Cross-comparison of convergence algorithms to solve trip-based dynamic traffic assignment problems

Mostafa Ameli, Jean-Patrick Lebacque, Ludovic Leclercq

Received 9 May 2019; | Revised 9 July 2019; | Accepted TBD

DOI: xxx/xxxx

ORIGINAL ARTICLE

Cross-comparison of convergence algorithms to solve trip-based dynamic traffic assignment problems

Mostafa Ameli[1,2] | Jean-Patrick Lebacque[1] | Ludovic Leclercq*[2]

[1]Univ. Gustave Eiffel, IFSTTAR, Paris, France
[2]Univ. Gustave Eiffel, Univ. Lyon, IFSTTAR, ENTPE, Lyon, France

***Correspondence**
Ludovic Leclercq, 3 Rue Maurice Audin, 69518 Vaulx-en-Velin cedex, France.
Email: ludovic.leclercq@ifsttar.fr

Abstract

Solving a dynamic traffic assignment problem in a transportation network is a computational challenge. This study first reviews the different algorithms in the literature used to numerically calculate the User Equilibrium (UE) related to dynamic network loading. Most of them are based on iterative methods to solve a fixed-point problem. Two elements must be computed: the path set and the optimal path flow distribution between all origin-destination pairs. In a generic framework these two steps are referred to as the outer and the inner loops, respectively. The goal of this study is to assess the computational performance of the inner loop methods that calculate the path flow distribution for different network settings (mainly network size and demand levels). Several improvements are also proposed to speed up convergence: four new swapping algorithms and two new methods for the step size initialization used in each descent iteration. All these extensions significantly reduce the number of iterations to obtain a good convergence rate and drastically speed up the overall simulations. The results show that the performance of different components of the solution algorithm is sensitive to the network size and saturation. Finally, the best algorithms and settings are identified for all network sizes with particular attention being given to the largest scale.

KEYWORDS:
Trip-based traffic assignment, User equilibrium, Large-scale network applications, Step size, Convergence

1 | INTRODUCTION

Dynamic Traffic Assignment (DTA) refers to the process of: (i) identifying the relevant paths between all Origin-Destination (OD) pairs in a transportation network, and (ii) determining the path flow distribution, considering the total OD flow demand and the time evolution of traffic states inside the network. For the first step, many researches have proposed multiple path selection models by considering the time and dynamics of the network, e.g., (Jayakrishnan, Tsai, Prashker, & Rajadhyaksha, 1994; Mahmassani, 2001; Xie, Nie, & Liu, 2018). The second step depends on the user behavior rule we want to adopt, which leads to different definitions of the network equilibrium. The best known is User Equilibrium (UE) when all users try to minimize their own travel time selfishly. It corresponds to Wardrop's first principle (Wardrop, 1952), where users are assumed to be perfectly rational and have perfect information on the network's status (Miaou, Summers, & Lieu, 1999), i.e., the predicted travel time on all the relevant alternatives is known at the beginning of all the users trips (Ng & Waller, 2012). Implementing this simple behavioral rule for Dynamic Network Loading (DNL) is far from trivial (Lin, Valsaraj, & Waller, 2011). DNL is the combination of DTA with a traffic simulator that calculates network states and travel times (Yu,

[0]**Abbreviations:** UE, user equilibrium; DTA, dynamic traffic assignment; OD, origin-destination

Ma, & Zhang, 2008). The critical issue is that the simulator needs to know the path flow distribution in order to predict the travel time accurately while the DTA process requires this information to estimate the path flow distribution (Bekhor, Toledo, & Reznikova, 2009). Mathematically, this problem corresponds to a fixed-point search, which requires an iterative solution method to converge. Transforming the DTA problem into a fixed-point problem allow using a large number of algorithms. The main idea stems from the theory of fixed-point re-statement (Xu, 2002). Since one run of the traffic simulator is computationally expensive, in particular for a large-scale network, in the field of transportation, it is essential to use an efficient algorithm to solve the fixed-point problem.

The general principle of the algorithms to solve the fixed-point problem is to reassign a fraction of the users at each step (Friesz & Han, 2019). The algorithm usually reassigns the part of the users who have chosen a non-optimal path because the travel time estimation was misleading (Sancho, Ibáñez Marí, & Bugeda, 2015). The critical issue is to reach a given level of convergence while minimizing the number of iterations. This study aims to investigate the performance of convergence algorithms to solve the DTA problem.

Multiple convergence algorithms have been proposed in the literature to solve this problem. There are two main approaches to determine how many users at each step should be reassigned: analytical and simulation-based. The analytical approach, e.g., (Jiang & Xie, 2014; Mounce & Carey, 2011; Wang, Szeto, Han, & Friesz, 2018), is very accurate but can only be applied in practice to small networks with few ODs. Several studies proposed exact decomposition techniques to reduce the computational complexity of the traffic assignment problems in static (Jafari, Pandey, & Boyles, 2017) and dynamic cases (Mehrabipour, Hajibabai, & Hajbabaie, 2019). However, congestion patterns are almost intractable analytically due to multiple non-linear interactions inside the network (Taale & Pel, 2015). Simulation-based approaches can match any given network, but obviously the simulation time increases with the number of nodes/links and vehicles inside the network.

Traffic simulators can be divided into two classes: Flow-based models, which consider the flow on each path and Trip-based models, which define how many travelers take each path. Macroscopic traffic flow models fall into the first category while microscopic models belong to the second. In other words, the flow-based models have a continuous solution space while trip-based ones have a discrete solution space (Ramadurai & Ukkusuri, 2011). Both kinds of simulators can be coupled with convergence algorithms. The macroscopic approach and flow-based models usually converge faster as the path flow discipline is more flexible (flows are not necessarily equivalent to vehicle units), but without adding integrality constraints, they are less realistic for OD pairs with low demand as vehicles are split into parts in practice. In this study, we decide to focus on the trip-based approach because it is more realistic as each vehicle is reproduced individually. Microscopic traffic simulators are now widely used for operational studies, and we have chosen to focus on DTA performance for this kind of model. In summary, trip-based DNL attempts to assign particle-discretized time-dependent origin/destination (OD) flows in a dynamic network equilibrium framework (Jayakrishnan & Rindt, 1999).

Regarding the real-world application, there are several computer packages in the literature. Jeihani (2007) reviewed the DTA models used in the most well-known. Indeed, a lot of effort has been paid in providing a new mathematical representation to solve the DTA problem; however, it is also essential to have an idea about the overall efficiency of different computational methods. Here, we consider the simulator as a black box to make this study compatible with any existing traffic simulation software. We then focus on solution methods. This study reviews existing algorithms to solve the fixed-point problem and proposes multiple improvements to speed-up the convergence for a given level of accuracy. Particularly, for large-scale networks as computation times usually count in hours or even days; therefore, slight improvements may be valuable. This study aims to address the following questions:

- Which solution algorithm is efficient to find the DTA solution by considering the size and loading of the network?
- Is there a way to improve the existing methods to find a good quality solution in terms of optimality and feasible computation time for large-scale?

This study considers common convergence algorithms in the literature and attempts to overcome the drawbacks of existing methods to improve the performance of the solution algorithm for simulation-based DTA problems. We aim to provide guidance to practitioners about the best settings to solve simulation-based DTA problems.

The next section, Problem statement, provides a discussion on the mathematical conditions for the UE solution. It also presents the two indicators that will be used to assess the algorithm's performance. The benchmark of the solution algorithm for finding the UE is presented in the Methodology section. The improvements made to the solution algorithm with the new swapping algorithms are presented in the section Investigating the solution algorithm. The experimental design is presented in the section Numerical experiment. The results obtained are discussed in the section Numerical results. Finally, we provide concluding remarks and introduce the future directions of work in the Conclusion section.

2 | PROBLEM STATEMENT

2.1 | Mathematical formula for UE

Let us consider a network $G(N, A)$ with a finite set of nodes N and a finite set of directed links A. The demand is given and time-dependent. The period of interest (planning horizon) of duration H is discretized into a set of small time intervals indexed by τ ($\tau \in T = \{\tau_0, \tau_0 + \eta, \tau_0 + 2\eta, ..., \tau_0 + M\eta\}$ and $\tau_0 + M\eta = H$). η is the duration of the time intervals. In an interval τ, travel times and traffic conditions are estimated on average and are assumed constant for the DTA. Note that the departure time of users are fixed in this study. In the sequel, the minimum cost path is considered as the shortest path.
The important notations which must be introduced for the dynamic equilibrium model are as follows:

W: OD pairs, subset of origin × destination nodes, $W \subset N \times N$.
w: index of origin-destination (OD) pair, $w \in W$.
$P_{w,\tau}$: set of paths for w in departure time interval τ.
$P^*_{w,\tau}$: set of shortest paths for w in departure time interval τ.
p: index of path, $p \in P_{w,\tau}$.
p^*: index of shortest path, $p^* \in P^*_{w,\tau}$.
D_w: total demand for w pair.
$Tr_{w,\tau}$: list of trips which travel for w in departure time interval τ.
$Tr_{p,\tau}$: list of trips which travel for w on path p in departure time interval τ, $Tr_{p,\tau} \subset Tr_{w,\tau}$.
tr: index of trip, $tr \in Tr_{w,\tau}$.
$C_{tr,p,\tau}$: experienced travel cost of trip tr on path p in departure time τ.
$C^*_{w,\tau}$: minimum experienced travel cost for w in departure time interval τ.
$\hat{C}_{p,\tau}$: mean travel cost of trips on path p in departure time τ.
$\hat{C}^*_{w,\tau}$: mean travel cost of trips on minimum cost path(s) of OD pair w in departure time τ.
$n(A)$: cardinality of a set A.

According to the definition, we have:

$$\hat{C}_{p,\tau} = \frac{\sum_{tr \in Tr_{p,\tau}} C_{tr,p,\tau}}{n(Tr_{p,\tau})} \quad ;\forall p \in P_{w,\tau},\ \tau \in T \tag{1}$$

$$\hat{C}^*_{w,\tau} = \frac{\sum_{p^* \in P^*_{w,\tau}} \sum_{tr \in Tr_{p^*,\tau}} C_{tr,p^*,\tau}}{n(Tr_{p^*,\tau})} \quad ;\forall w \in W,\ \tau \in T \tag{2}$$

Equation (1) presents the calculation of mean travel cost of path p and equation (2) is the same presentation for the shortest path p^*. For each OD pair $w \in W$ and for all paths $p \in P_w$, the dynamic traffic network equilibrium conditions with given travel demand and the users' departure time for the aforementioned traffic network equilibrium problem are:

$$\hat{C}_{p,\tau} - \hat{C}^*_{w,\tau} \geq 0 \quad ;\forall w \in W,\ p \in P_{w,\tau},\ \tau \in T \tag{3}$$

$$n(Tr_{p,\tau})(\hat{C}_{p,\tau} - \hat{C}^*_{w,\tau}) = 0 \quad ;\forall w \in W,\ p \in P_{w,\tau},\ \tau \in T \tag{4}$$

$$n(Tr_{p,\tau}) \geq 0 \quad ;\forall p \in P_{w,\tau},\ \tau \in T \tag{5}$$

According to constraint (3), the shortest path p^* has the minimum travel cost for the related OD pair. Equation (4) indicates that all users travel on shortest path with minimum travel cost at UE state and the flow of paths cannot be negative, according to constraint (5).

