Public Transportation in Rural Areas: The Clustered Dial-a-Ride Problem

Fabian Feitsch
November 16th, 2018

Attributions of third party images can be found on slide 12.
Public Transportation
Public Transportation
Public Transportation
Public Transportation

- House
- Person walking
- Bus stop
- Bus
Public Transportation

So far, so good?
Public Transportation

So far, so good?
Public Transportation

So far, so good?

walking distance?

weather?
Public Transportation

So far, so good?

- Walking distance?
- Weather?
- Waiting time?
Public Transportation

So far, so good? → Probably in the city, but not in villages!
Public Transportation

So far, so good?  → Probably in the city, but not in villages!

→ Doorstep Service in Rural Areas
The Dial-a-Ride Problem
The Dial-a-Ride Problem

A *Dial-a-Ride instance* is a triple $I = (n, [d_{i,j}], S)$. 
The Dial-a-Ride Problem

A *Dial-a-Ride instance* is a triple $I = (n, [d_{i,j}], S)$.

$n := \text{number of riders}$
The Dial-a-Ride Problem

A *Dial-a-Ride instance* is a triple \( I = (n, [d_{i,j}], S) \).

\( n := \text{number of riders} \)
The Dial-a-Ride Problem

A Dial-a-Ride instance is a triple $I = (n, [d_{i,j}], S)$.  

$n := \text{number of riders}$
A Dial-a-Ride instance is a triple $l = (n, [d_{i,j}], S)$.

$n$ := number of riders
Number of persons $m = n + 1$
The Dial-a-Ride Problem

A *Dial-a-Ride instance* is a triple \( I = (n, [d_{i,j}], S) \).

\( n := \text{number of riders} \)

Number of persons \( m = n + 1 \)
The Dial-a-Ride Problem

A *Dial-a-Ride instance* is a triple \( I = (n, [d_{i,j}], S) \).

\( n := \text{number of riders} \)

Number of persons \( m = n + 1 \)

\([d_{i,j}] := \text{distance matrix}\)
A Dial-a-Ride instance is a triple \( l = (n, [d_{i,j}], S) \).

\( n \) := number of riders
Number of persons \( m = n + 1 \)

\([d_{i,j}] := \) distance matrix
start with 0 (driver’s pickup)
The Dial-a-Ride Problem

A Dial-a-Ride instance is a triple $I = (n, [d_{i,j}], S)$.

$n := \text{number of riders}$

Number of persons $m = n + 1$

$[d_{i,j}] := \text{distance matrix}$

start with 0 (driver’s pickup)

enumerate pickups
The Dial-a-Ride Problem

A Dial-a-Ride instance is a triple \( l = (n, [d_{i,j}], S) \).

\( n := \) number of riders
Number of persons \( m = n + 1 \)

\( [d_{i,j}] := \) distance matrix
start with 0 (driver’s pickup)
enumerate pickups
enumerate dest’s in same order
A *Dial-a-Ride instance* is a triple $I = (n, [d_{i,j}], S)$.

$n := \text{number of riders}$

Number of persons $m = n + 1$

$[d_{i,j}] := \text{distance matrix}$

start with 0 (driver’s pickup)

enumerate pickups

enumerate dest’s in same order
The Dial-a-Ride Problem

A *Dial-a-Ride instance* is a triple $I = (n, [d_{i,j}], S)$.

$n := \text{number of riders}$

Number of persons $m = n + 1$

$[d_{i,j}] := \text{distance matrix}$

\begin{itemize}
  \item start with 0 (driver's pickup)
  \item enumerate pickups
  \item enumerate dest's in same order
\end{itemize}
The Dial-a-Ride Problem

A *Dial-a-Ride instance* is a triple $I = (n, [d_{i,j}], S)$.

$n :=$ number of riders

Number of persons $m = n + 1$

$[d_{i,j}] :=$ distance matrix

- start with 0 (driver’s pickup)
- enumerate pickups
- enumerate dest‘s in same order

$S :=$ number of seats in the vehicle
The Dial-a-Ride Problem

A *Dial-a-Ride instance* is a triple $I = (n, [d_{i,j}], S)$.

$n := \text{number of riders}$

Number of persons $m = n + 1$

$[d_{i,j}] := \text{distance matrix}$

- start with 0 (driver’s pickup)
- enumerate pickups
- enumerate dest’s in same order

$S := \text{number of seats in the vehicle}$

for this presentation: $S = \infty$
The Dial-a-Ride Problem

A *Dial-a-Ride instance* is a triple \( I = (n, [d_{i,j}], S) \).

- \( n \) := number of riders
- Number of persons \( m = n + 1 \)
- \( [d_{i,j}] \) := distance matrix
  - start with 0 (driver’s pickup)
  - enumerate pickups
  - enumerate dest‘s in same order
- \( S \) := number of seats in the vehicle
  - for this presentation: \( S = \infty \)

**Objective:** Feasible tour minimizing the sum of total distances.
The Dial-a-Ride Problem

A *Dial-a-Ride instance* is a triple \( I = (n, [d_{i,j}], S) \).

\( n := \text{number of riders} \)
Number of persons \( m = n + 1 \)

\([d_{i,j}] := \text{distance matrix} \)
start with 0 (driver’s pickup)
enumerate pickups
enumerate dest’s in same order

\( S := \text{number of seats in the vehicle} \)
for this presentation: \( S = \infty \)

**Objective:** Feasible tour minimizing the sum of total distances.
The Dial-a-Ride Problem

A *Dial-a-Ride instance* is a triple $I = (n, [d_{i,j}], S)$.

$n := \text{number of riders}$

Number of persons $m = n + 1$

$[d_{i,j}] := \text{distance matrix}$

- start with 0 (driver’s pickup)
- enumerate pickups
- enumerate dest’s in same order

$S := \text{number of seats in the vehicle}$

for this presentation: $S = \infty$

**Objective:** Feasible tour minimizing the sum of total distances.
An Exact Algorithm
An Exact Algorithm

There is an exact algorithm by Psaraftis, 1980.
An Exact Algorithm

There is an exact algorithm by Psaraftis, 1980. It works similar to the Held-Karp-algorithm.
An Exact Algorithm

There is an exact algorithm by Psaraftis, 1980. It works similar to the Held-Karp-algorithm. Running Time: $O^*(3^{n-1})$. 
An Exact Algorithm

There is an exact algorithm by Psaraftis, 1980. It works similar to the Held-Karp-algorithm.

