Approximation Algorithms for Conflict-Free Vehicle Routing

Kaspar Schupbach and Rico Zenklusen

Παπαηλίου Νικόλαος

CFVRP Problem

- Undirected graph of stations and roads
- Vehicles(k):
  - Source-Destination stations
- Discretized time
  - At each timestep every vehicle waits to the current position or moves to a neighbor station
- Conflicts:
  - No vehicles traverse the same edge at the same timestep
  - No vehicles are on the same station at a certain timestep
- Goal:
  - Conflict-free routing with minimum makespan(total routing time)

Sequential Routing Approaches

- Simple approach:
  - Sequentially send one vehicle after another on the shortest path to its destination
  - Makespan: O(k*L)
  - L: maximum s-t distance for vehicles
  - L<=OPT
  - O(k)-approximation
  - No efficient algorithm substantially beats this approach

Sequential Routing Approaches

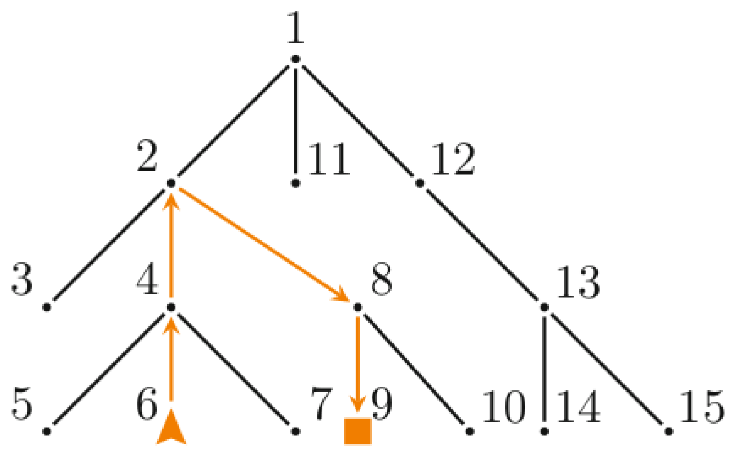
- Improved approach:
  - Greedy direct sequential routing
  - Greedy: Consider the vehicle in a given order
  - Direct: Vehicles never stop while advancing to their destination
  - For each vehicle find the earliest departure time that has no conflict with previously routed vehicles.
  - No theoretical improvements
  - O(k)-approximation

Complexity

- CFVRP is NP-hard even on paths
- Choosing a good ordering for greedy direct routing is also NP-hard
- Sub-linear in k approximation algorithms are known for grids
  - Takes advantage of the existence of two disjoint alternative paths for each s-t path
- This paper presents:
  - 4OPT+k approximation for trees
  - $O(\sqrt{k})$-approximation for general graphs
  - $O(\log^3(k))OPT+k$ randomized approximation algorithm based on tree embeddings

Tree Approximation

- DFS numbering on the tree nodes
- Increasing- Decreasing vehicles:
  - Increasing if label of destination is larger than the label of origin
  - Bending node: the node of the path that is closer to the root
  - in-label: last node before the bending node
  - out-label: first node after the bending node



Tree Approximation

- Sort vehicles using the following priorities:
  - Increasing vehicles have priority over decreasing ones
  - Among two increasing vehicles the higher out-label has priority
  - Among two decreasing vehicles the lower in-label has priority
  - Ties are broken using an arbitrary fixed vehicle ordering
- Apply greedy sequential routing using the above ordering
  - Makespan: 4L+k

Proof

- There exists a direct routing with at most 4L+k makespan
  - k^+: number of increasing vehicles
  - k^- : number of decreasing vehicles
- Examine vehicles using the ordering
  - The first vehicle has passage time from bending node: L
  - The second: L+1
  - ...
  - The last increasing: $L+k^+-1$
  - If this is conflict free all increasing vehicles can be routed with: $2L+k^+-1$
  - The same can be done for the decreasing leading to total makespan 4L+k

Proof

- We will show that the previous routing is conflict-free
- Let π, ψ be two vehicles and ψ has higher priority
  - Case 1: π, ψ don't share any node: no conflict
  - Case2: π, ψ share only one node v
    - v is the bending node of at least one of π, ψ
    - ψ passes first from v
  - Case 3: π, ψ use common subpath in the same direction
    - v the smallest node in the subpath
    - v is the bending node of at least one of π, ψ
    - ψ passes first from v
  - Case 4: π, ψ use common subpath in opposite directions
    - v the smallest node in the subpath
    - π, ψ can't bend in the subpath(increasing-decreasing)
    - ψ passes first from v
    - ψ leaves common path before π enters it

Hot Spot Routing

- General graphs
  - Congestion: maximum number of vehicles that pass from a node
  - Dilation: length of the longest path
  - Congestion, dilation = O(OPT)
- Generate paths with low congestion and dilation
  - Use of Sinivasan and Teo algorithm
- For each v if there are more than $\sqrt{k}$ vehicles not routed that pass from v
  - Find the shortest path tree routed at v
  - Use TreeRouting
- Route remaining vehicles using greedy direct sequential routing

Hot Spot Routing

- Approximation $O\left(\sqrt{k}OPT\right)$
- At most $\sqrt{k}$ TreeRouting steps
  - Each TreeRouting takes O(C+D)
  - The first phase is $O\left(\sqrt{k}OPT\right)$
- Second phase(greedy routing)
  - π any of the remaining vehicles(not routed in the first phase)
  - For every node in the path of π there are at most $\sqrt{k}$ previous routed vehicles.
  - This routing can stall π at most O(D$\sqrt{k}$)
  - The second phase is $O\left(\sqrt{k}OPT\right)$

Hot Spot Routing

- Approximation $O\left(\sqrt{k}\,OPT\right)$
- At most $\sqrt{k}$ TreeRouting steps
  - Each TreeRouting takes O(C+D)
  - The first phase is $O\left(\sqrt{k}\,OPT\right)$
- Second phase(greedy routing)
  - π any of the remaining vehicles(not routed in the first phase)
  - For every node in the path of π there are at most $\sqrt{k}$ previous routed vehicles.
  - This routing can stall π at most O(D$\sqrt{k}$)
  - The second phase is $O\left(\sqrt{k}\,OPT\right)$

Low-Strech Routing

- Find a collection of $O(polylog(k))$ trees such that
  - each s-t path in T is at most a $O(polylog(k))$-factor larger than the shortest path in G
  - Assign vehicles to trees
  - Use TreeRouting for each tree
- Randomized algorithm to find trees
  - Transform G=(V,E) to H(W,F) with size $O(k^2)$
  - Each vehicle has the same s-t distance on both graphs
  - Delete all nodes, edges of G that don't belong to shortest s-t paths
  - Every path of G is replaced by an edge in H if it doesn't contain another node of H
  - A random spanning tree of H has:

$$\mathbf{E}\left[d_{H[T]}(v,w)/d_H(v,w)\right] = O\left(\log^2|W|\right)$$

Low-Strech Routing

- Select p=2log(k) random spanning trees of H
- Find the respective trees (T) of G
- With probability 1-1/k there exists one tree T such that:

$$d_{G[T]}(s_\pi, t_\pi)/d_G(s_\pi, t_\pi) = O(\log^2 k)$$

- Each TreeRouting needs a makespan of:

$$4 \max\{d_{G[T_j]}(s_\pi, t_\pi) \mid \pi \in \Pi, i(\pi) = j\} + k_j = O(\log^2 k)L + k_j$$

- The total makespan is:

$$O(\log^2 k)pL + \sum_{i=1}^{p} k_j = O(\log^3 k)L + k$$

Ερωτήσεις?