Lu et al., 2009, extended the work of Smith, 1993, and reformulated the problem as a non-linear problem in order to minimize the gap function. The gap function is defined as the gap between average path travel time and average shortest path travel time. Consequently, the solution to this fixed-point problem is equivalent to finding the solution to the following variational inequality:

$$\sum_{w \in W} \sum_{\tau=1}^{T} \sum_{p \in P_{w,\tau}} {C^i_{w,p,\tau}}^* \left[n(Tr_{w,p,\tau}) - n({Tr_{w,p,\tau}}^*) \right] \geq 0 \tag{6}$$

where $n({Tr_{w,p,\tau}}^*)$ is the optimal number of trips from OD pair w that are assigned to path p at departure time τ and $n(Tr_{w,p,\tau}), n({Tr_{w,p,\tau}}^*) \in \mathcal{H}$ satisfy the equilibrium. $\mathcal{H}$ denotes the flow constraints based on D_w. In equation (6), both $n(Tr_{w,p,\tau})$ and $n(Tr^*_{w,p,\tau})$ are decision variables and hence the gap function is a function of both variables.

Then a vector of $n(Tr^*_{w,p,\tau})$ solves equation (6) if and only if it is a fixed-point of the following mapping (Marcotte, 1985):

$$\Gamma \triangleq \arg\min_{n(Tr_{p,\tau}) \in \mathcal{H}} [n(Tr_{p,\tau})]^T [\hat{C}_{p,\tau}] \tag{7}$$

where $[n(Tr_{p,\tau})]$ denotes the vector of $n(Tr_{p,\tau}); \forall p \in P_{w,\tau}, \tau \in T$ and $[\hat{C}_{p,\tau}]$ denotes the vector of $\hat{C}_{p,\tau}; \forall p \in P_{w,\tau}, \tau \in T$. For the proof and details, see Kaufman, Smith, & Wunderlich (1998). Before presenting the solution algorithm, we need to present the convergence indicators used in this study to evaluate the quality of the solution for the trip-based dynamic network user equilibrium.

2.2 | Convergence quality

In the trip-based DTA problem, the goal is to minimize the left side of the Equation (8) for all paths and OD pairs. In other words, finding the UE situation is equivalent to minimizing the delay of each user compared to the optimal option of the associated OD pair (shortest path) in the network. By this

definition we can define a quality indicator for solutions which is calculated as the average delay of each user (Janson, 1991):

$$AGap = \frac{\sum_{w\in W}\sum_{\tau=1}^{T}\sum_{p\in P(w,\tau)}\sum_{tr\in Tr_{p,\tau}}(C_{tr,p,\tau} - C^*_{tr,w,\tau})}{\sum_{w\in W}\sum_{\tau=1}^{T} n(Tr_{w,\tau})} \quad (8)$$

Note that this formula uses the experienced travel time to calculate the gap rather than the path travel time. The $AGap$ is zero for the perfect UE path flow distribution, so the best optimization algorithm obtains minimum $AGap$. Equation (8) has physical meaning for measuring the distance between solution and UE. The second indicator is a characterized assignment violation, i. e. users that are assigned on (a) non-optimal path(s). The violation indicator is calculated by the following steps:

1. Calculate the user violation: it is defined by considering the gap of each user (UV^{tr}_w):

$$UV^{tr}_w = \begin{cases} 1; & \text{if } \frac{C_{tr,w} - C^*_w}{C^*_w} \geq \epsilon \\ 0; & \text{o.w.} \end{cases} \quad (9)$$

   where $C_{tr,w}$ denotes the experienced travel time of the user tr who travels for OD pair w and C^*_w denotes the shortest path of OD pair w. If the gap between the user perceived travel time and the shortest path travel time is bigger than ϵ of the shortest path travel time, the user is in violation. $\epsilon = 0$ means perfect UE, but in practice with trip-based simulation, it is more appropriate to set up a margin to count users that are miss-assigned.

2. Compute the OD violation: the OD pair w is in violation when more than ϵ' of the users on w are in violation. The function ODV_w defines the OD violation:

$$ODV_w = \begin{cases} 1; & \text{if } \frac{\sum_{i\in I_w} UV^i_w}{n(Tr_w)} \geq \epsilon' \\ 0; & \text{o.w.} \end{cases} \quad (10)$$

   where Tr_w denotes the set of users, who travel for OD pair w.

3. The violation indicator of network G is the share of ODs in violation. The second indicator for the quality of solution (Violation) is defined as follows:

$$V(G) = \frac{\sum_{w\in W} ODV_w}{n(W)} \quad (11)$$

The value of ϵ and ϵ' are fixed at 10% in this study to evaluate the quality of the solution from a different angle as $AGap$. The violation index provides a more threat assessment of how the equilibrium is achieved on the OD-basis. Note that, similar to $AGap$, the perfect UE means $V(G) = 0$ with $\epsilon = \epsilon' = 0$.

While $AGap$ gives an idea about the average level of the convergence for the solution in a continuous way, the violation index defines how many out-liners we can accept. The two criteria represent the level of confidence we want to achieve at the OD level (ϵ) and then at the network level (ϵ'). Please note that we do not use $V(G)$ in the convergence loop and simply calculate it afterwards to assess how the process is doing.

3 | METHODOLOGY

In large-scale DTA problems, there are three costly steps in terms of computation in simulation-based DTA models: traffic simulation, shortest path discovery, and optimization. Here, we focus on the optimization step. According to the state of the art, it appears that the most advanced framework for solving the simulation-based DTA problem as a black-box optimization problem is that proposed by Lu, Mahmassani, & Zhou (2009) with a clear decomposition between outer and inner loops. The outer loop is responsible for path discovery and does not include any traffic simulation. The inner loop determines path flow distribution for all OD pairs in the network and includes multiple DNL. The classic approach executes both steps in one top loop. In large-scale network problems, it is extremely costly to keep the data of all possible paths between each OD pair. With Lu et al. (2009)'s framework, the simulator simply keeps the feasible paths discovered by the outer loops.

The outer loop is only about path discovery. Therefore, improving the computation time there means finding a more efficient shortest path algorithm, which is not in the scope of this study. This study focuses on path flow distribution, which is the role of the inner loop and plays a pivotal role in the overall UE calculation by reducing the number of simulation run we have to launch to get the equilibrium. The solution algorithm is presented in Figure 1 and is detailed in the following:

Step 1. **Initialization**: Load the network and the OD matrix, with real data or simple assignment models (e.g., All-or-nothing algorithm).

Step 2. **Time-dependent shortest path Calculation**: Calculate the time dependent shortest path(s) for each OD pair w.

Step 3. **Update the cost functions**: Read the information from the simulator output and update the cost of paths (links) on the network graph and update the quality indicators.

Step 4. **Outer loop convergence test**: Check the stop conditions:

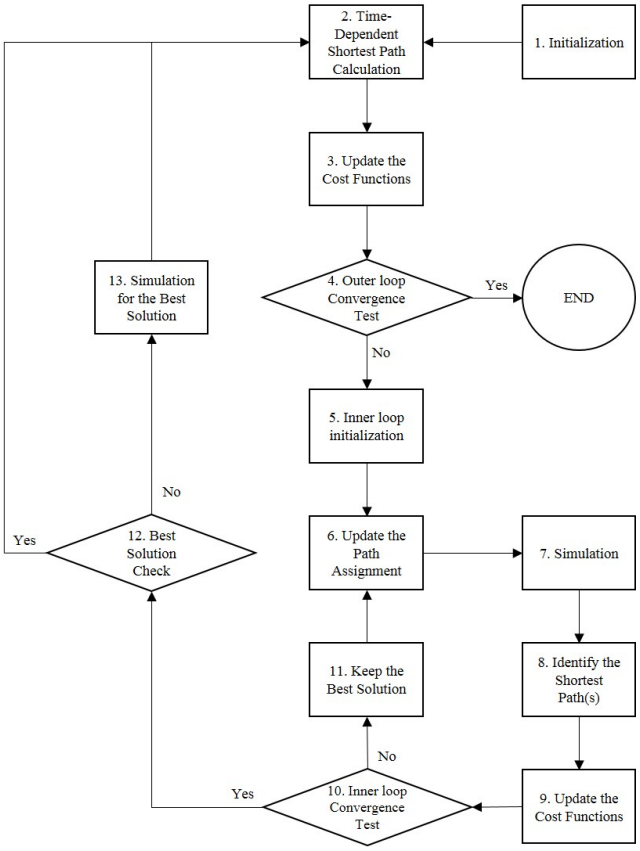


FIGURE 1 Solution algorithm for trip-based dynamic network equilibrium

a. Maximum number of outer loop iterations (j_{max}) is reached **OR**

b. No new shortest path for adding to path set **AND** good solution quality: $AGap^j \leq \lambda$; where $AGap^j$ is the $AGap$ of outer loop iteration j ($\forall j \in \{1, ..., j_{max}\}$). λ is a small and fixed value. Otherwise go to the next step.

If it has converged, the process is stopped (END). Note that the path set is a set of paths that contains used path(s) ($n(Tr_{p,\tau}) > 0$) and shortest path(s) which have been used or not ($n(Tr^*_{p,\tau}) \geq 0$) for all ODs which come from step 2.

Step 5. **Inner loop Initialization**: Load the path flow distribution of Step 1 by (Lu et al., 2009) or other initialization methods (this study) in order to generate the initial assignment to start the inner loop.

Step 6. **Update the path assignment**: Swapping trips from a path to another(s) based on an optimization method in order to load the flow to the different path candidates.

Step 7. **Simulation**: Command the simulator to simulate the new assignment pattern provided by Step 6.

Step 8. **Identify the shortest path(s)**: The simulator returns the experienced travel time of all the users on different ODs. The shortest path travel time can be changed, and it is possible that we have another shortest path from the path set based on the simulation results. Note that we have a fixed path set in this step and we do not need to use the shortest path algorithm.

Step 9. **Update the cost function**: Update the network data from the simulator run in Step 3.

Step 10. **Inner loop convergence test**: Check the stop conditions:

a. Maximum number of inner loop iterations (i_{max}) is reached **OR**

b. small enough variation in solution quality:

$$\frac{|AGap^i - AGap^{i-1}|}{AGap^{i-1}} \leq \Lambda \quad ; \forall i \in \{1, ..., i_{max}\} \tag{12}$$

where $AGap^i$ is the quality of the solution of inner loop iteration i and Λ is a fixed threshold ($\Lambda = 1\%$) for comparing the relative $AGap$.

At the end of each Inner Loop, if there is insufficient variation in solution quality, we converge and go to step 12. Otherwise, we continue to Step 11.

