Vehicle Routing With Time-Windows

Presentation

Anshul Sawant[1] Catalin-Stefan Tiseanu[2]

[1]University of Maryland,
asawant@cs.umd.edu

[2]University of Maryland,
ctiseanu@cs.umd.edu

7th of December 2011

Outline

1. Introduction
2. Point-to-Point Orienteering
3. Routing with Time-windows

Problem Definition

- We are given a metric $G = (V, E)$ and a reward on each node. The objective is to collect as much reward as possible by visiting a node after its release time and before its deadline.
- When all release times are zero, the problem is called Deadline-TSP.

Problem Definition

- We are given a metric $G = (V, E)$ and a reward on each node. The objective is to collect as much reward as possible by visiting a node after its release time and before its deadline.
- When all release times are zero, the problem is called Deadline-TSP.

Notation

- Let $G = (V, E)$ be the graph. Then r_i is the reward, R_i is the release time and D_i is deadline for node i.
- D_{max} is the maximum deadline.

Notation

- Let $G = (V, E)$ be the graph. Then r_i is the reward, R_i is the release time and D_i is deadline for node i.
- D_{max} is the maximum deadline.

Related Problems

- **k-TSP**. Visit k nodes on a graph. Minimize total distance travelled. 2-approx by **Garg**.
- **Minimum Excess Path**. Visit k cities on a path P from s to t, minimize the difference $d_P(s,t) - d(s,t)$. $2+\epsilon$ approximation by **Blum et al.**. Uses k-TSP as a sub-routine.
- **Point-to-Point Orienteering**. Find a path from s to t with $d_P(s,t) \leq D$ which maximizes the number of cities visited. Uses Minimum Excess Path as a sub-routine.

Related Problems

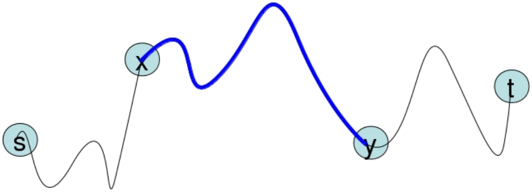
- **k-TSP**. Visit k nodes on a graph. Minimize total distance travelled. 2-approx by **Garg**.
- **Minimum Excess Path**. Visit k cities on a path P from s to t, minimize the difference $d_P(s,t) - d(s,t)$. $2+\epsilon$ approximation by **Blum et al.**. Uses k-TSP as a sub-routine.
- **Point-to-Point Orienteering**. Find a path from s to t with $d_P(s,t) \leq D$ which maximizes the number of cities visited. Uses Minimum Excess Path as a sub-routine.

Related Problems

- **k-TSP**. Visit k nodes on a graph. Minimize total distance travelled. 2-approx by **Garg**.
- **Minimum Excess Path**. Visit k cities on a path P from s to t, minimize the difference $d_P(s,t) - d(s,t)$. $2+\epsilon$ approximation by **Blum et al.**. Uses k-TSP as a sub-routine.
- **Point-to-Point Orienteering**. Find a path from s to t with $d_P(s,t) \leq D$ which maximizes the number of cities visited. Uses Minimum Excess Path as a sub-routine.

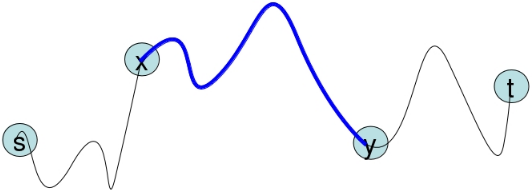
Orienteering Intuition

- Let reward be evenly spread among segments $s - x$, $x - y$ and $y - t$.
- In one of the segments, we can increase the excess 3 times if we minimize excess in the other segments.
- If we have a 3-approximation for min-excess problem we can get all the reward from segment $x - y$.



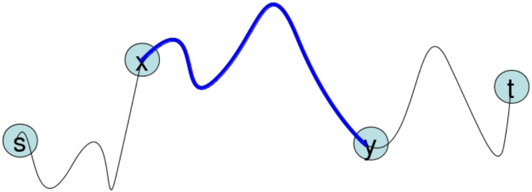
Orienteering Intuition

- Let reward be evenly spread among segments $s - x$, $x - y$ and $y - t$.
- In one of the segments, we can increase the excess 3 times if we minimize excess in the other segments.
- If we have a 3-approximation for min-excess problem we can get all the reward from segment $x - y$.