Running Time: $O^*(3^{n-1})$.

Can be generalized to solve partial instances:
An Exact Algorithm

There is an exact algorithm by Psaraftis, 1980. It works similar to the Held-Karp-algorithm. Running Time: $O^*(3^{n-1})$.

Can be generalized to solve partial instances:
An Exact Algorithm

There is an exact algorithm by Psaraftis, 1980. It works similar to the Held-Karp-algorithm. Running Time: $O^*(3^{n-1})$.

Can be generalized to solve partial instances:

Find best tour such that
a) girl is delivered
b) waiting customer is fetched
c) boy is still on board.
Back to Rural Areas . . .
Back to Rural Areas . . .

A rural Dial-a-Ride instance typically looks like this:
Back to Rural Areas . . .

A rural Dial-a-Ride instance typically looks like this:

Assumptions:
Back to Rural Areas . . .

A rural Dial-a-Ride instance typically looks like this:

Assumptions:
→ Locations are inside clusters.
Back to Rural Areas . . .

A rural Dial-a-Ride instance typically looks like this:

Assumptions:
→ Locations are inside clusters.
→ Bypasses do not exist.
A rural Dial-a-Ride instance typically looks like this:

Assumptions:
→ Locations are inside clusters.
→ Bypasses do not exist.
Back to Rural Areas . . .

A rural Dial-a-Ride instance typically looks like this:

Assumptions:
→ Locations are inside clusters.
→ Bypasses do not exist.
→ All riders head in the same direction.
Back to Rural Areas . . .

A rural Dial-a-Ride instance typically looks like this:

Assumptions:
→ Locations are inside clusters.
→ Bypasses do not exist.
→ All riders head in the same direction.

Seems to be simpler than the Dial-a-Ride Problem . . .
Back to Rural Areas . . .

A rural Dial-a-Ride instance typically looks like this:

Assumptions:
→ Locations are inside clusters.
→ Bypasses do not exist.
→ All riders head in the same direction.

Seems to be simpler than the Dial-a-Ride Problem . . .
Back to Rural Areas . . .

A rural Dial-a-Ride instance typically looks like this:

Assumptions:
→ Locations are inside clusters.
→ Bypasses do not exist.
→ All riders head in the same direction.

*Seems* to be simpler than the Dial-a-Ride Problem . . .
Back to Rural Areas . . .

A rural Dial-a-Ride instance typically looks like this:

Assumptions:
→ Locations are inside clusters.
→ Bypasses do not exist.
→ All riders head in the same direction.

Seems to be simpler than the Dial-a-Ride Problem . . .
→ $T^*$-algorithm
Back to Rural Areas . . .

A rural Dial-a-Ride instance typically looks like this:

\[\text{Diagram of rural Dial-a-Ride with locations: Springfield, Hogsmeade, Minas Tirith.}\]

**Assumptions:**
→ Locations are inside clusters.
→ Bypasses do not exist.
→ All riders head in the same direction.

*Seems* to be simpler than the Dial-a-Ride Problem . . .

→ $T^*$-algorithm

**Goal:**
Back to Rural Areas . . .

A rural Dial-a-Ride instance typically looks like this:

Assumptions:
→ Locations are inside clusters.
→ Bypasses do not exist.
→ All riders head in the same direction.

*Seems* to be simpler than the Dial-a-Ride Problem . . .
→ $T^*$ -algorithm

Goal:
Classify instances whose optimal tour is unidirectional.
Back to Rural Areas . . .

A rural Dial-a-Ride instance typically looks like this:

Assumptions:
→ Locations are inside clusters.
→ Bypasses do not exist.
→ All riders head in the same direction.

Seems to be simpler than the Dial-a-Ride Problem . . .
→ $T^*$-algorithm

Goal:
Classify instances whose optimal tour is unidirectional.
(without computing it)
A Classifier

Springfield

Hogsmeade

Minas Tirith
A Classifier

**Idea:** Distribute the costs of a tour to the clusters.
A Classifier

Idea: Distribute the costs of a tour to the clusters.
Idea: Distribute the costs of a tour to the clusters.
A Classifier

Idea: Distribute the costs of a tour to the clusters.
A Classifier

**Idea:** Distribute the costs of a tour to the clusters.

Let $C_1, \ldots, C_q$ be the clusters.
A Classifier

Idea: Distribute the costs of a tour to the clusters.

Let $C_1, \ldots, C_q$ be the clusters.

Let $r(T, C_i) \in \mathbb{R}^+$
A Classifier

**Idea:** Distribute the costs of a tour to the clusters.

Let $C_1, \ldots, C_q$ be the clusters.

Let $\mathcal{C}(T, C_i) \in \mathbb{R}^+$ such that $\forall T: \sum_{i=1}^{q} \mathcal{C}(T, C_i) = c(T)$
**A Classifier**

**Idea:** Distribute the costs of a tour to the clusters.

Let $C_1, \ldots, C_q$ be the clusters.

Let $\mathcal{R}(T, C_i) \in \mathbb{R}^+$ such that $\forall T: \sum_{i=1}^{q} \mathcal{R}(T, C_i) = c(T)$

Let $\Phi(C_i)$ be a lower bound on $\mathcal{R}(T^*, C_i)$. 
A Classifier

Idea: Distribute the costs of a tour to the clusters.

Let \( C_1, \ldots, C_q \) be the clusters.

Let \( \mathcal{R}(T, C_i) \in \mathbb{R}^+ \) such that \( \forall T: \sum_{i=1}^{q} \mathcal{R}(T, C_i) = c(T) \)

Let \( \Phi(C_i) \) be a lower bound on \( \mathcal{R}(T^*, C_i) \).

Theorem (= Classifier):
A Classifier

**Idea:** Distribute the costs of a tour to the clusters.

Let $C_1, \ldots, C_q$ be the clusters.