Step 11. **Keep the best solution**: Compare the solution quality ($AGap^i$) of the current inner loop iteration (i) with the best solution of the current outer loop ($AGap^j_{min}$). Note that if we are in the first inner loop iteration ($i = 1$), we consider the $AGap$ of Step 4 as the initial best and compare it with the current solution. If the solution has better quality, we replace the best solution by the current solution. Otherwise, the best solution is kept. The solution contains the path assignment pattern and solution's $AGap$. Afterward, we continue iterating in the inner loop by going back to Step 6.

Step 12. **Best solution check**: The goal of this step is to ensure that our last solution is the best solution of the current outer loop. We compare the last solution with the current best solution from Step 11. If the last solution is the best ($AGap^i \leq AGap^j_{min}$), go to the outer loop to explore the network and find (a) new shortest path(s) (Step 2). Otherwise, a final simulation is executed in the next step for the best assignment pattern which is not the last one we obtain.

Step 13. **Simulation of the best solution**: simulate the traffic state (like Step 7) with the best assignment pattern of

the inner loop provided by Step 11 to move to the next time period or the next outer loop iteration (Step 2).

Note that in the inner loop convergence test (Step 10), the stop conditions are the maximum number of iterations and variations of the gap, so the last iteration solution is not necessarily the best solution of the inner loop. Therefore, Steps 11-13 are keeping the best solution of the inner loop when the algorithm goes back to the outer loop. This will be even better when using trip-based approaches because their discrete nature makes them less stable and we will, therefore, have more cases where the solution of the last iteration is not the best over the current inner loop.

4 | INVESTIGATING THE SOLUTION ALGORITHM

In this section, we focus on improving the inner loop (in Figure 1). They correspond to steps 5 and 6.

4.1 | Swapping algorithms

Swapping algorithms corresponds to Step 6 in Figure 1, where the solution algorithm updates the path assignment. Here, we benchmark different algorithms that exist in the literature and also propose new ones. Table 1 gives a complete summary of all the algorithms with their references.

4.1.1 | Method of Successive Average (MSA)

The method of successive average (MSA) is the best known algorithm for solving the fixed-point problem. Most of the studies in the literature used MSA as a reference algorithm. The MSA was presented for the first time by Robbins & Monro (1951). The MSA is still widely used in simulation-based DTA (e.g., (Zhang, Liu, & Waller, 2019)), because it is simple to implement and does not require the derivative information of the flow cost function (Nagel & Flötteröd, 2012). The MSA updates the path flow by using a descent direction and a predetermined step size. Mounce & Carey (2014) investigates different descent direction methods for the MSA algorithm and concludes that MSA works well on large networks when combined with a swap towards the least costly route(s). Therefore, the classic descent direction (y^i_p) of the iteration (i) for path p is extracted from the auxiliary path assignments obtained by All-or-nothing discipline, i.e., everyone is put on the active shortest path. Consequently, in iteration i, the MSA algorithm swaps a fraction σ^i of users to the shortest path(s) from each non-shortest path (Equation (1)). Mathematically, a fraction σ^i of the total number of users on non-shortest paths is added to the shortest path(s).

$$n^{i+1}_{p,w} = n^i_{p,w} + \sigma^i_{MSA}(y^i_{p,w} - n^i_{p,w}) \tag{13}$$

where $n^i_{p,w}$ is the number of trips for OD pair w which travel on path p in iteration i at each time step. σ^i_{MSA} denotes the step size of the MSA algorithm. The MSA step size satisfies the following (Sheffi, 1985):

$$\sum_{i=1}^{\infty} \left(\sigma^i\right)^2 < \infty \tag{14}$$

$$\sum_{i=1}^{\infty} \sigma^i = \infty \tag{15}$$

The classic MSA uses one over the iteration index plus one as a step size (Equation (4)) to ensure the algorithm converges. The step size can be defined with respect to Equations (2) and (3).

$$\sigma^i_{MSA} = \frac{1}{i+1} \tag{16}$$

The first drawback of the MSA is that it swaps a fixed number of users from all non-shortest paths to the shortest one without considering the actual travel time on these paths; it does not consider the gap between the shortest and other paths. First, Sbayti, Lu, & Mahmassani (2007) implemented the classic MSA method in trip-based DTA, using the random selection technique in view to reducing memory requirements. Second, they attempted to overcome the first drawback by proposing a criterion-based selection that ranks the users based on experienced travel time. They showed that on a real network, both methodologies were observed to converge, and that the criterion-based technique also produced a better solution than MSA in terms of closeness to optimal. However, by increasing the number of users in a large-scale network, ranking them based on travel cost is a computationally costly approach. Moreover, Sbayti et al. (2007) uses the same predetermined step size as the MSA method.

The second drawback is about predetermining the step size. The step size rule pushes the process to stabilize. There may be a risk of stabilizing before reaching the optimal solution. The step size has a direct impact on the number of iterations (computation time) and convergence speed. There is no exact method for determining the step size in the literature (Huang & Lam, 2002; W. Szeto & Lo, 2005). The step size does not guarantee the quality of the solution at the end, which may not be the actual UE (Levin, Pool, Owens, Juri, & Waller, 2015). One of the goals of this study is to improve the performance of the optimization process by proposing new methods for step size determination in the simulation-based optimization framework. There are several extensions for MSA algorithm in the literature, which attempt to overcome the drawbacks of existing methods to improve the performance of the solution algorithm for simulation-based DTA problems.

4.1.2 | MSA Ranking

As mentioned in the previous section, the first drawback of the MSA algorithm for DTA application is that the MSA does not consider the travel time of the non-shortest path and simply swaps a fixed fraction of users to the shortest path. In other words, the MSA does not take into account the quality of the path in terms of travel time. To overcome this drawback of the MSA algorithm, there is an extension of the MSA in the literature called MSA ranking (Sbayti et al., 2007). The idea of MSA ranking is that the users are first ranked by experienced travel time then a maximum number of users with long experienced travel time are swapped to the shortest(s) path, based on the MSA algorithm's step size. The maximum number of swaps NS^i_{max} is observed when no users have been previously assigned to the shortest path.

$$NS^i_{max} = \sigma^i_{MSA}.D_w \tag{17}$$

The advantage of this algorithm is that it swaps users from the most expensive paths to the shortest path so the direction of solution searching can be improved to obtain a good solution in terms of quality for the trip-based UE problem. On the other hand, with a large number of users traveling between many ODs by many possible paths, ranking the users is a costly process in a large-scale problem. However, it is a good reference when taking all the MSA-based algorithms into account because it usually provides the best solutions (low *AGap* and Violation values).

4.1.3 | Projection-based algorithms

Here, we adapt two swapping algorithms with an extension of the MSA-based formula.

1- The within-day fixed-point algorithm is a projection-based algorithm designed for non-linear fixed-points of non-expanding maps. Friesz & Han (2019) applies this algorithm to flow-based (continuous) dynamic user equilibrium problem by differential variational inequalities. We adapt this algorithm as a Projection method (PM) to the trip-based DTA problem. The validation of the algorithm for the fixed-point problem is well defined in Halpern (1967). The swapping algorithm of the Projection method is based on the transformation of the cost to the flow by a constant (α), which is the time step size of the algorithm. From the standpoint of application, the unit of α is $\frac{time}{flow}$ and measures users' sensitivity to travel costs. The swapping algorithm for trip-based DTA is as follows:

$$NS^p_{PM} = min\ \left\{n(Tr_p),\ \alpha.(\hat{C}_{p,\tau} - \hat{C}_{w,\tau})\right\} \tag{18}$$

where $\hat{C}_{w,\tau}$ denotes the mean travel time of all paths of OD pair w in time interval τ. At every iteration, the algorithm attempts to swap users from the path with longer travel time compared to mean travel time to the shortest path and other low-cost paths. Note that α is determined in the light of the recent study of (K. Han, Eve, & Friesz, 2018) in which this algorithm is applied to a flow-based DTA problem. Note that in equation (18), we can have a negative number for swapping, meaning that users should be added to this path. It also gives an indication of how many should be added. This only concerns paths whose path travel time is below the mean travel time ($\hat{C}_{w,\tau}$).

2- Friesz, Kim, Kwon, & Rigdon (2011) extended the Projection algorithm by using a common method for speeding up the convergence in Hilbert space when solving the fixed-point problem. The idea is to retrieve the initial solution at each iteration by using a weighted coefficient based on a second step size for calculating the Projection algorithm solution. Let us consider z^i as the assignment pattern in iteration i of the Projection algorithm. The solution of this algorithm, labelled as Projection Initialization (PI) algorithm, will be:

$$z^i_{PI} = \sigma^i_{PI} z^0 + (1 - \sigma^i_{PI}) z^i \tag{19}$$

where z^0 is the initial flow pattern at the beginning of the process and σ^i_{PI} is the step size determined by the following formula:

$$\sigma^i_{PI} = (\frac{1}{1+i})^q \tag{20}$$

where q is a parameter to be designated and which must satisfy $0 < q < 1$.

3- The third algorithm in this category is developed based on the Projection Initialization algorithm. The same procedure is applied for the MSA algorithm to evaluate the impact of sticking to the initial solution on the optimization process. Therefore, the flow formula of the Initialization MSA (IMSA) algorithm is as follows:

$$z^i_{IMSA} = \sigma^i_{PI} z^0 + (1 - \sigma^i_{PI}) z^i{}_{MSA} \tag{21}$$

where z^i in equation (19) is replaced by z^i_{MSA}, which is the solution of the MSA algorithm in iteration i. As with the Projection initialization algorithm, σ^i_{PI} is determined by equation (20).

4.1.4 | Gap-based algorithm and Gap-based normalized algorithm

Lu et al. (2009) proposed the gap-based step size and proved that it satisfies the step size conditions. The volume of swapping is proportional to the gap (the difference between the non-shortest path and shortest path cost) over the path cost multiplied by MSA step size:

$$\sigma^i_{GB} = \frac{\hat{C}_{p,\tau} - \hat{C}^*_{w,\tau}}{\hat{C}_{p,\tau}}.\rho^j_{MSA} \tag{22}$$

$$\rho^j_{MSA} = \begin{cases} \sigma^j_{MSA}; & \text{if} \quad i = 0 \\ 1; & \text{o.w.} \end{cases} \tag{23}$$

We recall that j is the outer loop iteration index. This algorithm solves the problem of sorting and also circumvents the first drawback of the MSA formula (Section 4.1.1). However, it also uses the step-size which can induce the convergence of the algorithm to a non-optimal solution. Moreover, the multiplication of the gap indicator and the MSA step size can provide a small step size for this algorithm. Here we normalize the gap indicator to provide a relative fraction for the step size which gives the algorithm more flexibility in swapping at each iteration. The algorithm can swap more or fewer users from path p with respect to the gap of other paths of OD pair w:

$$\sigma^i_{GBN} = \frac{\hat{C}_{p,\tau} - \hat{C}^*_{w,\tau}}{\sum_{p \in P(w,\tau)} (\hat{C}_{p,\tau} - \hat{C}^*_{w,\tau})} . \rho^i_{MSA} \tag{24}$$

This swapping algorithm is applied for each OD pair. This algorithm attempts to normalize the gap indicator of the Gap-Based algorithm (GB), which is called the Gap-based normalization (GBN) algorithm in this study.

