The Journal of Transport and Land Use http://jtlu.org
Vol. 13 No. 1 [2020] pp. 539–558

JTLU

A multiple-path gradient projection method for solving the logit-based stochastic user equilibrium model

Heqing Tan
College of Civil and Transportation Engineering, Hohai University
tanheqing@hhu.edu.cn

Muqing Du
College of Civil and Transportation Engineering, Hohai University
dumqhhu@hhu.edu.cn

Chun-bin Yu
College of Civil and Transportation Engineering, Hohai University
yuchunbinjx@163.com

Abstract: This paper proposes a path-based algorithm for solving the well-known logit-based stochastic user equilibrium (SUE) problem in transportation planning and management. Based on the gradient projection (GP) method, the new algorithm incorporates a novel multiple-path gradient approach to generate the descent direction in consideration of many paths existing in every single origin-destination (O-D) pair. To apply the path-based algorithm, the SUE problem is reformulated as a variational inequality (VI), and a working path set is predetermined. The numerical experiments are conducted on the Winnipeg network where a large number of paths are provided. The results show the multiple-path gradient projection algorithm outperforms the original GP method. Three different step size strategies, including the fixed step size, self-regulated averaging and self-adaptive Armijo's strategies, are involved to draw a more general conclusion. Also, the effects of the path number on computational performance are analyzed. The multiple-path gradient projection (MGP) method converges much faster than the GP method when the path set size gets large.

Article history:
Received: April 18, 2019
Received in revised form: October 8, 2019
Accepted: April 8, 2020
Available online: November 10, 2020

1 Introduction

Traffic assignment is a key step of the travel demand forecasting or project evaluation. The results will be treated as an important basis of the decision making in urban transportation planning and management. To obtain high-quality forecast results, traffic assignment is a crucial need for transportation engineers and administrators. The stochastic user equilibrium (SUE) model is one of well-known traffic assignment models in the literature. Assuming a probabilistic route choice model, the SUE model is to find the link (and path) flow pattern on a traffic network given the travel demand between origins and destinations and the corresponding path choice sets (either explicit or implicit). Daganzo and Sheffi (1977) defined SUE as a state in which no driver can reduce his/her perceived travel time by unilaterally changing routes.

The difference between the SUE problem and deterministic user equilibrium (DUE) problem is that the SUE relaxes the perfect information assumption by incorporating a random error term in the route cost function to simulate travelers' imperfect perceptions of travel time. The imperfect perceptions are caused by a lack of information among the travelers about which routes are the shortest, travelers' different perceptions on path cost or different preferences (Sheffi, 1985).

The random error terms are usually assumed to follow the Gumbel distribution (Dial, 1971) or the Normal distribution (Daganzo & Sheffi, 1977), corresponding to the multinomial logit (MNL) and multinomial probit (MNP) discrete choice models, respectively. Compared with the MNP model, the MNL model is unable to account for overlapping and perception variance with respect to different path lengths (Sheffi, 1985; Chen, Pravinvongvuth, Xu, Ryu, & Chootinan, 2012; Chen, Ryu, Xu, & Choi, 2014). The inferiors result from assuming that the random error terms are independently and identically distributed (IID). The MNP model does not have these inferiors because it considers the covariance between the random error terms. However, the MNP model does not have a closed-form probability expression and it is computationally burdensome (Xu, Chen, Zhou, & Bekhor, 2012; Chen et al., 2014). Therefore, the MNL model is more suitable in practice.

To overcome the inabilities of the MNL SUE model, researchers have greatly extended the logit model. These extensions include the path-size logit (Ben-Akiva & Bierlaire, 2003), cross-nested logit paired (Prashker & Bekhor, 1998), combinatorial logit (Prashker & Bekhor, 1998), C-logit (Zhou, Chen, & Bekhor, 2012), etc. These extensions allow more realistic descriptions of travelers' path choice behaviors. For a more comprehensive review of these extended models, readers can refer to the excellent discussions provided by Prashker and Bekhor (1998), Chen, Pravinvongvuth, Xu, Ryu, and Chootinan (2012), and Kitthamkesorn and Chen (2013).

It is well known that the SUE problem can be formulated and solved either in the space of link flow or in the space of path flow (Fisk, 1980; Chen, & Alfa, 1991; Damberg, Lundgren, & Patriksson, 1995; Bekhor & Toledo, 2005; Bekhor, Toledo, & Reznikova, 2009; Xu & Chen, 2013; Chen, et al., 2014). In the early period, the link-based formulation was used widely. An important advantage of the link-based formulation is that they do not require explicit enumeration of the path choice set, so it can be applied to large-scale networks when the storage memory is limited. Recently, the path-based formulation is adopted more widely, because it allows a more flexible definition of the choice set (Damberg et al., 1995; Bekhor & Toledo, 2005; Xu et al., 2012). Besides, storage memory is also no longer a serious bottleneck to employ path-based algorithms for solving traffic assignment problems (Jayachrishnan, Tsai, & Prashker, 1994).

The Gradient projection (GP) method, a type of path-based algorithm, shows successful performance to solve various network equilibrium problems, e.g., the DUE problem (Jayachrishnan, Tsai, & Prashker, 1994; Chen, Lee, & Jayakrishnan, 2002), the non-additive traffic equilibrium problem (Chen, Zhou, & Xu, 2012), the elastic-demand traffic equilibrium problem (EDTEP) (Ryu, Chen, & Choi, 2014), the combined modal split and traffic assignment problem (CMSTA) (Ryu, Chen, & Choi, 2017), etc. However, the GP algorithm shifting flow to the current shortest path from all the other paths may be not very suitable to solve the SUE problem since there are usually plenty of paths in the given path set within each O-D pair.

The focus of this study is on developing an efficient path-based algorithm to solve the logit-based SUE problem. The new method shifts flow among all paths rather than to the current shortest path from all other paths. Some computational experiments are conducted in the test network. The numerical results show the new method has better performance to solve the SUE problem than the GP method. Also, the outperformance of the new proposed algorithm is analyzed under the situation in which different amounts of paths are involved within each O-D pair.

2 Formulation of the logit-based sue assignment problem

This section briefly reviews the formulation of the logit-based SUE problem in terms of path flow. Also, the mathematical program is reformulated into a variational inequality.

The transportation network can usually be abstracted into a directed graph in which the nodes represent the origins, destinations and intersections, and the links (or arcs) represent the streets and highways. In this paper, a link, connecting two nodes, is denoted by a. A and N denote the sets of links and nodes, respectively. Let $R \subset N$ be the origin set, and $S \subset N$ be the destination set. A path (or route), a sequence of directed links leading from one node to another, is denoted by k. The flow of link a, denoted by x_a, refers to the number of the travelers using link a (link flow). Similarly, the path flow represents the number of travelers using path k (path flow), which is denoted by f_k. Let t_a be the cost of link a, which represents the cost that a traveler takes to traverse link a (link cost). In traffic assignment, the link cost is a monotonically increasing function with respect to the link flow. The cost on path k is defined as the sum of the costs of all links comprising path k, and is denoted by c_k (path cost). Let q_{rs} denote the travel demand (number of total trips) from origin r to destination s. Let K_{rs} be the set of active paths (paths that are used) between r and s. In this paper, the notations are all scalars if there are not specific statements. The path flow pattern can be calculated by the logit model

$$f_k = q_{rs}\exp(-\theta c_k)/\textstyle\sum_{k\in K_{rs}}\exp(-\theta c_k)\,,\ \forall k\in K_{rs} \tag{1}$$

where θ is a positive dispersion parameter, which reflects an aggregate measure of drivers' perception of path cost (Sheffi, 1985).

