LAAS - DO - ROC

Efficient multiobjective and multimodal path computation

Marie-José Huguet - Visite ENS - 7 janvier 2015
Introduction

Users mobility in multimodal transportation networks

- large size network
- new usages and new problems

Bi-Objective Shortest Paths

- Min path duration and Min number of mode transfers
- Industrial transfer - CIFRE thesis (12/2010)

Synchronized Itineraries

- Synchronized Paths between 2 users / 2-Way Shortest Path
  - ANR Project (lead TSF): use case on dynamic carpooling (2012-)
  - GdR OR Project (collaborations: LIPN, LI Tours, Heudiasyc) (2011-2012)
- Open source software
Introduction

Users mobility in multimodal transportation networks

- large size network
- new usages and new problems

Bi-Objective Shortest Paths

- Min path duration and Min number of mode transfers
- Industrial transfer - CIFRE thesis (12/2010)

Synchronized Itineraries

- Synchronized Paths between 2 users / 2-Way Shortest Path
  - ANR Project (lead TSF): use case on dynamic carpooling (2012-)
  - GdR OR Project (collaborations: LIPN, LI Tours, Heudiasyc) (2011-2012)
- Open source software
Introduction

Users mobility in multimodal transportation networks
- large size network
- new usages and new problems

Bi-Objective Shortest Paths
- Min path duration and Min number of mode transfers
- Industrial transfer - CIFRE thesis (12/2010)

Synchronised Itineraries
- Synchronised Paths between 2 users / 2-Way Shortest Path
  - ANR Project (lead TSF): use case on dynamic carpooling (2012-)
  - GdR OR Project (collaborations: LIPN, LI Tours, Heudiasyc) (2011-2012)
- Open source software
Shortest Path Problems (SPP)

Standard Shortest Path from $o$ to $d$: Well-known Dijkstra Algorithm

- Two main variants
  - $A^*$ (guided by the destination $d$)
  - Bidirectional Search (forward from $o$ and backward from $d$)
- Preprocessing technics

Multimodal Shortest Path

- Labelled graph: mode on arcs
- Constraints on mode: regular language
Shortest Path Problems (SPP)

Standard Shortest Path from \( o \) to \( d \): Well-known Dijkstra Algorithm

- Two main variants
  - A* (guided by the destination \( d \))
  - Bidirectional Search (forward from \( o \) and backward from \( d \))
- Preprocessing technics

Multimodal Shortest Path

- Labelled graph: mode on arcs
- Constraints on mode: regular language
Shortest Path Problems (SPP)

Standard Shortest Path from $o$ to $d$: Well-known Dijkstra Algorithm

- Two main variants
  - A* (guided by the destination $d$)
  - Bidirectional Search (forward from $o$ and backward from $d$)
- Preprocessing technics

Multimodal Shortest Path

- Labelled graph: mode on arcs
- Constraints on mode: regular language
Shortest Path Problems (SPP)

Standard Shortest Path from $o$ to $d$: Well-known Dijkstra Algorithm

- Two main variants
  - A* (guided by the destination $d$)
  - Bidirectional Search (forward from $o$ and backward from $d$)
- Preprocessing technics

Multimodal Shortest Path

- Labelled graph: mode on arcs
- Constraints on mode: regular language
Shortest Path Problems (SPP)

Standard Shortest Path from $o$ to $d$: Well-known Dijkstra Algorithm

- Two main variants
  - A* (guided by the destination $d$)
  - Bidirectional Search (forward from $o$ and backward from $d$)
- Preprocessing techniques

Multimodal Shortest Path

- Labelled graph: mode on arcs
- Constraints on mode: regular language
Shortest Path Problems (SPP)

Standard Shortest Path from $o$ to $d$: Well-known Dijkstra Algorithm

- Two main variants
  - A* (guided by the destination $d$)
  - Bidirectional Search (forward from $o$ and backward from $d$)
- Preprocessing techniques

Multimodal Shortest Path

- Labelled graph: mode on arcs
- Constraints on mode: regular language
Shortest Path Problems (SPP)

Standard Shortest Path from \( o \) to \( d \): Well-known Dijkstra Algorithm

- Two main variants
  - A* (guided by the destination \( d \))
  - Bidirectional Search (forward from \( o \) and backward from \( d \))
- Preprocessing technics