4.1.5 | Probabilistic algorithm

We first introduced the Probabilistic algorithm in Ameli, Lebacque, & Leclercq (2017) for solving the multi-class multimodal equilibrium, but we never investigated it in detail. This algorithm calculates for each user the probability of swapping (SP_{tr}) by equation 25. Then the Bernoulli trial is implemented in simulation for a user tr in order to decide to swap or not according to the result of the trial.

$$SP_{tr} = P(tr_{swap} = 1) = \frac{C_{tr,\tau} - C^*_{w,\tau}}{C_{tr,\tau}} \tag{25}$$

where tr_{swap} denotes the binary swap decision variable. This algorithm is the only one which does not use step size. Moreover, it avoids the ranking process and saves computation time. The Probabilistic algorithm can be considered as an implementation of the Gap-based algorithm to a trip-based approach if we adjust the probability of reducing the impact of the descent step. However, here we basically relax the step size. The Probabilistic algorithm is totally flexible when searching the solution space based on the probabilistic process.

4.1.6 | Hybrid algorithms

Hybrid algorithms are different combinations of Gap-based, Probabilistic and MSA algorithms for each individual step of the calculation.

1- Halat, Zockaie, Mahmassani, Xu, & Verbas (2016) applied a hybrid algorithm for a dynamic activity-based model. The algorithm is a similar to the Probabilistic method as it adds a step size to equation (25). This algorithm is called Step size Probabilistic (SSP), and calculates the swap probability by the following formula:

$$SP^{SSP}_{tr} = \frac{C_{tr,\tau} - C^*_{w,\tau}}{C_{tr,\tau}} . \sigma^i_{MSA} \tag{26}$$

The first hybrid algorithm uses a random number and compares it with SP^{SSP}_{tr} to make the swap decision for each user.

2- Verbas, Mahmassani, & Hyland (2015) applied the Gap-based algorithm for transit network assignment problems and used the probabilistic algorithm for each user on each path to swap more users with high travel cost without ranking. The algorithm is called Gap-based Probabilistic (GBP). For path p, the number of swapping users is determined by equation (22) and the users are selected by Equation (25) for swapping. Verbas, Mahmassani, & Hyland (2016) showed that this hybrid algorithm obtains better solutions than the MSA algorithm in large-scale transit assignment problems.

3- The third hybrid algorithm is the Boost-up Gap-based (BGB) algorithm and is proposed by this study. The idea is to boost step size of the Gap-based algorithm. For path p, we multiply the number of swaps by a fraction of the swap number of the Gap-based and MSA algorithms:

$$NS^p_{BGB} = min \left\{ n(Tr_p),\ [\sigma^i_{GB} . n(Tr_p)] . \frac{\sigma^i_{GB}}{\sigma^i_{MSA}} \right\} \tag{27}$$

This section presents different alternatives for Step 6 of the solution algorithm.

4.2 | Inner loop initialization

The solving process in the inner loop should start with an initial path flow distribution to first estimate travel times by simulation (Step 5 in Figure 1). The usual approach is to assign the total demand to the shortest path(s), using free-flow travel times (All-or-nothing initial assignment). In Step 5, at the beginning of each outer loop before entering the inner loop, the assignment pattern is initialized and reset to the assignment pattern in Step 1 (Lu et al., 2009). We investigate an alternative approach for the assignment pattern initialization.

Keeping the assignment pattern approach removes Step 5 from the solution algorithm. Consequently, the algorithm starts the outer loop j with the optimal solution from the previous outer loop $(j-1)$. Obviously, this will be very efficient for solving static situations, but we want to investigate its performance with dynamic loading.

4.3 | Initial step size selection

The third investigation is the definition of the descent step size. The initial step size (σ^1) defines how many users can be swapped during the first iteration. It is the largest step size during the inner loop and determines the exploration domain of

the solution space. In two-level simulation-based methodology (Lu et al., 2009), the initial step size of the first inner loop of the outer loop j ($\sigma^{1,j}$) is calculated by the iteration counter of the outer loop (j):

$$\sigma^{1,j}_{Initial} = \frac{1}{j+1} \tag{28}$$

This setting improves the speed of convergence because increasing j decreases $\sigma^{1,j}_{Initial}$, so the largest number of swaps for the current inner loops in outer loop $j+1$ begins with a smaller value in comparison with outer loop j. On the other hand, increasing j reduces the exploration domain of the inner loop. In order to overcome this drawback, this study proposes two new approaches to set up the step size.

4.3.1 | Re-initializing the step size

The idea of re-initializing the step size (Reset) method is to reset the step size by the inner loop iteration index:

$$\sigma^{i,j}_{Reset} = \frac{1}{i+1} \tag{29}$$

This approach at the beginning of each outer loop starts the optimization with $\sigma^{1,j}_{Reset} = \frac{1}{2}$ to have more flexibility for searching the solution space. In other words, this approach can increase the maximum number of swaps at each iteration in comparison to the initial approach ($\sigma^{1,j}_{Reset} \geq \sigma^{1,j}_{Initial}$).

4.3.2 | Smart step size

Here we design an approach that uses adaptive step size for each OD pair w. First, all the inner loops are initiated with the same step size whatever the OD pair. At the end of the first iteration ($i = 1$), the OD gap for OD pair w is calculated as follows:

$$Gap^{i}_{w} = \sum_{p \in P(w,\tau)} [n(Tr_{w,p}).(\hat{C}_p - \hat{C}^{*}_{w})] \tag{30}$$

Then, we keep the step size and run the second loop ($i = 2$). At the end of the second loop, we update the OD gap and compare it with the previous OD gap. If the OD gap is improved, we keep the step size to swap the same fraction of users for a possibly better solution, otherwise we decrease the OD step size to decrease the number of swaps. The step size for OD pair w at inner loop iteration i is:

$$\sigma^{i}_{w} = \begin{cases} \frac{\sigma^{i-1}_{w}}{\sigma^{i-1}_{w}+1}; & \text{if} \quad Gap^{i}_{w} \geq Gap^{i-1}_{w} \\ \sigma^{i-1}_{w}; & \text{o.w.} \end{cases} ; \quad \forall i \in \{2, ..., i_{max}\}, \quad \sigma^{1} = \frac{1}{2} \tag{31}$$

Equation (31) adapts the step size for each OD pair depending on how the quality of the solution is improved. This method mimics the Newton−Raphson method in numerical analysis (Ypma, 1995). It has been proven to be very efficient for continuous problems, but this is the first time it is used in the context of DTA and for the discrete formulation (trip-based model).

To conclude this section, we present all the methods considered in this study in Table 1. Before comparing the methods, we need to present the dynamic trip-based simulator and test cases.

TABLE 1 All the methods in the inner loop considered in this study

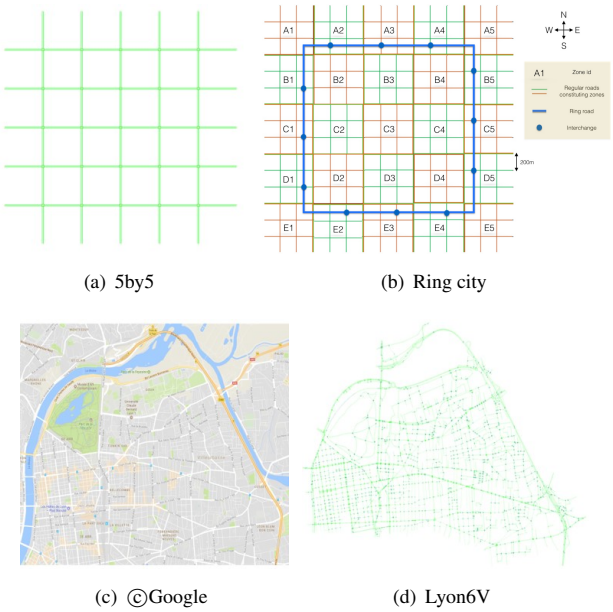
Method	Abbreviation	Reference
Swapping algorithms (Step 6)		
MSA	MSA	Robbins & Monro (1951)
MSA ranking	MSAR	Sbayti et al. (2007)
Projection method	PM	**Adapted** Halpern (1967)
Projection Initialization	PI	**Adapted** Friesz et al. (2011)
Initialization MSA	IMSA	**New**
Gap-based	GB	Lu et al. (2009)
Gap-based Normalized	GBN	**New**
Probabilistic	Prob.	**Ameli et al. (2017)**
Step size Probabilistic	SSP	Halat et al. (2016)
Gap-based Probabilistic	GBP	Verbas et al. (2015)
Boost up Gap-based	BGB	**New**
Inner loop initialization (Step 5)		
Default method	All-or-nothing	Lu et al. (2009)
Keep the assignment	Keep solution	**New**
Initial step size selection (Step 6)		
Default method	Initial	Lu et al. (2009)
Re-initializing the step size	Reset	**New**
Smart step size	Smart	**New**

5 | NUMERICAL EXPERIMENTS

In this work, we use Symuvia[1] as a trip-based simulator for calculating travel time in traffic networks. Symuvia is a microscopic simulator based on the Lagrangian resolution of the LWR model (Leclercq, Laval, & Chevallier, 2007). This car-following law has been further extended to account for all the features of urban traffic: bounded acceleration (Leclercq, 2007a), lane-changing with relaxation (Laval & Leclercq, 2008), multi-class (Leclercq & Laval, 2009), signalized and unsignalized intersections (Chevallier & Leclercq, 2009a), roundabouts (Chevallier & Leclercq, 2009b). The simulation time-step is equal to 1 second, and we retrieve the travel time information at the link and node level every 1 minute. This study considers three networks with different topologies to investigate if the network size influences the algorithm settings. Note that all the networks in this study are mono-modal, i.e., with car-traffic only. A 5×5 grid network (5by5) is used for the smallest scale network, see Figure 2(a). All the intersections are signalized, and the green and red light duration is set to 30 seconds. The simulation period is 2 hours for 19 origins and 16 destinations. The medium-scale network exemplifies a Manhattan type city with a ring road (Ring city), see Figure 2(b). This network corresponds to 14×14 two-way regular roads with a speed limit of 50km/h. These roads delimit blocks that are grouped 3 by 3 to form 5×5 zones. A two-way ring road with a speed limit of 90km/h has 12 interchanges with peripheral zones. All the intersections are signalized except the interchanges with the ring road, and the green and red light duration is set to 30 seconds. Ring city is simulated for 50 minutes with 26 origins and 24 destinations.

