Prize-Collecting Steiner Travelling Salesman Problem with Time Windows

Stephen Page

7 September 2012
The Travelling Salesman Problem

Figure: A Complete Graph
The Travelling Salesman Problem

- Complete graph

Figure: A Complete Graph
The Travelling Salesman Problem

- Complete graph
- Cost for travelling along each arc

Figure: A Complete Graph
The Travelling Salesman Problem

- Complete graph
- Cost for travelling along each arc
- Cheapest route visiting each node exactly once

**Figure:** A Complete Graph
The PCSTSPTW Variant

**Figure:** An Incomplete Graph
The PCSTSPTW Variant

- Steiner

**Figure: An Incomplete Graph**
The PCSTSPTW Variant

- Steiner
  - Graph may not be complete

Figure: An Incomplete Graph
The PCSTSPTW Variant

- Steiner
  - Graph may not be complete
  - Set of customers

Figure: An Incomplete Graph
The PCSTSPTW Variant

- Steiner
  - Graph may not be complete
  - Set of customers
  - Multiple visits

Figure: An Incomplete Graph
The PCSTSPTW Variant

- Steiner
  - Graph may not be complete
  - Set of customers
  - Multiple visits

- With Time Windows

Figure: An Incomplete Graph
The PCSTSPTW Variant

- **Steiner**
  - Graph may not be complete
  - Set of customers
  - Multiple visits

- **With Time Windows**
  - Times for arcs and nodes

*Figure: An Incomplete Graph*
The PCSTSPTW Variant

- **Steiner**
  - Graph may not be complete
  - Set of customers
  - Multiple visits

- **With Time Windows**
  - Times for arcs and nodes
  - Return time

*Figure: An Incomplete Graph*
The PCSTSPTW Variant

- **Steiner**
  - Graph may not be complete
  - Set of customers
  - Multiple visits

- **With Time Windows**
  - Times for arcs and nodes
  - Return time
  - Salesman may wait

*Figure: An Incomplete Graph*
The PCSTSPTW Variant

- **Steiner**
  - Graph may not be complete
  - Set of customers
  - Multiple visits

- **With Time Windows**
  - Times for arcs and nodes
  - Return time
  - Salesman may wait

- **Prize-Collecting**

*Figure: An Incomplete Graph*
The PCSTSPTW Variant

- Steiner
  - Graph may not be complete
  - Set of customers
  - Multiple visits

- With Time Windows
  - Times for arcs and nodes
  - Return time
  - Salesman may wait

- Prize-Collecting
  - Prize for servicing
The PCSTSTPW Variant

- **Steiner**
  - Graph may not be complete
  - Set of customers
  - Multiple visits

- **With Time Windows**
  - Times for arcs and nodes
  - Return time
  - Salesman may wait

- **Prize-Collecting**
  - Prize for servicing
  - Profit minus cost

*Figure: An Incomplete Graph*
Formulation for the PCSTSPTW

A mixed 0-1 linear programming formulation for the Steiner travelling salesman problem with time windows (Letchford et al., 2012) can be adapted to the PCSTSPTW.

The following variables are used:
Formulation for the PCSTSP-TW

A mixed 0-1 linear programming formulation for the Steiner travelling salesman problem with time windows (Letchford et al., 2012) can be adapted to the PCSTSP-TW.

The following variables are used:

- $\tilde{x}_a^k$ is a binary variable which is 1 if and only if arc $a$ is traversed after exactly $k$ customers have been serviced.
### Formulation for the PCSTSPTW

A mixed 0-1 linear programming formulation for the Steiner travelling salesman problem with time windows (Letchford et al., 2012) can be adapted to the PCSTSPTW.

The following variables are used:

- $\tilde{x}^k_a$ is a binary variable which is 1 if and only if arc $a$ is traversed after exactly $k$ customers have been serviced.
- $y^k_i$ is a binary variable which is 1 if and only if customer $i$ is the $k$th customer to be serviced.
A mixed 0-1 linear programming formulation for the Steiner travelling salesman problem with time windows (Letchford et al., 2012) can be adapted to the PCSTSPTW.