Multimodal Shortest Path

- Labelled graph: mode on arcs
- Constraints on mode: regular language
Shortest Path Problems (SPP)

Standard Shortest Path from $o$ to $d$: Well-known Dijkstra Algorithm

- Two main variants
  - A* (guided by the destination $d$)
  - Bidirectional Search (forward from $o$ and backward from $d$)
- Preprocessing technics

Multimodal Shortest Path

- Labelled graph: mode on arcs
- Constraints on mode: regular language
**Bi-Objective Multimodal SPP**

### Objectives
- Minimize the travel time
- Minimize the number of transfers
- Pareto set of solutions

![Graph showing trade-off between duration and number of transfers between TLS and MQLS]

### Contributions
- Complexity: polynomial
- Two main algorithms
- Dominance rule based on automata
- A* and bidirectional variants
**Objectives**
- Minimize the travel time
- Minimize the number of transfers
- Pareto set of solutions

**Contributions**
- Complexity: polynomial
- Two main algorithms
- Dominance rule based on automata
- A* and bidirectional variants
Bi-Objective Multimodal SPP: Algorithm and Dominance Rule

Impact of Dominance Rule for TLS-A

Graph: 60,000 nodes; 160,000 edges
CPU time: from 300 to 150 ms
Pareto dom.: speed-up around 38% vs TLS-A
State dom.: speed-up around 48% vs TLS-A

*without dominance*
Bi-Objective Multimodal SPP: Algorithm and Dominance Rule

Impact of Dominance Rule for TLS-A

- Graph: 60,000 nodes; 160,000 edges
- CPU time: from 300 to 150 ms
- Pareto dom.: speed-up around 38% vs TLS-A (∗ without dominance)
- State dom.: speed-up around 48% vs TLS-A (∗ without dominance)
Bi-Objective Multimodal SPP: Algorithm and Dominance Rule

\[ [s_0, 1, 18] \]

\[ x_1 \] \hspace{1cm} [s_0, 1, 22], Pareto dom.

\[ x_2 \] \hspace{1cm} [s_0, 1, 20] [s_0, 2, 17]

\[ s_0, 2, 15] \]

Impact of Dominance Rule for TLS-A

Graph: 60 000 nodes; 160 000 edges

CPU time: from 300 to 150 ms

Pareto dom.: speed-up around 38% vs TLS-A

State dom.: speed-up around 48% vs TLS-A
Bi-Objective Multimodal SPP: Algorithm and Dominance Rule

Impact of Dominance Rule for TLS-A

Graph: 60 000 nodes; 160 000 edges

CPU time: from 300 to 150 ms

Pareto dom.: speed-up around 38% vs TLS-A

State dom.: speed-up around 48% vs TLS-A

\[s_0, 1, 18\]
\[s_0, 1, 20\] \[s_0, 2, 17\]
\[s_2, 2, 17\], State dom

\[s_2, 2, 15\]
Impact of Dominance Rule for TLS-A*

- Graph: 60,000 nodes; 160,000 edges
- CPU time: from 300 to 150 ms
- Pareto dom.: speed-up around 38% vs TLS-A* without dominance
- State dom.: speed-up around 48% vs TLS-A* without dominance
Bi-Objective Multimodal SPP: Outcomes

Publications


- Gueye, Artigues, Huguet, Schettini, Dezou. Bi-objective multimodal time-dependent shortest viable path algorithms. *In Seven Triennial Symposium on Transportation Analysis (TRISTAN 2010)*, Tromso (Norway), June 20-24, 2010

DEMO

- MobiAnalyst Software
- PhD (CIFRE) - MOBIGIS
2-Way Multimodal SPP

- Minimize the total travel time
  - from \(o\) to \(d\) and \(d\) to \(o\)
  - through a synchronization point to be determined (parking)

2-Synchronisation Points SPP

- A driver and a pedestrian
- Minimize the arrival time for both user
  - through 2 synchronization points: pick-up and drop-off to be determined

Polynomial Problems

- Several Dijkstra algorithms
  - Enumeration of synchronization points (> 30 min for 80 parking nodes)
- Proposed Approach: several forward and backward algorithms (≈ 4s without limit on parking)
SPP with Synchronization Points

2-Way Multimodal SPP
- Minimize the total travel time
  - from $o$ to $d$ and $d$ to $o$
  - through a synchronization point to be determined (parking)