The large-scale network of this study is the network of two French cities: Lyon 6e + Villeurbanne (Lyon6V). This network has 1,883 Nodes, 3,383 Links, 94 Origins, 227 Destinations. All the signalized intersections in the real field have been implemented in the simulator with their actual signal timing. It is illustrated in Figures 2(c) and 2(d). The network is loaded with travelers of all ODs with given departure times in order to represent 2.5 hours of the network with the demand of three levels of saturation based on the study of (Krug, Burianne, & Leclercq, 2017).

The Macroscopic Fundamental Diagram (MFD) shows the rapid evolution and gives a synthetic overview of network states. It usually distinguishes three situations. First, the curve increases from (0, 0) and traffic states remain under-saturated when demand is light. This is referred to as the Under Saturation (US) scenario. Travel production, which is equivalent to the total travel distance for a given period of time, stabilizes while the accumulation (or total travel time) still increases,



(a) 5by5 (b) Ring city

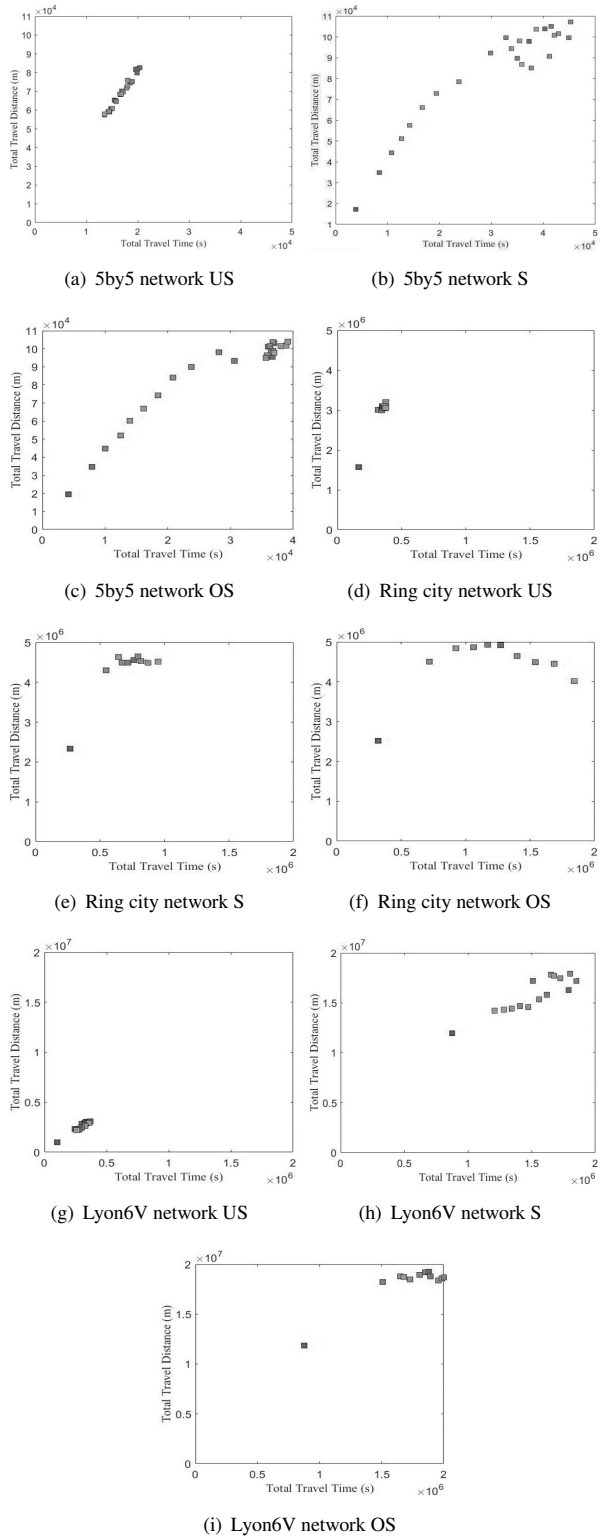
(c) ©Google (d) Lyon6V

FIGURE 2 The three traffic networks of this study. (a): Small-scale network. (b): medium-scale network. (c) and (d): Large-scale network.

i.e., the network progresses to maximal capacity and then it quickly becomes unloaded. This corresponds to the Saturation (S) scenario when we reach to the maximal capacity of the network, and then unload it. Finally, production decreases when the total travel time continues increasing. This is the Over Saturation (OS) scenario where the network is heavily congested and remains at the maximum capacity of the network for a long time. We tune the demand to observe all three levels of saturation in numerical experiments to assess the impact of network loading on the performance of the fixed-point algorithm. Table 2 presents the total demand for all the test cases of this study. The MFDs of each demand scenario for all the networks are presented in Figure 3. The points in the figures represent the state of the network at each successive 5-min time period. The MFD of the US scenarios shows the evolution of the state of the networks with almost free-flow travel time during the simulation for each network (Figures 3(a), 3(d) and 3(g)). The saturated MFD diagrams for each network (Figures 3(b), 3(e) and 3(h)) show that the S demand scenarios put the states of the networks in saturation level during the simulation. Finally, the OS scenario moves the network states to a heavily congested situation in which the saturation level is very high (Figures 3(c), 3(f) and 3(i)).

Demand is constant, but the network state evolves dynamically due to spreading congestion in all three scenarios.

[1]Note that Symuvia is an open-source simulator which will be made freely available during summer 2019 (http://www.licit.ifsttar.fr).



(a) 5by5 network US (b) 5by5 network S

(c) 5by5 network OS (d) Ring city network US

(e) Ring city network S (f) Ring city network OS

(g) Lyon6V network US (h) Lyon6V network S

(i) Lyon6V network OS

FIGURE 3 The macroscopic fundamental diagram of 9 demand scenarios of the three traffic networks. There are 3 different saturation levels per network: Under Saturation (US), Saturation (S) and Over Saturation (OS).

TABLE 2 Total demand for all test cases

Level of saturation	5by5	Ring city	Lyon6V
Under Saturation (US)	3,520	12,000	20,000
Saturation (S)	5,197	19,000	54,190
Over Saturation (OS)	8,100	22,500	100,000

6 | RESULTS

All the experiments are first initiated with the All-or-nothing assignment algorithm (see Step 1 in Figure 1). In this study, we impose a limit on the maximum number of iterations and compare the final solutions obtained by the different algorithms. This is to contain the computational cost and to bring the experiments closer to usual practice in which traffic engineers try to control maximum computation times. Each scenario is executed for five outer loops ($j_{max} = 5$). This means that users have to finally choose between a minimum of six paths (the first path comes from Step 1) for each OD pair. Note that the maximum number of the outer loop iterations is not the limitation. At one point, increasing the number of discovered paths will simply add close alternatives that do not improve the solution. To confirm this intuition, we rerun the large-scale scenario with the saturated level of demand with an increasing number of the outer loop. It appears that the final gap values, even when the number of outer loop iterations reaches 20, remain very close to the original one with five outer loop iterations only. So, we keep $j_{max} = 5$ for all scenarios.

The inner loops run for a maximum of ten iterations ($i_{max} = 10$) for a small network, twenty for Ring city ($i_{max} = 20$), and thirty for a large-scale network ($i_{max} = 30$). We restrict the number of inner loop iterations for each network to obtain a complete comparison of the performances of the algorithms in the same settings. The different values of i_{max} are determined as a function of the size of the network, based on the studies of Halat et al. (2016); Verbas et al. (2016). All the experiments are conducted on a 64-bit personal computer with a 12-core central processing unit of 2.10 GHz speed, and a memory of 128 GB. The experiments are conducted in three stages. First, we compare all the swapping algorithms. Then the best swapping algorithms are chosen for the second stage. The initialization methods are tested in the second stage. Finally, the best combinations of swapping algorithms and initialization methods are considered for the third stage. The step size methods are examined in the third stage.

6.1 | Comparison of swapping algorithms

The first stage of the numerical experiments entails finding the best swapping algorithm. We track *AGap* and Violation indicators to assess the performance of each swapping algorithm.

The final values are presented in Table 3 for the nine scenarios. Metal colors highlight the top three algorithms (Gold = first, Silver = second, Bronze = third) for each scenario. Moreover, Figure 4 presents a bar chart of the computation times.

Regarding the uniqueness of the UE, this is the case for the DNL problem when users have the same characteristic (homogeneous demand) and travel time functions are increasing (Ameli, Lebacque, & Leclercq, 2018; Iryo & Smith, 2017). Monotonicity exists at the link level for the traffic simulator of this study (Leclercq, 2007b) but not at the node because most of the intersections are signalized. Therefore, we cannot claim, on the basis of the literature, that we have unicity; however, we have a similar solution in terms of path flow distribution whatever the algorithm used in each case. In particular, in the small-scale network (5by5), MSA ranking, Probabilistic, and Projection initialization algorithms lead to optimal values for the US scenario (Table 3). Furthermore, the path flow distributions are equal, possibly providing the optimal UE for this scenario.

