MOOCTA WORKSHOP 2026

Time-Dependent Optimization for Shortest Path Routing Under Peak-Hour Congestion in Antananarivo

Jean Michel Menjanahary
Department of Mathematics and Informatics
University of Antananarivo, Madagascar

MOOCTA Workshop 2026
Pretoria, South Africa

Collaborators and Affiliations

University of Antananarivo



Lecturer at the Math Department

International Institute for Social Sciences



Researcher at IISS Economics Lab

Joliot Rarinony Manandratra (Economist)
Mamy Arilala Andrianambinina (Economist)
Patriarck Randriamanarina (Economist)
Anthony Ramanantsalama (Computer Scientist)

Outline

- **Motivation and Context**
- **Research Problem and TDSPP Model**
- **Solution Methods**
- **Experimental Results**
- **Conclusion**

Urban Mobility in Antananarivo

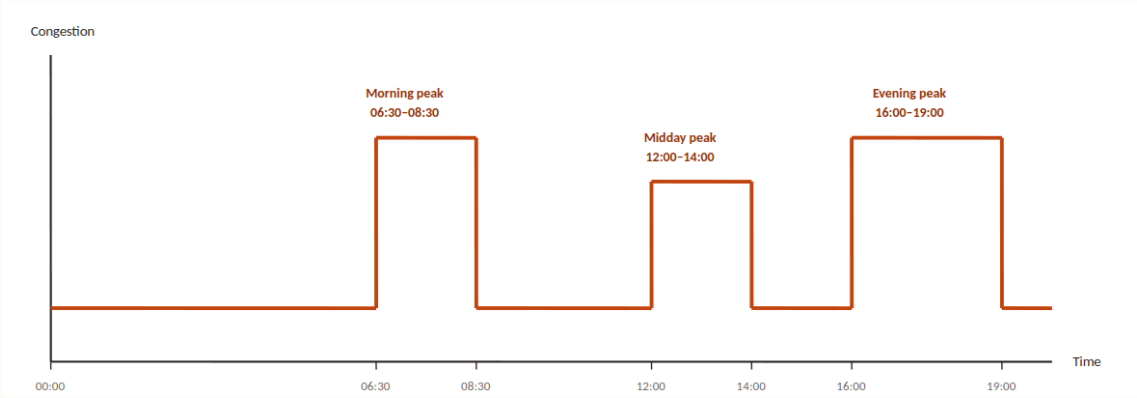
Severe and Unpredictable Traffic Congestion

- Heavy commuter departures during peak hours
- Mixed traffic on narrow, crowded roads
- Short distances but long, unpredictable travel times
- Intersections are manually managed by traffic police instead of traffic lights

The physical shortest path is frequently *not* the fastest path during peak hours.



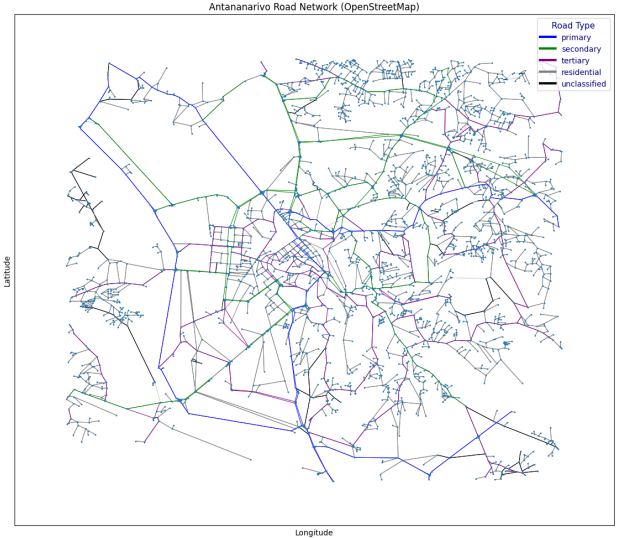
Congestion Profile



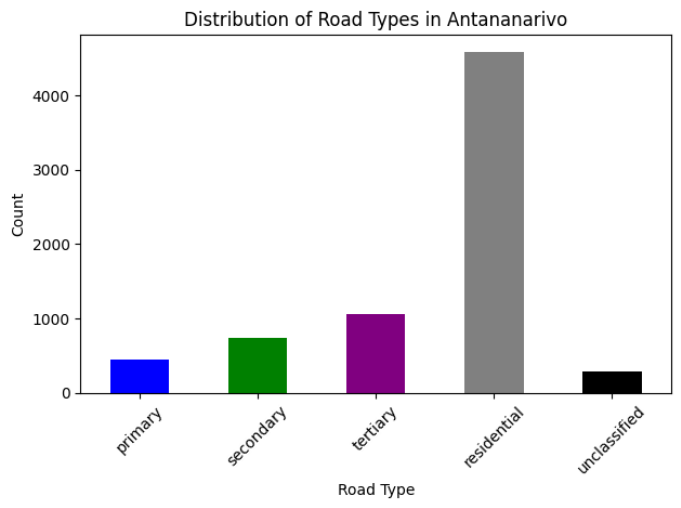
Antananarivo's Road Network

- Nodes: 3377 (intersections + dead-ends)
- Edges: 7172

The network can be expanded by adding other points of interest (tourist attractions, pharmacies, etc.).



