Nonlinear optimization for traffic assignment problem

Yu Wang

April 13, 2020

1 Introduction and Paper Organization

With the fast-growing population and the increasing travel demands, transportation engineers face various levels of difficulties, including designing, maintaining, and improving transportation networks. To deal with these difficulties as well as satisfying the existed and future travel demands, solving traffic assignment problems [3], i.e., estimating traffic flow of road links, is of great importance. With such a background, this paper is to formalize the traffic assignment problem as a nonlinear optimization problem and apply several current-existing algorithms to solve the problem and obtain the traffic user equilibrium.

The rest of the paper is organized as follows. In Section 2, we mathematically formalize the traffic assignment problem as a nonlinear optimization problem. The existence of the minimizer and the convexity of the objective function are proved. In Section 3, various implemented algorithms for finding the minimizer are introduced. Several network examples are included in Section 4 to test the existing algorithms. The user manual is given in Section 5 to guide the future use of the program for readers' convenience.

2 Problem Formulation

This section introduces the mathematical formulation of the traffic assignment problem and the notation to be used throughout the paper.

A transportation network is defined as a directed graph $\boldsymbol{G}(\boldsymbol{N}, \boldsymbol{A})$ where $\boldsymbol{N}$ is the node set and $\boldsymbol{A}$ is the arc set. Specifically, the undirected edge is modeled by two opposite arcs. There are several origin-destination pairs (O-D pairs) and the users of the transportation networks travel between each of the O-D pairs. Let D_{od_i} denote the travel demand value of the O-D pair $od_i \in \boldsymbol{OD}$, where $\boldsymbol{OD}$ is the set of all O-D pairs. The demand value represents how many people or vehicles are travelling from an origin to a destination.

The critical factor in analyzing the traffic assignment problem is to consider the traffic congestion effect that occurs in road networks [7]. Due to the congestion effect, when the number of vehicles traveling on a particular road link increases, the users require more time to travel through that link. To incorporate the traffic congestion effect, link cost functions are introduced where the travel time (travel cost) of each link is a function of the number of vehicles (traffic flow) traveling along that link. Typically, the link cost is a monotonically increasing function of the traffic flow, and in applications, one may follow the suggestion of the Federal Highway Administration [8] to choose the link performance function as follows:

$$t_{l_i}(f_{l_i}) = t_{l_i}^0(1 + \alpha(\frac{f_{l_i}}{C_{l_i}})^\beta) \tag{1}$$

where f_{l_i} represents the traffic flow on link $l_i \in \boldsymbol{A}$, i.e., the number of vehicles per time slice; C_{l_i} denotes the capacity of link l_i which is usually given by history data; $t_{l_i}^0$ is the free time of link l_i indicating the traveling time of a vehicle along link l_i when the link is free of other vehicles; α, β are parameters determined based on the specific transportation network. Usually, they are set to be $0.15, 4$. From Eq.(1), it is clear that there is a basic traveling time t_0 and an extra traveling time due to the traffic congestion effect.

Let $\boldsymbol{F} = [F_{p_1}, F_{p_2}, ..., F_{p_k}], p_i \in \boldsymbol{P}$ denotes a vector of path flows where $\boldsymbol{P} = [p_1, p_2, ..., p_k]$ is the set of all simple paths of graph $\boldsymbol{G}(\boldsymbol{N}, \boldsymbol{A})$. Link flows are determined by path flows following the relationship:

$$f_{l_i} = \sum_{od_i \in OD} \sum_{p_j \in K_{od_i}} \delta_{l_i}^{p_j} F_{p_j} \tag{2}$$

where $K_{od_i} \subseteq K$ is the set of paths between O-D pair od_i and $K = \{K_{od_i} | od_i \in OD, \forall i\}$.

To satisfy the traveling demands of the transportation network, the sum of path flows should equal to the demand value of their corresponding O-D pair, which gives us:

$$\sum_{p_j \in K_{od_i}} F_{p_j} = D_{od_i} \tag{3}$$

The model for solving the traffic assignment problem also depends on the additivity of path cost. The additivity feature of path cost indicates that the travel time on each path is the sum of travel times of links belonging to this path, which is shown as follows:

$$t_{p_j} = \sum_{l_i \in A} \delta_{l_i}^{p_j} t_{l_i} \tag{4}$$

After introducing the basic concepts and relationships, the following subsection illustrates the user equilibrium state and the famous principle, Wardrop's first principle.

2.1 User Equilibrium

In a transportation network, each user noncooperatively seeks to minimize his or her own cost by taking the path with the least cost from their corresponding origins to destinations. Therefore users prefer to choose paths with least travel time. Due to the congestion effect, the travel cost of these paths increases, and then users change their path choice. Such changes from flow to cost and back to flow keeps going iteratively until users can not reduce his or her own cost by shifting to other alternative routes, which means all possible paths between the same O-D pair, if they are used, have the same and the least traveling time. This specific state is defined as the user equilibrium of the transportation network.

Meanwhile, Wardrop has come up the Wardrop's First Principle to give an equivalent definition of the user equilibrium: For each O-D pair, at user equilibrium, the travel time on all used paths is less than or equal to the travel time that would be spent by a single vehicle on any unused path [4].

For each specific flow solution $\boldsymbol{f} = \{f_{l_i} | l_i \in \boldsymbol{A}\}$, the traveling time of each link can be computed using Eq.(1) and by additivity of path costs given from Eq.(4), the traveling time of each path can be calculated. By Wardrop's first principle, if the specific flow solution gives the user equilibrium, then for each fixed $od_i \in \boldsymbol{OD}$, there exists a constant positive traveling time $t_{od_i} \in \mathbf{R}$ such that $t_{p_j} = t_{od_i}$for all $p_j \in K_{od_i}, f_{l_i}^{od_i} > 0, \delta_{l_i}^{p_j} = 1$, and $t_{p_j} \geq t_{od_i}$ for all $p_j \in K_{od_i}, f_{l_i}^{od_i} = 0, \delta_{l_i}^{p_j} = 1$.