The following variables are used:

- $\tilde{x}^k_a$ is a binary variable which is 1 if and only if arc $a$ is traversed after exactly $k$ customers have been serviced.
- $y^k_i$ is a binary variable which is 1 if and only if customer $i$ is the $k$th customer to be serviced.
- $g^k_a$ is the total time elapsed when the salesman begins to traverse arc $a$ after exactly $k$ customers have been serviced if this time exists and is 0 otherwise.
Example Data

Table: Customer Data

<table>
<thead>
<tr>
<th>Node</th>
<th>Prize</th>
<th>Time Window</th>
<th>Service Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>6</td>
<td>[2, 4]</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>11</td>
<td>[1, 12]</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>[10, 12]</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>[10, 20]</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>[2, 7]</td>
<td>2</td>
</tr>
</tbody>
</table>

Table: Arc Data

<table>
<thead>
<tr>
<th>Arc</th>
<th>Cost</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,2)</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>(1,8)</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>(1,9)</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>(2,3)</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Arc</th>
<th>Cost</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>(3,9)</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>(3,10)</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>(4,5)</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>(4,8)</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Arc</th>
<th>Cost</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>(5,6)</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>(6,7)</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>(7,10)</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
The optimal route is 1 - 2 - 3 - 9 - 1 - 8 - 4 - 8 - 1 servicing customers 3, 9 and 4 with objective value 5.
What Are Vehicle Routing Problems?
What Are Vehicle Routing Problems?

- Similar to TSP
What Are Vehicle Routing Problems?

- Similar to TSP
- Multiple vehicles
What Are Vehicle Routing Problems?

- Similar to TSP
- Multiple vehicles
- Deliver goods
What Are Vehicle Routing Problems?

- Similar to TSP
- Multiple vehicles
- Deliver goods
- Can be time-constrained
Solving Time-Constrained VRPs
Solving Time-Constrained VRPs

- Methods assume cheapest paths are quickest
Solving Time-Constrained VRPs

- Methods assume cheapest paths are quickest
- Use this to complete graph
Solving Time-Constrained VRPs

- Methods assume cheapest paths are quickest
- Use this to complete graph
- Doesn’t work for STSPTW (Nasiri et al., 2012)
Solving Time-Constrained VRPs

- Methods assume cheapest paths are quickest
- Use this to complete graph
- Doesn’t work for STSPTW (Nasiri et al., 2012)
- Doesn’t work for PCSTSPTW either
Cheapest Paths for the Example

<table>
<thead>
<tr>
<th>Arc</th>
<th>Cost</th>
<th>Time</th>
<th>Arc</th>
<th>Cost</th>
<th>Time</th>
<th>Arc</th>
<th>Cost</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,3)</td>
<td>2</td>
<td>5</td>
<td>(3,4)</td>
<td>7</td>
<td>7</td>
<td>(4,7)</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>(1,4)</td>
<td>5</td>
<td>2</td>
<td>(3,6)</td>
<td>8</td>
<td>6</td>
<td>(4,9)</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>(1,6)</td>
<td>10</td>
<td>11</td>
<td>(3,7)</td>
<td>5</td>
<td>3</td>
<td>(6,7)</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>(1,7)</td>
<td>7</td>
<td>8</td>
<td>(3,9)</td>
<td>1</td>
<td>2</td>
<td>(6,9)</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>(1,9)</td>
<td>1</td>
<td>3</td>
<td>(4,6)</td>
<td>6</td>
<td>4</td>
<td>(7,9)</td>
<td>6</td>
<td>5</td>
</tr>
</tbody>
</table>
Solution Given for Completed Graph

The route given is 1 - 9 - 1 - 4 - 1 which corresponds to the route 1 - 9 - 1 - 8 - 4 - 8 - 1 on the original graph with objective value 4.
What is Dynamic Programming?

- If a route between two states is optimal then so are all subroutes.
- Can find optimal way to get to some state by looking at possible previous states.
What is Dynamic Programming?

- If a route between two states is optimal then so are all subroutes
What is Dynamic Programming?

- If a route between two states is optimal then so are all subroutes
What is Dynamic Programming?

- If a route between two states is optimal then so are all subroutes.
What is Dynamic Programming?

- If a route between two states is optimal then so are all subroutes
What is Dynamic Programming?

- If a route between two states is optimal then so are all subroutes
What is Dynamic Programming?

- If a route between two states is optimal then so are all subroutes.
- Can find optimal way to get to some state by looking at possible previous states.
What is Dynamic Programming?

- If a route between two states is optimal then so are all subroutes.
- Can find optimal way to get to some state by looking at possible previous states.
What is Dynamic Programming?