Fisk (1980) developed the following mathematical programming formulation for the logit-based SUE problem

$$min\, z(x)=\sum_{a\in A}\int_0^{x_a} t_a(\omega)\,d\omega + \frac{1}{\theta}\sum_{r\in R,\, s\in S}\ \sum_{k\in K_{rs}} f_k ln f_k \tag{2}$$

subject to

$$\sum_{k\in K_{rs}} f_k = q_{rs},\ \forall r\in R,\ s\in S \tag{3}$$

$$f_k \geq 0,\ \forall k\in K_{rs},\ r\in R,\ s\in S \tag{4}$$

$$x_a = \sum_{r\in R}\sum_{s\in S}\sum_{k\in K_{rs}} f_k \delta_{a,k}^{rs}\,,\ \forall a\in A \tag{5}$$

where $z(x)$ is the objective function; $\delta^{rs}_{a,k}$ is a link/path indicator, which equals 1 if link a is on path $k \in K_{rs}$, and 0 otherwise. Note that $\lim_{f_k \to 0} f_k \ln f_k = 0$ and therefore $f_k \ln f_k$ can be assumed to be zero for $f_k = 0$.

To solve the SUE model by using the path-based algorithm, we reformulated the above MP into a variational inequality (VI) in terms of path flow. That is, find $\mathbf{f}^* \in \mathbf{\Omega}$, such that

$$\mathbf{C}(\mathbf{f}^*)(\mathbf{f} - \mathbf{f}^*) \geq 0, \forall \mathbf{f} \in \mathbf{\Omega} \tag{6}$$

where the mapping $\mathbf{C}$ is the vector form of the perceived path cost function with respect to path flow $\mathbf{f}$. The perceived path cost function is given by

$$C_k(f_k) = c_k(f_k) + \frac{1}{\theta}(1 + \ln f_k) \tag{7}$$

Also, the feasible set $\mathbf{\Omega}$ consists of Eq. (3)-(5). The next section presents a novel solution algorithm for the SUE problem.

3 A multiple-path gradient projection method

3.1 A brief description of the GP algorithm

Jayakrishnan et al. (1994) adopted the Goldstein-Levitin-Polyak gradient projected algorithm to solve the user equilibrium traffic assignment problem with fixed demand. During the past two decades, this algorithm has been applied to different traffic assignment models on realistic networks (see Section 1), and the numerical experiment results of these applications are quite encouraging.

The GP algorithm starts with a given estimate of the optimal solution (the objective function value is lowest at the optimal solution). At each iteration, the algorithm first generates a descent direction and then determines a suitable step size to produce a new estimate of the optimal solution (also called as an approximate solution). The descent directions are such vectors along which the objective function value can be decreased with the current approximate solution moving. A good descent direction allows the objective function value to decrease sharply. The step size is used to guide how far the current approximate solution should move along the descent direction. After that, the GP algorithm employs the projection operation to make the newly obtained approximate solution feasible (i.e., satisfy the constraints). This procedure is repeated until the algorithm obtains an approximate solution with some level of precision (the precision indicates how close the current approximate solution gets to the optimal solution). For path-based traffic assignment solution algorithms, the approximate solutions are produced by transferring the flow from some paths to the others, which is called flow transfer (or shift) in this paper. In traffic assignment, the convergence speed is widely used to evaluate how far the solution algorithms converge to the optimal solution. The convergence speed can be measured by the computing time required to obtain an approximate solution with some level of precision.

As a typical path-based algorithm, the GP algorithm exploits the O-D pair decomposition scheme. The original problem is separated into several subproblems, each with respect to only one O-D pair. Then, these subproblems are solved sequentially. Consequently, the focus is on every single O-D subproblem. Given an O-D pair rs, the objective function of the traffic assignment problem for a specific O-D pair rs can be approximated by the second-order Taylor expansion as follows

$$Z_{rs}(\mathbf{f}) \approx Z_{\mathrm{rs}}(\mathbf{f}^n) + \nabla Z_{rs}(\mathbf{f}^n)^{\mathrm{T}}(\mathbf{f}-\mathbf{f}^n) + \frac{1}{2}(\mathbf{f}-\mathbf{f}^n)^{\mathrm{T}}\mathbf{H}_{rs}(\mathbf{f}^n)(\mathbf{f}-\mathbf{f}^n) \tag{8}$$

where $\mathbf{f}$ denotes the path flow vector; $\mathbf{f}^n$ is the path flow vector at nth iteration; $\mathbf{H}_{rs}(\mathbf{f}^n)$ is the Hessian matrix at $\mathbf{f}^n$.

Some terms in Eq. (8) are constant and the problem can be simplified as follows

$$min\ Z_{rs} = \nabla Z_{rs}(\mathbf{f}^n)^{\mathrm{T}}\mathbf{f} + \frac{1}{2}(\mathbf{f}-\mathbf{f}^n)^{\mathrm{T}}\mathbf{H}_{rs}(\mathbf{f}^n)(\mathbf{f}-\mathbf{f}^n) \tag{9}$$

subject to

$$\mathbf{f}^{\mathrm{T}}\mathbf{E} = q_{rs} \tag{10}$$

$$f_k \geq 0, \forall k \in K_{rs} \tag{11}$$

where $\mathbf{E}$ is the unit column vector $(1, \ldots, 1)^{\mathrm{T}}$.

In the GP algorithm, the demand conservation constraint is eliminated by expressing the shortest path flow with the others, which is

$$f_{\bar{k}} = q_{rs} - \sum_{k \in K_{rs},\ k \neq \bar{k}} f_k \tag{12}$$

where $\bar{k}$ is the shortest path.

Thus, the problem is transformed into

$$min\ \tilde{Z}_{rs} \tag{13}$$

subject to

$$f_k \geq 0, \forall k \in K_{rs} \tag{14}$$

where $\tilde{Z}_{rs}$ is the new objective function in terms of non-shortest path flow. The first-order derivative of the new problem can be derived

$$\frac{\partial \tilde{Z}_{rs}}{\partial f_k} = \frac{\partial Z_{rs}}{\partial f_k} - \frac{\partial Z_{rs}}{\partial f_{\bar{k}}} \quad \forall k \in K_{rs}, k \neq \bar{k} \tag{15}$$

Note that the first-order derivative of Z_{rs} with respect to path k is simply the sum of the link costs on that path, which is

$$\frac{\partial Z_{rs}}{\partial f_k} = c_k = \sum_{a \in \mathrm{A}} t_a(x_a)\, \delta_{a,k}^{rs} \tag{16}$$

where c_k denotes the path cost on path k. Besides, the second-order derivative is

$$\frac{\partial^2 Z_{rs}}{\left(\partial f_k\right)^2}=\sum_{a\in A} t_a'(x_a)\,\delta_{a,k}^{rs} \tag{17}$$

where $t_a'(x_a)$ is the first-order derivative of the travel time function on link a.

If the non-negative constraint is not considered contemporarily, the transformed problem turns into an unconstraint optimization problem and the stationary point can be easily calculated by

$$\nabla\tilde{Z}_{rs}=\nabla\tilde{Z}_{rs}(\tilde{\mathbf{f}}^n)+\mathbf{H}_{rs}(\tilde{\mathbf{f}}^n)(\tilde{\mathbf{f}}-\tilde{\mathbf{f}}^n)=0 \tag{18}$$

where $\tilde{\mathbf{f}}$ is the path flow vector excluding the shortest path $\bar{k}$.