2.2 Mathematical Formulation

The most famous mathematical formulation for user equilibrium is formalized as follows [4]:

$$\min_{\boldsymbol{f} > 0} z(\boldsymbol{f}) = \sum_{l_i \in A} \int_0^{f_{l_i}} t_{l_i}(x) dx \tag{5}$$

$$\sum_{p_j \in K_{od_i}} F_{p_j} = D_{od_i}, \forall od_i \in \mathbf{OD} \tag{6}$$

$$f_{l_i} = \sum_{od_i \in OD} \sum_{p_j \in K_{od_i}} \delta_{l_i}^{p_j} F_{p_j} \tag{7}$$

$$F_{p_j} \geq 0, \forall p_j \in \boldsymbol{P} \tag{8}$$

As mentioned in previous sections, Eq.(6) guarantees that the demands of all O-D pairs are satisfied, and Eq.(7) specifies the relationship between path flow and link flow. Eq.(8) ensures the positivity of the path flow, which naturally leads to the positivity of link flow given Eq.(7).

The objective function stated by Eq.(5) is the Beckman function, and it should be noticed that the objective function does not have any intuitive interpretation. [4] has proved the equivalence between minimizing the Beckman function and finding the user equilibrium according to Wardrop's first principle. Here for the forward direction that the minimizer of the Beckman function corresponds exactly to a flow solution of the user equilibrium, we provide another rough but concise proof.

Let $\boldsymbol{f}$ be the minimizer of the Beckman function expressed in Eq.(5). Suppose by contradiction that $\boldsymbol{f}$ can not ensure the user equilibrium, which means there is a path p_m of which the travel cost t_{p_m} is less than the travel cost of another user-used path t_{p_n}. Assume the link set $L_{p_m p_n}$ contains all links shared by these two paths p_m, p_n, then the link set $L_{p_m} \backslash L_{p_m p_n}$ contains links belonging to path p_m but not path p_n while $L_{p_n} \backslash L_{p_m p_n}$ contains links belonging to path p_n but not path p_m. Since t_{p_n} is more than t_{p_m} and by additivity of travel cost, we have:

$$\begin{aligned} t_{p_n}(f) = \sum_{l_i \in L_{p_n}} t_{l_i}(f_{l_i}) = \sum_{l_i \in L_{p_m p_n}} t_{l_i}(f_{l_i}) + \sum_{l_i \in (L_{p_n} \backslash L_{p_m})} t_{l_i}(f_{l_i}) \\ > t_{p_m}(f) = \sum_{l_i \in L_{p_m}} t_{l_i}(f_{l_i}) = \sum_{l_i \in L_{p_m p_n}} t_{l_i}(f_{l_i}) + \sum_{l_i \in (L_{p_m} \backslash L_{p_n})} t_{l_i}(f_{l_i}) \end{aligned} \tag{9}$$

which gives us:

$$\sum_{l_i \in (L_{p_n} \backslash L_{p_m})} t_{l_i}(f_{l_i}) > \sum_{l_i \in (L_{p_m} \backslash L_{p_n})} t_{l_i}(f_{l_i}) \tag{10}$$

Shifting a pretty small flow δf from path p_n to path p_m and denoting the new flow state as $\boldsymbol{f'}$, the difference between the Beckman function value of $\boldsymbol{f}, \boldsymbol{f'}$ is expressed as:

$$D = \sum_{l_i \in A} \int_0^{f_{l_i}} t_{l_i}(x)dx - \sum_{l_i \in A} \int_0^{f'_{l_i}} t_{l_i}(x)dx \tag{11}$$

Decomposing the link set A into $A \backslash (L_{p_m} \cup L_{p_n}), L_{p_m} \cap L_{p_n}, L_{p_m} \backslash L_{p_n}, L_{p_n} \backslash L_{p_m}$ and D can be expressed as follows:

$$\begin{aligned} D = & \sum_{l_i \in A \backslash (L_{p_m} \cup L_{p_n})} \int_0^{f_{l_i}} t_{l_i}(x)dx + \sum_{l_i \in (L_{p_m} \cap L_{p_n})} \int_0^{f_{l_i}} t_{l_i}(x)dx \\ & + \sum_{l_i \in (L_{p_m} \backslash L_{p_n})} \int_0^{f_{l_i}} t_{l_i}(x)dx + \sum_{l_i \in (L_{p_n} \backslash L_{p_m})} \int_0^{f_{l_i}} t_{l_i}(x)dx \\ & - \sum_{l_i \in A \backslash (L_{p_m} \cup L_{p_n})} \int_0^{f'_{l_i}} t_{l_i}(x)dx - \sum_{l_i \in (L_{p_m} \cap L_{p_n})} \int_0^{f'_{l_i}} t_{l_i}(x)dx \\ & - \sum_{l_i \in (L_{p_m} \backslash L_{p_n})} \int_0^{f'_{l_i}} t_{l_i}(x)dx - \sum_{l_i \in (L_{p_n} \backslash L_{p_m})} \int_0^{f'_{l_i}} t_{l_i}(x)dx \end{aligned} \tag{12}$$

Since the flow variation only happens on links $l \in (L_{p_n} \backslash L_{p_m}) \cup (L_{p_m} \backslash L_{p_n})$, we have:

$$\sum_{l_i \in A \backslash (L_{p_m} \cup L_{p_n})} \int_0^{f'_{l_i}} t_{l_i}(x)dx = \sum_{l_i \in A \backslash (L_{p_m} \cup L_{p_n})} \int_0^{f_{l_i}} t_{l_i}(x)dx \tag{13}$$

