TECH 4

Obligatory assignment

November 2025

Aksel Johan Aasehaug Søraas

Jørgen Lund Johannessen

Contents

See attached files for Python code.

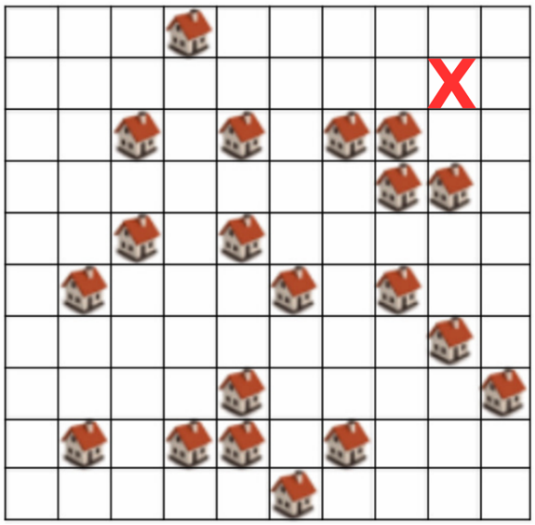
Problem A1.

Note on indexing:

In this task, we have decided to forego the coordinate system given in the task (E5, A10, etc.).

Instead, we use more useful indexes from 1 to 10. The coordinates will start at the rows, and end at the columns. For example, the marked square will be called (2, 9) instead of I9 from now on.

Note that our code in Python will start at 0 and end at 9. Other than this, the indexes between this paper and the code will be identical. (The marked example will be noted as (1, 8) in Python.)



Indexes:

$r, c \in \{1,2, \dots ,10\}$

$\Rightarrow$ ***Comment***: *The r and c index represent the rows and columns of the coordinate system going from 1 to 10.*

$j \in \{1,2, \dots ,20\}$

$\Rightarrow$ ***Comment***: *The j index represents the neighborhoods going from 1 to 20, if the number of neighborhoods was different, we would change this index.*

Parameters:

$$F_{r,c} = \begin{cases} 2500 \; for \; (r, \;\; c) \in \{(5,5),(5,6),(6,5),(6,6)\} \\ 2000 \; for \; other \; (r,c) \end{cases}, \qquad \forall r, c$$

$\Rightarrow$ ***Comment***: *Fixed cost for installing an electrical station at all coordinates (r, c)*

$N_j = coordinates \; (r,c) \; of \; neighborhood \; j \; , \qquad \forall j$

$\Rightarrow$ ***Comment***: *Location of all given neighborhoods*

$D_{j,r,c} = distance \; between \; N_j \; and \; (r,c), \qquad \forall j, r, c$

$\Rightarrow$ ***Comment***: *Distance from all neighborhoods to all possible coordinates, calculated using the formula given in the task. (in km)*

$V = 100$

$\Rightarrow$ ***Comment***: *Variable cost of installing 1 km of electrical wire*

Decision variables:

$$s_{r,c} = \begin{cases} 1 \; if \; a \; station \; is \; installed \; on \; coordinates \; (r,c) \\ 0 \; otherwise \end{cases}, \qquad \forall r, c$$

⇒ ***Comment***: *Binary variable showing if there is a station at every coordinate. (100 variables)*

$$w_{j,r,c} = \begin{cases} 1 \; if \; N_j \; is \; connected \; coordinate \; (r,c) \\ 0 \; otherwise \end{cases}, \qquad \forall j, r, c$$

⇒ ***Comment***: *Binary variable showing if a neighborhood is connected to a coordinate by wire. (2000 variables)*

Objective function:

$$Minimize \; Z = \sum_{r=1}^{10}\sum_{c=1}^{10} F_{r,c}s_{r,c} + V\sum_{j=1}^{20}\sum_{r=1}^{10}\sum_{c=1}^{10} D_{j,r,c}w_{j,r,c}$$

⇒ ***Comment***: *Objective function minimizing the sum of fixed and variable costs.*

Constraints:

$$\sum_{r=1}^{10}\sum_{c=1}^{10} w_{j,r,c} = 1, \qquad \forall j$$

⇒ ***Comment***: *Constraint ensuring that all neighborhoods are only connected to one electrical station.*

$$w_{j,r,c} \leq s_{r,c}, \qquad \forall j, r, c$$

⇒ ***Comment***: *Constraint ensuring all connections from neighborhoods go to existing electrical stations.*

$$s_{r,c}, w_{j,r,c} \in \{0, 1\}$$

⇒ ***Comment***: *Binary constraints*

Problem A2:

With the model formulated in A1, implemented in Python, the solver gives the following answer:

One station at **(4, 7)** connected to neighborhoods (1, 4), (3, 3), (3, 5), (3, 7), (3, 8), (4, 8), (4, 9), (5, 3), (5, 5), (6, 6), (6, 8), (7, 9), (8, 10).