Let $\mathcal{Y}(T, C_i) \in \mathbb{R}^+$ such that $\forall T: \sum_{i=1}^{q} \mathcal{Y}(T, C_i) = c(T)$

Let $\Phi(C_i)$ be a lower bound on $\mathcal{Y}(T^*, C_i)$.

**Theorem (Classifier):** $\forall C_i: \Phi(C_i) = \mathcal{Y}(\overrightarrow{T^*}, C_i) \Rightarrow T^* = \overrightarrow{T^*}$
A Classifier

Idea: Distribute the costs of a tour to the clusters.

Let $C_1, \ldots, C_q$ be the clusters.
Let $\Upsilon(T, C_i) \in \mathbb{R}^+$ such that $\forall T: \sum_{i=1}^{q} \Upsilon(T, C_i) = c(T)$
Let $\Phi(C_i)$ be a lower bound on $\Upsilon(T^*, C_i)$.

Theorem (= Classifier): $\forall C_i: \Phi(C_i) = \Upsilon(T^*, C_i) \Rightarrow T^* = \overrightarrow{T^*}$

Proof. Via exchange argument. □
A Classifier

Idea: Distribute the costs of a tour to the clusters.

Let $C_1, \ldots, C_q$ be the clusters.

Let $\mathcal{R}(T, C_i) \in \mathbb{R}^+$ such that $\forall T : \sum_{i=1}^q \mathcal{R}(T, C_i) = c(T)$

Let $\Phi(C_i)$ be a lower bound on $\mathcal{R}(T^*, C_i)$. TODO!

**Theorem (Classifier):** $\forall C_i : \Phi(C_i) = \mathcal{R}(\overrightarrow{T^*}, C_i) \Rightarrow T^* = \overrightarrow{T^*}$

*Proof.* Via exchange argument. \qed
Distribute Costs to Clusters

$m = 6$
Distribute Costs to Clusters

Assign the parts of a tour to clusters.
Distribute Costs to Clusters

Assign the parts of a tour to clusters.

Obs.: Edges of a tour are weighted.
Distribute Costs to Clusters

Assign the parts of a tour to clusters.

Obs.: Edges of a tour are weighted.
Distribute Costs to Clusters

Assign the parts of a tour to clusters.

Obs.: Edges of a tour are weighted.
Distribute Costs to Clusters

Assign the parts of a tour to clusters.

Obs.: Edges of a tour are weighted. → Count atomic journeys!
Distribute Costs to Clusters

Assign the parts of a tour to clusters.

Obs.: Edges of a tour are weighted. → Count atomic journeys!

Every cluster $C_i$ has four counters:
Assign the parts of a tour to clusters.

Obs.: Edges of a tour are weighted. → Count atomic journeys!
Every cluster $C_i$ has four counters:

$\alpha := \#\text{rightbound persons with } p_r \leq i$. 
Distribute Costs to Clusters

Assign the parts of a tour to clusters.

Obs.: Edges of a tour are weighted. \( \rightarrow \) Count atomic journeys!

Every cluster \( C_i \) has four counters:

\[
\alpha := \# \text{rightbound persons with } p_r \leq i.
\]
Distribute Costs to Clusters

Assign the parts of a tour to clusters.

\[ m = 6 \]

\textbf{Obs.:} Edges of a tour are weighted. \( \rightarrow \) Count atomic journeys!

Every cluster \( C_i \) has four counters:

\[ \alpha := \# \text{rightbound persons with } p_r \leq i. \]

\[ \beta := \# \text{leftbound persons with } d_r \geq i. \]
Distribute Costs to Clusters

Assign the parts of a tour to clusters.

Obs.: Edges of a tour are weighted. → Count atomic journeys!

Every cluster \( C_i \) has four counters:

\[ \alpha := \# \text{rightbound persons with } p_r \leq i. \]

\[ \beta := \# \text{leftbound persons with } d_r \geq i. \]
Assign the parts of a tour to clusters.

**Obs.:** Edges of a tour are weighted. → Count atomic journeys!

Every cluster $C_i$ has four counters:

- $\alpha := \#\text{rightbound persons with } p_r \leq i.$
- $\beta := \#\text{leftbound persons with } d_r \geq i.$
- $\gamma := \#\text{left-entering persons with } p_r \geq i.$
Distribute Costs to Clusters

Assign the parts of a tour to clusters.

Obs.: Edges of a tour are weighted. → Count atomic journeys!

Every cluster $C_i$ has four counters:

$\alpha := \# \text{rightbound persons with } p_r \leq i.$

$\beta := \# \text{leftbound persons with } d_r \geq i.$

$\gamma := \# \text{left-entering persons with } p_r \geq i.$

$\delta := \# \text{right-entering persons with } d_r \leq i.$
Distribute Costs to Clusters

Assign the parts of a tour to clusters.

Obs.: Edges of a tour are weighted. → Count atomic journeys!

Every cluster $C_i$ has four counters:

$\alpha := \#\text{rightbound persons with } p_r \leq i.$

$\beta := \#\text{leftbound persons with } d_r \geq i.$

$\gamma := \#\text{left-entering persons with } p_r \geq i.$

$\delta := \#\text{right-entering persons with } d_r \leq i.$

$\Upsilon(T, C_i) = \text{in}(C_i)$
Distribute Costs to Clusters

Assign the parts of a tour to clusters.

Obs.: Edges of a tour are weighted. → Count atomic journeys!

Every cluster $C_i$ has four counters:

- $\alpha := \#\text{rightbound persons with } p_r \leq i.$
- $\beta := \#\text{leftbound persons with } d_r \geq i.$
- $\gamma := \#\text{left-entering persons with } p_r \geq i.$
- $\delta := \#\text{right-entering persons with } d_r \leq i.$

$$\Upsilon(T, C_i) = \text{in}(C_i) + \alpha C_i C_{i+1}$$
Distribute Costs to Clusters

Assign the parts of a tour to clusters.

![Diagram of a tour with labeled clusters and points](image)

$m = 6$

**Obs.:** Edges of a tour are weighted. → Count atomic journeys!

Every cluster $C_i$ has four counters:

- $\alpha := \#\text{rightbound persons with } p_r \leq i$.
- $\beta := \#\text{leftbound persons with } d_r \geq i$.
- $\gamma := \#\text{left-entering persons with } p_r \geq i$.
- $\delta := \#\text{right-entering persons with } d_r \leq i$.

$$\Upsilon(T, C_i) = \text{in}(C_i) + \alpha C_i C_{i+1} + \beta C_i C_{i-1}$$
Distribute Costs to Clusters

Assign the parts of a tour to clusters.

