Link-based algorithms for solving UE

Pramesh Kumar

IIT Delhi

September 15, 2025

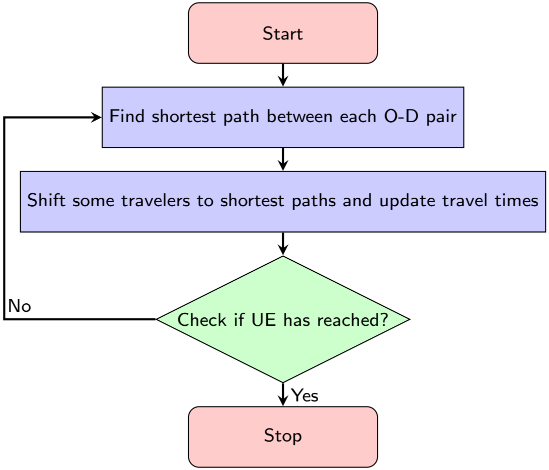
Algorithms for solving the traffic assignment

- Under a few assumptions, we formulated the UE traffic assignment problem as a convex optimization problem.
- Although one can use general-purpose algorithms designed to solve convex optimization problems for the UE traffic assignment problem, we need to consider the following issues:
  - The problem size is large for realistic networks.
  - The computational time should be reasonable.
- After several decades of research, there are many efficient algorithms developed for solving UE traffic assignment problem. Each algorithm has its own merits and demerits.

Types of algorithms

1. Link-based algorithms
   - keeps track of link flows only
   - requires less computer memory
   - easy to implement
   - slow convergence especially when high accuracy is required
2. Path-based algorithms
   - keeps track of path flows
   - requires high computer memory
   - requires more programming skills than link-based methods
   - faster convergence especially when high accuracy is required
3. Bush-based algorithms
   - keeps track of bush (acyclic subgraphs) flows
   - requires higher computer memory than link-based algorithms but lesser memory than path-based algorithms
   - requires sophisticated programming skills
   - faster convergence as compared to both link-based and path-based algorithms

General structure of traffic assignment algorithms


Start
Find shortest path between each O-D pair
Shift some travelers to shortest paths and update travel times
Check if UE has reached?
No
Yes
Stop

Convergence criteria

- Relative gap The gap becomes zero if and only if the flow values $\mathbf{x}$ satisfy UE

$$\text{gap} = \frac{\text{TSTT} - \text{SPTT}}{\text{SPTT}} = \frac{\sum_{(i,j)\in A} x_{ij}t_{ij}(x_{ij}) - \sum_{(r,s)\in Z^2} d^{rs}k^{rs}}{\sum_{(r,s)\in Z^2} d^{rs}k^{rs}} \tag{1}$$

- Objective function gap Let $f(\mathbf{x})$ be the Beckmann's objective function.

$$\text{gap} = \frac{\bar{f} - \underline{f}}{\underline{f}} \tag{2}$$

  where, $\bar{f}$ and $\underline{f}$ are the upper and lower bound respectively.
- Average excess cost

$$AEC = \frac{\sum_{(i,j)\in A} x_{ij}t_{ij}(x_{ij}) - \sum_{(r,s)\in Z^2} d^{rs}k^{rs}}{\sum_{(r,s)\in Z^2} d^{rs}} \tag{3}$$

- Maximum excess cost

$$MEC = \max_{(r,s)\in Z^2} \left\{ \max_{\pi\in\Pi^{rs}:h^\pi>0} \{c^\pi - k^{rs}\} \right\} \tag{4}$$

Frank-Wolfe method

In order to find the search direction, Frank-Wolfe method (also known as conditional gradient method) solves the following linear program in iteration k:

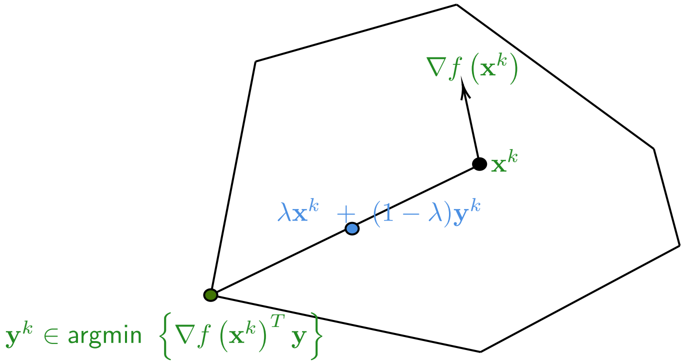
$$\min_{\mathbf{y} \in X} \nabla f(\mathbf{x}^k)^T (\mathbf{y} - \mathbf{x}^k) \tag{5}$$

