A Greedy Path-Based Algorithm for Traffic Assignment

Jun Xie, Yu Nie, Xiaobo Liu

March 13, 2023

Introduction

Setup and Notation

- $G(N, A)$ – strongly connected directed transportation network. N set of nodes, A set of links.
- $R, S \subseteq N$, collection of origin and destination nodes.
- d_{rs}, the travel demand between $r \in R$ and $s \in S$.
- H_{rs} – set of simple paths between r and s.
- (i, j) – link with head node i and tail node j. $t_{ij}(x_{ij})$ is the cost of traversing (i, j). The function $t_{ij}(\cdot)$ is strictly positive and monotonically increasing.
- For path $h \in H_{rs}$, the flow in h will be denoted by f_h.

Wardrop's Principle

Definition (UE-Condition)

For each O-D pair (r, s), every used path between r and s must have equal and minimal travel-time.

This can be formulated as an optimization problem.

$$\min z(f) = \sum_{(i,j)\in A} \int_0^{x_{ij}} t(\omega)d\omega$$

where

$$x_{ij} = \sum_{r\in R} \sum_{s\in S} \sum_{h\in H_{rs}} f_h \delta_{ij}^h$$

Subject to:

$$f_h \geq 0 \text{ and } \sum_{h\in H_{rs}} f_h = d_{rs} \quad \forall\, r \in R, s \in S$$

Approximation

- Solution to the above problem satisfies UE-condition.
- Will use quadratic approximation of the Beckman function as the objective function.
- Consider the case of a single O-D pair.
- Second order Taylor approximation at path flow solution g:

$$\hat{z}_{rs}(f) \approx z_{rs}(g) + \sum_{h \in H_{rs}} \left.\frac{\partial z_{rs}}{\partial f_h}\right|_g (f_h - g_h) + \sum_{h \in H_{rs}} \left.\frac{\partial^2 z_{rs}}{\partial f_h^2}\right|_g (f_h - g_h)^2$$

- Denote $\left.\frac{\partial z_{rs}}{\partial f_h}\right|_g =: v_h^g$ and $\left.\frac{\partial^2 z_{rs}}{\partial f_h^2}\right|_g =: s_h^g$.
- After removing constants, objective becomes:

$$\sum_{h \in H_{rs}} \left[(v_h^g - s_h^g g_h) f_h + \frac{1}{2} s_h^g f_h^2 \right]$$

KKT Conditions

These constants can be calculated easily by taking the derivative of the Beckman function.

$$\left.\frac{\partial z_{rs}}{\partial f_h}\right|_g = \sum_{(i,j)\in A} \delta_{ij}^h t_{ij}(x_{ij}); \quad \left.\frac{\partial^2 z_{rs}}{\partial f_h^2}\right|_g = \sum_{(i,j)\in A} \delta_{ij}^h t'_{ij}(x_{ij})$$

The KKT conditions of the modified problem with the same constraints as before imply:

$$v_h^g + s_h^g(f_h - g_h) - w_{rs} \geq 0$$

and

$$f_h(v_h^g + s_h^g(f_h - g_h) - w_{rs}) = 0$$

Here the Lagrange multiplier (for the equality constraint), w_{rs} is interpreted as the least travel cost between r and s.

More Notations

Note that

$$\hat{v}_h = v_h^g + s_h^g(f_h - g_h)$$

is a first order approximation of the path travel cost.
Put

$$c_h^g := v_h^g - s_h^g g_h$$

then the conditions seen in previous slide for used paths ($\hat{H}_{rs}$) become

$$c_h^g + s_h^g f_h = w_{rs} := \bar{w}_{rs} \quad \forall h \in \hat{H}_{rs}$$

Path Flow Updates

Dividing the above expression by s_h^g and summing over all paths,

$$\bar{w}_{rs} = \frac{d_{rs} + \sum_{h \in \hat{H}_{rs}} (c_h^g / s_h^g)}{\sum_{h \in \hat{H}_{rs}} 1/s_h^g}$$

And the new flows are

$$f_h = \frac{\bar{w}_{rs} - c_h^g}{s_h^g}$$

The paths in $H_{rs} \setminus \hat{H}_{rs}$ recieve zero flow.