Orienteering Intuition

- Let reward be evenly spread among segments $s-x$, $x-y$ and $y-t$.
- In one of the segments, we can increase the excess 3 times if we minimize excess in the other segments.
- If we have a 3-approximation for min-excess problem we can get all the reward from segment $x-y$.



Algorithm

- For each pair of nodes (x, y) and value k, we compute a minimum excess path from x to y which visits k nodes.
- We then select the triple (x, y, k) with the maximum k, such that the computed path has excess $D - d(s, x) - d(x, y) - d(y, t)$ or smaller.
- We return the path which travels from s to x via shortest path, then x to y via the computed path, then y to t via shortest path.

Algorithm

- For each pair of nodes (x, y) and value k, we compute a minimum excess path from x to y which visits k nodes.
- We then select the triple (x, y, k) with the maximum k, such that the computed path has excess $D - d(s, x) - d(x, y) - d(y, t)$ or smaller.
- We return the path which travels from s to x via shortest path, then x to y via the computed path, then y to t via shortest path.

Algorithm

- For each pair of nodes (x, y) and value k, we compute a minimum excess path from x to y which visits k nodes.
- We then select the triple (x, y, k) with the maximum k, such that the computed path has excess $D - d(s, x) - d(x, y) - d(y, t)$ or smaller.
- We return the path which travels from s to x via shortest path, then x to y via the computed path, then y to t via shortest path.

Time-windows Intuition

- If we can start after release time, we don't have to worry about release time. Problem is reduced to deadline-TSP.
- If some nodes above a deadline d are captured by time d, then we can collect the same reward by reducing deadlines of all such nodes to d.
- If deadlines are constant for all vertices, then deadline-TSP can be solved using point-to-point orienteering.

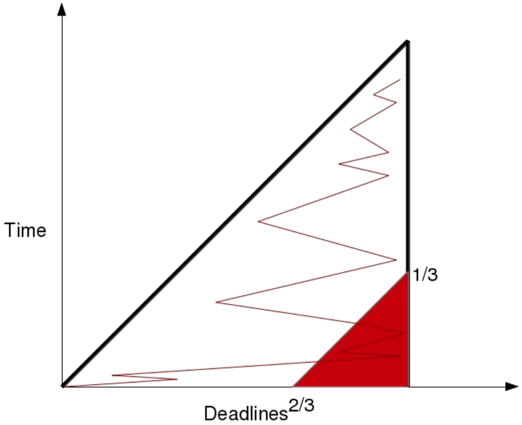
Time-windows Intuition

- If we can start after release time, we don't have to worry about release time. Problem is reduced to deadline-TSP.
- If some nodes above a deadline d are captured by time d, then we can collect the same reward by reducing deadlines of all such nodes to d.
- If deadlines are constant for all vertices, then deadline-TSP can be solved using point-to-point orienteering.

Time-windows Intuition

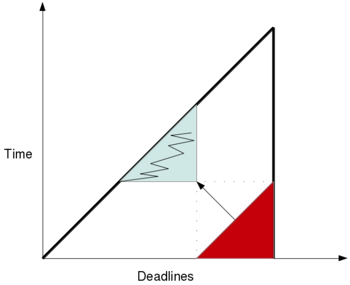
- If we can start after release time, we don't have to worry about release time. Problem is reduced to deadline-TSP.
- If some nodes above a deadline d are captured by time d, then we can collect the same reward by reducing deadlines of all such nodes to d.
- If deadlines are constant for all vertices, then deadline-TSP can be solved using point-to-point orienteering.

Optimal Path



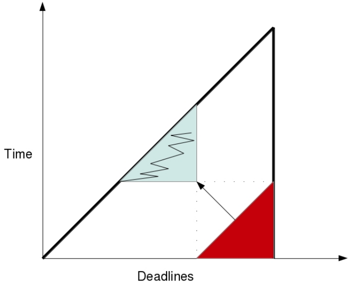
A Path That Does Well

- Starts after release time of nodes in red area.
- Ends at at time equal to least deadline in the red area.
- Gets a constant factor of reward by using Point-to-Point Orienteering.



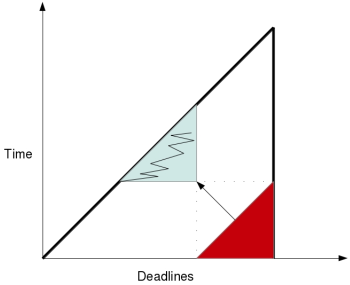
A Path That Does Well

- Starts after release time of nodes in red area.
- Ends at at time equal to least deadline in the red area.
- Gets a constant factor of reward by using Point-to-Point Orienteering.



