# Graph optimization
# Lab Project

Ewan Decima

May 2025

## Contents

# Question 1

### Model

Let's write down the model. First we have the following sets and parameters:

- $N$ set of nodes in the network.
- $E$ set of edges in the network.
- $c_i$ cost of installing a device at node $i \in N$.
- $t_{i,j}$ propagation delay on edge $(i,j) \in E$.
- $d_{i,j}$ shortest path distance from node $i$ to $j$.
- $M = (M_{ij})_{i,j \in N}$ a reachability matrix, such that $M_{ij}$ is equal to one if $y_{i,j} d_{i,j} \leq T$, zero otherwise.

and those decision variables:

- $x_i \in \{0,1\}$, a binary variable equal to one if a device is installed at node $i \in N$, zero otherwise.
- $y_{i,j} \in \{0,1\}$, a binary variable equal to one if $i \in N$ is served by a device at node $j \in N$.

The objective function is then

$$\min \sum_{i \in N} c_i x_i \tag{1}$$

and is subject to the following constraints:

- Coverage constraint: Each node must be served by exactly one device:

$$\sum_{j \in N | M_{ij} = 1} y_{i,j} = 1 \qquad \forall i \in N$$

- Service constraint: A node can only be served by an installed device:

$$y_{i,j} \leq x_j \qquad \forall i,j \in N$$

**Result**

We obtain the following

| **Instance** | **Optimal value** | **Selected nodes** | **Solver time (s)** |
|---|---|---|---|
| 1 | 2 | 2; 5 | 0.1 |
| 2 | 25 | 13; 17; 19 | 0.092 |
| 3 | 129 | 1; 7; 13; 15; 21; 22; 28; 29; 34 | 0.171 |
| 4 | 256 | 11; 12; 14; 16; 17; 18; 26; 27; 36; 43; 46; 51; 53; 54; 57; 60; 66; 71; 75; 76 | 0.942 |

Table 1: Results for each instance

# Question 2

The heuristic we designed takes an adaptive greedy approach to find a feasible solution to the problem. As in the previous question we premcompute the reachability matrix $M_{ij}$ using the Floyd-Warshall algorithm. The algorithm starts with an empty solution and iteratively assigns a device to a node until a feasible solution is found where all nodes are served by a device. It keeps track of the nodes that have been searched $N^{Unsearched}$ and the nodes that are not yet served by a device in the solution $N^{Unserved}$. At each iteration, the algorithm selects the node $i$ that minimizes the ratio where $c_i$ is the cost of installing a device at node $i$, and the denominator is the number of new nodes that would be served by installing a device at $i$. In other words, we select the node where the cost per newly served node is minimized. Formally the node to be considered is determined:

$$\text{select i} \in N^{Unsearched} | min(\frac{c_i}{1 + \sum k \in N^{Unserved} M_{ij}})$$

These are the steps taken by the algorithm at each iteration:

- If solution is feasible end iteration.
- Consider the cheapest node using our defined heuristic.
- If installing a device on that node can serve 1 or more unserved nodes then add it to the solution.

A feasible solution will always be found as in worst case the algorithm will install a device on all nodes and this case is always a feasible solution.

**Results**

| Instance | Solution Cost | Selected nodes | Solver time (s) |
|---|---|---|---|
| 1 | 2 | 2; 5 | 0.002 |
| 2 | 36 | 1; 5; 8; 10; 11 | 0.013 |
| 3 | 147 | 7; 8; 10; 14; 15; 16; 25; 26; 29; 34; 40 | 0.083 |
| 4 | 287 | 1; 3; 10; 11; 12; 18; 23; 24; 25; 27; 29; 33; 53; 56; 57; 59; 62; 64; 71; 79; 80 | 0.599 |

Table 2: Results for each instance

# Question 3

This problem can be represented as a *minimum-weighted dominating set problem (MDSP)*.

