Multi-path route guidance with walking distance integration

Antonis F. Lentzakis, Chunyang Sun

OCT 2018

NANYANG TECHNOLOGICAL UNIVERSITY

SECURE CONNECTIONS FOR A SMARTER WORLD

Motivation

- Travelers unfamiliar with an area are frequently unaware of route options available
- Familiar travelers possess only limited route options knowledge
- Congestion occurrence can be unpredictable (incidents)
- Recurrent congestion also contains randomness due to variability in travel demand levels and network performance

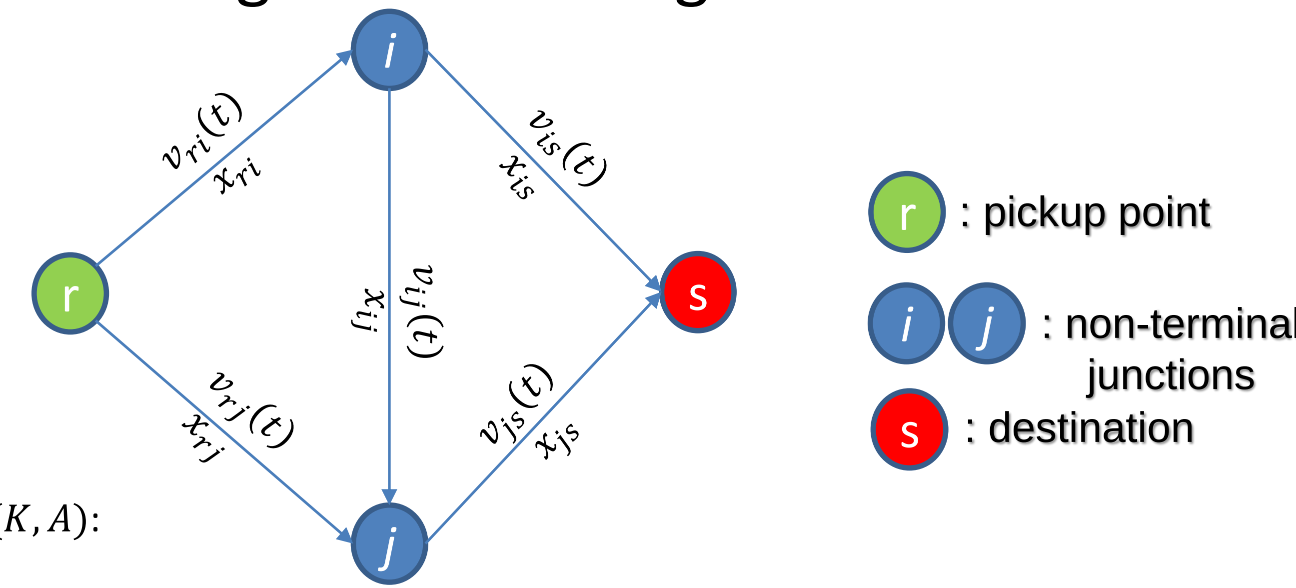
Goals

- Up-to-date information enables travelers to make improved route choice decisions
- Reduction of individual travelers' travel times (**user oriented approach**)
- Better utilization of existing infrastructure

Shortest Path Algorithms for Route Planning

- Basic Techniques (Dijkstra, Bellman-Ford, Floyd-Warshall)
- Goal-Directed Techniques (A* Search, Geometric Containers, Arc Flags, Precomputed Cluster Distances, Compressed Path Databases)
- Separator-Based Techniques (Vertex/Arc Separators)
- Hierarchical Techniques (Contraction hierarchies, Reach as centrality measure on vertices)
- Bounded-Hop Techniques (Labeling Algorithms, Transit Node Routing, Pruned Highway Labeling

Speed-profile dependent, Dijktra-based, SP algorithm with walking distance integration



Where, for a directed graph $G = (K, A)$:

- K : set of junctions
- A : set of links
- $r \in K_o$: pickup point r belonging to set of origin region junctions K_o
- $s \in K_d$: “true” destination s belonging to set of destination region junctions K_d
- $i, j \in K | (i, j) \in \mathrm{A}$: neighboring junctions i and j
- $v_{ij}(t) = a_n t^n + a_{n-1} t^{n-1}$+…+$a_1 t + a_0$: a polynomial approximation of the Link Speed Profile Function at time t for link between neighboring junctions i and j
- $D = [d_{ij}]^{|K| \times |K|}$: the adjacency matrix for graph G, where $\mathrm{d}_{ij} = 1$ if $(i, j) \in A$, 0 otherwise
- x_{ij}: the geographical distance between neighboring junctions i and j
- t_{ij}: travel time on link between neighboring junctions i and j

NANYANG TECHNOLOGICAL UNIVERSITY

NXP SECURE CONNECTIONS FOR A SMARTER WORLD

(1) Speed Profile Function Extension to Dijkstra's Algorithm

Input:

- Network Topology (adjacency matrix *D*)
- $(\forall i, j \in K | (i, j) \in A)\ v_{ij}(t), x_{ij}$

Steps:

For all paths $p \coloneqq \{r, k_1, \ldots, k_w, s)|[\forall l \in \{1, \ldots, w\}, k_l \in K \backslash \{r, s\}] \wedge (r, k_1) \in A \wedge (k_w, s) \in A \wedge [\forall m \in \{1, \ldots, w-1\}, (k_m, k_{m+1}) \in A] \wedge l \neq m \rightarrow k_l \neq k_m\}$

1. Set pickup point *r* as current and place all other junctions in the unvisited set.
2. For origin *r* set as current, calculate tentative travel times for all neighboring junctions
    - $t_{rk_1} \leftarrow \int_0^{t_{rk_1}} v_{rk_1}(\tau)d\tau - x_{rk_1}$

Compare the newly calculated tentative travel times to the current assigned value and assign the minimum

3. For all subsequent junctions, other than pickup point *r* calculate the tentative travel times as follows:

- **if** k_l=k_1

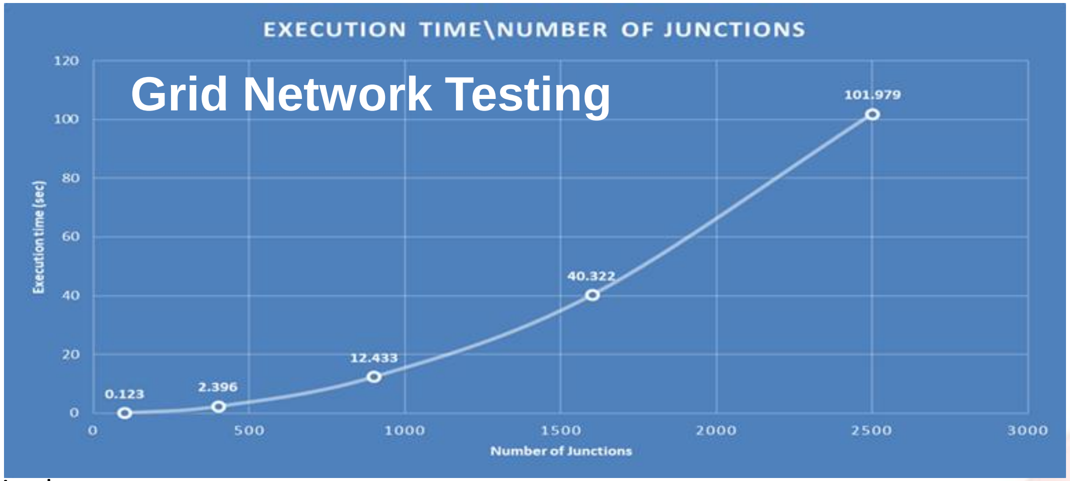
$(\forall k_m | (k_l, k_m) \in A)\ \ t_{k_l k_m} \leftarrow \int_{t_{rk}}^{t_{k_l k_m}} v_{k_l k_m}(\tau)d\tau - x_{k_l k_m}$

- **if** $k_m \neq\ k_1$ && $k_m \neq k_w$

$(\forall k_l | (k_l, k_m) \in A)\ \ t_{k_l k_m} \leftarrow \int_{t_{k_{l-1} k_l}}^{t_{k_l k_m}} v_{k_l k_m}(\tau)d\tau - x_{k_l k_m}$