Algorithm – Single O-D Pair

Takes some initial path flow vector $\{g_h : h \in H_{rs}\}$ as input and outputs updated path flow vector. The steps involved are:

1. Calculate the constants v_h^g, s_h^g and c_h^g.
2. Sort paths based on increasing c_h^g.
3. Add the path with smallest c_h^g to $\hat{H}_{rs}$ and update $\bar{w}_{rs}$.
4. Keep adding subsequent paths until c_h^g exceeds $\bar{w}_{rs}$.
5. Update the flows.
6. The new path flow vector $\hat{H}_{rs}$ is given as output.

Algorithm – Single O-D Pair

1: **Input:** The current solution $\{g_h, \forall h \in H_{rs}\}$ is given.
2: **Initialization:** lines 3–9.
3: **for** each $h \in H_{rs}$ **do**
4: Compute v_h^g and s_h^g according to formulation (16) and formulation (17), respectively.
5: Compute c_h^g according to formulation (18).
6: **end for**
7: Sort all paths $h \in H_{rs}$ according to the increasing order of c_h^g, i.e., $H_{rs} = \{1, 2, 3, \ldots\}$ with $c_1^g \leqslant c_2^g \leqslant c_3^g \leqslant \ldots$.
8: Let $B = 1/(s_1^g d_{rs})$, $C = c_1^g/(s_1^g d_{rs})$ and $\bar{w}_{rs} = (1.0 + C)/B$.
9: Set $h = 2$, $\hat{H}_{rs} = \{1\}$.
10: **Main Loop:** lines 11–17.
11: **while** $h \leqslant |H_{rs}|$ and $c_h^g < \bar{w}_{rs}$ **do**
12: Set $C = C + c_h^g/(s_h^g d_{rs})$.
13: Set $B = B + 1/(s_h^g d_{rs})$.
14: Set $\bar{w}_{rs} = (1.0 + C)/B$.
15: Let $\hat{H}_{rs} = \hat{H}_{rs} \cup \{h\}$.
16: Set $h = h + 1$.
17: **end while**
18: **Flow Update:** lines 19–31.
19: **end for**
20: **for** each $h \in \hat{H}_{rs}$ **do**
21: Set $f_h = (\bar{w}_{rs} - c_h^g)/s_h^g$.
22: **for** each $h \in H_{rs} \backslash \hat{H}_{rs}$ **do**
23: Set $f_h = 0$.
24: **end for**
25: **for** each $h \in H_{rs}$ **do**
26: **if** $f_h \neq g_h$ **then**
27: Update the link flow by $x_{ij} = x_{ij} + (f_h - g_h), \forall (i,j) \in h$.
28: Update $t_{ij}(x_{ij})$ and $\dfrac{\partial t_{ij}(x_{ij})}{\partial x_{ij}}$ for each link $(i,j) \in h$.
29: **end if**
30: **end for**
31: Let $H_{rs} = \hat{H}_{rs}$.
32: **Output:** A new solution $\{f_h, \forall h \in H_{rs}\}$.

Initialization

1: **Input:** The current solution $\{g_h, \forall h \in H_{rs}\}$ is given.
2: **Initialization:** lines 3–9.
3: **for** each $h \in H_{rs}$ **do**
4: Compute v_h^g and s_h^g according to formulation (16) and formulation (17), respectively.
5: Compute c_h^g according to formulation (18).
6: **end for**
7: Sort all paths $h \in H_{rs}$ according to the increasing order of c_h^g, i.e., $H_{rs} = \{1, 2, 3, \ldots\}$ with $c_1^g \leqslant c_2^g \leqslant c_3^g \leqslant \ldots$.
8: Let $B = 1/(s_1^g d_{rs})$, $C = c_1^g/(s_1^g d_{rs})$ and $\bar{w}_{rs} = (1.0 + C)/B$.
9: Set $h = 2$, $\hat{H}_{rs} = \{1\}$.

Main Loop

10: **Main Loop:** lines 11–17.
11: **while** $h \leqslant |H_{rs}|$ and $c_h^g < \bar{w}_{rs}$ **do**
12: Set $C = C + c_h^g/(s_h^g d_{rs})$.
13: Set $B = B + 1/(s_h^g d_{rs})$.
14: Set $\bar{w}_{rs} = (1.0 + C)/B$.
15: Let $\hat{H}_{rs} = \hat{H}_{rs} \cup \{h\}$.
16: Set $h = h + 1$.
17: **end while**