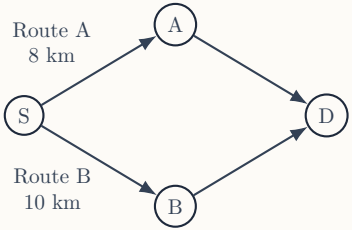
Road Types Distribution



Failure of Static Shortest Path

Consider Route A (8 km) vs. Route B (10 km):

- **Normal/Off-Peak Traffic**:
  - $T_A = 20$ minutes ← **Optimal Choice**
  - $T_B = 25$ minutes
- **Morning Peak Hour**:
  - $T_A = 65$ minutes (Severe bottlenecks)
  - $T_B = 32$ minutes ← **Optimal Choice**



Traffic changes over time, so the best route depends on when you leave.

A TDSPP Model Formulation

- **Road Network Topology**: Directed graph $G = (V, E)$, where V represents intersections and E represents directed road segments.
- **Dynamic Cost Function**: Each edge $e \in E$ is assigned a time-dependent cost function:

$$C_e(t) = C_e^0 \cdot [1 + \alpha_e \cdot p(t)]$$

where:

  - C_e^0 represents the free-flow travel time;
  - α_e represents the congestion sensitivity of road e;
  - $p(t)$ is the temporal congestion profile reflecting three daily peaks:

Morning: 06:30 – 08:30 **Midday**: 12:00 – 14:00 **Evening**: 16:00 – 19:00

Path Cost and the FIFO Property

For a path $P = (e_1, e_2, \ldots, e_k)$ and departure time t_0, arrival times are computed recursively:

$$t_i = t_{i-1} + C_{e_i}(t_{i-1})$$

The optimization goal is to solve

$$\min_{P:s\to d} T(P, t_0)$$

over all feasible paths.

The FIFO Property:

$$t_1 < t_2 \implies t_1 + C_e(t_1) \leq t_2 + C_e(t_2)$$

Leaving earlier means arriving earlier (no overtaking). This property allows the use of efficient polynomial-time label-setting algorithms.

Why Consider Non-FIFO?

- Limited use of traffic lights at junctions
- Mixed road use by cars, motorcycles, and pedestrians
- Rapid changes from normal traffic to heavy congestion

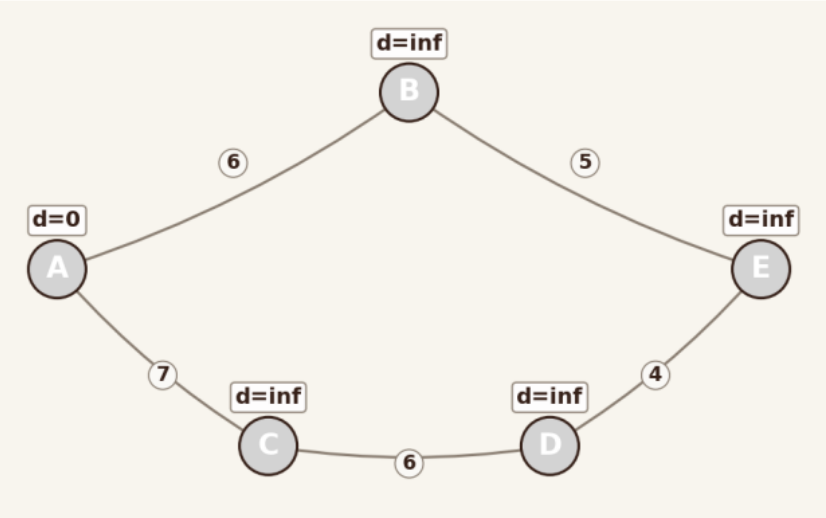
Exact Method on Static Network: Dijkstra

Algorithm 1: Classical Dijkstra

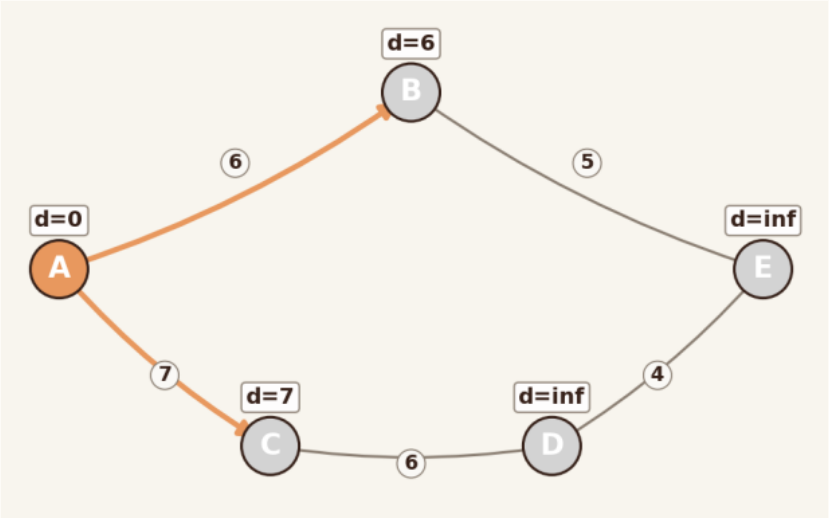
- **Input:** Graph $G = (V, E)$, source $s \in V$, static weights $w(u, v) \geq 0$
- **Output:** Array of shortest-path distances $d[\cdot]$, predecessor array $\pi[\cdot]$