Another station at **(9, 5)** connected to neighborhoods (6, 2), (8, 5), (9, 2), (9, 4), (9, 5), (9, 7), (10, 6).

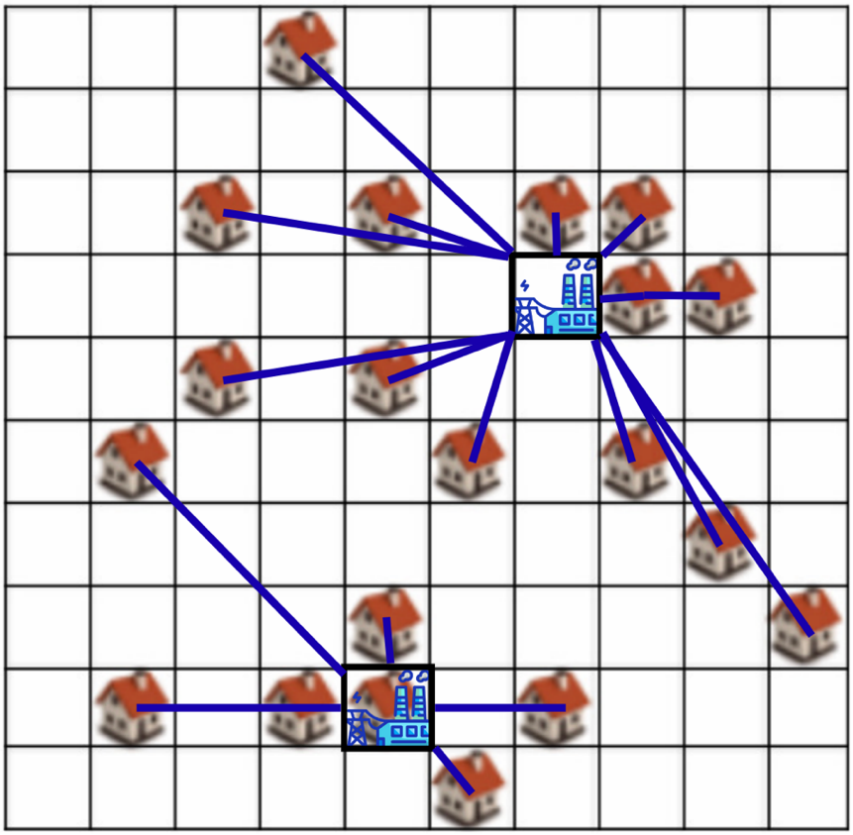
This solution will cost **$ 8810.97**, according to the objective function Z.

Because the solver only gives this solution, it means there exists no cheaper solution.

Upon examination, we see that the solution limits the amount of electrical station, as these are relatively expensive. Stations are then connected to the nearest neighborhoods.

It's also worth noting that the solver avoids stations in the expensive zone in the middle of the grid. This behavior is consistent with our model.

Overall, the solution seems correct, and the answer makes sense.



Problem A3

To modify the model so that the Northernmost Council can only locate one station, we add the following constraint. Note that the rest of the model is identical to that in task A1.

$$\sum_{r=1}\sum_{c=1} s_{rc} = 1$$

⇒ ***Comment***: *This constraint ensures that the binary value of s is true for only one coordinate, thereby restricting the number of stations to 1.*

Implementing the new model, we get the best coordinates for the station at **(7, 6)**, where this station is obviously connected to every neighborhood.

The solution will cost **$8840.19**, according to the objective function Z.