By the construction rule of ***f'***, $f_{l_i}^{'} = f_{l_i} - 2\delta f, l_i \in l_i \in (L_{p_n} \backslash L_{p_m})$ and $f_{l_i}^{'} = f_{l_i} + 2\delta f, l_i \in l_i \in (L_{p_m} \backslash L_{p_n})$, the Eq.(12) becomes:

$$\begin{aligned} D &= (\sum_{l_i \in (L_{p_m} \backslash L_{p_n})} \int_0^{f_{l_i}} t_{l_i}(x)dx - \sum_{l_i \in (L_{p_m} \backslash L_{p_n})} \int_0^{f_{l_i}+2\delta f} t_{l_i}(x)dx) \\ &+ (\sum_{l_i \in (L_{p_n} \backslash L_{p_m})} \int_0^{f_{l_i}} t_{l_i}(x)dx - \sum_{l_i \in (L_{p_n} \backslash L_{p_m})} \int_0^{f_{l_i}-2\delta f} t_{l_i}(x)dx) \\ &= \sum_{l_i \in (L_{p_n} \backslash L_{p_m})} \int_{f_{l_i}-2\delta f}^{f_{l_i}} t_{l_i}(x)dx - \sum_{l_i \in (L_{p_m} \backslash L_{p_n})} \int_{f_{l_i}-2\delta f}^{f_{l_i}} t_{l_i}(x)dx \end{aligned} \tag{14}$$

When δf is small enough compared with f, we can simplify D furthermore as:

$$\begin{aligned} D &= \sum_{l_i \in (L_{p_n} \backslash L_{p_m})} t_{l_i}(f_{l_i})2\delta f - \sum_{l_i \in (L_{p_m} \backslash L_{p_n})} t_{l_i}(f_{l_i})2\delta f \\ &= 2\delta f(\sum_{l_i \in (L_{p_n} \backslash L_{p_m})} t_{l_i}(f_{l_i}) - \sum_{l_i \in (L_{p_m} \backslash L_{p_n})} t_{l_i}(f_{l_i})) \overset{Eq.\ (10)}{>} 0 \end{aligned} \tag{15}$$

which means there is another flow solution ***f'*** such that the Beckman function value is even smaller than the value given by ***f***, contradicting to the fact that ***f*** is the minimizer. So we cannot find any other user-used path with less traveling cost, which means the minimizer of the Beckman function corresponds exactly to the user equilibrium. But it doesn't mean the user equilibrium corresponds exactly to the minimizer of the Beckman function. Maybe one user equilibrium corresponds to several values of the Beckman function, and one of them is the minimum. The complete proof is given in [4].

2.2.1 Existence of the Minimizer

After we show the equivalence between finding the minimizer of the Beckman function and solving the user equilibrium problem, we have to prove the existence of the minimizer.

Theorem 1: If $S = \mathbf{R}^n$ and f is continuous and coercive ($\lim_{\|x\| \to \infty} f(x) = +\infty$), then a global minimizer of f exists at a critical point.

The solution space of the traffic assignment problem is $\mathbf{R}^{|\boldsymbol{A}|}$. If $\|f\| \to \infty$, then there exists $l_i \in \boldsymbol{A}$ such that $f(l_i) \to \infty$ and obviously $t_{l_i}(f_{l_i}) \to \infty$ by Eq.(1). Furthermore, from Eq. (1), $t_{l_i}(f_{l_i})$ is nonnegative, monotonic and continuity of f_{l_i}, so we have:

$$\sum_{l_i \in A} \int_0^{f_{l_i}} t_{l_i}(x)dx \geq \int_0^{f_{l_i}} t_{l_i}(x)dx \to \infty \tag{16}$$

To prove the continuity of the Beckman function, we just need to prove the continuity of a single term $\int_0^{f_{l_i}} t_{l_i}(x)dx$ in the summation, which is an integral upper limit function. Notice that $t_{l_i}(x)$ is integrable. Assuming $G(x) = \int_a^x t(x)dx$ and thus the problem turns into proving the continuity of $G(x), x \in R$. For $\forall x \in [a,b]$,

$$\begin{aligned} G(x+\delta x) - G(x) &= \int_a^{x+\delta x} t(x)dx - \int_a^x t(x)dx \\ &= \int_a^x t(x)dx + \int_x^{x+\delta x} t(x)dx - \int_a^x t(x)dx \\ &= \int_x^{x+\delta x} t(x)dx \end{aligned} \tag{17}$$

Since $t(x)$ is integrable on $[a,b]$, $t(x)$ is bounded on $[a,b]$, assuming $\forall x \in [a,b], |t(x)| \leq M$, then $\int_x^{x+\delta x} t(x)dx \leq M\delta x$, so we have:

$$\lim_{\delta x \to 0} (G(x+\delta x) - G(x)) = \lim_{\delta x \to 0} M\delta x = 0 \tag{18}$$

which proves the continuity of $G(x)$ and concludes that the Beckman function is continuous and coercive. By **Theorem 1**, we conclude that a global minimizer of the Beckman function exists on the bounded closed set.

2.2.2 Convexity of the Equivalent Formulation

To prove the convexity of the Beckman function, we first calculate its Hessian Matrix,

$$H(z(f)) = \nabla^2 z(f) = \begin{bmatrix} \frac{dt_{l_1}(f_{l_1})}{df_{l_1}} & 0 & 0 & ... \\ 0 & \frac{dt_{l_2}(f_{l_2})}{df_{l_2}} & 0 & ... \\ 0 & 0 & ... & ... \\ ... & 0 & ... & \frac{dt_{l_{|A|}}(f_{l_{|A|}})}{df_{l_{|A|}}} \end{bmatrix} \tag{19}$$

Notice that the Hessian here is a diagonal matrix, so we can easily get the eigenvalue $\lambda_1, \lambda_2, ..., \lambda_{|A|}$, which are $\frac{dt_{l_1}(f_{l_1})}{df_{l_1}}, \frac{dt_{l_2}(f_{l_2})}{df_{l_2}}, ..., \frac{dt_{l_{|A|}}(f_{l_{|A|}})}{df_{l_{|A|}}}$ respectively. By Eq.(1), $t^0_{l_i}\alpha\frac{\beta}{C_{l_i}}(\frac{f_{l_i}}{C_{l_i}})^{\beta-1} > 0$ given the link performance function is increasing, which means the Hessian is positive definite. The convexity of the feasible region defined by the constraints is proved based on linear equality constraints [4]. So the global minimizer proved to be existent in the previous section is a unique strict global minimizer, which means the user equilibrium is unique.

3 Algorithms

As explained by the previous sections, getting the user equilibrium corresponds to solving a nonlinear optimization problem, and we further prove the existence as well as the uniqueness of the minimizer. Based on the above features, this section mainly focuses on introducing several algorithms that can solve this specific nonlinear optimization problem, i.e., find the minimizer and obtain the user equilibrium.