for each vertex $v \in V$ **do**
  $d[v] \leftarrow \infty$ // Tentative distance
  $\pi[v] \leftarrow \text{NIL}$ // Predecessor node
end for
$d[s] \leftarrow 0$
$Q \leftarrow V$ // Priority queue of all unvisited nodes
while $Q \neq \emptyset$ **do**
  $u \leftarrow \text{EXTRACT-MIN}(Q)$ // Node with minimum $d[u]$
  for each outgoing edge $(u, v) \in E$ **do**
    if $d[u] + w(u, v) < d[v]$ **then**
      $d[v] \leftarrow d[u] + w(u, v)$
      $\pi[v] \leftarrow u$
    end if
  end for
end while

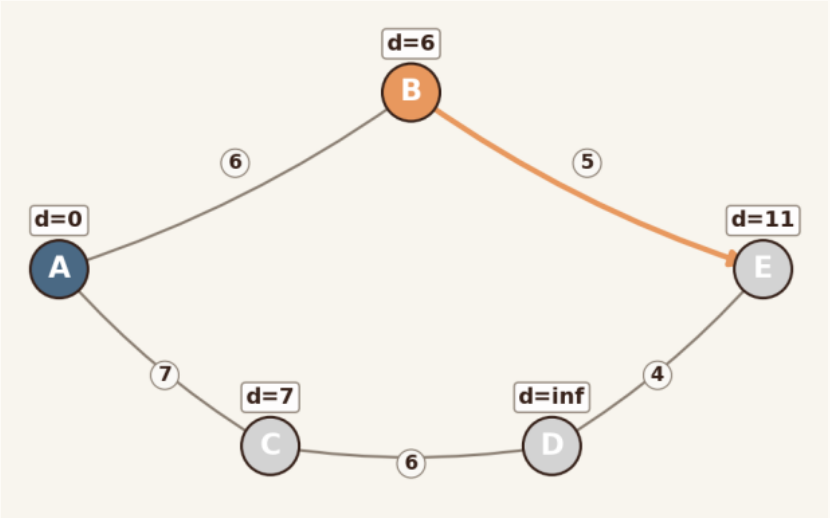
Example



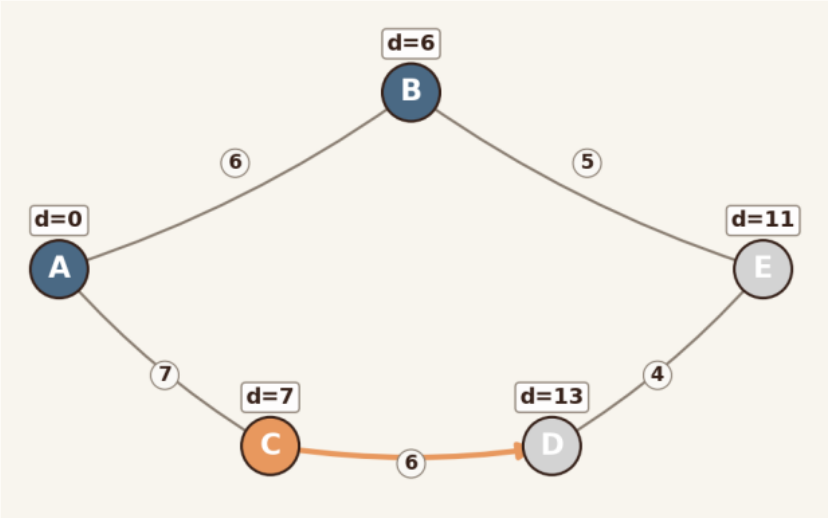
Example



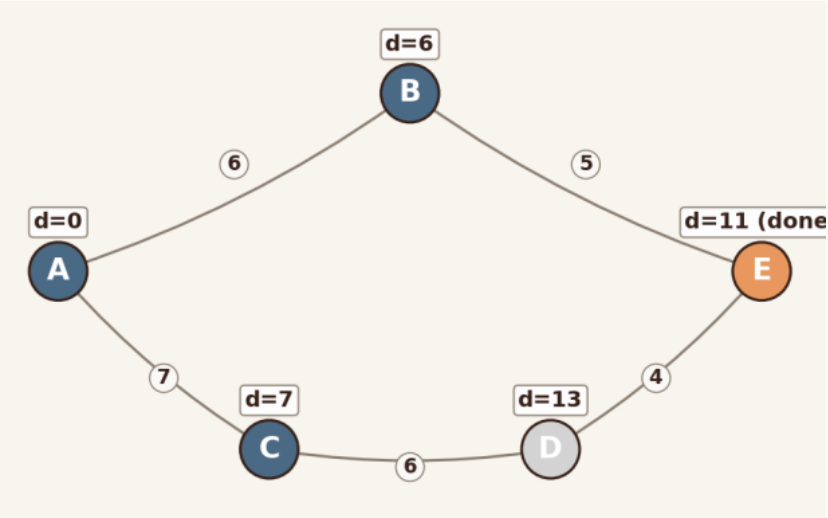
Example



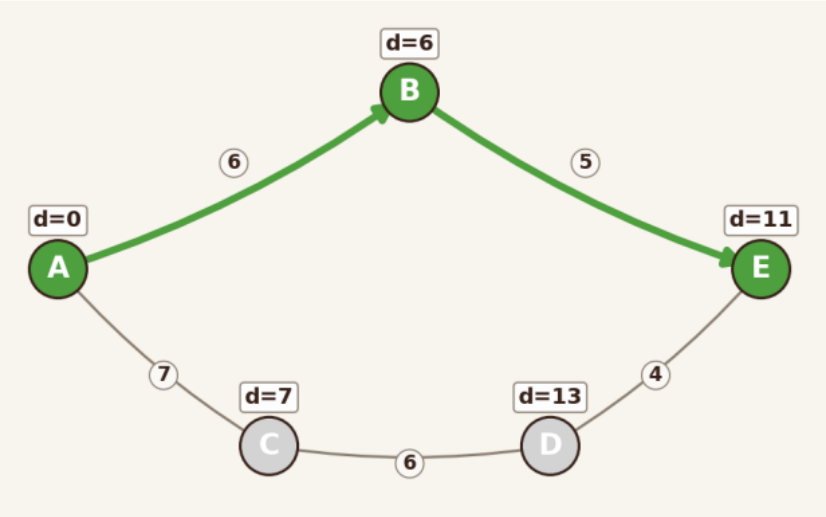
Example



Example



Example



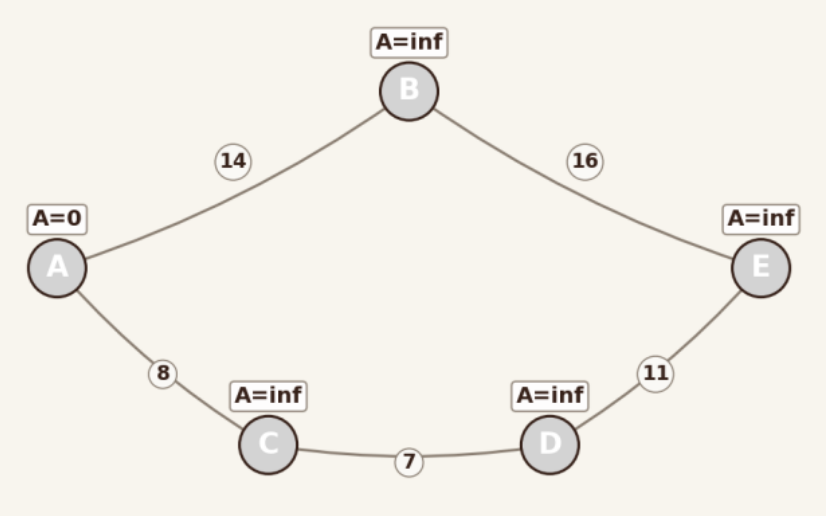
Dynamic FIFO Network: TD-Dijkstra

Algorithm 2: Time-Dependent Dijkstra (FIFO Only)