**Obs.:** Edges of a tour are weighted. → Count atomic journeys!

Every cluster $C_i$ has four counters:

- $\alpha := \#$rightbound persons with $p_r \leq i$.
- $\beta := \#$leftbound persons with $d_r \geq i$.
- $\gamma := \#$left-entering persons with $p_r \geq i$.
- $\delta := \#$right-entering persons with $d_r \leq i$.

$$\Upsilon(T, C_i) = \text{in}(C_i) + \alpha C_i C_{i+1} + \beta C_i C_{i-1} + \gamma C_{i-1} C_i$$

$m = 6$

[prompt]: Does this page contain any mathematical notations? Yes, it contains mathematical notations and formulas.

[prompt]: Is there any diagram or graph on this page? Yes, there is a diagram of a tour with nodes labeled Hogsmeade, Springfield, and Minas Tirith, and edges indicating the movement of persons.

[prompt]: What is the significance of the edges in the diagram? The edges represent the movement of persons from one location to another, with weights indicating the cost associated with each movement.

[prompt]: Are there any specific instructions or problems stated for the students? Yes, the students are asked to calculate the costs for each cluster $C_i$ using the formula $\Upsilon(T, C_i)$.

[prompt]: Is there any special notation or terminology used in the document? Yes, the notation $\text{in}(C_i)$ represents the in-degree of cluster $C_i$, and $C_0$ and $C_{n+1}$ are used to denote the pickup and dropoff clusters, respectively.

[prompt]: Is there any clarification or explanation for the diagram? Yes, the diagram shows the movement of persons from one location to another, and the edges are weighted to indicate the cost of movement. The pickup and dropoff clusters are marked with arrows and labels.
Distribute Costs to Clusters

Assign the parts of a tour to clusters.

Obs.: Edges of a tour are weighted. → Count atomic journeys!

Every cluster $C_i$ has four counters:

- $\alpha := \# \text{rightbound persons with } p_r \leq i.$
- $\beta := \# \text{leftbound persons with } d_r \geq i.$
- $\gamma := \# \text{left-entering persons with } p_r \geq i.$
- $\delta := \# \text{right-entering persons with } d_r \leq i.$

$(T, C_i) = \in(C_i) + \alpha \overline{C_i C_{i+1}} + \beta \overline{C_i C_{i-1}} + \gamma \overline{C_{i-1} C_i} + \delta \overline{C_{i+1} C_i}$

$m = 6$

See thesis for proof of $c(T) = \sum \gamma(T, C_i)$. 
Distribute Costs to Clusters

Assign the parts of a tour to clusters.

Obs.: Edges of a tour are weighted. → Count atomic journeys!

Every cluster $C_i$ has four counters:

- $\alpha := \#\text{rightbound persons with } p_r \leq i$. 
- $\beta := \#\text{leftbound persons with } d_r \geq i$. 
- $\gamma := \#\text{left-entering persons with } p_r \geq i$. 
- $\delta := \#\text{right-entering persons with } d_r \leq i$. 

$\gamma(T, C_i) = \text{in}(C_i) + \alpha C_i C_{i+1} + \beta C_i C_{i-1} + \gamma C_{i-1} C_i + \delta C_{i+1} C_i$

Todo: $\Phi(C_i) \leq \gamma(T^*, C_i)$
Lower Bound on $\mathcal{R}(T^*, C_i)$ (Sketch)
Lower Bound on $\mathcal{R}(T^*, C_i)$ (Sketch)

**Idea:** Any $T$ induces an ordered partition on every cluster.
Lower Bound on $\mathcal{R}(T^*, C_i)$ (Sketch)

**Idea:** Any $T$ induces an ordered partition on every cluster.

$m = 6$

Springfield

Hogsmeade

Minas Tirith

$\{8\}, \{4\}$
Lower Bound on $\Upsilon(T^*, C_i)$ (Sketch)

**Idea:** Any $T$ induces an ordered partition on every cluster.

$m = 6$

Other Possibilities?
Lower Bound on $\mathcal{R}(T^*, C_i)$ (Sketch)

**Idea:** Any $T$ induces an ordered partition on every cluster.

$m = 6$

Other Possibilities? $\mathcal{S} = [\{4\}, \{8\}]$  $\mathcal{S} = [\{4, 8\}]$
Lower Bound on \( \mathcal{R}(T^*, C_i) \) (Sketch)

**Idea:** Any \( T \) induces an ordered partition on every cluster.

\[ m = 6 \]

\[ \{8\}, \{4\} \]

Other Possibilities? \( S = [\{4\}, \{8\}] \quad S = [\{4, 8\}] \)

Additionally: List of Portals \( P \).
Lower Bound on $\mathcal{R}(T^*, C_i)$ (Sketch)

**Idea:** Any $T$ induces an ordered partition on every cluster.

Other Possibilities? $S = \{\{4\}, \{8\}\} \quad S = \{\{4, 8\}\}$

Additionally: List of Portals $P$. 

$$m = 6$$

$$[[l, l], [l, r]] \quad \{8\}, \{4\}$$
Lower Bound on $\Upsilon(T^*, C_i)$ (Sketch)

**Idea:** Any $T$ induces an ordered partition on every cluster.

$$m = 6$$

Other Possibilities? $S = \{\{4\}, \{8\}\}$

Additionally: List of Portals $P$.

Given $S$ and $P$ the lower bound can be estimated.
Lower Bound on $\mathcal{R}(T^*, C_i)$ (Sketch)

**Idea:** Any $T$ induces an ordered partition on every cluster.

Other Possibilities? $\mathcal{S} = \{\{4\}, \{8\}\}$  
Additionally: List of Portals $P$.

Given $\mathcal{S}$ and $P$ the lower bound can be estimated.

Solve internal tours.
Lower Bound on $\mathcal{Y}(T^*, C_i)$ (Sketch)

**Idea:** Any $T$ induces an ordered partition on every cluster.

Given $S$ and $P$ the lower bound can be estimated.