3.1 Link-based Algorithms

This subsection presents the Frank-Wolfe algorithm and two of its improved algorithms: conjugate and bi-conjugate Frank-Wolfe methods. All three algorithms begin with an initial feasible link flow assignment. It is generated by **all-or-nothing(AON)** assignment based on the initial zero flow. The travel time of the corresponding links is computed using Eq.(1). Then the shortest paths for each O-D pair are found, and all corresponding demand is assigned to those shortest paths. All these path flows are then projected back to all link flows given Eq.(2). Based on link flows in the current time step and previous time steps, the search direction can be determined [2][4]. These three link-based algorithms differ only in the way the search direction is defined. After we get the search direction, the step size θ^i is computed based on solving a one-dimensional minimization problem such that it shows a sufficient decrease of the function value. Since the function is already given by Eq.(5) and taking the derivative of the function over θ is not complicated, root-based methods like newton method or bisection method and function-based methods like golden-section method can all be applied to calculate the step size.

3.1.1 Frank-Wolfe Algorithm

Suppose the link flow at the current time step i is $\boldsymbol{f}^i$ and the link flow at the next step $\boldsymbol{f}^{AON,i}$ is generated by performing AON assignment, then $\boldsymbol{f}^{AON,i} - \boldsymbol{f}^i$ is a feasible direction. Moreover, this direction is also a direction of descent, which is shown in [4]. Step size is obtained using newton, bisection or golden-section method. The algorithm is detailed as follows:

Algorithm 1: Frank Wolfe algorithm for traffic assignment problem

```
Input: Network topology $\boldsymbol{G} = (\boldsymbol{N}, \boldsymbol{A})$, Link capacity $\boldsymbol{C}$, Link free time $\boldsymbol{T}_0$
    Generate a initial empty flow
    Perform AON assignment and get the initial feasible flow $\boldsymbol{f}^0$
    Set iteration counter $i = 0$
 1: while 1 do
 2:     Update the link cost using Eq.(1) and project it into path cost
 3:     Perform AON assignment based on the updated path cost to generate new feasible
        flow $\boldsymbol{f}^{\mathbf{AON},i}$
 4:     Define the search direction as $\boldsymbol{f}^{\mathbf{AON},i} - \boldsymbol{f}^i$
 5:     Compute the step size $\theta^i$ either using Newton, Bisection or Golden Section method
 6:     Update the link flow as $\boldsymbol{f}^{i+1} = \theta^i * \boldsymbol{f}^{\mathbf{AON},i} + (1 - \theta^i) * \boldsymbol{f}^i$
 7:     Calculate the convergence index $\gamma = \frac{TSTT}{SPTT} - 1$
 8:     if ($\gamma <$ Convergence Ratio) then
 9:         Terminate the iteration
10:     else
11:         $i = i + 1$
12:     end if
13: end while
```

Specifically, when we apply Newton or Bisection method to calculate the step size, we need to calculate the value of the 1st and 2nd derivative of the Beckman function over step α, which is as follows:

$$\frac{d}{d\theta}\sum_{l_j\in A}\int_0^{\theta f_{l_j}^{\text{AON, } i}+(1-\theta)f_{l_j}^{i}} t_{l_j}(x)dx = \sum_{l_j\in A} t_{l_j}(\theta f_{l_j}^{\text{AON},i}+(1-\theta)f_{l_j}^{i})(f_{l_j}^{\text{AON},i}-f_{l_j}^{i}) \tag{20}$$

$$\begin{aligned}\frac{d}{d^2\theta}\sum_{l_j\in A}\int_0^{\theta f_{l_j}^{\text{AON},i}+(1-\theta)f_{l_j}^{i}} t_{l_j}(x)dx &= \frac{d}{d\theta}\sum_{l_j\in A} t_{l_j}(\theta f_{l_j}^{\text{AON},i}+(1-\theta)f_{l_j}^{i})(f_{l_j}^{\text{AON},i}-f_{l_j}^{i}) \\ &= \sum_{l_j\in A} t_{l_j}^{'}(\theta f_{l_j}^{\text{AON, i}}+(1-\theta)f_{l_j}^{i})(f_{l_j}^{\text{AON},i}-f_{l_j}^{i})^2\end{aligned} \tag{21}$$

$$t_{l_j}^{'}(\theta f_{l_j}^{\text{AON},i}+(1-\theta)f_{l_j}^{i}) = t_{l_j}^{0}\alpha\frac{\beta}{C_{l_j}}(\frac{\theta f_{l_j}^{\text{AON},i}+(1-\theta)f_{l_j}^{i}}{C_{l_j}})^{\beta-1} \tag{22}$$

It should be noted that the line search framework applies to all three link-based algorithms as long as we get the feasible direction. Another thing worth paying attention to is when we apply the Newton method to find the minimizer, there is a chance that θ doesn't fall into the interval $[0,1]$ during the 1st few iterations. In this case, we might need to project it back to the feasible region. And such a problem is less likely to happen when we move closer to the minimizer as iteration goes on.

3.1.2 Conjugate and Bi-conjugate Frank-Wolfe Algorithms

Different from the original Frank-Wolfe algorithm, in CFW and BFW algorithms, the new search direction is obtained via mutually conjugate directions with respect to the Hessian of the objective. CFW considers the FW direction and the direction from the previous iteration while BFW takes into account one more direction from the last second iteration. For simplicity, procedures of obtaining the conjugate directions and updating the link flow are omitted here. For more details, please refer to [2]. The frameworks for CFW and BFW are listed in Alg.(2) and (3). To guarantee the feasibility of the solution, λ^i calculated by step 7 should fall in the interval $[0,1]$, and it is ensured by the following technique [2]:

$$\lambda^i = \frac{(f^{\text{C},i-1}-f^i)^{\text{T}}\nabla^2 z(f)(f^{\text{AON},i}-f^i)}{(f^{\text{C},i-1}-f^i)^{\text{T}}\nabla^2 z(f)(f^{\text{AON},i}-f^{\text{C},i-1})} = \frac{N_k}{D_k} \tag{23}$$

$$\lambda^i = \begin{cases} N_k/D_k, & if\ D_k \neq 0 \text{ and } N_k/D_k\ \in[0,1-\delta], \\ 1-\delta, & if\ D_k\neq 0 \text{ and } N_k/D_k > 1-\delta, \\ 0, & otherwise \end{cases} \tag{24}$$