Flow Update

18: **Flow Update:** lines 19–31.
19: **end for**
20: **for** each $h \in \hat{H}_{rs}$ **do**
21:     Set $f_h = (\bar{w}_{rs} - c_h^g)/s_h^g$.
22: **for** each $h \in H_{rs} \backslash \hat{H}_{rs}$ **do**
23:         Set $f_h = 0$.
24: **end for**
25: **for** each $h \in H_{rs}$ **do**
26:     **if** $f_h \neq g_h$ **then**
27:         Update the link flow by $x_{ij} = x_{ij} + (f_h - g_h), \forall (i,j) \in h$.
28:         Update $t_{ij}(x_{ij})$ and $\dfrac{\partial t_{ij}(x_{ij})}{\partial x_{ij}}$ for each link $(i,j) \in h$.
29:     **end if**
30: **end for**
31: Let $H_{rs} = \hat{H}_{rs}$.
32: **Output:** A new solution $\{f_h, \forall h \in H_{rs}\}$.

Greedy Path-Based Algorithm

The steps involved are:

0. Initialization: assign all passengers to shortest path between O-D pairs and set H_{rs} to contain the shortest path between r and s.
1. Perform flow adjustment on H_{rs} using the previous algorithm for each r, s.
2. (Inner loop) Update flows on O-D pairs with larger deviation:
   - If $\max v_h - \min v_h \geq RG^{k-1}/2$, for an O-D pair, H_{rs} is passed to the previous algorithm again.
   - If this update is not being done for any O-D pair (or if the iteration exceeds a limit), break from the loop.
3. Push the shortest path between each O-D pairs to H_{rs} and repeat from step 1 until convergence.

Greedy Path-Based Algorithm

1: **Initialize**: lines 2–7
2: **for** each O-D pair (r, s) **do**
3: Compute the shortest path $\hat{h}$ from r to s.
4: Assign all d_{rs} to $\hat{h}$ and push it into H_{rs}.
5: **end for**
6: Update link flows by $x_{ij} = \sum_{r \in R} \sum_{s \in S} \sum_{h \in H_{rs}} f_h \delta_{ij}^h, \quad \forall (i, j) \in A.$
7: Update the link cost $t_{ij}(x_{ij})$ and its derivative $\frac{\partial t_{ij}(x_{ij})}{\partial x_{ij}}$ for each link $(i, j) \in A$.

8: **Main Loop**: lines 9–34
9: **for** each $r \in R$ **do**
10: Compute the shortest path tree from r to all its destinations S_r.
11: **for** each $s \in S_r$ **do**
12: Build the shortest path $\hat{h}$ from r to s.
13: Push $\hat{h}$ into H_{rs} if $\hat{h} \notin H_{rs}$; otherwise delete $\hat{h}$.
14: Perform path flow adjustment for H_{rs} by Algorithm 1.
15: **end for**
16: **end for**

17: **Inner Loop**: lines 18–34
18: Set $I = 0, MaxI = 1000, FC = 0$.
19: **while** $I < MaxI$ **do**
20: Let $I = I + 1$ and $FC = 0$.
21: **for** each O-D pair (r, s) **do**
22: **if** $I\%100 = 0$ **then**
23: Compute $\Delta_{rs} = \max\{v_h\} - \min\{v_h\}, \forall h \in H_{rs}$.
24: **end if**
25: **if** $\Delta > RG^{k-1}/2.0$ **then**
26: Let $FC = FC + 1$.
27: Perform path flow adjustment for H_{rs} by Algorithm 1.
28: Update x_{ij}, $t_{ij}(x_{ij})$ and $\frac{\partial t_{ij}(x_{ij})}{\partial x_{ij}}$ for $\forall (i, j) \in h, \forall h \in H_{rs}$.
29: **end if**
30: **end for**
31: **if** $FC = 0$ **then**
32: Break the inner loop.
33: **end if**
34: **end while**

Initialization

1: **Initialize**: lines 2–7
2: **for** each O-D pair (r, s) **do**
3: Compute the shortest path $\hat{h}$ from r to s.
4: Assign all d_{rs} to $\hat{h}$ and push it into H_{rs}.
5: **end for**
6: Update link flows by
$x_{ij} = \sum_{r \in R} \sum_{s \in S} \sum_{h \in H_{rs}} f_h \delta_{ij}^h, \quad \forall (i,j) \in A.$
7: Update the link cost $t_{ij}(x_{ij})$ and its derivative $\frac{\partial t_{ij}(x_{ij})}{\partial x_{ij}}$ for each link $(i,j) \in A$.