To convert our problem into a *MDSP* we precompute the shortest distance between each pair of nodes in the graph, which can be easily done using the Floyd-Warshall algorithm in $\mathcal{O}(n^3)$ time, where $n$ is the number of nodes in the graph. Then we construct the reachability matrix by checking the distance between each pair of nodes.

$$M \in \mathcal{M}_{n\times n}(\{0,1\})$$
$$M_{ij} = \begin{cases} 1 & \text{if } d_{ij} \leq T \\ 0 & \text{otherwise} \end{cases}$$

We can solve the *MDSP* using the reachability matrix as the graph's adjacency matrix to obtain the solution to the original problem.

# Question 4

**Parameters**

- $G = (N, A)$: the directed graph with nodes $N$ and arcs $A$.
- $d_k$: data to send from node $k \in N$.
- $c_i$: cost of installing devices at node $i \in N$.
- $cap_i$: capacity of devices installed at node $i \in N$.
- $T$: maximum allowed propagation time for data.
- $t_{ij}$: propagation delay on arc $(i, j) \in A$.
- $g_{ij}$: cost of installing channels on arc $(i, j) \in A$.
- $u$: capacity of each channel installed on arcs $(i, j) \in A$.

**Decision Variables**

- $x_{ki}$ binary: one if data is sent from node $k$ to node $i$, zero otherwise.
- $y_i$ integer: number of devices installed at node $i$.
- $z_{ij}$ integer: number of channels installed on arc $(i, j)$.
- $w_{ij}^k$ binary: one if data from node $k$ is sent on arc $(i, j)$, zero otherwise.

**Constraints**

1. Each node must send its data to exactly one other node.
2. For each node, enough devices must be installed to handle the data sent to it.
3. For each arc, enough channels must be installed to handle the data sent on that arc.
4. The propagation time must not exceed the maximum allowed time.
5. Flow constraints: ensure that one and only one path from node $k$ to node $i$ exists if $x_{ki} = 1$; ensure that no data is sent on the network if a node sends data to itself ($x_{kk} = 1$).

**Model**

$$\min \quad \sum_{i \in N} c_i y_i + \sum_{(i,j) \in A} g_{ij} z_{ij}$$

s.t.

1. $\sum_{i \in N} x_{ki} = 1 \quad \forall k \in N$
2. $\sum_{k \in N} d_k \, x_{ki} \leq cap_i \, y_i \quad \forall i \in N$
3. $\sum_{k \in N} d_k \, w_{ij}^k \leq u \, z_{ij} \quad \forall (i,j) \in A$
4. $\sum_{(i,j) \in A} t_{ij} \, w_{ij}^k \leq T \quad \forall k \in N$
5. $$\sum_{j \in N | (i,j) \in A} w_{ij}^k - \sum_{j \in N | (j,i) \in A} w_{ji}^k = \begin{cases} (1 - x_{kk}) & \text{if } i = k \\ -x_{ki} & \text{if } i \neq k \end{cases} \quad \forall k \in N, \; i \in N$$

$x_{ki} \in \{0,1\} \quad \forall k \in N, i \in N$
$y_{ij} \in \mathbb{Z}^+ \quad \forall (i,j) \in A$
$z_i \in \mathbb{Z}^+ \quad \forall i \in N$
$w_{ij}^k \in \{0,1\} \quad \forall k \in N, (i,j) \in A$

**Results**

| Instance | Optimal value | Solver CPU time (s) |
|---|---|---|
| 1 | 85 | 0.154 |
| 2 | 328 | 0.099 |
| 3 | 376 | 0.053 |
| 4 | 149 | 7.550 |
| 5 | 154 | 0.702 |
| 6 | 424 | 15.152 |
| 7 | 420 | 9.739 |

### Question 5

*How can the cutset inequalities for network design be applied to the problem described in 4? Generate all the single node cutset-based inequalities (those that consider the cuts separating each single node from the others). Add them to the formulation and compute the continuous relaxation with and without them. Compare the two relaxations solving the instances. Are the inequalities effective? Upload the* `.run` *and* `.mod` *files, and a* `.pdf` *file with the updated model and the comparison of the two continuous relaxations.*

## Question 6

*How can the cover inequalities for the knapsack problem be applied to the problem described in 4? Generate heuristically some cover-based inequalities, add them to the formulation and compute the continuous relaxation with and without them for all the instances. Are they effective? Upload the* `.run` *and* `.mod` *files, and a* `.pdf` *with the results and the updated model.*