Solve internal tours.

Compute lower bounds for $\alpha$, $\beta$, $\gamma$ and $\delta$.
Lower Bound on $\Psi(T^*, C_i)$ (Sketch)

**Idea:** Any $T$ induces an ordered partition on every cluster.

Other Possibilities? $S = \{\{4\}, \{8\}\}$ $S = [\{4, 8\}]$

Additionally: List of Portals $P$.

Given $S$ and $P$ the lower bound can be estimated.

Solve internal tours.

Compute lower bounds for $\alpha, \beta, \gamma$ and $\delta$.

Add costs up and obtain lower bound $\Phi_{S,P}(C_i)$. 

$m = 6$

[[I, I], [I, r]] $\{8\}, \{4\}$
Lower Bound on $\Upsilon(T^*, C_i)$ (Sketch)

**Idea:** Any $T$ induces an ordered partition on every cluster.

Given $S$ and $P$ the lower bound can be estimated.

Solve internal tours.

Compute lower bounds for $\alpha$, $\beta$, $\gamma$ and $\delta$.

Add costs up and obtain lower bound $\Phi_{S,P}(C_i)$.

$\Rightarrow \min \Phi(C_i)_{S,P} = \Phi(C_i) \leq \Upsilon(T^*, C_i)$
Lower Bound on $\Upsilon(T^*, C_i)$ (Sketch)

Idea: Any $T$ induces an ordered partition on every cluster.

$\text{Other Possibilities? } S = [\{4\}, \{8\}]$  
$S = [\{4, 8\}]$

Additionally: List of Portals $P$.

Given $S$ and $P$ the lower bound can be estimated.

Solve internal tours.

Compute lower bounds for $\alpha, \beta, \gamma$ and $\delta$.

Add costs up and obtain lower bound $\Phi_{S,P}(C_i)$.

$\Rightarrow \min \Phi(C_i)_{S,P} = \Phi(C_i) \leq \Upsilon(T^*, C_i)$
**Lower Bound on** $\mathcal{Y}(T^*, C_i)$ **(Sketch)**

**Idea:** Any $T$ induces an ordered partition on every cluster.

![Diagram with cities and routes]

$m = 6$

$[\{4\}, \{8\}]$ $S = [\{4, 8\}]$

Other Possibilities? $S = [\{4\}, \{8\}]$ $S = [\{4, 8\}]$

Additionally: List of Portals $P$.

Given $S$ and $P$ the lower bound can be estimated.

Solve internal tours.

Compute lower bounds for $\alpha, \beta, \gamma$ and $\delta$.

Add costs up and obtain lower bound $\Phi_{S,P}(C_i)$.

$\Rightarrow \min \Phi(C_i)_{S,P} = \Phi(C_i) \leq \mathcal{Y}(T^*, C_i)$
Evaluation
Evaluation

→ First artificial instances, then realistic instances.
Evaluation

→ First artificial instances, then realistic instances.
Evaluation

for $n = 12$

→ First artificial instances, then realistic instances.
Evaluation

for $n = 12$

→ First artificial instances, then realistic instances.

Runtimes:
Evaluation

for $n = 12$

→ First artificial instances, then realistic instances.

Runtimes:

Exact: 120 s  \quad \overrightarrow{T^*}-Algorithm: 3 ms  \quad Classifier: 4 s
Evaluation

for $n = 12$

→ First artificial instances, then realistic instances.

Runtimes:

Exact: 120 s  \quad \overset{T^*}{-}\text{Algorithm}: 3 \text{ ms}  \quad \text{Classifier: } 4 \text{ s}

Classifier’s Accuracy:
Evaluation

for $n = 12$

→ First artificial instances, then realistic instances.

Runtimes:

Exact: 120 s  
$\overrightarrow{T^*}$-Algorithm: 3 ms  
Classifier: 4 s

Classifier’s Accuracy:

Ratio $T^* = \overrightarrow{T^*}$
Evaluation

for \( n = 12 \)

→ First artificial instances, then realistic instances.

Runtimes:

Exact: 120 s \( \vec{T} \)-Algorithm: 3 ms Classifier: 4 s

Classifier’s Accuracy:

\[
\text{Ratio } T^* = \vec{T}^* \]

Clusters close together (\( \sim 6\text{km} \)): 59 %
Evaluation

for $n = 12$

→ First artificial instances, then realistic instances.

Runtimes:

Exact: 120 s  \quad \overrightarrow{T^*\text{-Algorithm}}: 3 \text{ ms}  \quad \text{Classifier: } 4 \text{ s}

Classifier’s Accuracy:

$$\text{Ratio } T^* = \overrightarrow{T^*}$$

Clusters close together ($\sim 6\text{km}$): 59 %

far apart ($\geq 16 \text{ km}$): 100 %
Evaluation

for $n = 12$

→ First artificial instances, then realistic instances.

Runtimes:

Exact: 120 s  \quad \overrightarrow{T^*}-Algorithm: 3 ms  \quad \text{Classifier: 4 s}

Classifier’s Accuracy:

\[
\text{Ratio } \overrightarrow{T^*} = \overrightarrow{T^*} \quad \text{Recall}
\]

Clusters close together ($\sim 6\text{km}$): 59 %

far apart ($\geq 16\text{ km}$): 100 %
Evaluation

for $n = 12$

→ First artificial instances, then realistic instances.

Runtimes:

**Exact**: 120 s  
**$T^*$-Algorithm**: 3 ms  
**Classifier**: 4 s

Classifier’s Accuracy:

- Ratio $T^* = \overrightarrow{T^*}$
  - Clusters close together ($\sim 6\text{km}$): 59 %
  - far apart ($\geq 16 \text{ km}$): 100 %

Recall 0.4
Evaluation

for $n = 12$

→ First artificial instances, then realistic instances.

Runtimes:

Exact: 120 s  \hspace{1cm} \overrightarrow{T^*}\text{-Algorithm}: 3 ms  \hspace{1cm} \text{Classifier}: 4 s

Classifier’s Accuracy:

<table>
<thead>
<tr>
<th>Ratio $T^* = \overrightarrow{T^*}$</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clusters close together ($\sim 6\text{km}$):</td>
<td>59 %</td>
</tr>
<tr>
<td>far apart ($\geq 16 \text{ km}$):</td>
<td>100 %</td>
</tr>
</tbody>
</table>
Evaluation

for $n = 12$

→ First artificial instances, then realistic instances.