- If a route between two states is optimal then so are all subroutes.
- Can find optimal way to get to some state by looking at possible previous states.
What is Dynamic Programming?

- If a route between two states is optimal then so are all subroutes.
- Can find optimal way to get to some state by looking at possible previous states.
What is Dynamic Programming?

- If a route between two states is optimal then so are all subroutes.
- Can find optimal way to get to some state by looking at possible previous states.
What is Dynamic Programming?

- If a route between two states is optimal then so are all subroutes.
- Can find optimal way to get to some state by looking at possible previous states.
What is Dynamic Programming?

- If a route between two states is optimal then so are all subroutes.
- Can find optimal way to get to some state by looking at possible previous states.
What is Dynamic Programming?

- If a route between two states is optimal then so are all subroutes.
- Can find optimal way to get to some state by looking at possible previous states.
What is Dynamic Programming?

- If a route between two states is optimal then so are all subroutes.
- Can find optimal way to get to some state by looking at possible previous states.
What is Dynamic Programming?

- If a route between two states is optimal then so are all subroutes.
- Can find optimal way to get to some state by looking at possible previous states.
A dynamic programming algorithm for the STSPTW (Nasiri et al., 2012) can be adapted to the PCSTSPTW.

The following notation is used:

- $t_a$ is the time required to traverse arc $a$.
- $c_a$ is the cost of traversing arc $a$.
- $[a_i, b_i]$ is the time window in which the servicing of customer $i$ may begin.
- $s_i$ is the time required to service customer $i$.
- $p_i$ is the prize collected for servicing customer $i$.
- $T$ is the time in which the salesman must return to the depot.
Algorithm for the PCSTSPTW

\( f(S, i, t) \) is the maximum profit minus cost of a path servicing the customers in \( S \) leaving the depot at time 0 and reaching node \( i \) at time \( t \leq T \) an integer, or \(-\infty\) if there is no such path.

\( f(S, i, t) \) is the maximum of \(-\infty\) and:

- \( f(S, i, t - 1) \) if \( t \geq 1 \).
- \( \max_{a \in \delta^{-}(i): t \geq t_a} \{ f(S, i, t - t_a) - c_a \} \) if \( \{ a \in \delta^{-}(i) : t \geq t_a \} \neq \emptyset \).
- \( f(S \setminus \{ i \}, i, t - s_i) + p_i \) if \( i \in S \) and \( t \in [a_i + s_i, b_i + s_i] \).

The optimal solution has profit minus cost equal to
\( \max_{S \in V_R} \{ f(S, 1, \lfloor T \rfloor) \} \) and the route can be found by considering the previous step in the iteration.
Comparison of Solution Methods
Comparison of Solution Methods

- **Time**

  - LP solved in MPL in 35.63 seconds using CoinMP
  - DP solved in R in 4.20 seconds
  - Adaptable: Can vary data for the problem with time using DP
  - Assumes integer time
Comparison of Solution Methods

- **Time**
  - LP solved in MPL in 35.63 seconds using CoinMP

Assumptions:
- DP assumes integer time
Comparison of Solution Methods

- **Time**
  - LP solved in MPL in 35.63 seconds using CoinMP
  - DP solved in R in 4.20 seconds
Comparison of Solution Methods

- **Time**
  - LP solved in MPL in 35.63 seconds using CoinMP
  - DP solved in R in 4.20 seconds

- **Adaptability**
Comparison of Solution Methods

- **Time**
  - LP solved in MPL in 35.63 seconds using CoinMP
  - DP solved in R in 4.20 seconds

- **Adaptability**
  - Can vary data for the problem with time using DP
Comparison of Solution Methods

- **Time**
  - LP solved in MPL in 35.63 seconds using CoinMP
  - DP solved in R in 4.20 seconds

- **Adaptability**
  - Can vary data for the problem with time using DP
  - This includes times, prizes and customers
Comparison of Solution Methods

- **Time**
  - LP solved in MPL in 35.63 seconds using CoinMP
  - DP solved in R in 4.20 seconds

- **Adaptability**
  - Can vary data for the problem with time using DP
  - This includes times, prizes and customers

- **Assumptions**
Comparison of Solution Methods

- **Time**
  - LP solved in MPL in 35.63 seconds using CoinMP
  - DP solved in R in 4.20 seconds

- **Adaptability**
  - Can vary data for the problem with time using DP
  - This includes times, prizes and customers

- **Assumptions**
  - DP assumes integer time
Questions