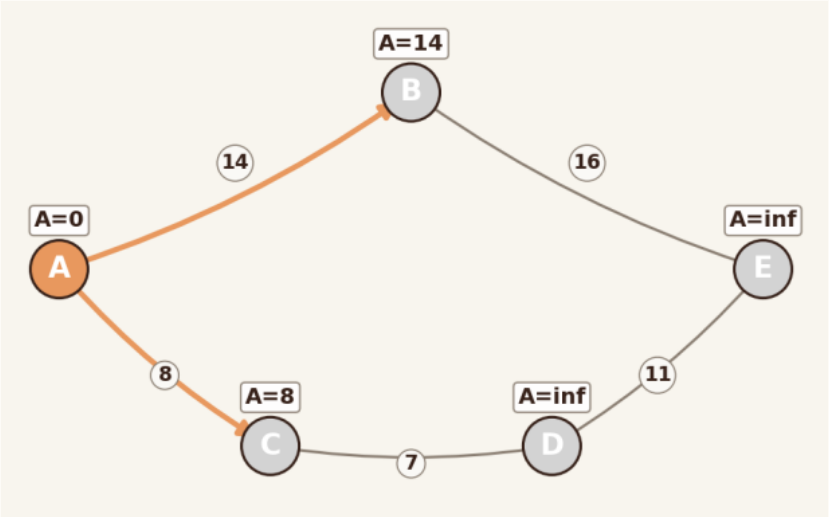
- **Input:** Graph $G = (V, E)$, source s, departure time t_0, time-dependent costs $c_{uv}(t)$
- **Output:** Earliest arrival times $A[\cdot]$, predecessor array $\pi[\cdot]$

for each vertex $v \in V$ **do**
  $A[v] \leftarrow \infty$ // Earliest arrival time at v
  $\pi[v] \leftarrow \text{NIL}$ // Predecessor node
end for
$A[s] \leftarrow t_0$
$Q \leftarrow V$ // Priority queue ordered by earliest arrival time $A[v]$
while $Q \neq \emptyset$ **do**
  $u \leftarrow \text{EXTRACT-MIN}(Q)$ // Node with minimum $A[u]$
  for each outgoing edge $(u, v) \in E$ **do**
    $t_{cand} \leftarrow A[u] + c_{uv}(A[u])$ // Evaluate travel time at arrival time
    if $t_{cand} < A[v]$ **then**
      $A[v] \leftarrow t_{cand}$
      $\pi[v] \leftarrow u$
    end if
  end for
end while

