Optimizing Urban Transport for Sustainability

Authors

Ion Mierluș Mazilu, Technical University of Civil Engineering Bucharest (Romania).

Type of activity

This problem applies Operational Research methods (graph algorithms) to support sustainable transport planning in urban environments. It can be implemented with algorithmic logic (e.g., Dijkstra's Algorithm) and software-based simulations.

Target educational level

It is suitable for first and second-year university students in civil engineering, environmental engineering, transport engineering, or applied mathematics.

Initial information

This approach is valuable for SDG 11: Sustainable Cities and Communities, as it promotes efficient, inclusive, and low-impact transport networks in urban settings. Students will model a real-world urban network and compute the optimal route using a mathematical algorithm to enhance transport sustainability.

Dijkstra's Algorithm

Transport planning often involves determining a route through a road (transport) network in such a way that transport costs or travel times are minimized. It is necessary to find the “shortest” path between two arbitrary nodes of the road network. This type of problem also occurs in the design of computer networks, in planning routes for public transport systems, etc.

Dijkstra's algorithm allows the computation of the shortest paths from a starting vertex s to all other vertices x in a connected graph $G = (U, X, l)$, provided that all arc lengths are non-negative.

Let $\Pi(x)$ represent the length of the shortest path from s to x.

Let S be the set of vertices for which Π has been computed. Then, $\Pi(x)$ represent the length of the shortest path from s to x having all intermediate nodes in S except x.

Let $\Gamma^+(x) = \{\text{the set of arcs that originate from node } x\}$, and

$\Gamma^-(x) = \{\text{the set of arcs that end in node } x\}$.

If the graph is undirected, then

$$\Gamma^+(x) = \Gamma^-(x) = \Gamma(x) = \{\text{the set of arcs incident to node } x\}$$

Dijkstra Algorithm

Step 1. *Initialization*

Let $S := \{s\}$; s is the start node, $\Pi(s) := 0$;

For each $x \in X - S$:

If $x \in \Gamma(s)$, then $\Pi(x) := l(s, x)$;

Otherwise, $\Pi(x) := +\infty$;

Step 2. *Current Iteration*

Repeat:

Find $y \in X - S$ such that $\Pi(y) = \min_{z \in X-S} \Pi(z)$;

If $\Pi(y) < \infty$, then $S := S \cup \{y\}$;

For each $z \in \Gamma^+(y)$:

$\Pi(z) := \min\{\Pi(z), \Pi(y) + l(z, y)\}$;

Until $S = X$ or $\Pi(y) = \infty$;

Step 3. Stop.

The values $\Pi(z)$ remain unchanged for all $z \epsilon S$, which can be used in implementing the algorithm on a computer.

Source

Problem adapted from a transport optimization example using Dijkstra's algorithm. Aligned with sustainable development goals.

Problem statement

A smart city planning team is designing a low-emission transport corridor between two key locations in the city:

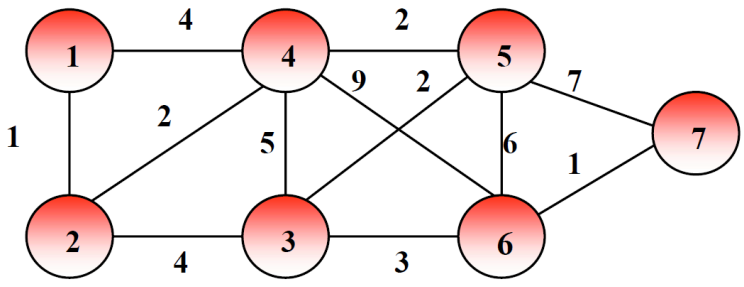
- Location 1: a residential zone
- Location 7: a green industrial area

The city network consists of 7 urban locations connected by travel routes, with known travel times in hours. The aim is to identify the optimal transport route—i.e., the shortest-time path—from location 1 to location 7. The graph representing the network is undirected, and all arc weights (times) are non-negative.