Runtimes:

**Exact:** 120 s

**$T^*$-Algorithm:** 3 ms

**Classifier:** 4 s

Classifier’s Accuracy:

<table>
<thead>
<tr>
<th>Ratio $T^* = \overrightarrow{\text{T^*}}$</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clusters close together ($\sim 6\text{km}$):</td>
<td>59 %</td>
</tr>
<tr>
<td>far apart ($\geq 16\text{ km}$):</td>
<td>100 %</td>
</tr>
</tbody>
</table>

$T^*$-Algorithm as Heuristic:
Evaluation

for $n = 12$

→ First artificial instances, then realistic instances.

Runtimes:

Exact: 120 s  
$\overline{T^*}$-Algorithm: 3 ms  
Classifier: 4 s

Classifier’s Accuracy:

<table>
<thead>
<tr>
<th>Ratio $T^* = \overline{T^*}$</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clusters close together ($\sim 6$km):</td>
<td>59 %</td>
</tr>
<tr>
<td>far apart ($\geq 16$ km):</td>
<td>100 %</td>
</tr>
</tbody>
</table>

$\overline{T^*}$-Algorithm as Heuristic:

Approximation Quality (empiric): $\leq 1.1$
Topology of Street Networks
Topology of Street Networks

Street Networks often do not meet the assumptions.
Topology of Street Networks

Street Networks often do not meet the assumptions.

Example #1:
Rural Instance
Topology of Street Networks

Street Networks often do not meet the assumptions.

Example #1: Rural Instance
Topology of Street Networks

Street Networks often do not meet the assumptions.

Example #1:
Rural Instance

$T^*$ bypasses a cluster!
Street Networks often do not meet the assumptions.

Example #1:
Rural Instance

$T^*$ bypasses a cluster!

Yet, no false positive.
Topology of Street Networks

Street Networks often do not meet the assumptions.

Example #1: Rural Instance

\(T^*\) bypasses a cluster!

Yet, no false positive.

\[\Rightarrow\text{Classifier is robust to some extent.}\]
Topology of Street Networks

Street Networks often do not meet the assumptions.

Example #2: Regional Instance
Topology of Street Networks

Street Networks often do not meet the assumptions.

Example #2:
Regional Instance
Really hard scenario . . .
Topology of Street Networks

Street Networks often do not meet the assumptions.

Example #2:
Regional Instance
Really hard scenario . . .
Topology of Street Networks

Street Networks often do not meet the assumptions.

Example #2:
Regional Instance

Really hard scenario . . .

False positives are to be expected in this case.
Conclusion
Conclusion

The Exact Algorithm considers unsensible tours.
Conclusion

The Exact Algorithm considers unsensible tours.
Conclusion

The Exact Algorithm considers unsensible tours.
Conclusion

The Exact Algorithm considers unsensible tours.

Intuition yields the $\overrightarrow{T^*}$-algorithm.
Conclusion

The Exact Algorithm considers unsensible tours.

Intuition yields the $T^*$-algorithm.

A **classifier** decides if the $T^*$-algorithm can be used.
Conclusion

The Exact Algorithm considers unsensible tours. Intuition yields the $T^*$-algorithm. A classifier decides if the $T^*$-algorithm can be used. If yes, only a fraction of time is needed to get $T^*$. 
Conclusion

The Exact Algorithm considers unsensible tours.

Intuition yields the $\overrightarrow{T^*}$-algorithm.

A **classifier** decides if the $\overrightarrow{T^*}$-algorithm can be used.

If **yes**, only a fraction of time is needed to get $T^*$.

If **no**, virtually no time is wasted.
Conclusion

The Exact Algorithm considers unsensible tours.

Intuition yields the $\overrightarrow{T^*}$-algorithm.

A **classifier** decides if the $\overrightarrow{T^*}$-algorithm can be used.

If **yes**, only a fraction of time is needed to get $T^*$.

If **no**, virtually no time is wasted.

**No false-positives:** Optimal route is guaranteed.
Conclusion

The Exact Algorithm considers unsensible tours.

Intuition yields the $\overrightarrow{T^*}$-algorithm.

A **classifier** decides if the $\overrightarrow{T^*}$-algorithm can be used. If **yes**, only a fraction of time is needed to get $T^*$. If **no**, virtually no time is wasted.

**No false-positives:** Optimal route is guaranteed.
Conclusion

The Exact Algorithm considers unsensible tours.

Intuition yields the $\overrightarrow{T^*}$-algorithm.

A **classifier** decides if the $\overrightarrow{T^*}$-algorithm can be used.

If **yes**, only a fraction of time is needed to get $T^*$.

If **no**, virtually no time is wasted.

**No false-positives:** Optimal route is guaranteed.
Attributions

The above icons are made by Freepik from flaticon.com

← CC 3.0 BY by SimpleIcon from flaticon.com

(c) Map Images from OpenStreetMap (osm.org)
The following slides were abandoned at some point and not officially shown at the presentation. They may contain errors or are incomplete. Maybe they help you nonetheless.
The Objective Function
A tour $T$ is a permutation of $[0, 2m - 1]$. 
The Objective Function

A tour $T$ is a permutation of $[0, 2m - 1]$. 

$T = [0, 3, 1, 5, 7, 2, 6, 4]$
The Objective Function

A tour $T$ is a permutation of $[0, 2m - 1]$.

$T$ feasible $\iff$

$T = \{0, 3, 1, 5, 7, 2, 6, 4\}$
The Objective Function

A tour $T$ is a permutation of $[0, 2m - 1]$. 

$T$ feasible $\iff T[1] = 0 \& T[2m] = m$

$T = [0, 3, 1, 5, 7, 2, 6, 4]$
The Objective Function

A tour $T$ is a permutation of $[0, 2m - 1]$.

$T$ feasible $\iff T[1] = 0$ & $T[2m] = m$
& precedences obeyed

$T = [0, 3, 1, 5, 7, 2, 6, 4]$
A tour $T$ is a permutation of $[0, 2m - 1]$.