Once the Hessian matrix, $\mathbf{H}_{rs}(\tilde{\mathbf{f}}^n)$, is assumed to be diagonal, the following equation can be obtained

$$\frac{\partial Z_{rs}}{\partial f_k^n}-\frac{\partial Z_{rs}}{\partial f_{\bar{k}}^n}+\left(\frac{\partial^2 Z_{rs}}{\left(\partial f_k^n\right)^2}+\frac{\partial^2 Z_{rs}}{\left(\partial f_{\bar{k}}^n\right)^2}-2\frac{\partial^2 Z_{rs}}{\partial f_k^n\partial f_{\bar{k}}^n}\right)(f_k'-f_k^n)=0,\ \forall k\in K_{rs},\ k\neq\bar{k} \tag{19}$$

where $f_k'-f_k^n$ is the flow increase on path k and denoted as Δf_k in the following.

A small increase in the flow on path k results in an equal decrease in the flow on the shortest path. Obviously, there is no change in the flow on the common part of the two paths. The flow shift between path k and the shortest path can be calculated as

$$\Delta f_k=\frac{c_k-c_{\bar{k}}}{\sum_{a\in A_{k,\bar{k}}} t_a'(x_a)},\ \forall k\in K_{rs},\ k\neq\bar{k} \tag{20}$$

where $A_{k,\bar{k}}$ is the set of links belong either to path k or to $\bar{k}$, but not both.

To avoid violating the non-negative constraint, the new solution will be projected onto the feasible region

$$f_k^{n+1}=f_k^n-\min\{\alpha\Delta f_k,f_k^n\},\ \forall k\in K_{rs},\ k\neq\bar{k} \tag{21}$$

$$f_{\bar{k}}^{n+1}=q_{rs}-\sum_{k\in K_{rs},\ k\neq\bar{k}} f_k \tag{22}$$

where α is the step size, $\alpha\in(0,1]$. Here we do not specify a particular step size determination strategy, because different strategies can be embedded into the GP algorithm.

To better understand the GP algorithm, we use the example in figure 1 to illustrate how this algorithm works. Consider that there are three paths for a given O-D pair, and path 3 is the shortest. The solid lines are the first-order approximation (by Taylor's expansion) of the path cost functions based on the current path flow (i.e., f_k^n, $k=1, 2, 3$). The purple line shows that the corresponding paths are equilibrated based on the linearized cost functions. The transferred flow (Δf_k^n, $k=1, 2, 3$) of the paths can be calculated as dividing the cost difference between non-shortest and shortest paths by the sum of the slopes of the corresponding path cost functions, as Eq. (20) presents.

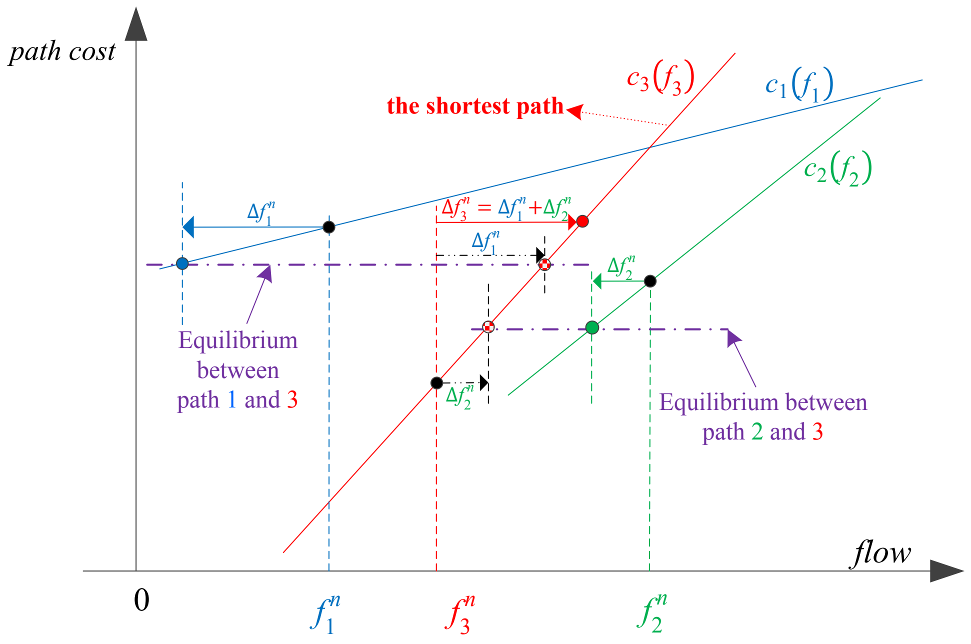

Figure 1. llustration of the path flow transfer strategy in the GP method

Remark 1: The flow transfer scheme of the GP algorithm is to shift the path flow from each non-shortest path to the shortest one (the flow on paths 1 and 2 decreases and the flow on path 3 increases). Since the path cost will not be updated right after flow transfer, the shortest path flow could be over high after the flow shifts of all paths (for example, the shortest path becomes the longest one in figure 1). Besides, the flow shifts between the non-shortest ones are not involved. This strategy can have good performance to equilibrate a few paths. But when the path number grows, which is usual for the SUE problem formulated in the space of path flows, it is difficult to calculate a good descent direction. In this paper, we develop a new path-based algorithm to deal with this issue, based on the GP algorithm.

3.2 The proposed multiple-path Gradient Projection algorithm

Since the GP algorithm only shifts path flows from each non-shortest path to the shortest one, it may not have good performance in achieving equilibration among a large number of paths. To avoid this, we seek a new approach to solve the problem Eq. (9) with constraints Eqs. (10)-(11).

Similar to the GP algorithm, we do not consider the non-negative constraint at first, which is one of the most fascinating characteristics for solving large-scale problems by the Goldstein-Levitin-Polyak algorithm. In contrast, for the algorithms like Rosen's gradient projection method (Rosen, 1960), the motion along the descent direction stops once a new constraint is encountered, which will slow the convergence speed in large-scale problems with many constraints binding at the optimal solution (Bertsekas, 1976). Then, the problem Eq. (9) with constraint Eq. (10) is an optimization problem with only the equation constraint, and can be directly solved by the method of Lagrange multipliers. The optimal solution will satisfy

$$\nabla Z_{rs}(\boldsymbol{f}^{n}) + \mathbf{H}_{rs}(\mathbf{f}^{n})(\mathbf{f}-\mathbf{f}^{n}) = \tau\mathbf{E} \tag{23}$$

Once the Hessian matrix is assumed to be diagonal, Eq. (23) can be expressed as follows

$$\frac{\partial Z_{rs}}{\partial f_k^n}+\frac{\partial^2 Z_{rs}}{\left(\partial f_k^n\right)^2}\Delta f_k = \tau \ \ \forall k\in K_{rs} \tag{24}$$

Eq. (24) has clear physical meanings. The left terms can be viewed as the linear approximation of the path cost function and the right term is the path cost after equilibration. It is indicated in Eq. (24) that the linearized path cost functions of all paths should be equal, which is consistent with the idea of the UE condition.

