Frank-Wolfe Algorithm for UE and SO Problems

Ximing Wang

1. Statement of the Problem

For the road network, the User Equilibrium (UE) is based on the assumption that each user wishes to minimize his/her travel time, so travel times on all used paths of each O-D pair are equal, and the travel time on any unused path is equal to or greater than the used travel time. In that case, no user can reduce his/her travel time by unilaterally changing path, so the network has become stationary, i.e. user equilibrium.

The mathematical expression for UE is:

$$\begin{aligned}
&f_k^{rs}\left(c_k^{rs}-c_{\min}^{rs}\right)=0,\forall k,r,s\\
&c_k^{rs}-c_{\min}^{rs}\geq 0,\forall k,r,s\\
&\sum_k f_k^{rs}=q_{rs},\forall r,s\\
&f_k^{rs}\geq 0,\forall k,r,s
\end{aligned}$$

The first two formula guarantee that the travel times on all the used paths from r to s are equal to the minimum path travel time, and the travel time on any unused path is equal to or greater than the minimum path travel time, which is in accordance with the user equilibrium assumption. The third formula is the O-D flow constraint. The fourth formula is the non-negative constraints of path flow.

Based on KKT conditions, the above expression can be transformed to BMW formulation:

$$\begin{aligned}
\min\ & Z(x)=\sum_a\int_0^{x_a}t_a(\varpi)\,\mathrm{d}\varpi\\
\text{s.t.}\ & \sum_k f_k^{rs}=q_{rs},\forall r,s\\
& f_k^{rs}\geq 0,\forall k,r,s\\
& x_a=\sum_{rs}\sum_k f_k^{rs}\delta_{a,k}^{rs},a\in A
\end{aligned}$$

Similar formulation can be achieved for System Optimum (SO) of the network. System optimum means that the total system travel time has reached the minimum, thus the SO formulation is:

$$\begin{aligned}
&\min Z(x) = \sum_a x_a t_a(x_a) \\
&\text{s.t.} \quad \sum_k f_k^{rs} = q_{rs}, \forall r, s \\
&\qquad f_k^{rs} \geq 0, \forall k, r, s \\
&\qquad x_a = \sum_{rs} \sum_k f_k^{rs} \delta_{a,k}^{rs}, a \in A
\end{aligned}$$

It is noticed that the difference between SO formulation and UE formulation is only in the objective function. By introducing the marginal link travel time:

$$\tilde{t}_a(x_a) = t_a(x_a) + x_a \frac{\partial t_a(x_a)}{\partial x_a} = \frac{\partial x_a t_a(x_a)}{\partial x_a}$$

The objective function of SO would become:

$$\sum_a x_a t_a(x_a) = \sum_a \int_0^{x_a} \frac{\partial x_a t_a(x_a)}{\partial x_a} \mathrm{d}\varpi = \sum_a \int_0^{x_a} \tilde{t}_a(\varpi) \mathrm{d}\varpi$$

Similar to the objective function of UE, except using marginal link travel time rather than link travel time. Therefore, the same solution procedure can be applied to both UE and SO.

From the system perspective, SO is a better solution. However, system optimum needs all drivers to act to minimize the total system travel time instead of their own travel time, so some drivers may be able to reduce their travel time by unilaterally switching routes, which makes the SO flow unstable. From the above analysis, SO can be transformed to some kind of UE when using marginal link travel time instead of link travel time. Therefore, if the user pay the full cost that his/her travel incurs to the system, i.e. marginal travel time, the system will be optimum and stable. The marginal-cost pricing is to charge the users the difference between marginal travel time and average travel time:

$$\tau_a = \tilde{t}_a\left(x_a^{\mathrm{SO}}\right) - t_a\left(x_a^{\mathrm{SO}}\right) = x_a \left.\frac{\partial t_a(x_a)}{\partial x_a}\right|_{x_a^{\mathrm{SO}}}$$

With the above toll rate at each link, users will be forced to choose their route based on marginal travel time. Consequently, the resulting UE is SO without toll.

2. Description of the Solution Procedures

Using the concept of marginal travel time, the SO and UE could have the same formulation, thus the same solution procedure can be applied to both UE and SO. Here Frank-Wolfe Algorithm is utilized. The basic idea behind Frank-Wolfe Algorithm is the iterative descent method. In the following, the decent direction and the step size calculation procedure is first stated, then the whole procedure of Frank-Wolfe Algorithm is shown.

2.1 The Decent Direction

The descent direction $\mathbf{y}^n$ is achieved by solving:

$$\begin{aligned}
&\min z^n(\mathbf{y}) = \sum_i \left(\frac{\partial z(\mathbf{x}^n)}{\partial x_i} \right) y_i \\
&\text{s.t.} \ \sum_i h_{ij} y_i \geq b_j, \forall j \in J
\end{aligned}$$

For the BMW formulation, this would become:

$$\begin{aligned}
&\min_y \nabla z(x^n)^T \cdot y = \min_y t(x^n)^T \cdot y = \min_y \sum_a t_a(x_a{}^n) \cdot y_a \\
&\text{s.t.} \ \sum_k f_k^{rs} = q_{rs}, \forall r, s \\
&\qquad f_k^{rs} \geq 0, \forall k, r, s \\
&\qquad y_a = \sum_{rs} \sum_k f_k^{rs} \delta_{a,k}^{rs}, a \in A
\end{aligned}$$

Within each iteration, $x_a{}^n$ is fixed, so $t_a(x_a{}^n)$ is also fixed, i.e. the travel time on each link is fixed. To minimize the total travel time, the O-D flow should be assigned to the shortest path connecting that O-D pair, i.e. "all-or-nothing" assignment. To find the shortest path, Dijkstra's Algorithm is applied.