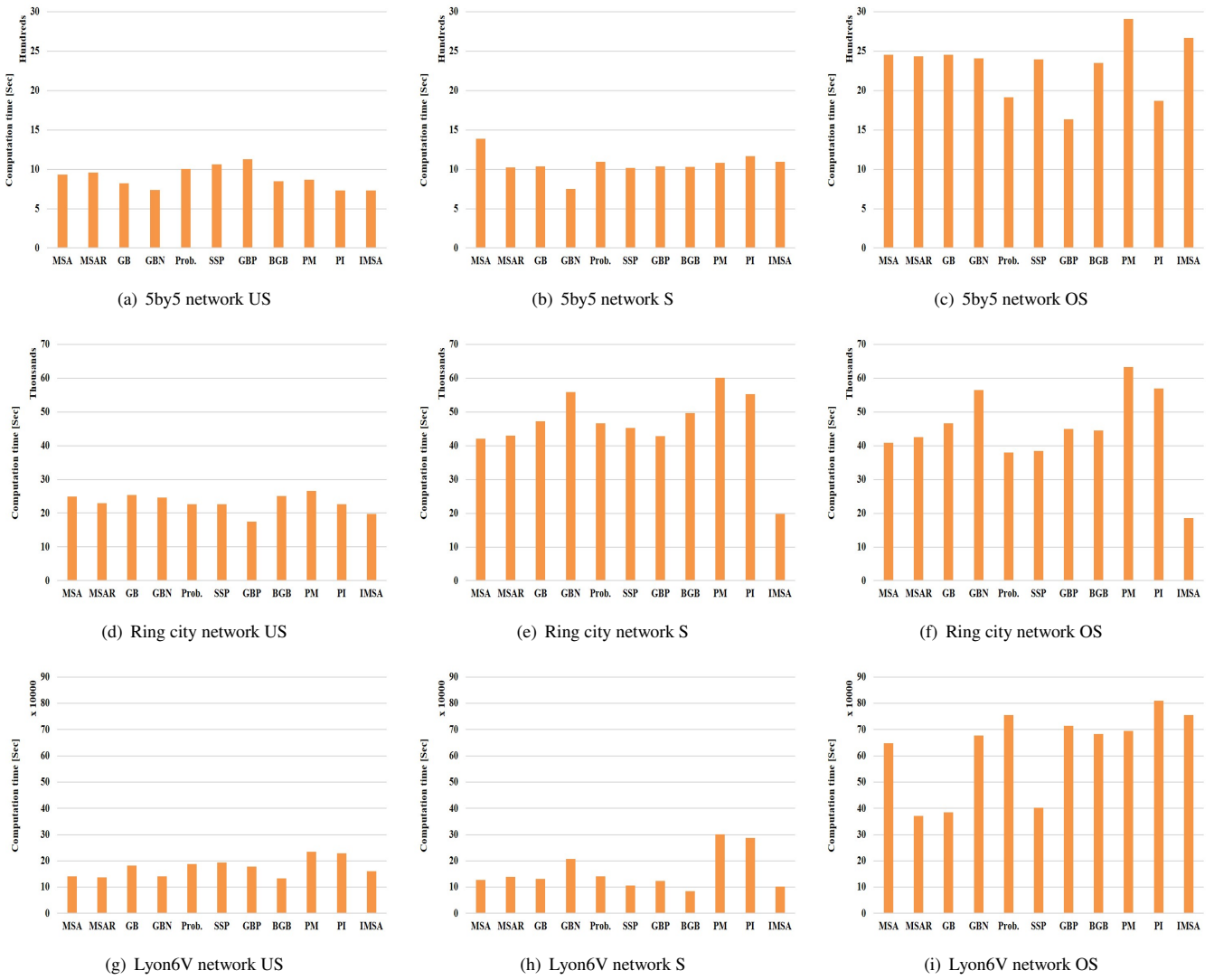
According to Figure 4(a), the Projection initialization algorithm converges fastest to the best solution. In the saturation scenario, the best solution is obtained by Probabilistic algorithm, but Gap-based normalization algorithm is the fastest algorithm to converge (Figure 4(b)) while the *AGap* by this algorithm is ranked second among all the algorithms. In the saturation scenario, the results provided by the Gap-based, Boost-up Gap-based, Projection initialization, and Initialization MSA algorithms are poor in comparison with the best solution. In the over-saturation scenario, the Gap-based normalization and Boost-up Gap-based algorithms perform well in terms of solution quality and CT.

Table 3 shows that the Probabilistic algorithm provides the best solutions in all the scenarios of the medium-scale network (Ring city). For the US scenario, the Gap-based Probabilistic algorithm converges fast (Figure 4(d)) and also to a good solution, with a *AGap* only 0.28 second longer than the best solution. In the S scenario, the Probabilistic algorithm is significantly better than the others in terms of solution quality. Initialization MSA algorithm converges fast (Figure 4(e)) but to a bad solution, which is more than 25 times bigger than the best solution.

In the large-scale network (Lyon6V), The results show that many algorithms with low CTs converge to solutions far from the best solution. This happens because the solution space for the large-scale is more complicated than the solution space of smaller networks, and most of the solution algorithm with step size can be stick to the local optimum and cannot explore very different path flow distribution. For instance, in the OS scenario, the MSA ranking, Gap-based and Step size Probabilistic algorithms converge faster than the others (Figure 4(i)), but the *AGap* and Violation are very poor in comparison with the

TABLE 3 Results of numerical experiments for eleven swapping algorithms [*AGap* (second)]

Network		/ algorithm	MSA	MSAR	GB	GBN	Prob.	SSP	GBP	BGB	PM	PI	IMSA
5by5	US	AGap	0.91	0.17	1.10	0.91	0.17	0.88	0.73	0.91	0.48	0.17	1.47
		Violation	0.05	0.05	0.08	0.05	0.05	0.08	0.07	0.06	0.06	0.05	0.05
	S	AGap	11.22	13.34	82.65	11.22	3.50	39.18	22.36	64.05	37.79	112.09	80.19
		Violation	0.13	0.13	0.16	0.13	0.13	0.18	0.16	0.18	0.18	0.16	0.17
	OS	AGap	0.70	0.85	1.18	0.70	0.93	1.44	2.37	0.71	1.05	1.56	1.19
		Violation	0.04	0.09	0.08	0.04	0.05	0.07	0.09	0.08	0.08	0.08	0.08
Ring city	US	AGap	5.79	2.99	8.66	4.30	2.74	5.47	3.02	5.13	4.41	3.27	38.13
		Violation	0.13	0.06	0.13	0.11	0.06	0.12	0.08	0.10	0.10	0.10	0.24
	S	AGap	30.96	12.81	18.31	16.35	8.33	40.13	12.33	24.72	200.71	206.70	205.24
		Violation	0.17	0.16	0.21	0.15	0.16	0.19	0.16	0.22	0.36	0.38	0.36
	OS	AGap	50.06	20.81	47.73	49.35	16.16	36.16	48.38	26.00	230.71	189.04	225.24
		Violation	0.26	0.16	0.17	0.26	0.14	0.21	0.20	0.18	0.36	0.32	0.33
Lyon 6V	US	AGap	109.47	18.78	37.01	41.09	12.30	16.15	13.59	35.47	20.01	41.09	39.75
		Violation	0.32	0.15	0.20	0.24	0.12	0.14	0.13	0.21	0.19	0.24	0.24
	S	AGap	79.89	39.54	237.35	29.14	24.72	47.73	121.45	62.98	1157.21	27.83	798.47
		Violation	0.25	0.24	0.35	0.16	0.13	0.25	0.26	0.25	0.35	0.15	0.43
	OS	AGap	180.38	233.44	183.83	80.79	106.83	339.87	153.00	240.95	108.45	73.43	47.18
		Violation	0.22	0.28	0.23	0.17	0.19	0.30	0.34	0.35	0.27	0.15	0.14



(a) 5by5 network US (b) 5by5 network S (c) 5by5 network OS

(d) Ring city network US (e) Ring city network S (f) Ring city network OS

(g) Lyon6V network US (h) Lyon6V network S (i) Lyon6V network OS

FIGURE 4 The computation time bar chart of 9 demand scenarios of the three traffic networks. There are 3 different saturation levels per network: Under Saturation (US), Saturation (S) and Over Saturation (OS)

best solution. The same observation can be made for the MSA, Gap-based, Gap-based Probabilistic, and Boost-up Gap-based algorithms in the saturation scenario and for the Gap-based Normalized in the US scenario. In the OS scenario, no algorithms except the Initialization MSA can find solutions with a *AGap* less than a minute. The CTs are also long for two top algorithms.

Moreover, the results in Table 3 show that the performance of the swapping algorithms depends not only on the scale of the traffic network but also on the saturation level. Because the solution space of the fixed-point problem becomes more complex when the level of saturation is increased. As a result, the intermediate solutions of the inner loop are not stable. In other words, when the level of congestion increases, the correlation of ODs, which share links, is increased. Therefore, any change in the reassignment process can change significantly the quality indicators. The performance analysis of the algorithms is analyzed as follows:

- The MSA method works well in small-scale, but by increasing the size of the traffic network, it obtains poor results for *AGap* and Violation in comparison with the other algorithms. It is noteworthy that the MSA algorithm is one of the worst algorithms on the large-scale, particularly for the US scenario. The reasons for this performance of the drawbacks of MSA algorithms (Section 4.1.1), but its CT is comparable with other algorithms in the US and S scenario.
- The MSA ranking algorithm is not among the top three algorithms in the large-scale. However, because of the

ranking process, it provides good enough solution compared to the best algorithm except in the OS scenario, where the MSA ranking algorithm converges faster than the others but to a bad solution. This algorithm is still suffering the drawback of predefined step size, which force this algorithm to converge.

- The Gap-based algorithm does not appear in the top three algorithms for all the scenarios, but it obtains solutions with small *AGap* for all the US demand scenarios. In these test cases, when increasing the level of network saturation, the Gap-based algorithm cannot converge to a high-quality solution. Using the gap indicator in the step size formula helps to converge to the local optimum while preventing the algorithm from exploring a wide range of solutions for the global optimum.

- The Gap-based Normalized algorithm performs better than the Gap-based algorithm in most of cases. The normalization techniques improve the performance of the Gap-based method, especially in the saturation and OS scenarios.

- Generally, the Probabilistic algorithm is the best algorithm in the US and S scenarios of all the networks and also it also provides good performance for the OS scenario in the small and the medium test cases. It should be recalled that the Probabilistic algorithm is the only step size free algorithm of this study. It can explore the solution space without limitation, which makes it more robust than other algorithms. However, in the large-scale and OS scenario, the CT of Probabilistic algorithm is very high, and it could not provide a good solution compared to the best algorithm. It means that this algorithm does not fully cover the solution space under the determined computation budget.

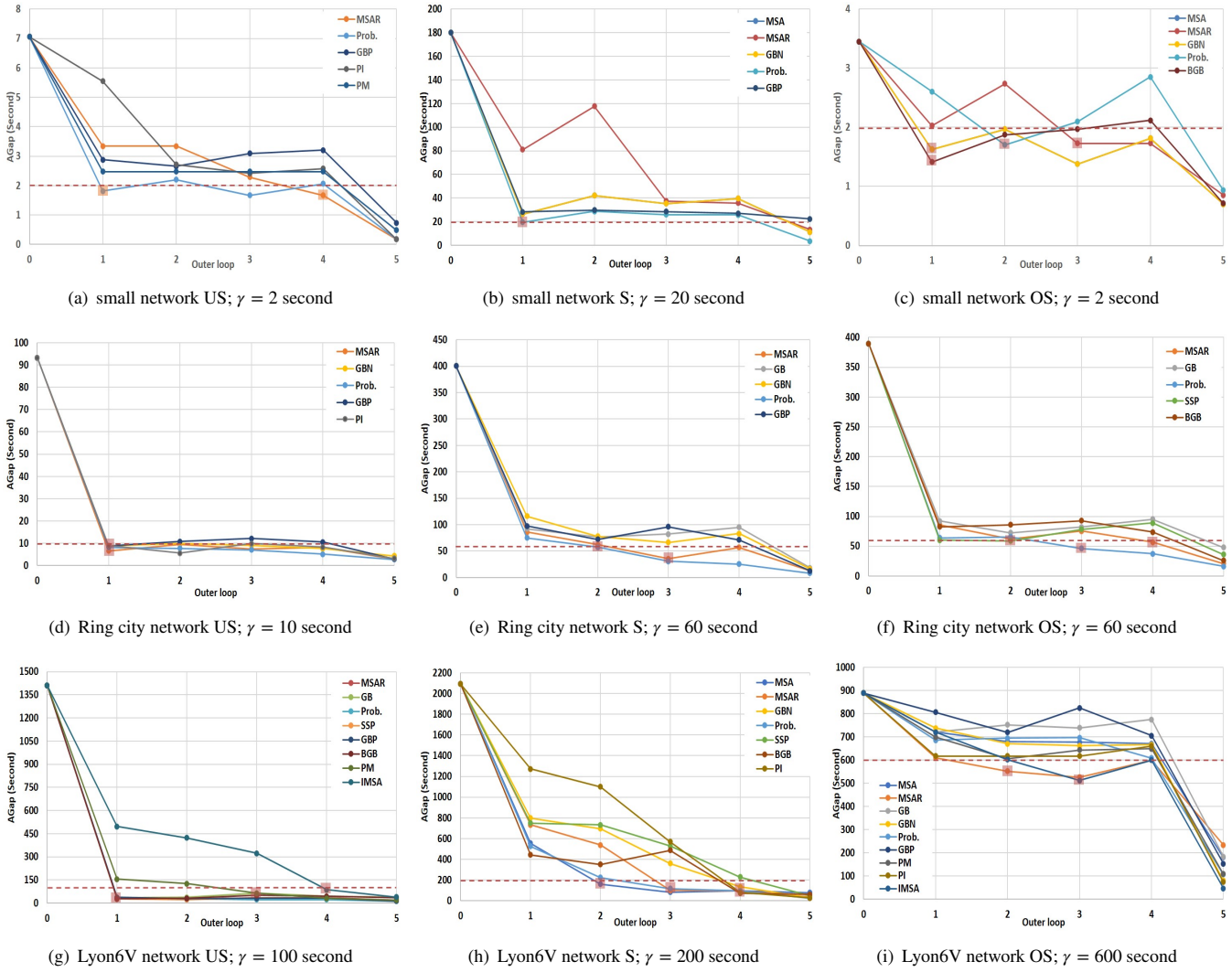
- The hybrid algorithms (Section 4.1.6) work better than MSA and Gap-based algorithms in the large-scale and US scenario, but they are dominated by Probabilistic algorithm because the step size still limits them. The Step size Probabilistic algorithm, which is a combination of MSA and Gap-based algorithms, provides better results than both methods in the medium- and the large-scale networks with US and saturation levels of demand. When the saturation level of the network increase, the Gap-based Probabilistic algorithm cannot provide a good solution. The Step size Probabilistic and Gap-based Probabilistic algorithms are dominated by the Probabilistic algorithm in all test cases. The Boost-up Gap-based algorithm, which used the boost-up step size performs better in OS scenarios where there is a large number of trips to optimize.