From Eq. (24), we have

$$\Delta f_k^n = \frac{\tau\text{-}c_k(f_k^n)}{s_k^n(f_k^n)} \ \ \forall k\in K_p^{rs} \tag{25}$$

where $s_k^n\left(f_k^n\right) = \frac{\partial c_k(f_k^n)}{\partial f_k^n} = \frac{\partial^2 Z_{rs}}{\left(\partial f_k\right)^2} = \sum_{a\in \mathrm{A}} t_a^{'}(x_a)\,\delta_{a,k}^{rs};$

Besides, according to the demand conservation condition, the sum of all path flows will not change after the equilibration. Therefore, we have

$$\sum_k \Delta f_k^n = \sum_k \frac{\tau\text{-}c_k\left(f_k^n\right)}{s_k^n\left(f_k^n\right)} = 0 \tag{26}$$

Rearrange Eq. (26), and obtain

$$\tau = \frac{\sum_k \frac{c_k\left(f_k^n\right)}{s_k^n\left(f_k^n\right)}}{\sum_k \frac{1}{s_k^n\left(f_k^n\right)}} \ \ \forall k\in K_p^{rs} \tag{27}$$

Substituting Eq. (27) for τ in Eq. (25),

$$\Delta f_k^n = \frac{\frac{\sum_k \frac{c_k\left(f_k^n\right)}{s_k^n\left(f_k^n\right)}}{\sum_k \frac{1}{s_k^n\left(f_k^n\right)}} \text{-} c_k\left(f_k^n\right)}{s_k^n\left(f_k^n\right)} \ \ \forall k\in K_p^{rs} \tag{28}$$

Then, an approximate solution can be obtained by

$$f_k^{n+1} = f_k^n + \alpha\Delta f_k^n \ \ \forall k\in K_p^{rs} \tag{29}$$

Where α is a predefined step size, $\alpha \in [0,1]$. Different step size strategies can be embedded into the proposed algorithm and some promising ones will be introduced in the next section.

The new method is referred to as "multiple-path gradient projection method" (for short, MGP) in this study. It attempts to shift the flow among all the paths. Figure 2 illustrates the flow transfer strategy in the MGP method. As indicated in Eq. (24), the goal of transferring path flow is to make all path costs equal. The path cost after flow transfer, τ (the purple horizontal line), can be calculated by Eq. (27), which indicates that the new flow on each path is the abscissa of the intersection point of the corresponding solid line and the purple horizontal dash line. As shown in figure 2, the cost of path 1 is greater than τ but the other two are not, so equilibrium can be achieved by shifting flow from path 1 to paths 2 and 3. The transferred flow can be calculated by dividing the difference between the path cost and τ by the solid line's slope, as Eq. (28) presents.

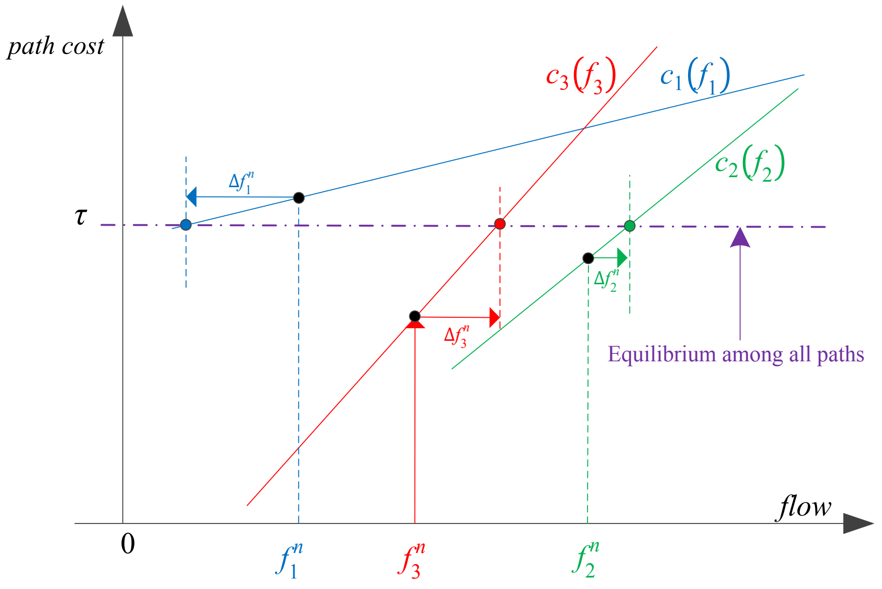

Figure 2. Illustration of the path flow transfer strategy in the MGP method

Remark 2: Comparing figures 1 and 2, we can see that the MGP algorithm intends to achieve equilibrium among all paths, while the GP algorithm tries to equilibrate each non-shortest path and the shortest one. Note that when there are only two paths between the O-D pair, these two algorithms are almost the same. The MGP method can define a better descent direction than the GP method with the growth of the path number within each O-D pair, because the latter does not focus on shifting flow among the non-shortest paths.

In addition, note that the above formulation does not take the non-negative constraint into account, so a projection strategy should be developed to avoid the violation of feasibility. When there is at least one path carrying flow in the new approximate solution, the path set K_p^{rs} is divided into two parts, $\underline{P}$ and $\bar{P}$. $\underline{P}$ consists of paths with the flow decreased $(\Delta f_k^n < 0)$ and $\bar{P}$ contains paths on which the flow did not decrease $(\Delta f_k^n < 0)$. Only the paths in $\underline{P}$are possible to carry negative flow, and the negative flow should be projected to 0, which is given by

$$f_k^{n+1} = max\{f_k^n + \alpha\Delta f_k^n, 0\} \ \forall k \in \underline{P} \tag{30}$$

where f_k^{n+1} is the new flow on path *k*;

The total path flow, required by paths in $\underline{P}$ to avoid the violation, is denoted as π which can be calculated by

$$\pi=\sum_k\{f_k^{n+1}-(f_k^n+\alpha\Delta f_k^n)\}\ \forall k\in\underline{P} \tag{31}$$

Besides, to satisfy the demand conservation condition, the total required path flow can be provided by the paths in $\bar{P}$. If the flow on each path in $\bar{P}$ decreases by the proportion of Δf_k^n, the path flow on paths in $\bar{P}$ can be calculated by

$$f_k^{n+1}=f_k^n+\alpha\Delta f_k^n-\pi\frac{\Delta f_k^n}{\sum_k\Delta f_k^n}\ \forall k\in\bar{P} \tag{32}$$

The projection process is illustrated in figure 3. Consider a special case in the above example shown in figure 2. As shown in figure 3, the flow on path 1 (i.e., the left black dot abscissa) is negative (assume that the step size, α , equals 1), which violates the non-negative constraint. So a projection process should be executed to satisfy the non-negative constraint. According to Eq. (30), the flow on path 1 will be projected to 0. For paths 2 and 3, the decreased flow on each path is proportional to the corresponding element of the descent direction (i.e., Δf_k^n), and the absolute sum of the decreased flow should be equal to the increased flow on path 1 according to the demand conservation constraint. Then, a new feasible approximate solution, $\{0,f_2^{n+1},f_3^{n+1}\}$, is provided. It is noteworthy that the negative path flow will be projected not to 0, but to a sufficiently small amount of flow in solving the SUE problem, since the logarithmic term (Eq. (7)) cannot admit zero path flow.