This task supports transport planning initiatives aiming to:

- Improve accessibility and reduce travel times for public and sustainable transport
- Minimize energy usage and carbon emissions
- Create data-informed policies for route design

Use Dijkstra's algorithm to determine the optimal path.



Network Data

The graph includes the following arcs (connections) and travel times (in hours):

Arc	Start	End	Time (h)
1	1	2	1
2	1	4	4
3	2	3	4
4	2	4	2
5	3	4	5
6	3	5	2
7	3	6	3
8	4	5	2
9	4	6	9
10	5	6	6
11	5	7	7
12	6	7	1

Solution

Use **Dijkstra's algorithm** to determine the shortest path from node 1 to node 7. The steps include:

1. Initialize the distance vector $\Pi(y)$ for all nodes, starting from node 1.
2. Iteratively update the distances using the minimum values from neighboring nodes.
3. Finalize the optimal route when all nodes are visited or when the target node is reached.

Step 1.

Initial data: $s = 1, S = \{1\}, \Gamma(1) = \{2, 4\}$

y	1	2	3	4	5	6	7
$\Pi(y)$	0	1	∞	4	∞	∞	∞

Step 2.

Iteration I:

$$\Pi(y) = \min\{\Pi(2), \Pi(3), \Pi(4), \Pi(5), \Pi(6), \Pi(7)\} = \Pi(2) = 1$$

New node added: $y = 2, S = \{1, 2\}, \Gamma(2) = \{1, 3, 4\}$

$$\Pi(3) = \min\{\infty, 1 + 4\} = 5$$
$$\Pi(4) = \min\{4, 1 + 2\} = 3$$

y	1	2	3	4	5	6	7
$\Pi(y)$	0	1	5	3	∞	∞	∞

Iteration II:

$y = 4, S = \{1, 2, 4\}, \Gamma(4) = \{1, 2, 3, 5, 6\}$

$$\Pi(3) = \min\{5, 3 + 5\} = 5$$
$$\Pi(5) = \min\{\infty, 3 + 2\} = 5$$
$$\Pi(6) = \min\{\infty, 3 + 9\} = 12$$

y	1	2	3	4	5	6	7
$\Pi(y)$	0	1	5	3	5	12	∞

Iteration III:

$y = 3, S = \{1, 2, 3, 4\}, \Gamma(3) = \{2, 4, 5, 6\}$

$$\Pi(5) = \min\{5, 5 + 2\} = 5$$
$$\Pi(6) = \min\{12, 5 + 3\} = 8$$

y	1	2	3	4	5	6	7
$\Pi(y)$	0	1	5	3	5	8	∞

Iteration IV:

$y = 5, S = \{1,2,3,4,5\}, \Gamma(5) = \{3,4,6,7\}$

$$\Pi(6) = \min\{8, 5+6\} = 8$$
$$\Pi(7) = \min\{\infty, 5+7\} = 12$$

y	1	2	3	4	5	6	7
$\Pi(y)$	0	1	5	3	5	8	12

Iteration V:

$y = 6, S = \{1,2,3,4,5,6\}, \Gamma(6) = \{3,4,5,7\}$

$$\Pi(7) = \min\{12, 8+1\} = 9$$

y	1	2	3	4	5	6	7
$\Pi(y)$	0	1	5	3	5	8	9

Iteration VI:

$y = 7, S = \{1,2,3,4,5,6,7\}$

No more updates are performed for nodes in $S \cap \Gamma(y)$.

Conclusion

The algorithm stops because $S = X$. The vector $\Pi(y)$ contains the shortest distances from node *1* to all other nodes. The shortest path from node 1 to node 7 is: $D = \{[1,2], [2,3], [3,6], [6,7]\}$ with a total distance of 9.

Discussion

This route minimizes travel time, making it ideal for eco-friendly public transportation and supporting low-carbon mobility strategies in smart cities.