Algorithm 2: Conjugate Frank Wolfe algorithm for traffic assignment problem

```
Input: Network topology $\boldsymbol{G} = (\boldsymbol{N}, \boldsymbol{A})$, Link capacity $\boldsymbol{C}$, Link free time $\boldsymbol{T}_0$
    Generate a initial empty flow
    Perform AON assignment and get the initial feasible flow $\boldsymbol{f}^1$
    Set iteration counter $i = 1$
1:  while 1 do
2:      Update the link cost using Eq.(1) and project it into path cost
3:      Perform AON assignment based on the updated path cost to generate new feasible
        flow $\boldsymbol{f}^{\mathbf{AON},i}$
4:      if ($i == 1$) then
5:          Define conjugate flow $\boldsymbol{f}^{\mathbf{C},i}$ to be just the auxiliary flow $\boldsymbol{f}^{\mathbf{AON},i}$
6:      else
7:          $\lambda^i = \frac{(\boldsymbol{f}^{\mathrm{C},i-1}-\boldsymbol{f}^i)^{\mathrm{T}}\nabla^2 z(\boldsymbol{f}^i)(\boldsymbol{f}^{\mathrm{AON},i}-\boldsymbol{f}^i)}{(\boldsymbol{f}^{\mathrm{C},i-1}-\boldsymbol{f}^i)^{\mathrm{T}}\nabla^2 z(\boldsymbol{f}^i)(\boldsymbol{f}^{\mathrm{AON},i}-\boldsymbol{f}^{\mathrm{C},i-1})}$
8:          Define conjugate flow $\boldsymbol{f}^{\mathrm{C,\ i}} = \lambda^i * \boldsymbol{f}^{\mathrm{C,\ i-1}} + (1-\lambda^i) * \boldsymbol{f}^{\mathrm{AON},i}$ to be
9:      end if
10:     Define the search direction as $\boldsymbol{f}^{\mathbf{C},i} - \boldsymbol{f}^i$
11:     Compute the step size $\theta^i$ either using Newton, Bisection or Golden Section method
12:     Update the link flow as $\boldsymbol{f}^{i+1} = \theta^i * \boldsymbol{f}^{\mathbf{C},i} + (1-\theta^i) * \boldsymbol{f}^i$
13:     Calculate the convergence index $\gamma = \frac{TSTT}{SPTT} - 1$
14:     if ($\gamma <$ Convergence Ratio) then
15:         Terminate the iteration
16:     else
17:         $i = i + 1$
18:     end if
19: end while
```

In step 18 of the Alg.(3), three coefficients $\beta_0^i, \beta_1^i, \beta_2^i$ need to be specified. To compute these three coefficients, one can refer [2] for details. In both algorithm 1 and 2, $TSTT, SPTT$ in the convergence index $\gamma = \frac{TSTT}{SPTT} - 1$ can be calculated as follows:

$$TSTT = \sum_{l_i \in A} t_{l_i}(f_{l_i}) f_{l_i} \tag{25}$$

$$SPTT = \sum_{l_i \in A} t_{l_i}(f_{l_i}^*) f_{l_i}^* \tag{26}$$

where $f_{l_i}^*$ is gained by performing one more **NOA** assignment using the link travel time calculated based on link flow given by current iteration.

Algorithm 3: Biconjugate Frank Wolfe algorithm for traffic assignment problem

```
Input: Network topology $\boldsymbol{G} = (\boldsymbol{N}, \boldsymbol{A})$, Link capacity $\boldsymbol{C}$, Link free time $\boldsymbol{T}_0$
    Generate a initial empty flow
    Perform AON assignment and get the initial feasible flow $\boldsymbol{f}^1$
    Set iteration counter $i = 1$
 1: while 1 do
 2:     Update the link cost using Eq.(1) and project it into path cost
 3:     Perform AON assignment based on the updated path cost to generate new feasible
        flow $\boldsymbol{f}^{\mathbf{AON},i}$
 4:     if $i == 1$ or $i == 2$ then
 5:         if $(i == 1)$ then
 6:             Define conjugate flow $\boldsymbol{f}^{\mathbf{B},i}$ to be just the auxiliary flow $\boldsymbol{f}^{\mathbf{AON},i}$
 7:         end if
 8:         if $(i == 2)$ then
 9:             $\lambda^i = \frac{(\boldsymbol{f}^{\mathrm{B},i-1}-\boldsymbol{f}^i)^{\mathrm{T}}\nabla^2 z(\boldsymbol{f}^i)(\boldsymbol{f}^{\mathrm{AON},i}-\boldsymbol{f}^i)}{(\boldsymbol{f}^{\mathrm{B},i-1}-\boldsymbol{f}^i)^{\mathrm{T}}\nabla^2 z(\boldsymbol{f}^i)(\boldsymbol{f}^{\mathrm{AON},i}-\boldsymbol{f}^{\mathrm{B},i-1})}$
10:             Define conjugate flow $\boldsymbol{f}^{\mathrm{B,\ i}} = \lambda^i * \boldsymbol{f}^{\mathrm{B,\ i-1}} + (1-\lambda) * \boldsymbol{f}^{\mathrm{AON},i}$
11:         end if
12:         Define the search direction as $\boldsymbol{f}^{\mathbf{B},i} - \boldsymbol{f}^i$
13:         Compute the step size $\theta^i$ either using Newton, Bisection or Golden Section
            method
14:         Update the link flow as $\boldsymbol{f}^{i+1} = \theta^i * \boldsymbol{f}^{\mathbf{B},i} + (1-\theta^i) * \boldsymbol{f}^i$
15:     end if
16:     if $i == 3$ then
17:         Define the search direction $D$ as
18:         $\beta_0^i(\boldsymbol{f}^{\mathrm{AON},i} - \boldsymbol{f}^i) + \beta_1^i(\boldsymbol{f}^{\mathbf{B},i-1} - \boldsymbol{f}^i) + \beta_2^i(\boldsymbol{f}^{\mathbf{B},i-2} - \boldsymbol{f}^i)$
19:         Compute the step size $\theta^i$ either using Newton, Bisection or Golden Section
            method
20:         Update the link flow as $\boldsymbol{f}^{i+1} = \theta^i * D + (1-\theta^i) * \boldsymbol{f}^i$
21:     end if
22:     Calculate the convergence index $\gamma = \frac{TSTT}{SPTT} - 1$
23:     if $(\gamma <$ Convergence Ratio$)$ then
24:         Terminate the iteration
25:     else
26:         $i = i + 1$
27:     end if
28: end while
```