- **if** k_m=k_w

$(\forall k_m | (k_m, s) \in A)\ \ t_{k_m s} \leftarrow t_{k_{m-1} k_m} + \frac{x_{k_m s}}{v_{walk}}$



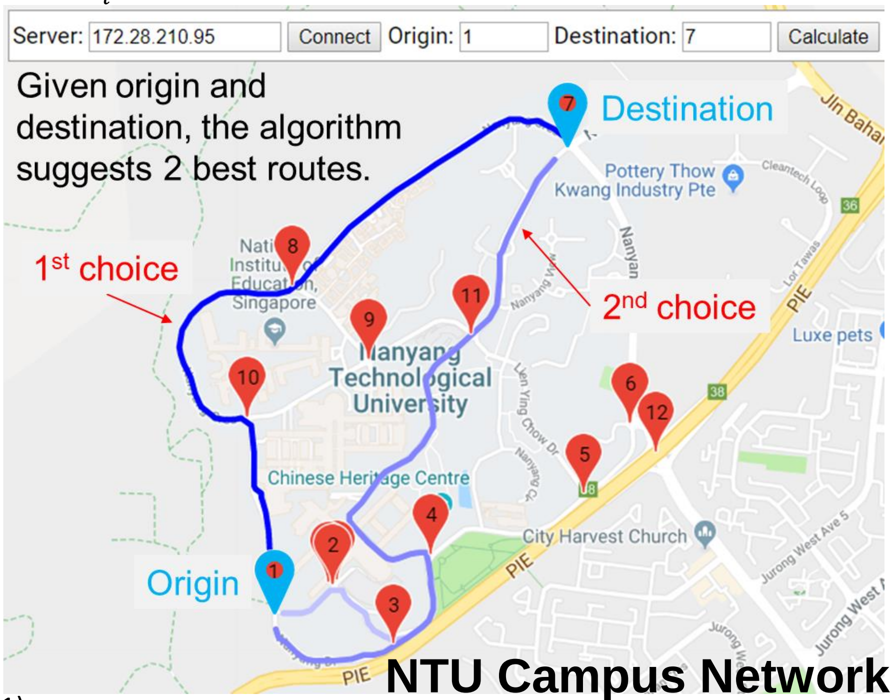
When we are done considering all of the neighboring junctions, mark the current waypoint as visited and remove from the unvisited set.

4. If the "true" destination *s* has been marked visited then stop. Otherwise select the neighboring node with the minimum tentative travel time, set as new current junction and go back to step 3.

Assuming walking speed $v_{walk} = 5\ km/h$, we designate the last non-terminal node k_m in our path as a drop-off point

(2) Speed Profile Function Extension to Yen's Algorithm for Multi-Path Route Guidance