Problem A4

To allow for neighborhoods to not be connected to the grid, we must first make a relaxation of constraints. The following constraint is modified from this:

$$\sum_{r=1}^{10}\sum_{c=1}^{10} w_{j,r,c} = 1, \qquad \forall j$$

To this:

$$\sum_{r=1}^{10}\sum_{c=1}^{10} w_{j,r,c} \leq 1, \qquad \forall j$$

⇒ ***Comment****: This constraint now allows a neighborhood to not be connected to any stations*

Still, we need at least 14, 16 or 18 of the neighborhoods to have a connection, so we introduce the following constraint:

$$\sum_{j=1}^{20}\sum_{r=1}^{10}\sum_{c=1}^{10} w_{j,r,c} \geq K, \qquad K = (14, 16, 18)$$

⇒ ***Comment****: We have now ensured that at least K neighborhoods are connected to the grid.*

Solving the model in Python for 14 neighborhoods the answer is as follows:

- One station at coordinates **(6, 7)**, with optimal cost at **$ 5846.71**

For 16 neighborhoods:

- One station at coordinates **(6, 7),** with optimal cost at **$ 6683.28**

For 18 neighborhoods:

- One station at coordinates **(6, 7)**, with optimal cost at **$ 7683.28**

Notice that the costs are reduced with a smaller number of neighborhoods served, but not proportionally to the decrease. This is because of the relatively high fixed cost of installing a station. The cost will approach 2000 as the number of neighborhoods approaches 1.

Problem B1

The names of the cities are abbreviated as follows.

Northern = N

Northernmost = NM

Eastern = E

Easternmost = EM

Southern = S

Southernmost = SM

Western = W

Westernmost = WM

The network G has a set of nodes:

$V(G) = \{N, NM, E, EM, S, SM, W, WM\}$

With a set of edges:

$E(G) = \{NMN, NMW, NE, NEM, NS, NW, EEM, ES, ESM, SSM, SW, SWM, SMW, WWM\}$,

The edges are undirected.

For each edge $e \in E(G), d(e)$ is the distance of the edge provided in table 1.

For each edge $e \in E(G), r(e)$ is the expected 5-year revenue for the edge provided in table 2.

The 5-year profit of each edge e is then calculated using d, r and 0.5 ($500 000).

$p(e) = r(e) - 0.5 \cdot d(e)$

$\Rightarrow$ ***Comment****: This number may be negative, indicating a loss.*

Problem formulation:

The input of the problem is (G, p) where (G, p) is the network described above, and $p: E(G)$ is calculated using the profit function described above.

The task is to find a maximum spanning tree T consisting of edges $e \in E(G)$. T maximizes profit, ergo maximizes $\sum_{e \in T} p(e)$.

The tree will have one less edge than the network has nodes, given by |V|. In this case |V| = 8, so |T| will need to be 7.

This gives the following mathematical statement:

$$max \sum_{e \in T} p(e), \;\; st. \; |T| = |V| - 1 \;\; and \; (V, T) \; connected$$

This is a *maximum spanning tree* problem, and we use Prim's algorithm to solve it for the spanning tree with the maximum weight (p).