3.2 Path-based Algorithm with Gradient Projection

The path-based algorithm exploits the O-D pair separability, and this group of methods operates in the space of path flows. Since the main idea of this paper is to apply nonlinear optimization to solve the transportation assignment problem, the details of the path-based algorithm is omitted here, and the framework of the path-based algorithm with gradient projection can be seen in [2].

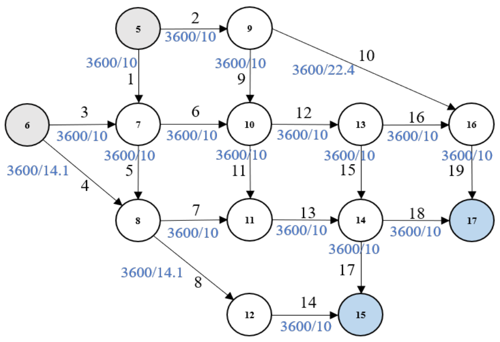
4 Test Result

Section 3 introduces two general types of user equilibrium algorithms, which are link-based and path-based algorithms. Among link-based algorithms, there are Frank-Wolfe

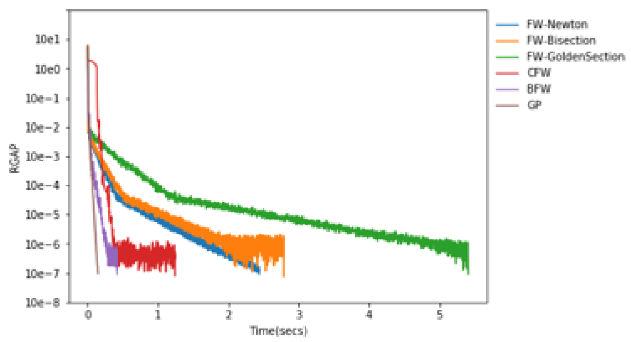
(FW), conjugate Frank-Wolfe (CFW), and biconjugate Frank-Wolfe (BFW) algorithms. Within the line search framework, newton, bisection and golden section methods are applied to find the minimizer along the line search direction. For path-based algorithms, only the gradient projection (GP) method is studied. Section 4 applies newton-FW, bisection-FW, golden section-FW, CFW, BFW and GP algorithms to solve user equilibrium state for different transportation networks. The performance of each algorithm is visualized in the form of the convergence ratio.

4.1 Case 1

Here is a network provided in [1]. The network topology as well as the parameters for solving the traffic assignment problem are shown in Fig.().



(a) Network adapted from [1]



(b) The convergence ratio of the six algorithms

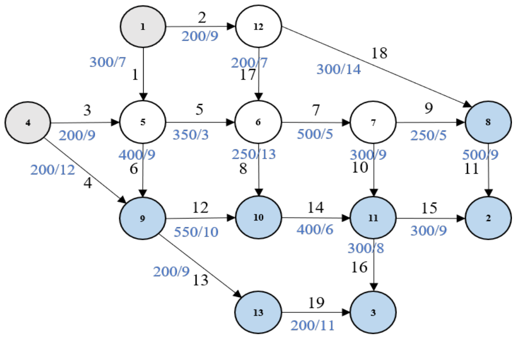
Figure 1: Solving traffic assignment problem for network 1

The nodes $5, 6$ in grey color are origins and nodes $15, 17$ are destinations. The demands of each O-D pair are $D_{5,15} = 6000, D_{5,17} = 6750, D_{6,15} = 7500, D_{6,17} = 5250$. The link capacity and free travel time are labeled on their corresponding arrows, e.g. 3600/10 indicates that the link capacity is 3600 and the free travel time is 10.

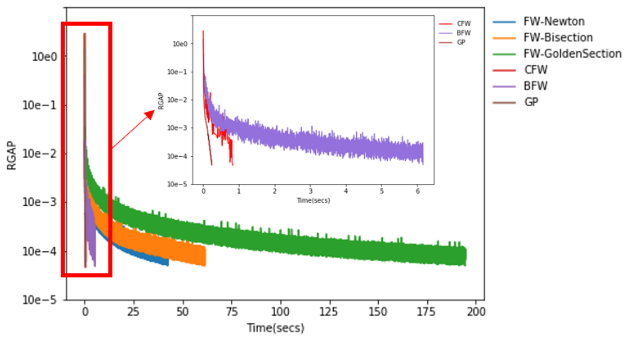
The flow along each link and the traveling time for paths between each O-D pair are listed in Appendix A. Comparing the traveling time of different paths between each O-D pair, it can be clearly seen that as long as two paths connect the same O-D pair and the flow on them are nonzero, their traveling costs are the same as required by the user equilibrium. From Fig.(1b), both GP, BFW and CFW have the better convergence performance compared with basic Frank-Wolfe algorithms with different line search methods. For the three FW algorithms, newton beats bisection and golden section ones, which makes sense since the convergence ratio of newton method is pretty faster than the other ones.

4.2 Case 2

Below is another network adapted from the Nguyen-Dupuis's 13-node network [6]. The demands of each O-D pair are $D_{1,2} = 660, D_{1,3} = 800, D_{1,10} = 800, D_{1,11} = 600, D_{4,2} = 412.5, D_{4,3} = 495, D_{4,8} = 700, D_{4,9} = 300, D_{4,10} = 300, D_{4,13} = 600$.