<u>Dijkstra's Algorithm</u>

Dijkstra's Algorithm finds the shortest path from original point to every node in the network, so the shortest path from o to d contains the shortest paths from o to every node on the path to d, which yields to the correctness of the algorithm. It divides the nodes into two groups: permanently labeled (S) and temporary labeled ($\tilde{S}$). Permanent label is the true shortest distance from origin. Temporary label is the shortest distance from origin using only permanently labeled nodes. At each iteration, the smallest temporary label becomes permanent. The algorithm stops when there is no more temporary labeled node. The pseudo-code of Dijkstra's Algorithm is as following:

```
begin
    $S \coloneqq \emptyset$;   $\tilde{S} \coloneqq N$
    $d(i) \coloneqq \infty$ for each node $i \in N$
    $d(o) \coloneqq 0$ and $pred(o) \coloneqq 0$
    while $|S| < n$ do
    begin
        let $i \in \tilde{S}$ be a node for which $d(i) = \min\{d(j): j \in \tilde{S}\}$;
        $S \coloneqq S \cup \{i\}$;
        $\tilde{S} \coloneqq \tilde{S} - \{i\}$;
        for each $(i, j) \in A(i)$ do
            if $d(j) > d(i) + c_{ij}$ then $d(j) := d(i) + c_{ij}$ and $pred(j) \coloneqq i$;
    end;
end;
```

With Dijkstra's Algorithm, the shortest path tree for the origin node could be found. Then the flows to each destination could be assigned to the corresponding shortest path. The procedure is repeated until flows from all origins have been assigned. This all-or-nothing assignment can help to get the decent direction flows y .

2.2 The Step Size

The step size can be determined by conducting the line search:

$$\begin{aligned} &\min_{\alpha} z\left(x^n + \alpha \cdot \left(y^n - x^n\right)\right) \\ &\text{s.t.} \quad 0 \le \alpha \le 1 \end{aligned}$$

To get the optimal step size, the bisection method is used.

<u>**Bisection Method**</u>

Bisection method searches for the point where the derivative of the objective function is zero. It is an iterative interval reduction procedure. The interval is reduced at each iteration while ensuring that the minimum lies within the current interval. The iteration continues until a good approximation is obtained. The scheme of bisection method is as following:

1. Let $[a^n\ b^n]$ be the current interval
2. $x^n = \frac{a^n+b^n}{2}$
3. If $\frac{dz(x^n)}{dx} < 0$, discard $[a^n\ x^n]$ and $[x^n\ b^n] \to [a^{n+1}\ b^{n+1}]$;

   Otherwise, discard $[x^n\ b^n]$ and $[a^n\ x^n] \to [a^{n+1}\ b^{n+1}]$
4. If stopping criterion $\frac{|b^N-a^N|}{2} \le \varepsilon$ met, the optimal point $x^* = \frac{a^N+b^N}{2}$, stop;

   Otherwise, go to step 2.

For the BMW formulation, the bisection method is used to find the optimal step size α that satisfies:

$$\frac{\partial z\left(x^n+\alpha\cdot\left(y^n-x^n\right)\right)}{\partial\alpha}=\frac{\partial\sum_a\int_0^{x_a{}^n+\alpha\cdot\left(y_a{}^n-x_a{}^n\right)}t_a\left(\varpi\right)\mathrm{d}\varpi}{\partial\alpha}$$
$$=\sum_a\left(y_a{}^n-x_a{}^n\right)\cdot t_a\left(x_a{}^n+\alpha\cdot\left(y_a{}^n-x_a{}^n\right)\right)=0$$

2.3 The Scheme of Frank-Wolfe Algorithm

The detailed calculation procedures of the descent direction and step size are stated above. The whole scheme of Frank-Wolfe Algorithm is as following:

Step 0. Initialization. Perform all-or-nothing assignment based on $t_a=t_a(0),\forall a$. This yields x^1. Set counter $n:=1$.

Step 1. Update. Set $t_a^n=t_a(x_a^n),\forall a$.

Step 2. Direction finding. Perform all-or-nothing assignment based on $t_a^n,\forall a$. This yields (direction) flows y^n.

Step 3. Line search. Find $0\le\alpha_n\le1$ that solves $\sum_a(y_a^n-x_a^n)\cdot t_a\big(x_a^n+\alpha_n\cdot(y_a^n-x_a^n)\big)=0$

Step 4. Move. Set $x^{n+1}=x^n+\alpha_n(y^n-x^n)$

Step 5. Convergence test. If $\frac{\sqrt{\sum_a\left(x_a^{n+1}-x_a^n\right)^2}}{\sum_a x_a^n}<\varepsilon$, stop; otherwise $n:=n+1$, and go to Step 1.