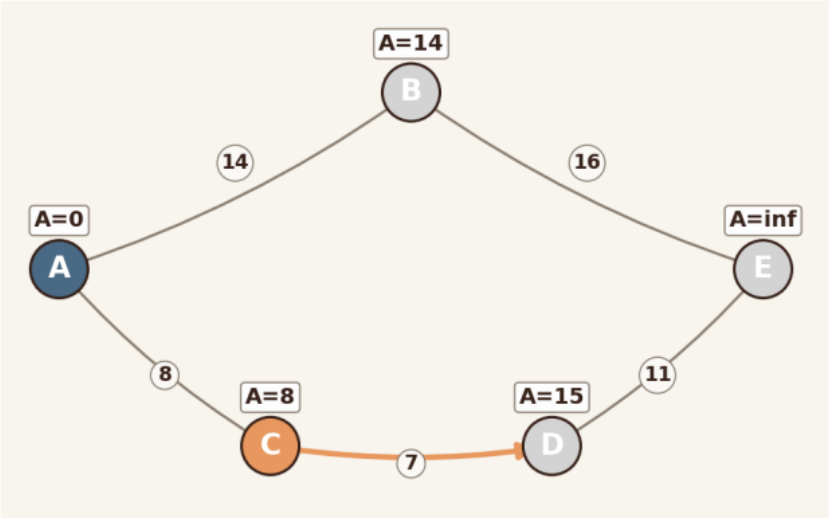
Example



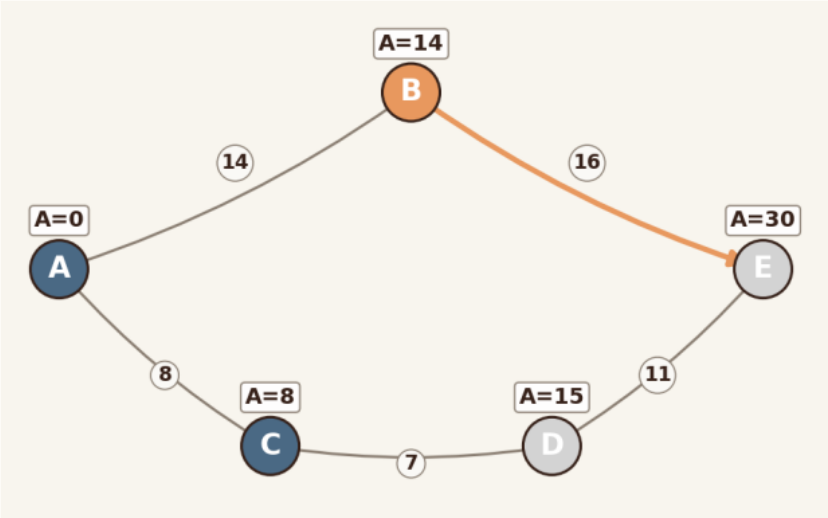
Example



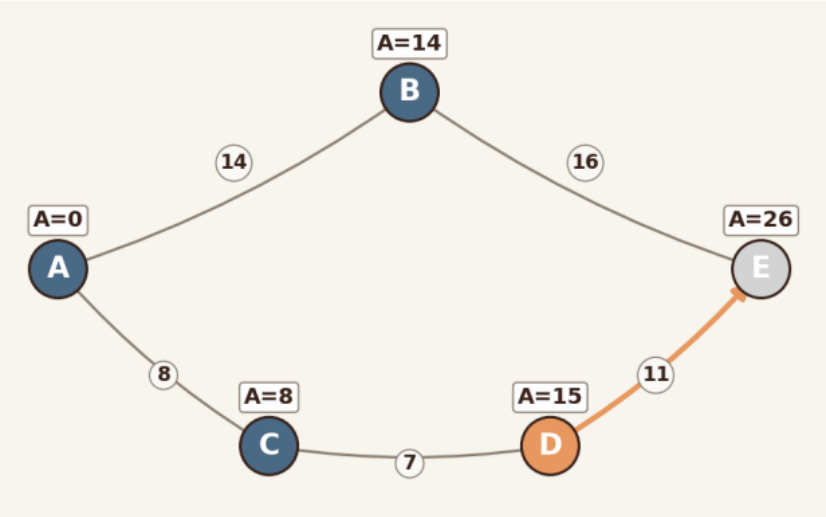
Example



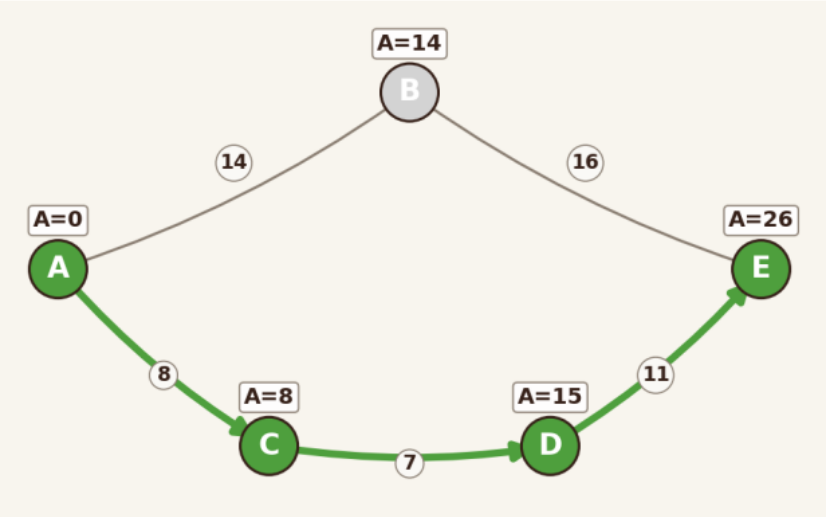
Example



Example



Example



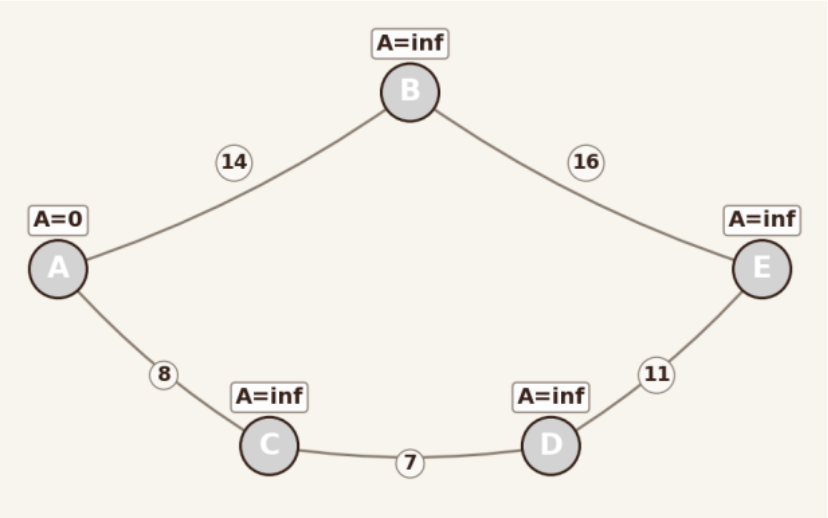
General Dynamic Network: Label-Correcting TDSPP

Algorithm 3: Label-Correcting TDSPP (FIFO/Non-FIFO)

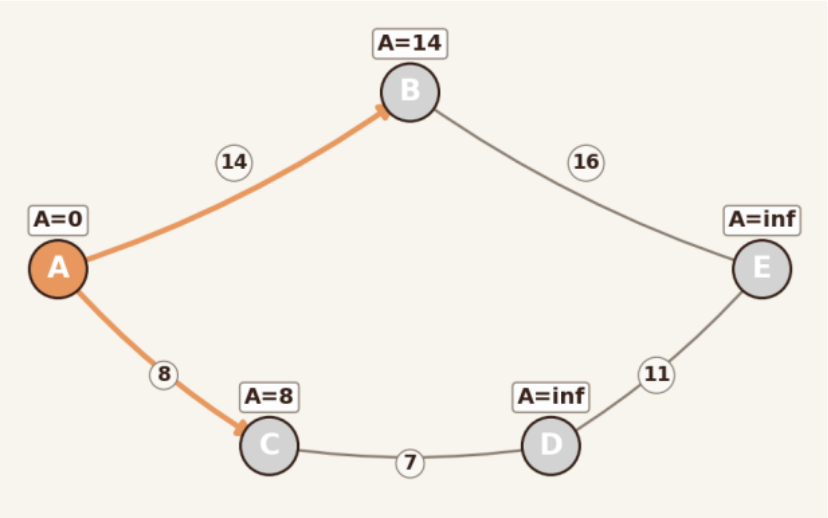
- **Input:** Graph $G = (V, E)$, source s, departure time t_0, time-dependent costs $c_{uv}(t)$
- **Output:** Earliest arrival times $A[\cdot]$, predecessor array $\pi[\cdot]$

for each vertex $v \in V$ **do**
  $A[v] \leftarrow \infty$ // Earliest arrival time at v
  $\pi[v] \leftarrow \text{NIL}$ // Predecessor node
end for
$A[s] \leftarrow t_0$
$Q \leftarrow \emptyset$ // Active node queue (FIFO list or Double-ended queue)
$\text{PUSH}(Q, s)$
while $Q \neq \emptyset$ **do**
  $u \leftarrow \text{POP}(Q)$
  for each outgoing edge $(u, v) \in E$ **do**
    $t_{cand} \leftarrow A[u] + c_{uv}(A[u])$ // Evaluate travel time
    if $t_{cand} < A[v]$ **then**
      $A[v] \leftarrow t_{cand}$
      $\pi[v] \leftarrow u$
      if $v \notin Q$ **then**
        $\text{PUSH}(Q, v)$ // Push back to correct downstream paths
      end if
    end if
  end for
end while