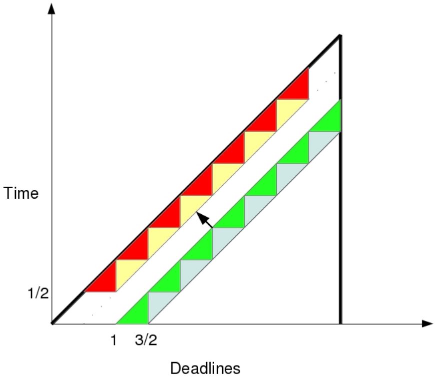
A Path That Does Well

- Starts after release time of nodes in red area.
- Ends at at time equal to least deadline in the red area.
- Gets a constant factor of reward by using Point-to-Point Orienteering.



Partitioning the Time-Deadline Space

- We want to get a constant factor of reward from each segment using DP.



Outline of Algorithm for a Segment

- Let the segment start at deadline d and let its width be k.
- For i = 0 to j such that $d + k * j \leq D_{max}$
  - Let S_i be the set of all vertices with deadlines in $[d + i * k, d + (i + 1) * k)$ and release times $\leq (i + 1) * k$.
  - Assign all vertices in $V - S_i$ a reward of 0. Let other rewards be the same as in original instance.
  - With the above reward assignment, let $S(G, x, y, l)$ be the approximate solution to p2p orienteering problem on graph G, with path length l, starting point x and end point y.
  - $A[i, x, y, l] = S(G, x, y, l)\ \forall x, y \in S_i,\ \forall l \in \{1, .., k\}$

Outline of Algorithm for a Segment

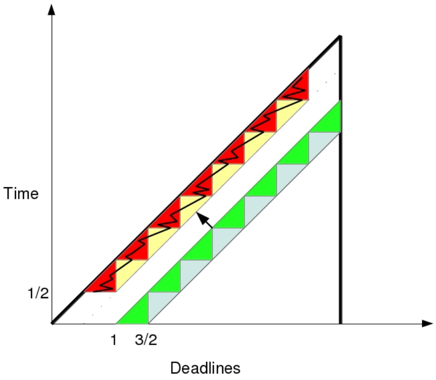
- Let the segment start at deadline d and let its width be k.
- For i = 0 to j such that $d + k * j \leq D_{max}$
  - Let S_i be the set of all vertices with deadlines in $[d + i * k, d + (i + 1) * k)$ and release times $\leq (i + 1) * k$.
  - Assign all vertices in $V - S_i$ a reward of 0. Let other rewards be the same as in original instance.
  - With the above reward assignment, let $S(G, x, y, l)$ be the approximate solution to p2p orienteering problem on graph G, with path length l, starting point x and end point y.
  - $A[i, x, y, l] = S(G, x, y, l)\ \forall x, y \in S_i,\ \forall l \in \{1, .., k\}$

Outline of Algorithm for a Segment

- Let the segment start at deadline d and let its width be k.
- For i = 0 to j such that $d + k * j \leq D_{max}$
  - Let S_i be the set of all vertices with deadlines in $[d + i * k, d + (i + 1) * k)$ and release times $\leq (i + 1) * k$.
  - Assign all vertices in $V - S_i$ a reward of 0. Let other rewards be the same as in original instance.
  - With the above reward assignment, let $S(G, x, y, l)$ be the approximate solution to p2p orienteering problem on graph G, with path length l, starting point x and end point y.
  - $A[i, x, y, l] = S(G, x, y, l)\ \forall x, y \in S_i,\ \forall l \in \{1, .., k\}$

Outline of Algorithm for a Segment

- Let the segment start at deadline d and let its width be k.
- For i = 0 to j such that $d + k * j \leq D_{max}$
  - Let S_i be the set of all vertices with deadlines in $[d + i * k, d + (i + 1) * k)$ and release times $\leq (i + 1) * k$.
  - Assign all vertices in $V - S_i$ a reward of 0. Let other rewards be the same as in original instance.
  - With the above reward assignment, let $S(G, x, y, l)$ be the approximate solution to p2p orienteering problem on graph G, with path length l, starting point x and end point y.
  - $A[i, x, y, l] = S(G, x, y, l)\ \forall x, y \in S_i,\ \forall l \in \{1, .., k\}$


Time
1/2
1
3/2
Deadlines