(a) Network adapted from [7]



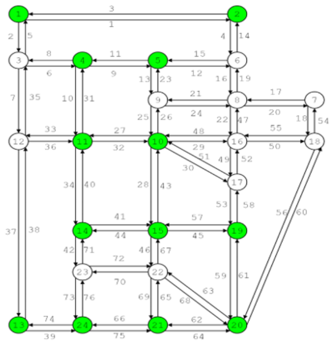
(b) The convergence ratio of the six algorithms

Figure 2: Solving traffic assignment problem for network 2

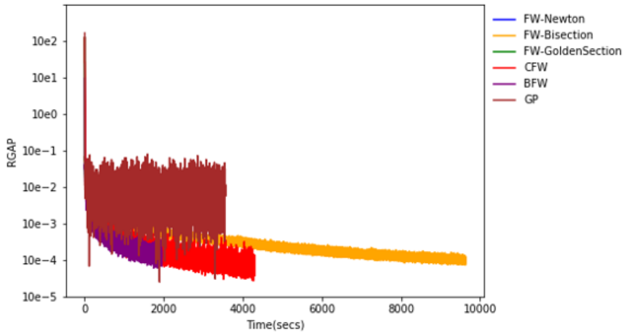
Similarly, the flow along each link and the traveling time for paths between each O-D pair are presented in Appendix B. And the performance of gradient projection method is much better than any other method. One thing worth noticing in this case is that the conjugate direction method beats the bi-conjugate direction method, which is opposite in case 1 where bi-conjugate direction method is faster than conjugate direction method.

4.3 Case 3

The following network is a classic SiouxFalls transportation network [5]. It has 24 nodes, 76 links and 576 O-D pairs. The network topology and the convergence ratio are as shown here:



(a) Network adapted from [5]



(b) The convergence ratio of the six algorithms

Figure 3: Solving traffic assignment problem for network 3

5 Program User Manual

This section guides you through the source code, and help you to set up your own transportation network and solve the user equilibrium problem. There are six .py files, three of which serve the leading role while other three .py files are used to input network data of the three previous examples. All codes are modified based on [1].

File **'graph.py'** creates a Graph class with the adjacent list as the input and inherits the Graph class to TrafficNetwork class. The code for the Graph class is revised from the classic graph package in Python. File **'model.py'** creates a TrafficFlowModel based on the TrafficNetwork class and adds methods exclusively used for solving the traffic assignment problem. The methods 'FW_solver_linesearch' (newton, bisection and golden section), 'solve_CFW', 'solve_BFW' and 'solve_GP' corresponds to the 6 optimization algorithms implemented in previous sections. File **'data.py'** is used for specifying information of the transportation network to be modeled. The information includes graph adjacent list, link capacity, link free travel time, list of 0-D pairs and their demand values. File **'main.py'** loads the two packages **'model.py'** and **'data.py'**, specifies the basic optimization parameters, runs the optimization algorithms.

To run the code, simply enter the information of your transportation network in the **data.py** file and then run the **'main.py'** file. The program will print the optimization information such as convergence ratio and the link flow during each iteration. When convergence ratio is less than the criterion, the program will print out the link flow, link time, path flow and path time information as well as the convergence ratio of the 6 algorithms.

References

[1] li, zheng. *Program for obtaining the user equilibrium solution with Frank-Wolfe Algorithm in urban traffic assignment.* Oct. 2019. URL: https://github.com/ZhengLi95/User-Equilibrium-Solution.

[2] Mitradjieva, Maria and Lindberg, Per Olov. "The stiff is moving—Conjugate direction Frank-Wolfe methods with applications to traffic assignment". In: *Transportation Science* 47.2 (2013), pp. 280–293.

[3] Perederieieva, Olga et al. "A framework for and empirical study of algorithms for traffic assignment". In: *Computers & Operations Research* 54 (2015), pp. 90–107.

[4] Sheffi, Y. "Equilibrium Analysis with Mathematical Programming Methods". In: *Massachusetts: Institute of Technology* (1985).

[5] Stabler, Ben, Bar-Gera, Hillel, and Sall, Elizabeth. *Transportation networks for research core team.* 2018.

[6] Xu, Xiangdong, Chen, Anthony, and Cheng, Lin. “Reformulating environmentally constrained traffic equilibrium via a smooth gap function”. In: *International Journal of Sustainable Transportation* 9.6 (2015), pp. 419–430.

[7] Zhang, Xiaoge and Mahadevan, Sankaran. “A bio-inspired approach to traffic network equilibrium assignment problem”. In: *IEEE transactions on cybernetics* 48.4 (2017), pp. 1304–1315.

[8] Zou, Qiling and Chen, Suren. “Resilience Modeling of Interdependent Traffic-Electric Power System Subject to Hurricanes”. In: *Journal of Infrastructure Systems* 26.1 (2020), p. 04019034.

A Solution of user equilibrium for network1

Link	Free-flow travel time (min/trip)	Capacity (veh/h)	Final link flow(veh/h)	Final link time(min)
1	10	3600	5632.68	18.99
2	10	3600	7117.32	32.92
3	10	3600	6048.31	21.95
4	14.1	3600	6701.69	39.50
5	10	3600	5392.05	17.55
6	10	3600	6288.95	23.97
7	10	3600	5191.43	16.49
8	14.1	3600	6902.30	42.68
9	10	3600	1481.14	10.04
10	22.4	3600	5363.18	42.59
11	10	3600	1648.04	10.07
12	10	3600	6122.05	22.54
13	10	3600	6839.47	29.54
14	10	3600	6902.30	30.27
15	10	3600	5303.10	17.06
16	10	3600	818.95	10.00
17	10	3600	6597.70	26.92
18	10	3600	5544.87	18.44
19	10	3600	6455.13	25.51