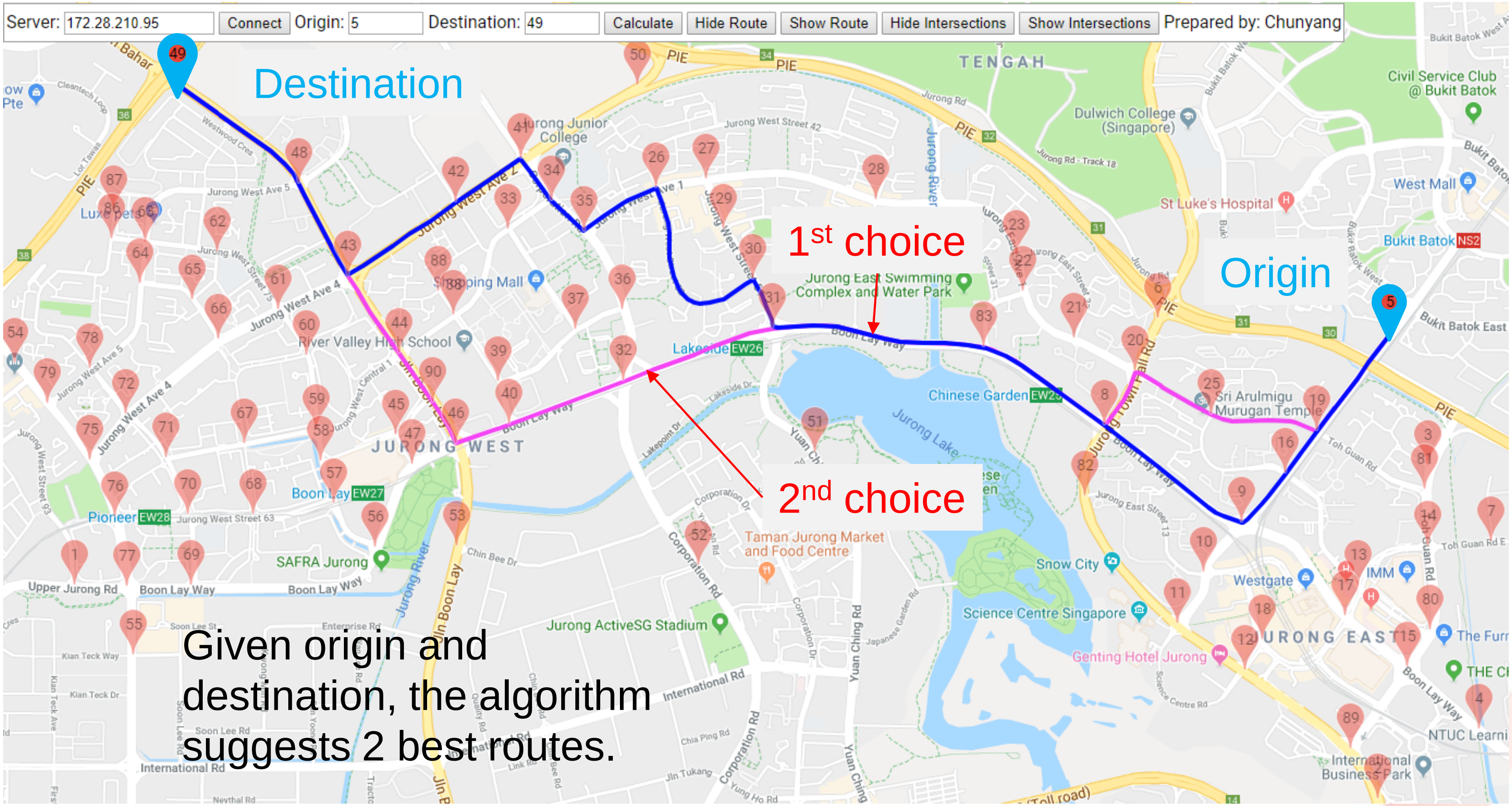
- $p^q := \{r, k_1^q, \ldots, k_w^q, s\}$ representing a shortest path
- d_i^q representing the deviation from k^{q-1} at junction k_i^{q-1}
- $e_i^q := \{r, k_1^q, \ldots, k_i^q\}, f_i^q := \{k_{i+1}^q, \ldots, k_w^q, s\}$, representing the root and spur of d_i^q respectively



Steps:

- Find shortest path p_1 derived from Algorithm (1).
- For $q = 2, 3, \ldots$, find k^q as follows
  1. Let $B^q = B^{q-1}$, the set of candidate paths from iteration q-1
  2. For $1 \leq i \leq |p^{q-1}|$ **do**
     - Let $y = k_i^{q-1}$
     - Hide incoming edges to y for the remainder of iteration q
     - For each z s.t. the first i junctions in p^z match p^{q-1} **do**
       - Hide edge (y, k_{i+1}^z) for remainder of iteration q
     - e_i^q are the first i junctions of k^{q-1}
     - f_i^q is the shortest path from y to s, derived by algorithm (1)
     - Concatenate e_i^q and f_i^q to form d_i^q
     - Add candidate path d_i^q to B^q
     - Return the shortest path from B^q

Multi-path route guidance for Jurong area network



Given origin and destination, the algorithm suggests 2 best routes.

Concluding Remarks

- A multi-path route guidance system with walking distance integration was developed
- The part of multi-path path calculation has already been tested on NTU campus and Jurong area network

Future Considerations

- Walking distance could be determined using Manhattan distance rather than Euclidean distance for practical applications

Video Links

- https://www.youtube.com/watch?v=3mnf5J4oB8I
- https://www.youtube.com/watch?v=tRP420yvqD0

NANYANG TECHNOLOGICAL UNIVERSITY

SECURE CONNECTIONS FOR A SMARTER WORLD