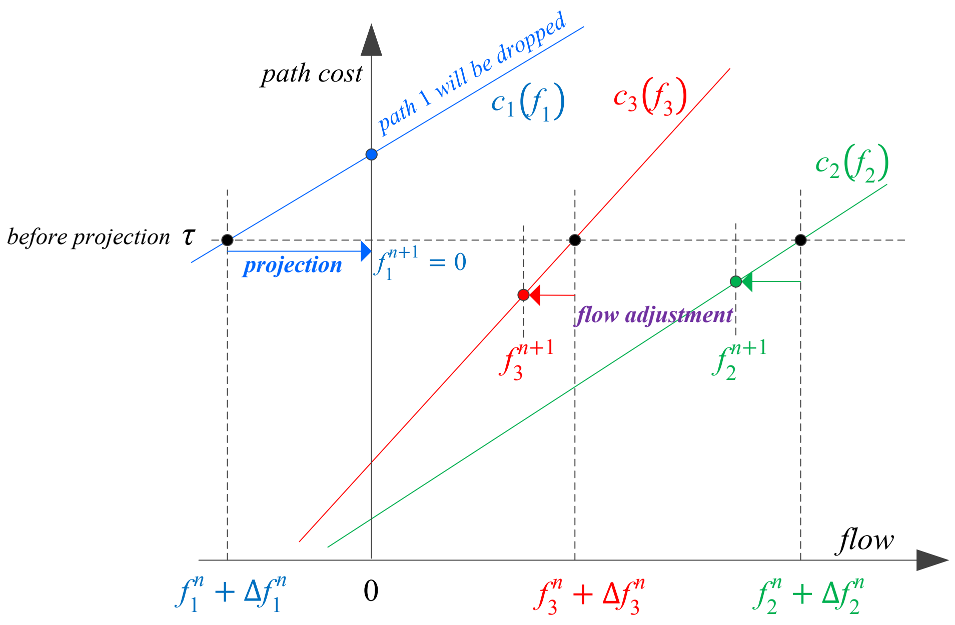

Figure 3. Illustration of the projection process of the MGP method

The GP method and proposed MGP method have a lot in common. For instance, both methods calculate the flow shift with the second-order derivative of the objective function and assume the Hessian matrix to be diagonal; the projection approach is also similar, in which the motion along the direction does not stop when a new constraint is encountered. The difference between them is the way to solve the approximated problem (Eqs. (9)-(10)). The GP method transforms it into a problem without the equation constraint by expressing the shortest path flow with the others. This strategy can do well in equilibrating a few paths within each O-D pair, but may perform poorly while the path number grows, which is usual for the SUE problem formulated in the space of path flow. On the other hand, the MGP method tries to achieve equilibrium directly among all the paths under the circumstance where the original problem is approximated as a quadratic program. The path flow shifts are no longer limited to between each non-shortest path and the shortest one. As a result, the new method can converge faster than the GP method in solving the logit-based SUE problem.

4 Adaption of the MGP algorithm for solving SUE problems

4.1 Determination of step size

Step size determination is a significant step for solving the SUE problem. In the literature, there are three main types of step size determination schemes: predetermined, adaptive and optimized step sizes (Gibb, 2016).

The predetermined step size schemes provide a sequence step size without considering any properties of specific problems. The most widely used predetermined step size scheme is the method of successive averages (MSA). The MSA generates a predetermined sequence $\{\alpha_n\}$ that satisfies Blum's theorem (Blum, 1954) to guarantee the convergence (i.e., $\alpha_n \rightarrow 0, \sum_{n=0}^{\infty} \alpha_n = \infty$). The MSA is quite simple to implement, but it suffers from a sublinear convergence rate (Liu, He, & He, 2009; Chen et al., 2014). To improve the convergence speed of the MSA, some researchers have proposed alternative predetermined schemes to slow down the decrease of the step size. For eaxmple, Polyak (1990) suggested a new predetermined step size sequence as $\alpha_n = n^{-2/3}$, and Nagurney and Zhang (1996) proposed to determine the sequence as $\{\alpha_n\} = \{1, \frac{1}{2}, \frac{1}{2}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \ldots, n \text{ repetitions of } \frac{1}{n}\}$. Unfortunately, these inspiring works lead to only minor improvement in convergence speed, as indicated in the results presented by Liu et al. (2009). Recently, Bar-Gera and Boyce (2006), and Boyce, O'Neill, and Scherr (2008) empirically found appropriately chosen fixed (constant) step sizes can converge much faster than the MSA for various models.

The adaptive step size schemes try to distinguish whether the iterations tend to diverge or converge based on certain measures, and then adaptively adjusts the step size (the step size will be decreased if the iterations tend to diverge; otherwise, the step size should be increased). The most widely used adaptive step size scheme is the self-regulated averaging (SRA) method (Liu et al., 2009). The SRA method is developed on the basis of the MSA, but an adaptive step size adjustment rule is incorporated.

The optimized step size schemes determine the step size based on the principle that reduces the objective function along the given descent direction. Chen and Alfa (1991) suggested the exact line search method (e.g., the golden section and bi-section methods) for determining the step size, while Maher (1998) proposed the optimal step length algorithm (OSLA) for solving the SUE problem. Bekhor and Toledo (2005) and Chen et al. (2014) employed the Armijo's rule for solving various SUE models and both reported promising results. Recently, Chen, Xu, Ryu, and Zhou (2013) proposed the self-adaptive Armijo step size strategy to accelerate the convergence of the Armijo's scheme.

Three promising step size strategies, including the fixed, self-regulated averaging and self-adaptive Armijo's step size schemes (corresponding to three types of step size), are detailed below.

4.1.1 Fixed step size scheme

The fixed step size scheme defines an appropriate constant step size for all iterations. This scheme is quite simple to implement and an appropriate step size can indeed accelerate the convergence (Bar-Gera & Boyce, 2006; Boyce et al., 2008). However, it is not an easy task to choose an appropriate constant step size, since the choice of step size is problem-specific and too large a step size will lead to divergence. It usually requires to conduct a series of experiments (trial-and-error method) for an appropriate choice of step size (Perederieieva, Ehrgott, Raith, & Wang 2015; Gibb, 2016).

4.1.2 Self-regulated averaging (SRA) scheme

This scheme utilized the information between iterations to guide the choice of step size either "speeding up" or "slowing down." The absolute residual error, $\| y_k - x_k \|$, is treated as the monitor of the convergence. The SRA scheme satisfies Blum's theorem (Blum, 1954) to guarantee the convergence. The step size is calculated by

$$\alpha_k = \frac{1}{\mu_k}, \mu_k = \begin{cases} \mu_{k-1} + \psi, \psi > 1, \text{if } \|\mathbf{y}_k - \mathbf{x}_k\| \geq \|\mathbf{y}_{k-1} - \mathbf{x}_{k-1}\| \\ \mu_{k-1} + \varphi, \varphi < 1, \text{if } \|\mathbf{y}_k - \mathbf{x}_k\| < \|\mathbf{y}_{k-1} - \mathbf{x}_{k-1}\| \end{cases} \tag{33}$$

The SRA scheme has been adopted to solve various SUE models, e.g., the MNL SUE model (Liu et al., 2009), the C-logit SUE problem with elastic demand (Xu & Chen, 2013), the pair combinatorial logit SUE model (Chen et al., 2014), the unconstrained weibit-based SUE problem (Kitthamkesorn & Chen, 2014), and the weibit-based SUE problem with elastic demand (Kitthamkesorn, Chen, & Xu, 2015). These studies reported that the SRA scheme is quite promising.