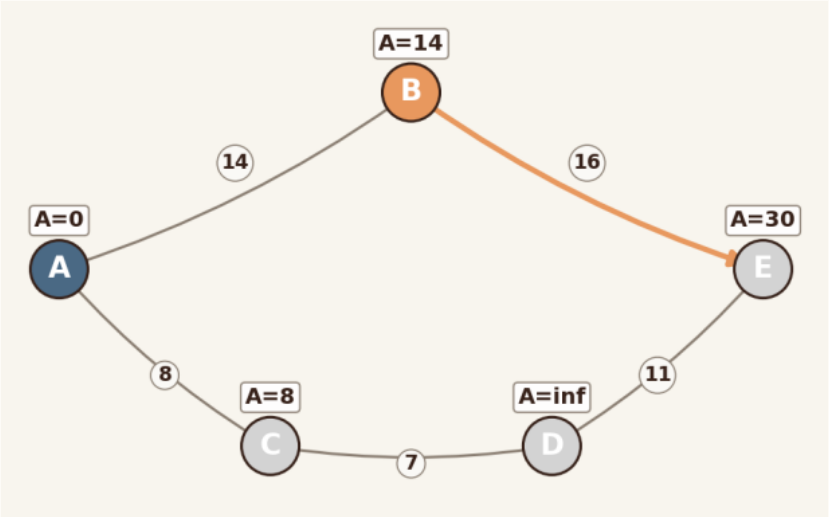
Example



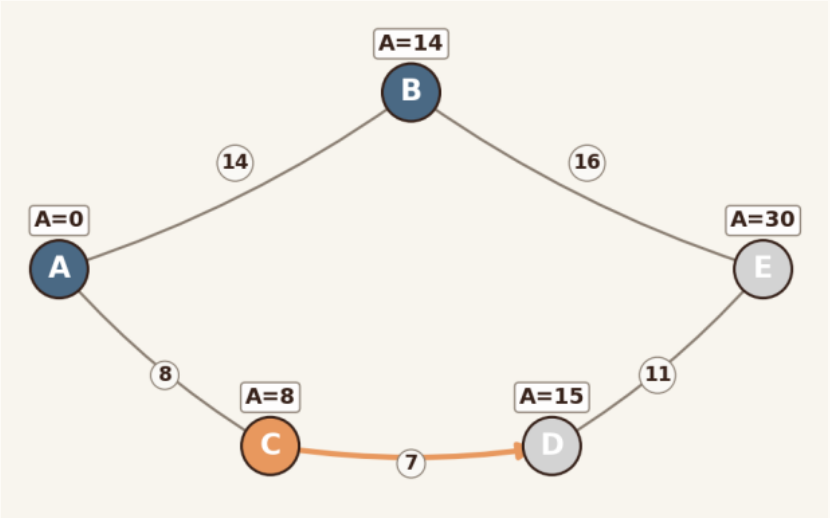
Example



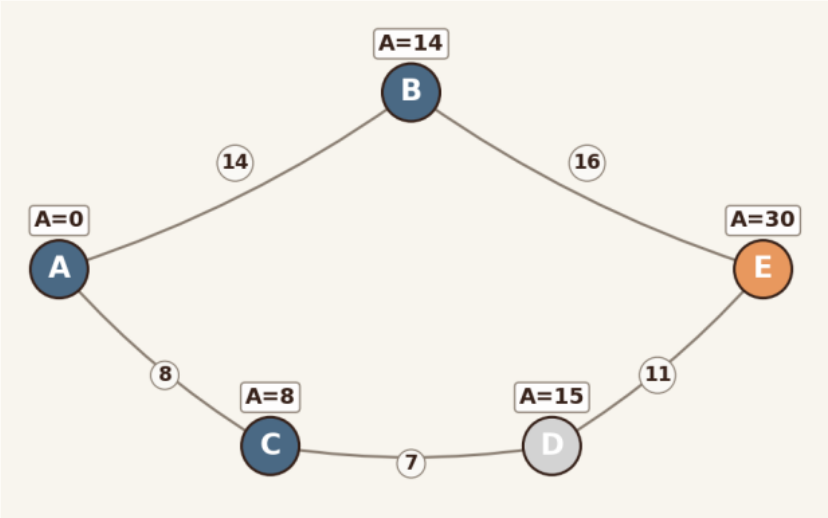
Example



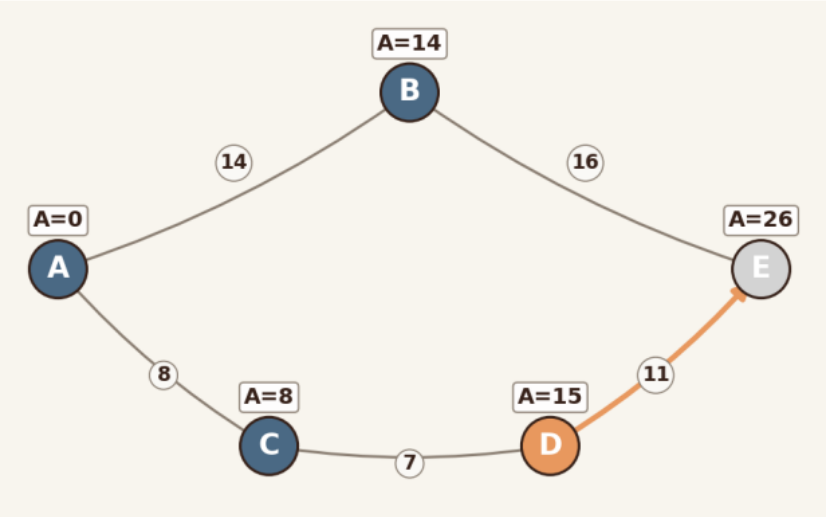
Example



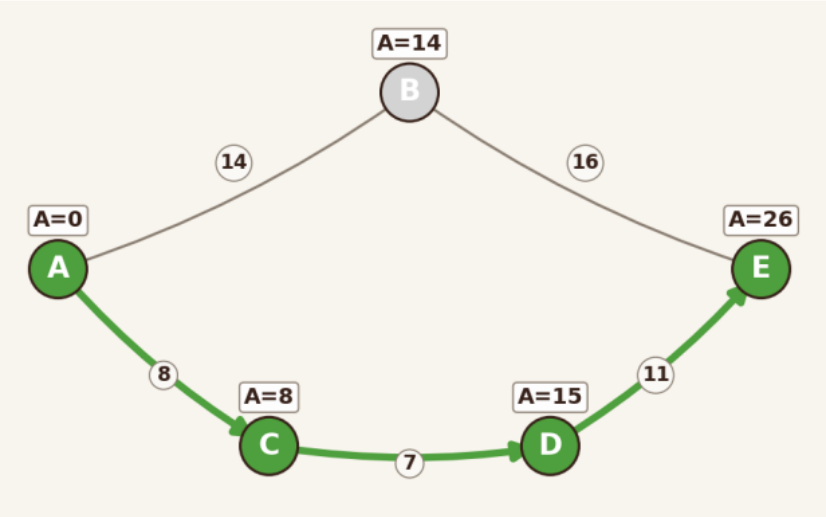
Example



Example



Example



Metaheuristic: Simulated Annealing (SA)

Algorithm 4: Simulated Annealing for TDSPP

- **Input:** Graph $G = (V, E)$, source s, target t_{target}, departure time t_0, time-dependent costs $c_{uv}(t)$, initial temperature T_0, cooling rate $\alpha \in (0, 1)$, stopping temperature T_{min}, max iterations per temp K
- **Output:** Best valid path $P^* = (s = v_0, v_1, \ldots, v_k = t_{target})$, minimal arrival time A^*

$P \leftarrow \text{GENERATE-INITIAL-PATH}(G, s, t_{target})$
$A \leftarrow \text{EVALUATE-PATH-ARRIVAL-TIME}(P, t_0, c)$
$P^* \leftarrow P, A^* \leftarrow A$
$T \leftarrow T_0$
while $T > T_{min}$ **do**
  for $k \leftarrow 1$ **to** K **do**
    $P_{cand} \leftarrow \text{GENERATE-NEIGHBOR-PATH}(P, G)$
    // Local perturbation (e.g., node insertion/removal)
    $A_{cand} \leftarrow \text{EVALUATE-PATH-ARRIVAL-TIME}(P_{cand}, t_0, c)$
    $\Delta \leftarrow A_{cand} - A$
    if $\Delta < 0$ **or** $\text{RANDOM}(0, 1) < \exp\left(-\frac{\Delta}{T}\right)$ **then**