where X is the feasible region.
If we use the Beckmann's formulation of UE, (5) turns into $|Z^2|$ mincost flow problems in the network $G(N, A)$ with link costs $\{t_{ij}(x_{ij}^k)\}_{(i,j) \in A}$. These problems can be solved by finding the shortest path between each O-D pair $(r, s) \in Z^2$ and loading the demand d^{rs} on it. The corresponding link flow values $\{y_{ij}^k\}_{(i,j) \in A}$ is referred to as auxiliary flows or all or nothing assignment since all the travelers are assigned to shortest paths. This is the direction in which the travelers feel pressure to move. However, if we move all the travelers to these paths, they will no longer remain shortest paths. Therefore, we shift only a fraction λ of travelers, i.e.,

$$\mathbf{x}^{k+1} = (1 - \lambda)\mathbf{x}^k + \lambda \mathbf{y}^k \tag{6}$$

The new flows $\mathbf{x}^{k+1}$ will remain feasible since X is a convex set.

Frank-Wolfe method


$\nabla f\left(\mathbf{x}^k\right)$
$\mathbf{x}^k$
$\lambda\mathbf{x}^k + (1-\lambda)\mathbf{y}^k$
$\mathbf{y}^k \in \text{argmin} \left\{\nabla f\left(\mathbf{x}^k\right)^T \mathbf{y}\right\}$

Frank-Wolfe method

Note that $(\mathbf{y}^k - \mathbf{x}^k)$ is a descent direction. Why? For Beckmann's function $\nabla f(\mathbf{x})^T(\mathbf{y}-\mathbf{x}) = \mathbf{t}(\mathbf{x})^T(\mathbf{y}-\mathbf{x}) \leq 0$.
We can use the exact line search to compute λ, i.e.,

$$\lambda \in \text{argmin}_{\lambda \in [0,1]}\{f((1-\lambda)\mathbf{x}^k + \lambda \mathbf{y}^k)\} \tag{7}$$

This can be treated unconstrained minimization problem of one variable, which can solved using $\frac{df}{d\lambda} = 0 \implies$.

$$\boxed{\sum_{(i,j)\in A} t_{ij}((1-\lambda)x_{ij} + \lambda y_{ij})(y_{ij} - x_{ij}) = 0} \tag{8}$$

If required, the final value of λ can be projected back to region $[0,1]$. Let us summarize the F-W algorithm now.

Frank-Wolfe algorithm

```
1:  procedure FW($G, \mathbf{t}, \mathbf{d}, \mathsf{tol}$)
2:      $k = 1$, $x_{ij}^k = 0$, $t_{ij}^k = t_{ij}(0), \forall (i,j)$, and $\mathsf{gap} = \infty$        ▷ Initialization
3:      while $\mathsf{gap} > \mathsf{tol}$ do
4:          $y_{ij}^k = 0, \forall (i,j) \in A$                                        ▷ Auxiliary flows
5:          for $r \in Z$ do
6:              $l^*, pred \leftarrow$ DIJKSTRA($G, \mathbf{t}^k, r$)
7:              for $s \in Z$ do
8:                  $\pi_{rs}^* \leftarrow$ TRACEPREDS($G, pred, s$)
9:                  for $(i,j) \in \pi_{rs}^*$ do
10:                     $y_{ij}^k \leftarrow y_{ij}^k + d^{rs}$
11:                 end for
12:             end for
13:             Evaluate $\lambda$ using (8)
14:             for $(i,j) \in A$ do
15:                 $x_{ij}^{k+1} \leftarrow (1-\lambda)x_{ij}^k + \lambda y_{ij}^k$                ▷ Update link flows
16:                 $t_{ij}^{k+1} \leftarrow t_{ij}(x_{ij}^{k+1})$                       ▷ Update link travel times
17:             end for
18:             Evaluate $\mathsf{gap}$
19:             $k \leftarrow k + 1$
20:         end for
21:     end while
22: end procedure
```

Method of successive averages (MSA)

- In each iteration k, the method uses step size $\lambda = \frac{1}{k+1}$.
- This means it shifts more travelers to shortest paths in initial iterations but fewer travelers as the algorithm progresses until convergence.
- It is easy to implement method.