4.1.3 Self-adaptive Armijo (SAA) step size strategy

Armijo's rule (Armijo, 1967), one of the most popular inexact line search strategies, was originally proposed for unconstrained optimization problems. Bertsekas (1976) proposed a generalized Armijo strategy for constrained optimization problems. Recently, Chen et al. (2013) suggested the self-adaptive Armijo step size strategy, allowing the starting step size of each iteration to increase by exploiting the information derived from former iterations. The self-adaptive Armijo's step size can be described as:

Given that x_k is not an optimal solution, set

$$\alpha_k = \beta^m \gamma_k \tag{34}$$

Where m_k is the first nonnegative integer such that

$$Z(\mathbf{x}_k) - Z\big(\mathbf{x}_k(\beta^m \gamma_k)\big) \geq \sigma \nabla Z(\mathbf{x}_k)^{\mathrm{T}} \big(\mathbf{x}_k - \mathbf{x}_k(\beta^m \gamma_k)\big) \tag{35}$$

Where $\sigma \in (0,1/2)$ and $\beta \in (0,1)$ are fixed scalars, and $\gamma_k > 0$. Eq. (35) is to find a suitable step size in iteration k. To set a "smart" initial step size for iteration $k+1$, the following criterion is used:

If

$$Z(\mathbf{x}_k) - Z(\mathbf{x}_k(\beta^m\gamma_k)) \geq \eta \nabla Z(\mathbf{x}_k)^{\mathrm{T}}(\mathbf{x}_k - \mathbf{x}_k(\beta^m\gamma_k)) \tag{36}$$

Where $\eta \in (0,1)$ is a fixed scalar, then

$$\gamma_{k+1} := min\{\alpha_k\rho, \bar{\gamma}\} \tag{37}$$

Where $\rho > 1$ is a fixed scalar and $\bar{\gamma} > 0$ is a predetermined constant as the upper bound of step size. Otherwise, set

$$\gamma_{k+1} := \alpha_k \tag{38}$$

Compared with the original Armijo's rule, the self-adaptive one can reduce the number of evaluating the objective function and derivative values efficiently, which comes from the observation that the step sizes of two consecutive iterations are very close for most of the iterations (Chen et al., 2013). The numerical results shown by Chen et al. (2013) are quite promising.

4.2 Algorithm procedures

According to Eq. (7), the perceived path cost in the SUE problems can be represented as the summation of the path cost and an associated logarithm. Note that the logarithmic term cannot admit zero path flow at any iteration, so the path flow less than ϵ should be set to ϵ (where the demand conservation condition can be satisfied by decreasing the flow on another path by ϵ and ϵ is a sufficiently small amount of flow).

Consequently, the SUE problem can be solved easily by the path-based algorithms and the detailed procedures of the MGP algorithm are summarized as follows.

Initialization: Set $t_a = t_a\ (0), a \in A$ and calculate the path costs $c_k\ (f_k)$ (not $c_k\ (f_k)$). Get the initial path flows pattern by the route choice probabilities calculated by Eq. (1). Update all links flow, cost and first-order derivative of cost function.

Main Loop: Step 0: Update all paths augmented cost function $c_k(f_k)$ by Eq. (7) and its first-order derivative.

<u>Step 1: Calculate the descent direction.</u> Calculate the descent direction for all O-D pairs, i.e., $\Delta f_k^n\ \forall\ k \in K_p^{rs}, \forall r \in R, s \in S$.

<u>Step 2: Compute the step size</u>. Based on the descent direction, search a proper step size.

<u>Step 3: Transfer flow.</u> For some O-D pair, check if $f_k^n + \alpha \Delta f_k^n \geq 0$ for all paths. If it does, obtain the new solution and visit next O-D pair; otherwise, go to step 4.

<u>Step 4: Projection.</u> Project the infeasible solution onto the feasible region. Update the path flow by Eqs. (30)-(32). Visit next O-D pair.

After all O-D pairs are visited, update all links flow, cost and first-order derivative of the cost function, go to step 5.

<u>Step 5: Convergence test.</u> If the convergence criterion is met, or the iteration number reaches the upper limit, terminate the algorithm; otherwise, go to step 0.

5 Numerical experiments

This section presents several experiments to compare the performance of the GP and MGP algorithms and analyzes the performance of the MGP method with different path set sizes. The experiments are conducted on the Winnipeg network whose configuration is available at http://www.bgu.ac.il/~bargera/tntp. The Winnipeg network consists of 154 zones, 1,067 nodes, 2,535 links, and 4,345 origin-destination (O-D) pairs. A working path set from Bekhor, Toledo, and Prashker (2008) is used. In this path set, there is a total of 174,491 paths, and the maximum number of paths for any O-D pair is 50. The dispersion parameter θ equals to 1.0.

The convergence measurement used in this paper is based on the relative gap (RGAP). It is calculated as follows

$$\text{RGAP}=1-\frac{\sum_{r\in R}\sum_{s\in S}q_{rs}\cdot C_{min}^{rs}}{\sum_{r\in R}\sum_{s\in S}\sum_{k\in K_{rs}}f_k\cdot C_k} \tag{39}$$

where C_k denotes the perceived path cost on path k and is calculated by Eq. (7); C_{min}^{rs} denotes the cost of the perceived shortest path from origin r to destination s. The RGAP is a magnitude of the "deviation" of the perceived cost on paths. If the RGAP is near to 0, the perceived path costs are almost to the same for each O-D pair. Particularly, the RGAP is equal to 0 if and only if the perceived costs on all paths are the same for each O-D pair, which is exactly the SUE principle. Hence, if the RAGP is nearer to 0, the flow pattern gets closer to the SUE condition. The algorithm codes are conducted on the Microsoft Windows 8.1 operating system and with Intel Core i5-5200U CPU @ 2.20 GHz, 8GB RAM. All of the algorithms are coded in Visual C#.

5.1 Computational result comparison with different step size schemes

To draw more general conclusions, the experiments are conducted under three step size search strategies: 1) fixed step size scheme; 2) self-regulated averaging (SRA) scheme, in which the information between iterations is used; 3) self-adaptive Armijo (SAA) step size scheme, in which the objective function and derivative values are evaluated. The parameter setting of the SRA and SAA step size schemes follows Liu et al. (2009) and Chen et al. (2013). The parameters ψ and φ are set to 1.9 and 0.01 in the SRA step size scheme. For the SAA step size strategy, the initial step size is 1.0 and the other parameters are set as: $\bar{\gamma}$ = 1.0 , σ = 0.45, β = 0.7, ρ =2 and η = 0.9.

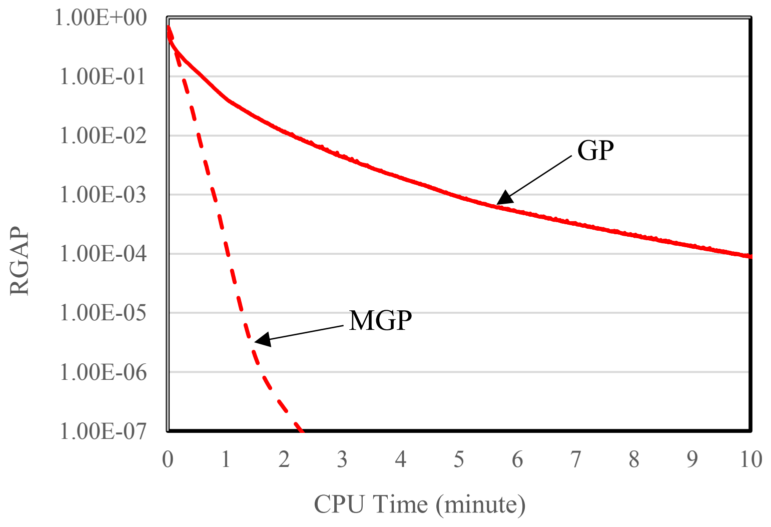

Figure 4. RGAP versus CPU time under fixed step size

Firstly, the fixed step size scheme is applied and the computational efforts of the GP and MGP algorithm are compared. The fixed step size should be set to guarantee the convergence of both algorithms. Based on the trial-and-error strategy, we set the fixed step size to 0.05.