      $P \leftarrow P_{cand}$
      $A \leftarrow A_{cand}$
      if $A < A^*$ **then**
        $P^* \leftarrow P$
        $A^* \leftarrow A$
      end if
    end if
  end for
  $T \leftarrow \alpha \cdot T$ // Temperature reduction
end while

Synthetic Experiments

$\lvert V\rvert$	TD-Dijkstra (ms)	Label-Correcting (ms)	SA (ms)	SA Gap	Static Penalty
20	0.36	0.71	13.93	14.58%	+16.41%
50	0.97	1.77	29.67	9.75%	+15.26%
100	2.09	3.64	40.55	2.70%	+3.56%
500	8.85	19.82	75.34	3.33%	+3.36%
1000	10.26	28.19	63.58	1.91%	+2.17%
3377	42.50	107.15	112.16	2.05%	+2.05%

- TD-Dijkstra solves networks with 3377 nodes in only 42.50 ms.
- Static routing can increase travel time by up to **16.41%**.
- Simulated Annealing finds solutions only **2.05%** worse than the best solution on 3377 nodes.

Data Collection

- **Road Network**: Extracted from OpenStreetMap
- **Travel Times**: Will be collected from multi-stop bus surveys
- **Traffic Context**: Time of day, seasons, and rainfall

Conclusion

- Traffic changes over time, so static routing is not enough
- Non-FIFO algorithms can handle changing traffic more reliably
- Heuristics can provide fast solutions for larger networks

“When traffic changes with time, routing must change with time.”