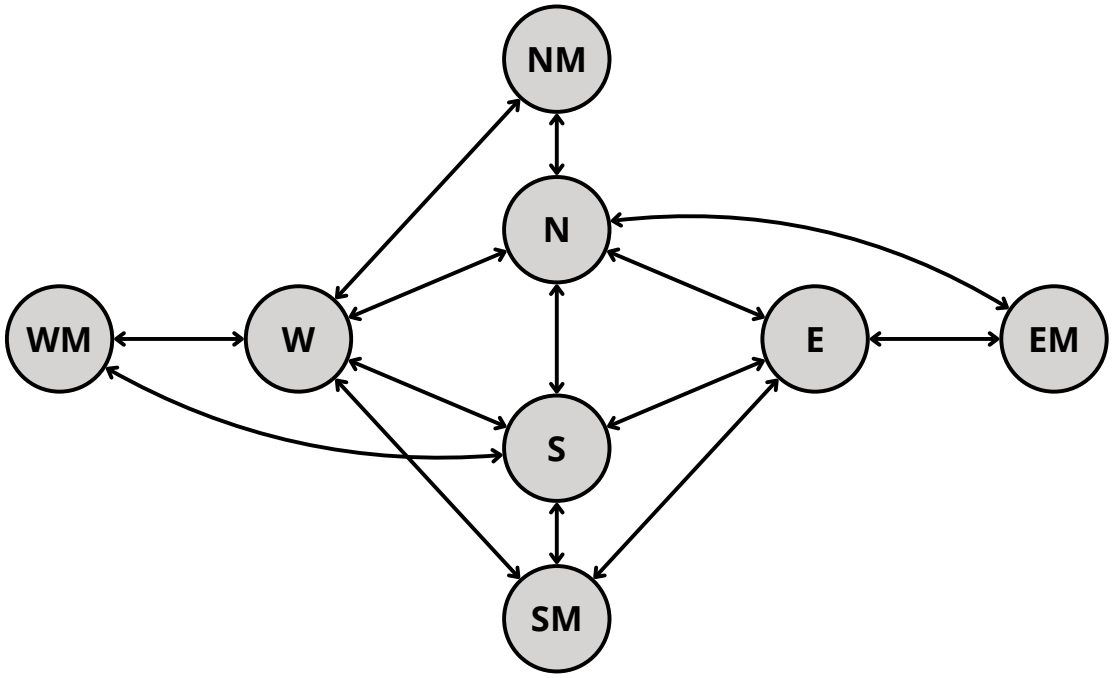


Figure 1: The network G without edge weights

Problem B2

When solving the MaxST in Python we get the following solution:

$T = \{\text{NEM}, \text{SW}, \text{WWM}, \text{ES}, \text{SME}, \text{SN}, \text{NNM}\}$

So, the roads built should be NEM, SW, WWM, ES, SME, SN and NNM.

The network will look like the illustration in Figure 2

The net profit after 5 years is **$ 66 million**.

If the roads generate toll revenue for 10 years, the net profit will be **$ 347 million.**

For 20 years the profit is **$ 948.5 million.**

We can also confirm that |T| = |V| - 1, as there are 8 nodes in the network and 7 edges in the solution.

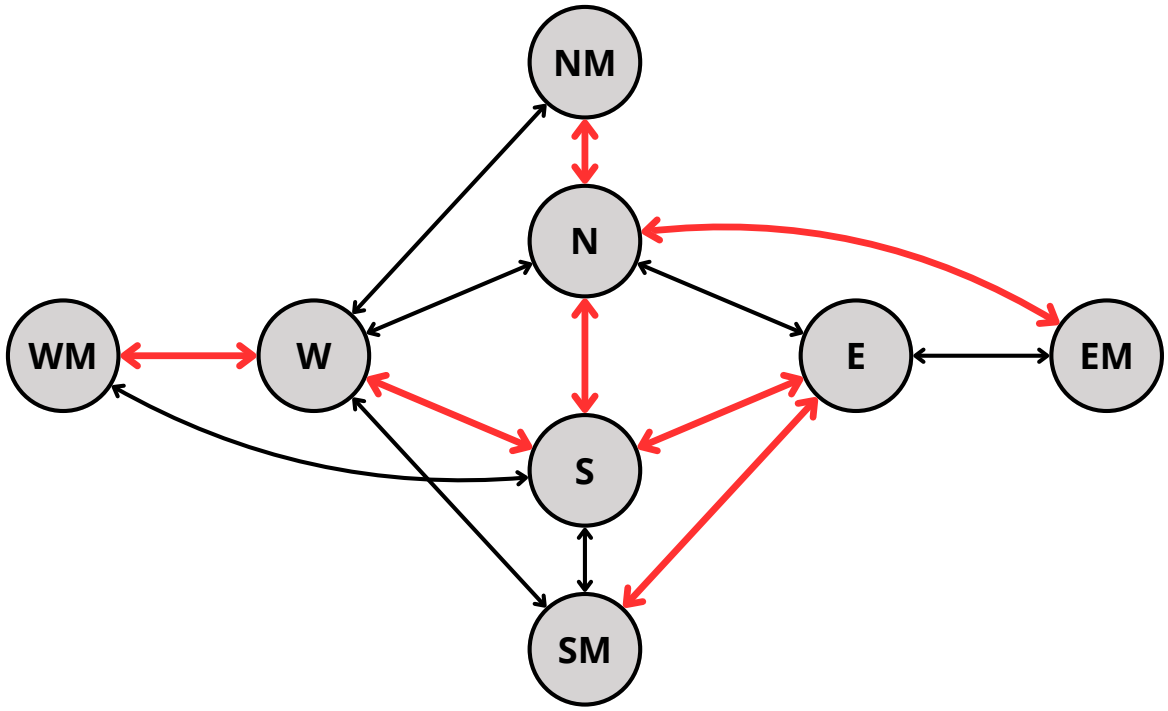


Figure 2: The optimal road network, with roads built represented by red edges

Problem B3

If the edge NS is forbidden, the rest of the network will look like the illustration in Figure 3.