Under the circumstance described in the above, the computational result of the GP and MGP methods with fixed step size is shown in figure 4. The testing algorithms are stopped if RGAP achieves 1E-7 or 10 minutes is taken. As shown in figure 3, the MGP method can achieve RGAP=1E-7 within 10 minutes, while the GP method can only achieve RGAP=1E-4. Besides, it takes the MGP method only 62 seconds to achieve RGAP=1E-4, which is almost 9 times faster than the GP method. Also, the MGP algorithm is ahead of the GP algorithm in the whole convergence process. Table 1 shows some detailed results. The objective function value that the MGP takes 97 iterations to reach costs the GP 900 iterations and the time they cost is 394.88 seconds and 32.54 seconds respectively.

Table 1. Computational Efforts of the GP and MGP methods

GP			MGP		
Iterations	Obj. function	Time(sec)	Iterations	Obj. function	Time(sec)
100	1070560.0	40.67	5	1068885.6	1.57
300	1057364.8	130.53	19	1057213.8	5.60
500	1055478.8	222.30	68	1055456.6	22.69
700	1055209.6	304.22	85	1055201.6	28.42
900	1055142.9	394.88	97	1055142.4	32.54

Secondly, two practical step size schemes, the SRA and SAA, are embedded into these two algorithms. The computational result is shown in figure 5. The testing algorithms are stopped once RGAP achieves 1E-7 or 20 minutes is taken. Obviously, the MGP algorithm is much faster than the other under both step size schemes. Embedded with the SAA scheme, both algorithms have a linear convergence speed and the MGP method is about 19 times faster than the other. Incorporated with the SRA scheme, the MGP algorithm can achieve RGAP= 1E-7 within less than 12 minutes and the other converges at a very slow speed.

It can be observed that the algorithms incorporated with the SRA scheme seem to perform poorly (especially for the GP algorithm). Note that the projected process of both algorithms will allow the motion along the descent direction even if a new constraint is encountered. At the beginning of the convergence process, the frequent projection process does not make the distance decrease between the auxiliary solutions of two sequential iterations. Hence, the step size gets very small after several iterations at the beginning, which leads to the slow convergence. Consequently, the SRA scheme may be not very suitable for these algorithms employing the projected process of the Goldstein-Levitin-Polyak algorithm.

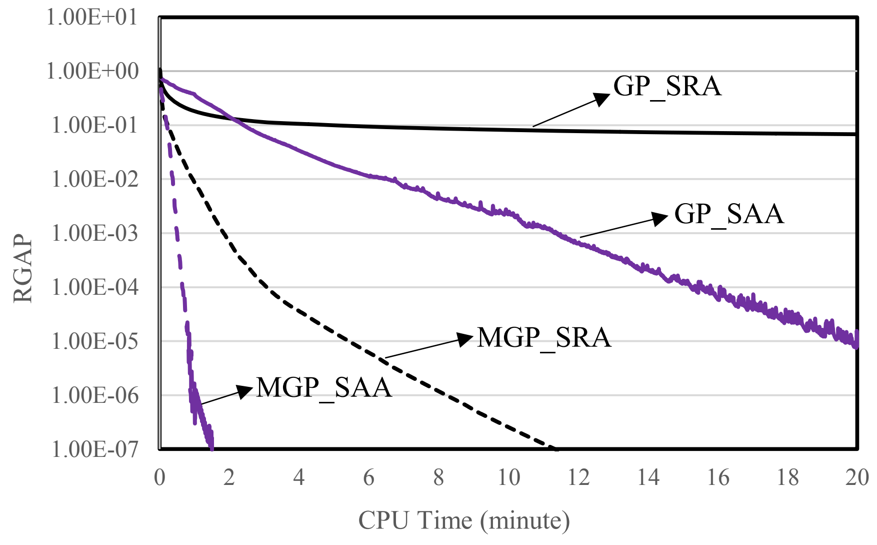

Figure 5. RGAP versus CPU time under the SRA and SAA step sizes

It can be seen that the new proposed algorithm has much better performance than the GP algorithm under these three step size strategies. Overall, the MGP algorithm is greatly faster than the GP algorithm to solve the SUE problem.

5.2 Effects of the path set size on computational performance

This subsection is designed to demonstrate that the MGP method is more suitable than the GP method to handle the situation in which the path set size grows.

In this experiment, three path set sizes are involved, which are 5, 20 and 40 paths contained within each O-D pair on average. The fixed step size scheme is employed and the step size is set to 0.05. In addition, the convergence criterion is that 10 minutes is taken or RGAP=1E-7 is achieved. The computational result comparison is shown in figure 6.

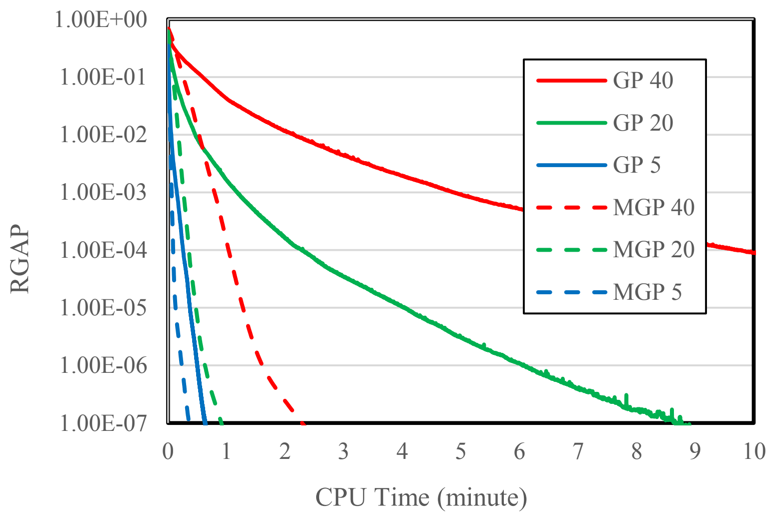

Figure 6. Effects of the path set size on computational performance

As shown in figure 6, the curves are colored differently for different path set sizes. The solid and dash curves represent the performance of the GP and MGP algorithms, respectively. It is clearly shown that the computation cost grows with the growth of the path set size for both algorithms. Also, the MGP algorithm performs better than the other with all path set sizes. Moreover, the outperformance of the MGP method is relevant to the path set size. To achieve RGAP=1E-4, it takes the GP and MGP algorithms 580.8 and 62.1 seconds when 40 paths are involved within each O-D pair; 136.9 and 21.6 seconds when 20 paths are involved; 15.5 and 5.4 seconds when only 5 paths are involved. In other words, the MGP algorithm is 8.4, 5.3 and 1.9 times faster than the GP algorithm, when 40, 20 and 5 paths are involved within each O-D pair, respectively. Obviously, the MGP algorithm performs much better than the GP algorithm for large-size path sets. As analyzed in Section 3, once there are only two paths, these two algorithms will generate almost identical directions. Theoretically, the MGP algorithm can perform better when over 2 paths are involved within each O-D pair.

Resulting from solving the objective problem by expressing the shortest path flow with the others, the GP algorithm shifts flows from the non-shortest paths to the shortest one and does not take the flow shifts among the non-shortest paths into account. Consequently, the GP algorithm performs poorly to equilibrate a large number of paths, which has been also demonstrated by the experiment results.

In summary, the new proposed MGP algorithm shows better performance than the GP algorithm in solving the SUE problem especially when there are lots of equilibrated paths.

6 Conclusions and perspectives

Based on the well-known gradient projection method, this study proposes a multiple-path gradient projection method to solve the SUE problem. These two algorithms utilize a similar projection process, in which the motion along the descent direction will not stop immediately once a new constraint is encountered. This projection process makes it possible to take a more aggressive motion along the descent direction. Hence, solving the original problem can be separated into two steps: moving along the direction and projection. The first step can be regarded as an optimization problem with only the equation constraint and the second step is to make the new solution feasible. For solving the first step, the GP method eliminates the equation constraint by expressing the shortest path flow with the others. However, it leads to the path flow shifted only between each non-shortest path and the shortest and it performs poorly to achieve equilibrium within a large-size path set.