- The projection-based algorithms (Section 4.1.3) are good for small scale. Their CT is increased significantly by the saturation level of the network. In addition, they have a high CT in the large-scale network compared to other algorithms. Using the initialization technique in the swapping formula increases the algorithm CT significantly, particularly in saturated level and does not help that much the algorithm to provide a better solution.

Figure 5 presents the convergence pattern of swapping algorithms. Top five algorithms in terms of the *AGap* indicator for each saturation level of small- and medium-scale networks are presented in Figures 5(a)-5(f). For the large-scale network, we present more swapping algorithm convergence pattern (Figure 5(g)-5(i)) to analyze more algorithms.

According to Figure 5, the saturation level has an impact on the scale of *AGap* at each level of the outer loop during the optimization process. In the US scenario, the value decreases suddenly for the first outer loop but for more saturated scenarios adding the new shortest path changes *AGap* scale of users, particularly in the large-scale network. As shown in figure 5(i), the gap value slumps for the outer loop five, where users have six alternatives per OD to choose.

In addition to the quality of the final solution, converging with the less number of outer loops ($j < 5$ in this study) to a solution with good quality is important in practice. In other words, the algorithm with a minimum number of outer loops and good solution in terms of *AGap* can be more efficient. Note that the solution with good quality means the *AGap* of the solution is below a predetermined satisfaction threshold. Here, we consider a given satisfaction threshold (γ) for the quality of the solution at outer loop iteration j ($AGap_j$) in order to investigate which algorithms can reach faster this threshold ($AGap_j \leq \gamma$). The values of γ for US and S scenarios are approximately equal to $\lceil 2 \times AGap^* \rceil + 1$ where $AGap^*$ denotes the best quality of the final solution obtained by swapping algorithm in second. For the OS scenarios in large-scale, we set the γ to 9 minutes based on the initial *AGap* ($\approx$ 15 min) which is obtained by the All-or-nothing algorithm. Figure 5 presents the γ values for each scenario. We can see in this figure, the first time that each algorithm is positioned below γ threshold before the last iteration and then determine the minimal number of the outer loop to guaranty a given level of performance. The Probabilistic algorithm can converge faster in the small and medium networks according to the γ value. For instance, about the Ring city network saturated scenario, if we want to reach the level of $AGap_j = 1$ minute ($\gamma = 60$), we need to do only one outer loop iteration with the Probabilistic algorithm. In the large-scale considering the γ threshold helps to save CT. In the large-scale network, the MSA algorithm for the saturated scenario, and the MSA ranking algorithm for OS scenario are much faster than others and need only two outer loops to reach the threshold.



(a) small network US; $\gamma = 2$ second (b) small network S; $\gamma = 20$ second (c) small network OS; $\gamma = 2$ second

(d) Ring city network US; $\gamma = 10$ second (e) Ring city network S; $\gamma = 60$ second (f) Ring city network OS; $\gamma = 60$ second

(g) Lyon6V network US; $\gamma = 100$ second (h) Lyon6V network S; $\gamma = 200$ second (i) Lyon6V network OS; $\gamma = 600$ second

FIGURE 5 Convergence patterns for the swapping algorithms of all scenarios. There are 3 different saturation levels per each network: Under Saturation (US), Saturation (S) and Over Saturation (OS). γ denotes the satisfaction threshold.

TABLE 4 Best swapping algorithms

Network	5by5			Ring city			Lyon 6V		
Scenario	US	S	OS	US	S	OS	US	S	OS
MSA		✓	✓						
MSAR	✓	✓	✓	✓	✓	✓	✓	✓	✓
GB									
GBN		✓	✓					✓	✓
Prob.	✓	✓		✓	✓	✓	✓	✓	✓
SSP							✓	✓	
GBP				✓	✓		✓		
BGB			✓			✓			
PM							✓	✓	✓
PI	✓								✓
IMSA									✓

All our analyses give good hints about which are the best swapping algorithms. As we discussed in Section 4, there are different components in the optimization process, which can help the swapping algorithm to find a better solution. Consequently, the swapping algorithms with good quality indicators ($AGap$ and Violation) for the final solution and CT in each scenario (all the golden cells in Table 3) are chosen for the next stage of the numerical experiment. We also consider all algorithms which obtain a solution with a similar value for $AGap$ and Violation to the best algorithms solution for the next stage. For more investigation in the large-scale, we choose at the swapping algorithms including all colored cells in Table 3 and algorithms with $AGap_j \leq \gamma;\ \forall j < j_{max}$ in Figure 5 for each saturation level. In other words, we also consider the swapping algorithms which converge to the γ before the last outer loop.

Table 4 presents which algorithms are selected in each scenario for further investigation. MSA ranking and Probabilistic algorithms are suitable for all scenarios. MSA algorithm is efficient for small network, and Projection method and Step size Probabilistic algorithm are efficient for the large-scale network. Gap-based Normalized and Boost-up Gap-based algorithms provide good results in more saturated scenarios, while Gap-based Probabilistic algorithm is good for US scenarios and larger networks.

6.2 | Initialization methods

The second stage of the numerical experiments corresponds to Step 5 in the solution algorithm (Figure 1). The methodology improvements to this step are discussed in Section 4.2. Now, an alternative initialization method (Keep solution; Table 1) for the inner loop is applied for all numerical experiments. The results for combinations of swapping algorithms and initialization methods are presented in Tables 5 and 6. Table 5 presents the quality indicators of final solutions for initialization methods in all networks. About 5by5 network, in the US scenario, it is shown that Keep solution initialization obtains the optimal solution for all best three algorithms. Also, in S and OS scenarios, this method improves the performance of all swapping algorithms in comparison with the same swapping algorithm and All-or-nothing initialization. The results on the small network show that the Probabilistic and Keep solution combination provides the best solution in all saturation levels.

In Ring city network, same as 5by5, the Keep solution method improves the performance of swapping algorithms in particular for the Probabilistic method. The differences between All-or-nothing and Keep solution violation value of two closed *AGap* solution in Table 5, box of MSA ranking and S scenario, shows the different direction of searching by swapping algorithm and the impact of the starting point at the beginning of each outer loop on the final result of optimization methods.

In the large-scale test case, the impact of the initialization is significant. It means that the quality of the solution by keep solution method is always better than All-or-nothing with a considerable difference of *AGap*, e.g., the Keep solution method obtains a better solution (more than 50% lower *AGap*) for Step size Probabilistic and Probabilistic algorithms in US and saturation levels.

Table 6 presents the CT of all experiments in this stage. First, the results prove that the initialization method has an impact on CT. Second, the Keep solution method which is indicated by "Keep" in the Table 6 improves the speed of convergence for all swapping algorithms in comparison with the All-or-nothing method which is the default method in the literature. Third, the combination of Probabilistic and Keep

TABLE 5 Results of initialization methods [*AGap* (second)]

Network		5by5						Ring city						Lyon 6V					
Scenario		US		S		OS		US		S		OS		US		S		OS	
Algorithm / Indicator		AGap	V(G)	AGap	V(G)	AGap	V(G)	AGap	V(G)	AGap	V(G)	AGap	V(G)	AGap	V(G)	AGap	V(G)	AGap	V(G)
MSA	AoN	-		11.22	0.13	0.70	0.04	-		-		-		-		-		-	
	Keep			5.60	0.21	0.68	0.03												
MSAR	AoN	0.17	0.05	13.34	0.13	0.85	0.09	2.99	0.06	12.81	0.16	20.81	0.16	18.78	0.15	39.54	0.24	233.44	0.28
	Keep	0.17	0.05	7.78	0.15	1.10	0.05	1.89	0.08	12.34	0.17	15.25	0.12	13.51	0.13	16.42	0.08	162.60	0.28
GBN	AoN	-		11.22	0.13	0.70	0.04	-		-		-		-		29.14	0.16	80.79	0.17
	Keep			6.48	0.21	0.46	0.04									22.47	0.11	69.77	0.15
Prob.	AoN	0.17	0.05	3.50	0.13	-		2.74	0.06	8.33	0.16	16.16	0.14	12.30	0.12	24.72	0.13	106.83	0.19
	Keep	0.17	0.05	0.86	0.10			1.47	0.05	5.07	0.09	6.35	0.10	5.96	0.09	10.59	0.06	92.83	0.19
SSP	AoN	-		-		-		-		-		-		16.15	0.14	47.73	0.25	-	
	Keep													6.03	0.09	13.58	0.08		
GBP	AoN	-		-		-		3.02	0.08	12.33	0.16	-		13.59	0.13	-		-	
	Keep							1.53	0.05	6.15	0.13			6.79	0.09				
BGB	AoN	-		-		0.71	0.08	-		-		26.00	0.18	-		-		-	
	Keep					0.50	0.01					10.48	0.14						
PM	AoN	-		-		-		-		-		-		20.01	0.19	27.83	0.15	108.45	0.27
	Keep													7.33	0.09	21.58	0.11	101.49	0.19
PI	AoN	0.17	0.05	-		-		-		-		-		-		-		73.43	0.17
	Keep	0.17	0.05															62.60	0.15
IMSA	AoN	-		-		-		-		-		-		-		-		47.18	0.14
	Keep																	40.89	0.15