Frank-Wolfe's "zig-zagging" behavior

FW searches for the target vector $\mathbf{y}$ by solving LP (5). Since the optimal solution to an LP is an extreme point, the target vector is restricted to be one of those extreme points. This results in a trajectory of points shown by thin line in the following figure.The algorithm is not able to take a direct step shown by thick line. This causes slow performance of the algorithm when it reaches near to the equilibrium solution, behavior popularly known as zigzagging behavior.

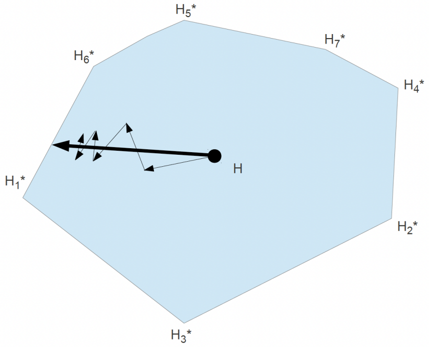


Figure: Credits: BLU book Chapter 7, Fig. 7.4

Conjugate Frank-Wolfe method

- Conjugate Frank-Wolfe finds the target vector in more intelligent way than F-W method.
- It chooses the target vector $\mathbf{y}$ so that the current search direction $(\mathbf{y} - \mathbf{x})$ is conjugate to the previous iteration search direction $(\mathbf{y}_{\text{prev}} - \mathbf{x})$.

$$(\mathbf{y}_{\text{prev}} - \mathbf{x})^T H(\mathbf{y} - \mathbf{x}) = 0 \tag{9}$$

  and $\mathbf{y}$ remains feasible which could be achieved as below (since X is a convex set).

$$\mathbf{y} = \lambda \mathbf{y}_{\text{prev}} + (1 - \lambda)\mathbf{y}^{\text{AON}} \tag{10}$$

  where, $H = \text{diag}(\frac{dt_{ij}(x_{ij})}{x_{ij}})$ is the Hessian of the Beckmann's function evaluated at the current solution $\mathbf{x}$, and $\mathbf{y}^{\text{AON}}$ refers to the all or nothing assignment vector.
- By plugging $\mathbf{y}$ from (10) into (9), we get the following value of λ

$$\boxed{\lambda = \frac{(\mathbf{y}_{\text{prev}} - \mathbf{x})^T H(\mathbf{y}^{\text{AON}} - \mathbf{x})}{(\mathbf{y}_{\text{prev}} - \mathbf{x})^T H(\mathbf{y}^{\text{AON}} - \mathbf{y}_{\text{prev}})}} \tag{11}$$

Conjugate Frank-Wolfe method

In summation, we get,

$$\boxed{\lambda = \mathsf{proj}_{[0,1-\epsilon]}\left(\frac{\sum_{(i,j)\in A}((y_{\mathsf{prev}})_{ij} - x_{ij})(y_{ij}^{\mathsf{AON}} - x_{ij})t_{ij}^{'}}{\sum_{(i,j)\in A}((y_{\mathsf{prev}})_{ij} - x_{ij})(y_{ij}^{\mathsf{AON}} - (y_{\mathsf{prev}})_{ij})t_{ij}^{'}}\right)} \quad (12)$$

where, $t_{ij}^{'}$ is the travel time derivative evaluated at x_{ij}.
Remark.

- If the denominator of (12) is zero, then we can set $\lambda = 0$ forcing the target vector to be all or nothing assignment $\mathbf{y} = \mathbf{y}^{\mathsf{AON}}$.
- For the first iteration, we do not have any $\mathbf{y}_{\mathsf{prev}}$ in which case, we also select $\lambda = 0$.
- We also don't λ to take the value 1, otherwise the new target vector $\mathbf{y}$ will be same as the previous $\mathbf{y}_{\mathsf{prev}}$ making the algorithm stuck in an infinite loop. In this case, we can take $\lambda = 1 - \epsilon$, where ϵ is small positive number.

Suggested reading

- BLU Book Chapter 7 Section 1 and 2
- Sheffi Chapter 5
- Mitradjieva, Maria, and Per Olov Lindberg. "The stiff is moving—Conjugate direction Frank-Wolfe methods with applications to traffic assignment." Transportation Science 47.2 (2013): 280-293.

Thank you!