In order to handle this issue, the proposed algorithm shifts flow not only between each non-shortest path and the shortest one but also among the non-shortest ones. It is realized by solving the problem with the method of Lagrange multipliers. Several experiments are conducted on the test network with a given path set. To draw a more general conclusion, three practical step size schemes, the fixed step size, the SRA scheme and the SAA strategy, are embedded into the GP and MGP algorithms. The computational efforts are compared. The results show that the MGP algorithm has much better performance than the GP algorithm to solve the SUE problem. Furthermore, the effects of the path set size on computational performance are also analyzed. It is indicated that the MGP method is much faster than the GP method when the path set size gets larger. Hence, the MGP method may also have good performance to solve other traffic assignment problems with a large of number paths existing within each O-D pair based on the conclusion of this paper, which is the focus of further studies.

Acknowledgments

This research is supported by the Natural Science Foundation of China (No. 71801079) and the Fundamental Research Funds for the Central Universities (No. B200202079). These supports are gratefully acknowledged.

References

Armijo, L. (1966). Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of Mathematics, 16*(1), 1–3.

Blum, J. (1954). Multidimensional stochastic approximation methods. *Annals of Mathematical Statistics, 4,* 737–744.

Bar-Gera, H., & Boyce, D. (2006). Solving a non-convex combined travel forecasting model by the method of successive averages with constant step sizes. *Transportation Research Part B, 40*(5), 351–367.

Bertsekas, D. P. (1976). On the Goldstein-Levitin-Polyak Gradient Projection method. *IEEE Transactions on Automatic Control, 21*(2), 174–184.

Ben-Akiva, M., & Bierlaire, M. (2003). *Handbook of transportation science.* Boston: Springer.

Bekhor, S., & Toledo, T. (2005). Investigating path-based solution algorithms to the stochastic user equilibrium problem. *Transportation Research Part B, 39*, 279–295.

Bekhor, S., Toledo, T., & Prashker J. N. (2008). Effects of choice set size and route choice models on path-based traffic assignment. *Transportmetrica, 4,* 117–133.

Bekhor, S., Toledo, T., & Reznikova, L. (2009). A path-based algorithm for the cross-nested logit stochastic user equilibrium traffic assignment. *Computer-Aided Civil and Infrastructure Engineering, 24*(1), 15–25.

Boyce, D., O'Neill, C. R., & Scherr, W. (2008). Solving the sequential travel forecasting procedure with feedback. *Transportation Research Record, 2077*(1), 129–135.

Chen, A., Lee, D. H., & Jayakrishnan, R. (2002). Computational study of state-of-the-art path-based traffic assignment algorithms. *Mathematics and Computers in Simulation, 59*(6), 509–518.

Chen, A., Pravinvongvuth, S., Xu, X., Ryu, S., & Chootinan, P. (2012) Examining the scaling effect and overlapping problem in logit-based stochastic user equilibrium model. *Transportation Research Part A, 46,* 1343–1358.

Chen, A., Ryu, S., Xu, X., & Choi, K. (2014). Computation and application of the paired combinatorial logit stochastic user equilibrium problem. *Computers & Operations Research, 43*, 68–77.

Chen, A., Xu, X., Ryu, S., & Zhou, Z. (2013). A self-adaptive Armijo step size strategy with application to traffic assignment models and algorithms. *Transportmetrica, 9*(8), 695–712.

Chen, A., Zhou, Z., & Xu, X. (2012). A self-adaptive gradient projection algorithm for the nonadditive traffic equilibrium problem. *Computer & Operation Research, 39*(2), 127–138.

Chen, M., & Alfa, A. (1991). Algorithms for solving Fisk's stochastic traffic assignment model. Transportation Research Part B, 25(6), 405–412.

Dial, R. (1971). A probabilistic multipath traffic assignment model which obviates path enumeration. *Transportation Research, 5*(2), 83–111.

Daganzo, C. F., & Sheffi, Y. (1977). On stochastic models of traffic assignment. *Transportation Science, 11,* 351–372.

Damberg, O., Lundgren, J., & Patriksson, M. (1995). An algorithm for the stochastic user equilibrium problem. *Transportation Research Part B, 30,* 115–131.

Fisk, C. (1980). Some developments in equilibrium traffic assignment. *Transportation Research 14B*, 243–255.

Gibb, J. (2016). *Solving travel demand model equilibrium with Barzilai-Browein step sizes.* Paper presented at 95th Annual Meeting of the Transportation Research Board, Washington, DC.

Jayakrishnan, R., Tsai, W. K., & Prashker J. N. (1994). A faster path-based algorithm for traffic assignment. *Transportation Research Record, 1554,* 75–83.

Kitthamkesorn, S., & Chen, A. (2013). A path-size Weibit stochastic user equilibrium model. *Transportation Research Part B, 57*, 378–397.

Kitthamkesorn, S., & Chen, A. (2014). Unconstrained weibit stochastic user equilibrium model with extensions. *Transportation Research Part B, 59,* 1–21.

Kitthamkesorn, S., Chen, A., & Xu, X. (2015). Elastic demand with weibit stochastic user equilibrium flows and application in a motorized and non-motorized network. *Transportmetrica A: Transport Science, 11*(2), 158–185.

Liu, H., He, X., & He, B. (2009). Method of successive weighted averages and self-regulated averaging schemes for solving stochastic user equilibrium problem. *Networks and Spatial Economics, 9*, 485–503.

Maher, M. (1998). Algorithm for logit-based stochastic user equilibrium assignment. *Transportation Research Part B, 32*(8), 539–549.

Nagurney, A., & Zhang, D. (1996). *Projected dynamical systems and variational inequalities with applications.* Boston: Kluwer.

Perederieieva, O., Ehrgott, M., Raith, A., & Wang, J. Y. T. (2015). A framework for and empirical study of algorithms for traffic assignment. *Computers and Operations Research, 54,* 90–107.

Polyak, B. (1990). New method of stochastic approximation type. *Automation and Remote Control, 51*(7), 937–946.

Prashker, J. N., & Bekhor, S. (1998). Investigation of stochastic network loading procedures. *Transportation Research Board, 1645,* 94–102.

Rosen, J. (1960). The gradient projection method for nonlinear programming. Part I. Linear Constraints. *Journal of the Society for Industrial and Applied Mathematics, 8*(1), 181–217.

Ryu, S., Chen, A., & Choi, K. (2014). A modified gradient projection algorithm for solving the elastic demand traffic assignment problem. *Computers and Operations Research, 47*, 61–71.

Ryu, S., Chen A., & Choi K. (2017). Solving the combined modal split and traffic assignment problem with two types of transit impedance function. *European Journal of Operational Research, 257*, 870–880.

Sheffi, Y. (1985). *Urban transportation networks: Equilibrium analysis with mathematical programming methods.* Upper Saddle River, NJ: Prentice-Hall.

Xu, X., & Chen, A. (2013). C-logit stochastic user equilibrium problem with elastic demand. *Transportation Planning and Technology, 36*(5), 463–478.

Xu, X., Chen, A., Zhou, Z., & Bekhor, S. (2012). Path-based algorithms to solve C-logit stochastic user equilibrium assignment problem. *Transportation Research Record, 2279,* 21–30.

Zhou, Z., Chen, A., & Bekhor, S. (2012). C-logit stochastic user equilibrium model: Formulations and solution algorithm. *Transportmetrica, 8*(1), 17–41.