$T$ feasible $\iff T[1] = 0$ & $T[2m] = m$ & precedences obeyed & $S$ not violated

$T = [0, 3, 1, 5, 7, 2, 6, 4]$
A *tour* $T$ is a permutation of $[0, 2m - 1]$. 

$T$ feasible $\iff T[1] = 0$ & $T[2m] = m$ 
& precedences obeyed

& $S$ not violated

$T = [0, 3, 1, 5, 7, 2, 6, 4]$
The Objective Function

A *tour* $T$ is a permutation of $[0, 2m - 1]$.

$T$ feasible $\iff T[1] = 0 \& T[2m] = m \&$ precedences obeyed $\& S$ not violated

Objective:

$$\min_{T \text{ feasible}} \sum_{i=2}^{2m} k(i - 1) \cdot d[T[i-1], T[i]]$$

$T = [0, 3, 1, 5, 7, 2, 6, 4]$
The Objective Function

A tour $T$ is a permutation of $[0, 2m - 1]$.

$T$ feasible $\iff T[1] = 0 \& T[2m] = m$ & precedences obeyed & $S$ not violated

Objective:

$$\min_{T \text{ feasible}} \sum_{i=2}^{2m} k(i - 1) \cdot d[T[i - 1], T[i]]$$

$k(j)$ is the number of persons after step $j$ of $T$. 

$T = [0, 3, 1, 5, 7, 2, 6, 4]$
An Exact Algorithm
An Exact Algorithm
An Exact Algorithm

Find a tour with 6 steps:
An Exact Algorithm

Find a tour with 6 steps:

\[
\begin{array}{cccccc}
1 & 2 & 3 & 4 & 5 & 6 \\
\end{array}
\]
An Exact Algorithm

Find a tour with 6 steps:

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
</table>

```latex
\begin{tabular}{c}
1 \\
2 \\
3 \\
4 \\
5 \\
6
\end{tabular}
```

![Diagram](image.png)

Cost: 0
An Exact Algorithm

Find a tour with 6 steps:

1 | 2 | 3 | 4 | 5 |
---|---|---|---|---|
0 | 2 |  |  |  | cost: 20
   |   | 1 |  |  | cost: 9
   |   |   |  |  | cost: 0
An Exact Algorithm

Find a tour with 6 steps:

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

- Cost: 0
- Cost: 9
- Cost: 20
- Cost: 41
- Cost: 49

```
1 5 3
```
An Exact Algorithm

Find a tour with 6 steps:

\[
\begin{array}{cccccc}
0 & 1 & 2 & 3 & 4 & 5 \\
\text{cost: 0} & \text{cost: 9} & \text{cost: 20} & \text{cost: 41} & \text{cost: 49} & \text{cost: 49} \\
\end{array}
\]

1
2
3
4
5
0
1
2
3
4
5

1
5
3
2
4
An Exact Algorithm

Find a tour with 6 steps:

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (cost: 0)</td>
<td>1 (cost: 9)</td>
<td>4 (cost: 41)</td>
<td>2 (cost: 57)</td>
<td></td>
</tr>
<tr>
<td>2 (cost: 20)</td>
<td>1 (cost: 60)</td>
<td>5 (cost: 49)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4 (cost: 49)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The tour with the minimum cost is 0-1-2-4-5-3, with a total cost of 49.
An Exact Algorithm

Find a tour with 6 steps:

1. Cost: 0
2. Cost: 9
3. Cost: 20
4. Cost: 41
5. Cost: 49
6. Cost: 60
7. Cost: 92
8. Cost: 49
9. Cost: 92
10. Cost: 57

Diagram shows a network of nodes with costs between them.
An Exact Algorithm

Find a tour with 6 steps:

1 2 3 4 5

0

1 cost: 9

2 cost: 20

4 cost: 41

2 cost: 49

4 cost: 60

5 cost: 49

2 cost: 98

5 cost: 92

84

1

5

3

4

2
An Exact Algorithm

Find a tour with 6 steps:

1. Begin at node 0.
2. Move to node 1 at a cost of 9.
4. Move to node 3 at a cost of 49.
5. Move to node 4 at a cost of 41.
6. Move to node 5 at a cost of 57.

Total cost: 0 + 9 + 20 + 49 + 41 + 57 = 178.

The tour from node 0 to node 5 to node 3 to node 2 to node 4 to node 0 is a valid tour with a total cost of 178.

Graph representation:

- Node 0 connects to node 1 (cost 9).
- Node 1 connects to node 2 (cost 20).
- Node 2 connects to node 3 (cost 49).
- Node 3 connects to node 4 (cost 41).
- Node 4 connects to node 5 (cost 57).
- Node 5 connects to node 0 (cost 57).

The sequence of moves is: 0 → 1 → 2 → 3 → 4 → 5 → 0.
An Exact Algorithm

Find a tour with 6 steps:

1 2 3 4 5

0 1 2 3 4 5

1 2 3 4 5

Cost: 0

Cost: 9

Cost: 41

Cost: 57

Cost: 86

Cost: 20

Cost: 60

Cost: 57

Cost: 49

Cost: 98

Cost: 92

Cost: 49

Cost: 49
An Exact Algorithm

Find a tour with 6 steps:

1. 2. 3. 4. 5. 6.

- 0 → 1 (cost: 9)
- 1 → 2 (cost: 20)
- 2 → 3 (cost: 41)
- 3 → 4 (cost: 49)
- 4 → 5 (cost: 57)
- 5 → 6 (cost: 86)
An Exact Algorithm

Find a tour with 6 steps:

1. Start at 0.
2. Go to 1 with cost 9.
3. Go to 2 with cost 49.
4. Go to 4 with cost 41.
5. Go to 5 with cost 57.
6. Go to 3 with cost 98.