2-Synchronisation Points SPP
- A driver and a pedestrian
- Minimize the arrival time for both user
  - through 2 synchronization points: pick-up and drop-off to be determined

Polynomial Problems
- Several Dijkstra algorithms
  - Enumeration of synchronization points ($> 30$ min for 80 parking nodes)
- Proposed Approach: several forward and backward algorithms ($\approx 4s$ without limit on parking)
Solving the 2-Synchronisation Points SPP

The Best Origin Problem
- Set of origins and destinations
- Find the Best Origin for each destination
- Dijkstra algorithm in static case
Solving the 2-Synchronisation Points SPP

The Best Origin Problem

- Set of origins and destinations
- Find the Best Origin for each destination
- Dijkstra algorithm in static case

Dijkstra algorithm in static case

The Best Origin Problem
Sub-Problem: The Best Origin Problem

In Time-Dependent Case

- No consistency between costs and arrival times

\[(c = 41, t = 8:05)\]

\[(c = 39, t = 8:06)\]

Impact of heuristic dominance

- CPU time: decr. 91.6%; Sol cost: incr. 0.6%
Sub-Problem: The Best Origin Problem

In Time-Dependent Case

- No consistency between costs and arrival times

\[ (c = 41, t = 8:05) \]
\[ \odot_1 \quad 8:05 \rightarrow \Delta = 2 \]

\[ (c = 39, t = 8:06) \]
\[ \odot_2 \quad 8:06 \rightarrow \Delta = 2 \]

\[ (c = 41, t = 8:08) \]
Sub-Problem: The Best Origin Problem

In Time-Dependent Case

- No consistency between costs and arrival times

\[ (c = 41, t = 8:05) \]
\[ (c = 39, t = 8:06) \]
\[ (c = 43, t = 8:07) \]
\[ (c = 41, t = 8:08) \]

Multi-Label Dijkstra

Dominance rule: \((c_x, t_x)\) dominates \((c'_x, t'_x)\) if and only if:

- Exact:
  \[ t_x \leq t'_x \text{ and } c_x - c'_x \leq t_x - t'_x \]
- Heuristic:
  \[ t_x \leq t'_x \text{ and } c_x \leq c'_x \]

Impact of heuristic dominance

- CPU time: decrease 91.6%; Sol cost: increase 0.6%
Sub-Problem: The Best Origin Problem

In Time-Dependent Case

- No consistency between costs and arrival times

\[ (c = 41, t = 8:05) \]
\[ (c = 39, t = 8:06) \]

\[ (c = 41, t = 8:08) \]
\[ (c = 53, t = 8:20) \]
Sub-Problem: The Best Origin Problem

In Time-Dependent Case

- No consistency between costs and arrival times

\[
\begin{align*}
(c &= 41, t = 8:05) & \quad (c &= 43, t = 8:07) & \quad (c &= 52, t = 8:16) \\
(c &= 39, t = 8:06) & \quad (c &= 41, t = 8:08) & \quad (c &= 53, t = 8:20)
\end{align*}
\]
Sub-Problem: The Best Origin Problem

In Time-Dependent Case

- No consistency between costs and arrival times

\[ (c = 41, t = 8:05) \quad (c = 43, t = 8:07) \quad (c = 52, t = 8:16) \]

\[ (c = 39, t = 8:06) \quad (c = 41, t = 8:08) \quad (c = 53, t = 8:20) \]

Multi-Label Dijkstra

- Dominance rule: \((c_x, t_x)\) dominates \((c'_x, t'_x)\) if and only if:
  - **Exact:** \(t_x \leq t'_x\) and \(c_x - c'_x \leq t_x - t'_x\)
  - **Heuristic:** \(t_x \leq t'_x\) and \(c_x \leq c'_x\)

- Impact of heuristic dominance
  - CPU time: decr. 91.6%; Sol cost: incr. 0.6%
SPP with Synchronization Points: Outcomes

Publication


DEMO

- MuPaRo (Multi-Participant Routing)
- Open source (http://projects.laas.fr/MuPaRo/MuPaRo/)
Future works

- Alternative and Multi-Objective itineraries
- Synchronization points (multi-users) and Privacy preserving
- Robustness of itineraries in agile transportation networks