	110	111	112	113	114	115	O	J	R	M									
O	110	111	112	113	114	115	N	R	S	Q									
P	110	111	112	113	114	115	Q	R	S	M									
Q	110	111	112	113	114	115	P	R	S	O									
R	110	111	112	113	114	115	S	Q	P	O									
S	110	111	112	113	114	115	R	Q	P	O									
5																			
A	110	111	112	113	114	115	B	C	J	I	N								
B	110	111	112	113	114	115	C	J	A	N	I								
C	110	111	112	113	114	115	B	J	A	I	N								
D	110	111	112	113	114	115	E	H	L	K	F								
E	110	111	112	113	114	115	D	H	L	K	M								
F	110	111	112	113	114	115	G	K	H	L	E								
G	110	111	112	113	114	115	F	K	H	L	E								
H	110	111	112	113	114	115	E	L	K	D	F								
I	110	111	112	113	114	115	C	B	J	A	O								
J	110	111	112	113	114	115	B	C	N	M	O								
K	110	111	112	113	114	115	F	G	H	L	E								
L	110	111	112	113	114	115	H	E	K	M	D								
M	110	111	112	113	114	115	L	E	H	N	D								
N	110	111	112	113	114	115	O	J	R	M	Q								
O	110	111	112	113	114	115	N	R	S	Q	J								
P	110	111	112	113	114	115	Q	R	S	M	L								
Q	110	111	112	113	114	115	P	R	S	O	N								
R	110	111	112	113	114	115	S	Q	P	O	N								
S	110	111	112	113	114	115	R	Q	P	O	N								

Figure 13. Frequency Assignment Consistency Check

Figure 13. Frequency Assignment Consistency Check illustrates the manual application of the frequency assignment process, while Figure 10. Frequency Assignment, **Error! Reference source not found.** illustrates the results of the application computations. Both results were compared, which did confirm the application consistency in the algorithm's application.

Using both the manual calculations illustrated in Figure 12. Manual Calculations and Figure 13. Frequency Assignment Consistency Check, the algorithm's correctness and optimality in frequency assignments were evaluated. This evaluation confirmed the algorithm's reliability and effectiveness in assigning frequencies. However, it is important to note that the optimality of frequency assignments remains subjective.

The lack of predefined distance metrics to indicate interference ranges, along with the absence of results to compare the frequency assignments to, introduced uncertainty. Furthermore, the manual evaluation process was prone to human error, which could have impacted the assessment.

Nonetheless, every effort was made to ensure that the frequency assignment for each Cell Tower was as accurate as possible within these limitations.

Conclusion

This project was initiated in response to a task provided by WIM Technologies to assess a candidate's potential in software development (Technologies, 2025). More specifically, the task required the creation of an application that can optimally assign frequencies to a range of Cell Towers, with the goal of assigning frequencies in such a manner that interference is minimised between Cell Towers. To address this, a console application was developed, which included functionality ranging from dynamic input to Cell Tower network graph display.

To address the original problem of optimal frequency assignments, an algorithm was developed that considered the closest N towers to a given target tower before assigning a frequency. Through meticulous evaluation, it was confirmed to the greatest extent possible that the algorithm consistently and optimally applied frequencies.

Therefore, the original task can be considered addressed. However, there are some recommendations for possible improvements that could further enhance the application's applicability. These recommendations include using persistent storage such as a database to store the Cell Tower information, which would allow more flexibility in designing application functionality, employing more complex mathematical techniques to ensure the correctness of the algorithm's frequency assignments to address the problems listed previously, and creating a UI to allow a simpler and more visually appealing illustration of the Cell Tower network in an understandable way.

References

- An, T., Chen, X., Mohan, P. & Lao, B.-Q. 2017. Radio frequency interference mitigation. *arXiv preprint arXiv:1711.01978*,
- Bischl, B., Binder, M., Lang, M., Pielok, T., Richter, J., Coors, S., ... Boulesteix, A.L. 2023. Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 13(2):e1484.
- Chou, C.K., Guy, A.W. & Galambos, R. 1982. Auditory perception of radio-frequency electromagnetic fields. *The Journal of the Acoustical Society of America*, 71(6):1321-1334.
- Dauni, P., Firdaus, M., Asfariani, R., Saputra, M., Hidayat, A. & Zulfikar, W. 2019. Implementation of Haversine formula for school location tracking. In. *Journal of Physics: Conference Series*. IOP Publishing. p 077028.
- KeRNeLith. 2022. *QuickGraph*. <https://www.nuget.org/packages/QuikGraph> Date of access: 13 January.
- Krislock, N. & Wolkowicz, H. 2012. *Euclidean distance matrices and applications*. Springer.
- Langley, R.B. 1998. The UTM grid system. *GPS world*, 9(2):46-50.
- Liang, Y.D. 2015. *Introduction to Java Programming, Comprehensive Version*. Tenth Edition. Upper Saddle River, New Jersey, USA: Pearson Education, Inc.
- Malkauthekar, M. 2013. Analysis of euclidean distance and manhattan distance measure in face recognition. In. *Third International Conference on Computational Intelligence and Information Technology (CIIT 2013)*. IET. pp. 503-507.
- Mawrey, R.S. 1998. Radio Frequency interference and Antenna sites. *Unisite*, sitio web: <http://www.unisite.com>,
- Microsoft. 2024a. *C# language documentation*. <https://learn.microsoft.com/en-us/dotnet/csharp/> Date of access: 10 October.
- Microsoft. 2024b. *Microsoft Excel*. <https://www.microsoft.com/en-za/microsoft-365/excel> Date of access: 7 May.
- Microsoft & Lamb, A. 2023. *Microsoft.Msagl*. <https://www.nuget.org/packages/Microsoft.Msagl> Date of access: 13 January.
- MTL, M.T.S. 2024. *Calculate distance, bearing and more between Latitude/Longitude points*. <https://www.movable-type.co.uk/scripts/latlong.html> Date of access: 13 January.
- Nasteski, V. 2017. An overview of the supervised machine learning methods. *Horizons. b*, 4(51-62):56.

Ookii. 2021. *Ookii.Dialogs.WinForms*. <https://www.nuget.org/packages/Ookii.Dialogs.WinForms>
Date of access: 13 January.

Porko, J.-P.G. 2011. *Radio frequency interference in radio astronomy*. Aalto University.

Shyam, R. & Chakraborty, R. 2021. Machine learning and its dominant paradigms. *Journal of Advancements in Robotics*, 8(2):1-10p.

Technologies, W. 2025. *Developing solutions that address real-world telecoms complexities*.
<https://www.wimtechnologies.com/> Date of access: 13 January.

Young, G. & Householder, A.S. 1938. Discussion of a set of points in terms of their mutual distances. *Psychometrika*, 3(1):19-22.