Upon inspection we find there are two "islands" in the network that are no longer connected to each other.

Island 1 consists of nodes WM, W, S, E and SM.

Island 2 consists of nodes EM, N and NM.

It is easy to think that the restriction of only making one change, may lead to a sub-optimal road network in the end, but that is not the case.

The algorithm will simply include the most profitable edge between the islands. In this case the options are -51.5, -2, -9 and -3.

The edge NW with p = -2 will be chosen. This solution holds due to the "cut property" of a MaxST.

We would get the same result by running Prim's algorithm "from the start" without edge NS.

In this case, the obvious answer of choosing NW is confirmed by our Python code.

The result is shown in Figure 5.

Note that the profit changes significantly, from 66 million to 9 million. This means NS provided an important source of profit in the original solution.

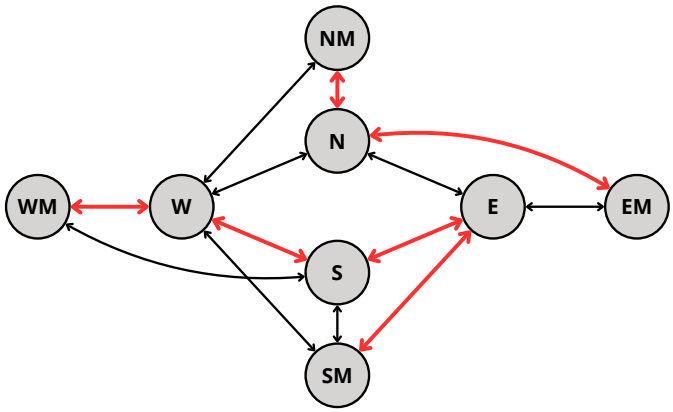


Figure 3: The road network after the edge NS is removed

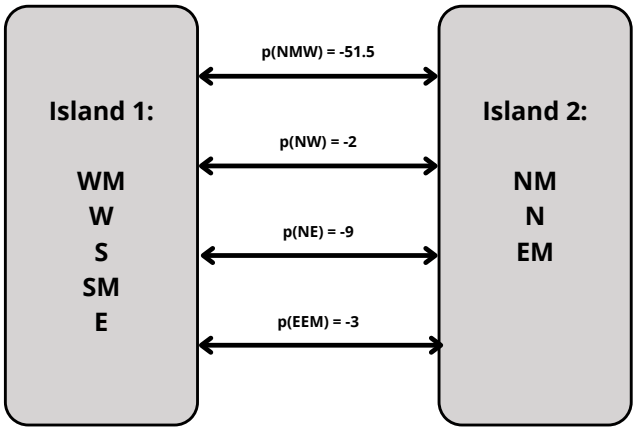


Figure 4: The two parts of the network with edges and edge weights to choose from

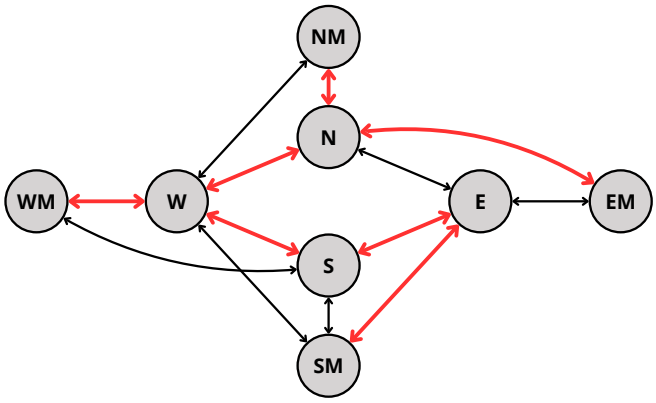


Figure 5: The new optimal network given the constraints in the task