O-D pair	Path	Time	O-D pair	Path	Time
(5, 15)	5 - 7 - 8 - 11 - 14 - 15	109.487	(6, 15)	6 – 7 – 8 – 11 – 14 – 15	112.448
(5, 15)	5 – 7 – 8 – 12 – 15	109.486	(6, 15)	6 – 7 – 8 – 12 – 15	112.448
(5, 15)	5 – 7 – 10 – 11 – 14 – 15	109.486	(6, 15)	6 – 7 – 10 – 11 – 14 – 15	112.448
(5, 15)	5 – 7 – 10 – 13 – 14 – 15	109.488	(6, 15)	6 – 7 – 10 – 13 – 14 – 15	112.450
(5, 15)	5 – 9 – 10 – 11 – 14 – 15	109.486	(6, 15)	6 – 8 – 11 – 14 – 15	112.448
(5, 15)	5 – 9 – 10 – 13 – 14 – 15	109.488	(6, 15)	6 – 8 – 12 – 15	112.448
(5, 17)	5 – 7 – 8 – 11 – 14 – 17	101.006	(6, 17)	6 – 7 – 8 – 11 – 14 – 17	103.968
(5, 17)	5 – 7 – 10 – 11 – 14 – 17	101.006	(6, 17)	6 – 7 – 10 – 11 – 14 – 17	103.968
(5, 17)	5 – 7 – 10 – 13 – 14 – 17	101.008	(6, 17)	6 – 7 – 10 – 13 – 14 – 17	109.970
(5, 17)	5 – 7 – 10 – 13 – 16 – 17	101.008	(6, 17)	6 – 7 – 10 – 13 – 16 – 17	103.970
(5, 17)	5 – 9 – 10 – 11 – 14 – 17	101.006	(6, 17)	6 – 8 – 11 – 14 – 17	103.968
(5, 17)	5 – 9 – 10 – 13 – 14 – 17	101.008			
(5, 17)	5 – 9 – 10 – 13 – 16 – 17	101.008			
(5, 17)	5 – 9 – 16 – 17	101.025			

B Solution of user equilibrium for network2

Link	Free-flow travel time(min/trip)	Capacity (veh/h)	Final link flow(veh/h)	Final link time(min)
1	7	300	1727.06	1160.265
2	9	200	1132.94	1399.108
3	9	200	1443.71	3674.514
4	12	200	1363.79	3903.717
5	3	350	1741.27	278.681
6	9	400	1429.49	229.203
7	5	500	1593.07	82.289
8	13	250	621.15	87.312
9	5	250	781.66	76.676
10	9	300	811.41	81.244
11	9	500	741.66	15.535
12	10	550	1667.65	136.784
13	9	200	825.63	401.064
14	6	400	1188.80	76.217
15	9	300	330.84	10.997
16	8	300	1069.37	201.735
17	7	200	472.97	39.833
18	14	300	660.00	63.194
19	11	200	225.63	13.673

O-D pair	Path	Time	O-D pair	Path	Time
(1, 2)	1 - 5 - 6 - 7 - 8 - 2	1613.447	(4, 2)	4 – 5 – 6 – 7 – 8 – 2	4127.695
(1, 2)	1 - 5 - 6 - 7 - 11 - 2	1613.476	(4, 2)	4 – 5 – 6 – 7 – 11 – 2	4127.724
(1, 2)	1 - 5 - 6 - 10 - 11 - 2	1613.472	(4, 2)	4 – 5 – 6 – 10 – 11 – 2	4127.720
(1, 2)	1 – 5 – 9 – 10 – 11 – 2	1613.466	(4, 2)	4 – 5 – 9 – 10 – 11 – 2	4127.715
(1, 2)	1 – 12 – 6 – 7 – 8 – 2	1613.442	(4, 2)	4 – 9 – 10 – 11 – 2	4127.714
(1, 2)	1 – 12 – 6 – 7 – 11 – 2	1613.471	(4, 3)	4 – 5 – 6 – 7 – 11 – 3	4318.462
(1, 2)	1 – 12 – 6 – 10 – 11 – 2	1613.467	(4, 3)	4 – 5 – 6 – 10 – 11 – 3	4318.458
(1, 2)	1 – 12 – 8 – 2	1477.838	(4, 3)	4 – 5 – 9 – 10 – 11 – 3	4318.453
(1, 3)	1 – 5 – 6 – 7 – 11 – 3	1804.214	(4, 3)	4 – 5 – 9 – 13 – 3	4318.454
(1, 3)	1 – 5 – 6 – 10 – 11 – 3	1804.210	(4, 3)	4 – 9 – 10 – 11 – 3	4318.452
(1, 3)	1 – 5 – 9 – 10 – 11 – 3	1804.204	(4, 3)	4 – 9 – 13 – 3	4318.453
(1, 3)	1 – 5 – 9 – 13 – 3	1804.205	(4, 8)	4 – 5 – 6 – 7 – 8	4112.159
(1, 3)	1 – 12 – 6 – 7 – 11 – 3	1804.209	(4, 9)	4 – 5 – 9	3903.717
(1, 3)	1 – 12 – 6 – 10 – 11 – 3	1804.205	(4, 9)	4 – 9	3903.717
(1, 10)	1 – 5 – 6 – 10	1526.258	(4, 10)	4 – 5 – 6 – 10	4040.506
(1, 10)	1 – 5 – 9 – 10	1526.253	(4, 10)	4 – 5 – 9 – 10	4040.501
(1, 10)	1 – 12 – 6 – 10	1526.253	(4, 10)	4 – 9 – 10	4040.501
(1, 11)	1 – 5 – 6 – 7 – 11	1602.479	(4, 13)	4 – 5 – 9 – 13	4304.781
(1, 11)	1 – 5 – 6 – 10 – 11	1602.475	(4, 13)	4 – 9 – 13	4304.781
(1, 11)	1 – 5 – 9 – 10 – 11	1602.470			
(1, 11)	1 – 12 – 6 – 7 – 11	1602.474			
(1, 11)	1 – 12 – 6 – 10 – 11	1602.470			

It should be noticed that the travel time of path $1-12-8-2$ indeed is less than all other paths between O-D pair 1-2. At first glance it seems to be inconsistent with the Wardrop's first principle. However, if we look at the link 18 connecting node 12 and 8, the flow on it equails exactly the demand value of O-D pair 1-2, which means the flow on all other paths between O-D pair 1-2 is added due to other O-D pairs. All users who start from node 1 to node 2 only use path $1 - 12 - 8 - 2$ and since it takes the least time, the Wardrop's first principle still holds.