The total cost of the tour is 209.
An Exact Algorithm

Find a tour with 6 steps:

0 → 1 → 4 → 2 → 1 → 5 → 3

Costs:
- 0: 0
- 1: 9
- 2: 49
- 3: 86
- 4: 41
- 5: 57
- 6: 98
An Exact Algorithm

Find a tour with 6 steps:

→ Generalizes to an algorithm with exchangeable objective
An Exact Algorithm

Find a tour with 6 steps:

0 → 1 → 4 → 2 → 5 → 3

→ Generalizes to an algorithm with exchangeable objective
→ DFS-like traversal also possible  [Psaraftis 1980]
An Exact Algorithm

Find a tour with 6 steps:

0 → 1 → 2 → 3 → 4 → 5 → 6

→ Generalizes to an algorithm with exchangeable objective
→ DFS-like traversal also possible [Psaraftis 1980]
→ BFS-like traversal can save storage
Running Time and Partial Execution

Diagram showing a network with nodes labeled 0 to 5 and edges connecting them with costs associated with each edge. The costs are shown next to the edges and nodes.
At every step, a rider can have three steps: *wait, travel, finish.*
Running Time and Partial Execution

At every step, a rider can have three steps: wait, travel, finish. ⇒ for a fixed location ∉ \{0, m\} there are 3^{n-1} states.
At every step, a rider can have three steps: wait, travel, finish.

⇒ for a fixed location $\neq \{0, m\}$ there are $3^{n-1}$ states.
Running Time and Partial Execution

At every step, a rider can have three steps: wait, travel, finish. 

⇒ for a fixed location $\not\in \{0, m\}$ there are $3^{n-1}$ states. 

⇒ $2n3^{n-1} + 2$ vertices, which yields $O^*(3^{n-1})$ running time.
Running Time and Partial Execution

At every step, a rider can have three steps: wait, travel, finish.

⇒ for a fixed location \( \neq \{0, m\} \) there are \( 3^{n-1} \) states.

⇒ \( 2n3^{n-1} + 2 \) vertices, which yields \( O^*(3^{n-1}) \) running time.

Given \( S \) and \( S' \), the instance can be solved partially.
At every step, a rider can have three steps: *wait*, *travel*, *finish*. 

⇒ for a fixed location \( \notin \{0, m\} \) there are \( 3^{n-1} \) states.

⇒ \( 2n3^{n-1} + 2 \) vertices, which yields \( O^*(3^{n-1}) \) running time.

Given \( S \) and \( S' \), the instance can be solved *partially*. 
At every step, a rider can have three steps: wait, travel, finish.

for a fixed location $\not\in \{0, m\}$ there are $3^{n-1}$ states.

$2n3^{n-1} + 2$ vertices, which yields $O^*(3^{n-1})$ running time.

Given $S$ and $S'$, the instance can be solved partially.
At every step, a rider can have three steps: wait, travel, finish.

⇒ for a fixed location $\notin \{0, m\}$ there are $3^{n-1}$ states.

⇒ $2n3^{n-1} + 2$ vertices, which yields $O^*(3^{n-1})$ running time.

Given $S$ and $S'$, the instance can be solved partially.
Running Time and Partial Execution

At every step, a rider can have three steps: wait, travel, finish.

⇒ for a fixed location \( \not \in \{0, m\} \) there are \( 3^{n-1} \) states.

⇒ \( 2n3^{n-1} + 2 \) vertices, which yields \( O^*(3^{n-1}) \) running time.

Given \( S \) and \( S' \), the instance can be solved partially.
Evalution of Realistic Examples
Evalution of Realistic Examples

Rural Scenario
Evalution of Realistic Examples

Rural Scenario

Regional Scenario
Evalution of Realistic Examples

Rural Scenario

Regional Scenario

Intercity Scenario
Evalution of Realistic Examples

Rural Scenario
Six small villages with $\varnothing 1.2$ km distance.

Regional Scenario

Intercity Scenario
Evalutation of Realistic Examples

Rural Scenario
Six small villages with $\varnothing 1.2$ km distance.

Regional Scenario
Six small towns with $\varnothing 7.2$ km distance.

Intercity Scenario
Evalution of Realistic Examples

Rural Scenario
Six small villages with Ø1.2 km distance.

Regional Scenario
Six small towns with Ø7.2 km distance.

Intercity Scenario
Six major german cities with Ø129 km distance.
Evalutation of Realistic Examples

Rural Scenario
Six small villages with $\varnothing 1.2$ km distance.

Regional Scenario
Six small towns with $\varnothing 7.2$ km distance.

Intercity Scenario
Six major german cities with $\varnothing 129$ km distance.

All optimal tours are unidirectional, recall $> 0.9$. 
Evalution of Realistic Examples

**Rural Scenario**
Six small villages with $\varnothing 1.2$ km distance.
> 70% optimal tours unidirectional, recall < 0.1.

**Regional Scenario**
Six small towns with $\varnothing 7.2$ km distance.

**Intercity Scenario**
Six major german cities with $\varnothing 129$ km distance.
All optimal tours are unidirectional, recall > 0.9.
Evalution of Realistic Examples

**Rural Scenario**
Six small villages with $\mathcal{O}$1.2 km distance.
> 70% optimal tours unidirectional, recall $< 0.1$.
Bad, distances are too small.

**Regional Scenario**
Six small towns with $\mathcal{O}$7.2 km distance.

**Intercity Scenario**
Six major german cities with $\mathcal{O}$129 km distance.
All optimal tours are unidirectional, recall $> 0.9$. 
Evalution of Realistic Examples

**Rural Scenario**
Six small villages with $\varnothing 1.2$ km distance.
> 70\% optimal tours unidirectional, recall < 0.1.
Bad, distances are too small.

**Regional Scenario**
Six small towns with $\varnothing 7.2$ km distance.
> 50\% optimal tours unidir., recall > 0.55, precision 0.61.

**Intercity Scenario**
Six major german cities with $\varnothing 129$ km distance.
All optimal tours are unidirectional, recall > 0.9.
Evalution of Realistic Examples

**Rural Scenario**
Six small villages with $\varnothing 1.2$ km distance.
> 70% optimal tours unidirectional, recall < 0.1.
Bad, distances are too small.

**Regional Scenario**
Six small towns with $\varnothing 7.2$ km distance.
> 50% optimal tours unidir., recall > 0.55, precision 0.61.

Wait . . . What?!

**Intercity Scenario**
Six major german cities with $\varnothing 129$ km distance.
All optimal tours are unidirectional, recall > 0.9.