Main Loop

```
8:  Main Loop: lines 9–34
9:  for each r ∈ R do
10:     Compute the shortest path tree from r to all its
        destinations S_r.
11:     for each s ∈ S_r do
12:         Build the shortest path ĥ from r to s.
13:         Push ĥ into H_rs if ĥ ∉ H_rs; otherwise delete ĥ.
14:         Perform path flow adjustment for H_rs by Algorithm 1.
15:     end for
16: end for
```

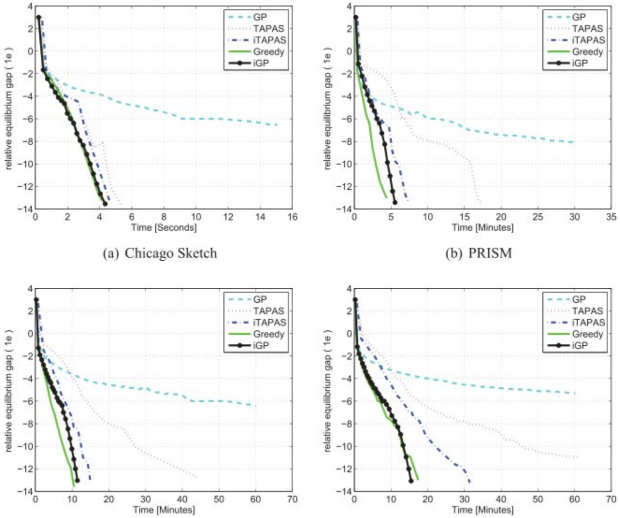
Inner Loop

17: **Inner Loop**: lines 18–34
18: Set $I = 0, MaxI = 1000, FC = 0$.
19: **while** $I < MaxI$ **do**
20:     Let $I = I + 1$ and $FC = 0$.
21:     **for** each O-D pair (r, s) **do**
22:         **if** $I\%100 = 0$ **then**
23:             Compute $\Delta_{rs} = \max\{v_h\} - \min\{v_h\}, \forall h \in H_{rs}$.
24:         **end if**
25:         **if** $\Delta > RG^{k-1}/2.0$ **then**
26:             Let $FC = FC + 1$.
27:             Perform path flow adjustment for H_{rs} by Algorithm 1.
28:             Update x_{ij}, $t_{ij}(x_{ij})$ and $\frac{\partial t_{ij}(x_{ij})}{\partial x_{ij}}$ for $\forall (i, j) \in h, \forall h \in H_{rs}$.
29:         **end if**
30:     **end for**
31:     **if** $FC = 0$ **then**
32:         Break the inner loop.
33:     **end if**
34: **end while**

Results

- Numerical results were produced on Windows 10 workstation with 3.3*GHz* Intel Xenon processor and 16*GB* RAM using C++.
- The greedy algorithm performs better than GP, TAPAS, iTAPAS in all the experiments.
- iGP performs almost as good as the greedy method.

Results

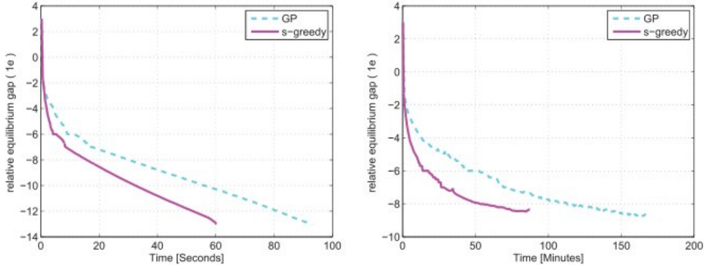


(a) Chicago Sketch (b) PRISM

(c) Chicago Regional (d) Philadelphia

Results

Even without the inner loop, the greedy algorithm performs better than GP.



(a) Chicago Sketch (b) Chicago Regional

Results – Memory Usage

Table 2. Path Information for Different Algorithms

Networks	Information	GP	greedy	iGP
Chicago Regional	path number	1,985,744	1,991,055	1,920,298
	memory size	489 M	491 M	473 M
Philadelphia	path number	1,712,502	1,709,818	1,376,289
	memory size	583 M	562 M	437 M

Note: M = megabyte.

The path-based algorithms generates less than 2 million used paths, which consumes less than 600 megabytes of memory.

Conclusion