TABLE 6 Computation time of initialization methods (second)

Network		5by5			Ring city			Lyon 6V		
Scenario		US	S	OS	US	S	OS	US	S	OS
MSA	AoN	-	1384	2451	-	-	-	-	-	-
	Keep		713	1184						
MSAR	AoN	956	1021	2430	22962	42973	42447	137527	139232	371695
	Keep	730	624	2306	18130	39262	39418	154138	132750	333751
GBN	AoN	-	754	2406	-	-	-	-	206912	677682
	Keep		739	1965					197719	556489
Prob.	AoN	1003	1093	-	22586	46536	38015	187394	141885	754640
	Keep	595	664		15431	28517	36462	138046	111598	291486
SSP	AoN	-	-	-	-	-	-	194636	111824	-
	Keep							135540	106705	
GBP	AoN	-	-	-	17541	42839	-	177725	-	-
	Keep				23140	36431		121653		
BGB	AoN	-	-	2349	-	-	44461	-	-	-
	Keep			1899			27358			
PM	AoN	-	-	-	-	-	-	234809	287847	693828
	Keep							155722	179288	653254
PI	AoN	731	-	-	-	-	-	-	-	809785
	Keep	492								741501
IMSA	AoN	-	-	-	-	-	-	-	-	754350
	Keep									671548

solution method is faster than other methods in most of the cases as we can observe that this method is always in the top two fastest methods in all scenarios (gold and silver cells in Table 6).

In order to have a better comparison between the performance of different initialization methods, the inner loop convergence patterns of MSA ranking and Probabilistic algorithms with both initialization methods are presented in Figure 6. The results correspond to the saturation scenario for the large-scale network. The default setting, which is the "All-or-nothing" method, starts at each outer loop with the same assignment pattern. It works like as handbrake at the beginning of each outer loop and forces the algorithm to start from the assignment with a high *AGap*. Therefore, the swapping algorithm should attempt to come back to the optimal region by several iterations.

It is obvious that "All-or-nothing" method needs more iteration (CT) compared to the Keep solution method to find the new optimal solution for each outer loop. The Keep solution method converges faster than the other one and also provides a better solution (Table 5). For instance, in Figure 6, The Keep solution method prevents the MSA ranking method from spending several iterations in order to come back to the lower range of *AGap* and also helps this swapping algorithm to converge faster than the initial version. Note that the first outer loop is the same for both methods and then from the second one "MSA ranking + Keep solution" continue with previous path flow distribution, but the initial version of the MSA ranking starts the second outer loop with the All-or-nothing assignment.

Moreover, Figure 6 shows the flexibility of probabilistic algorithm to search the solution space. The probabilistic algorithm moves in a larger range of *AGap* during each outer loop, and it does not improve the solution sequentially except when the probability of swapping (equation (25)) is very low for all users, i.e., the algorithm is close to the optimal solution.

6.3 | Step size methods

In the previous stage, we presented and discussed the results of the initialization method, i.e., Step 5 in Figure 1. Here, we will present and discuss the results for the step size methods (Step 6). For the final stage, we select the best methods (colored in Tables 5 and 6) except the Probabilistic algorithm. The

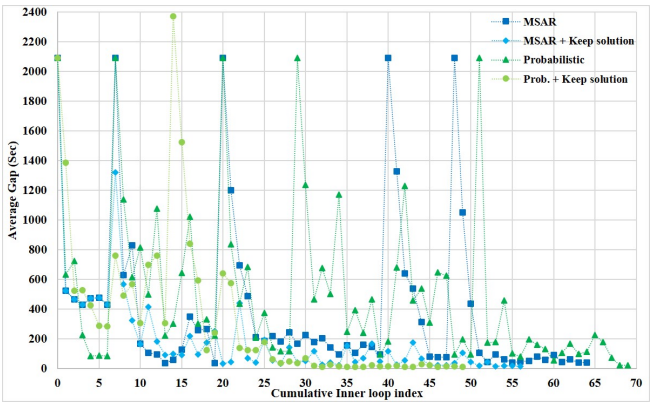


FIGURE 6 Convergence patterns of the inner loops with initialization methods for Saturation (S) scenario on Lyon6V network. [default is All-or-nothing initialization]

Probabilistic method is excluded simply because it does not have a step size (Section 4.1.5). For methods with step size, the one or two best combinations in all test cases are chosen. The new methods for step size initialization are compared with the initial method (Section 4.3). Note that we fix the seed for probability functions in the Step size Probabilistic and Gap-based Probabilistic in order to compare the impact of step sizes. The results of applying the three methods on the three networks are presented in Tables 7 and 8. Medal colors highlight the top three combinations for each scenario.

Table 7 presents the quality indicators for the final solution of all experiments for all networks. Smart step size improves the performance of the MSA and Gap-based Normalized algorithms. The Reset method improves the MSA for the small network. For the medium-scale network (Ring city), the results show that the Reset method improves the MSA ranking algorithm, and the Smart method works well with Gap-based Probabilistic algorithm (Table 7). In the Boost-up Gap-based algorithm, the step size of this method is already modified at each inner loop by boost-up techniques. The result shows that applying the alternative step size methods cannot improve the performance of this algorithm.

The results in Table 7 for the large-scale show that the Smart method improves the solution of Gap-based Probabilistic and Initialization MSA algorithms compared to the Initial method which is the default setting for step size initialization and Reset method. Same as Ring city network, the Reset method improves the performance of Step size Probabilistic algorithm. The CT for all experiments of this stage is presented in Table 8. The Smart and Reset methods converge slowly in the small network, but they converge faster than the Initial method in the large-scale network particularly the Smart method for Initialization MSA and oversaturated scenario compared to other methods with which we can save a minimum of one day of computation.

TABLE 7 Results of initial step size methods [*AGap* (second)]

Network	5by5						Ring city						Lyon 6V					
Scenario	US		S		OS		US		S		OS		US		S		OS	
Algorithm / Indicator	AGap	V(G)	AGap	V(G)	AGap	V(G)	AGap	V(G)	AGap	V(G)	AGap	V(G)	AGap	V(G)	AGap	V(G)	AGap	V(G)
MSA Initial			5.60	0.21														
MSA Reset	-		2.36	0.19	-		-		-		-		-		-		-	
MSA Smart			3.77	0.18														
MSAR Initial	0.17	0.05					1.89	0.08	12.34	0.17	15.25	0.12						
MSAR Reset	0.17	0.05	-		-		1.47	0.05	9.57	0.15	8.05	0.10	-		-		-	
MSAR Smart	0.17	0.05					1.71	0.08	11.25	0.18	12.38	0.12						
GBN Initial					00.46	0.04												
GBN Reset	-		-		0.62	0.03	-		-		-		-		-		-	
GBN Smart					0.45	0.03												
SSP Initial													6.03	0.09	13.58	0.08		
SSP Reset	-		-		-		-		-		-		5.96	0.09	11.44	0.08	-	
SSP Smart													12.13	0.12	17.75	0.09		
GBP Initial							1.53	0.05	6.15	0.13			6.79	0.09				
GBP Reset	-		-		-		1.53	0.05	5.73	0.09	-		6.50	0.10	-		-	
GBP Smart							1.47	0.05	5.49	0.09			6.42	0.09				
BGB Initial					0.50	0.08					10.48	0.14						
BGB Reset	-		-		0.45	0.02	-		-		21.38	0.19	-		-		-	
BGB Smart					0.69	0.04					11.31	0.11						
PI Initial	0.17	0.05																
PI Reset	1.83	0.07	-		-		-		-		-		-		-		-	
PI Smart	0.17	0.05																
IMSA Initial																	40.89	0.15
IMSA Reset	-		-		-		-		-		-		-		-		35.49	0.14
IMSA Smart																	34.10	0.14

7 | CONCLUSIONS

This paper focuses on improving the solution algorithms for finding the user equilibrium considering trip-based dynamic network loading. We highlight the current drawbacks of existing swapping algorithms and propose several solutions to overcome them, and speed up the convergence. A significant contribution of this paper is the full benchmark of all algorithms for different network size and level of saturation. We compare the performance of the solution algorithms (see Table 1 as a synthesis of main algorithms and methods in this study) based on the quality of solutions and computation times. Table 9 presents the best configuration of the solution algorithm that

TABLE 8 Computation time of initial step size methods (second)

Network		5by5			Ring city			Lyon 6V		
Scenario		US	S	OS	US	S	OS	US	S	OS
MSA	Initial	-	713	-	-	-	-	-	-	-
	Reset		337							
	Smart		705							
MSAR	Initial	730	-	-	18130	39262	39418	-	-	-
	Reset	2430			9086	34439	35562			
	Smart	3381			11033	24550	19506			
GBN	Initial	-	-	1965	-	-	-	-	-	-
	Reset			2337						
	Smart			2219						
SSP	Initial	-	-	-	-	-	-	135540	106705	-
	Reset							81498	101141	
	Smart							103695	89258	
GBP	Initial	-	-	-	23140	36431	-	121653	-	-
	Reset				3878	37892		107142		
	Smart				13927	25805		92989		
BGB	Initial	-	-	1899	-	-	27358	-	-	-
	Reset			2796			26114			
	Smart			2747			27557			
PI	Initial	492	-	-	-	-	-	-	-	-
	Reset	1908								
	Smart	3270								
IMSA	Initial	-	-	-	-	-	-	-	-	671548
	Reset									571493
	Smart									485980

makes the best compromise between quality and computation time for all network size and saturation levels.

According to Table 9 and the results, this study shows that:

- The network size and saturation level has an impact on the performance of solution algorithms to solve the DTA problem.
- The combination of Probabilistic approach (without step size) and Keep solution initialization appears in most of the cases in Table 9 as the best algorithm.
- The initial assignment and step size at the beginning of the outer loop have a significant impact on the final solution and convergence speed of the algorithm.
- An alternative method, proposed by this study, to initialize the assignment pattern at the beginning of the outer loop (Section 4.2) improves the performance of all swapping algorithms compared to the recent methodology in the literature (Table 9).
- Two new methods, proposed by this study, for the initialization of the step size (Section 4.3) ensures that the algorithm converges and it has a direct impact on the speed of convergence. However, the step size cannot guarantee the quality of the final solution.

The analysis for the impact of The network size and saturation level on the convergence process shows that the classic algorithms (e.g., MSA) exhibit good performance in the small-scale network, but they do not provide a good solution in the large-scale network. The computational cost of the classic algorithms are also prohibitive for the large-scale network. One of the hybrid algorithms, Step size Probabilistic (see section 4.1.6), works faster in the large-scale network (Table 9). The MSA ranking is efficient for small- and medium- scale networks, but it cannot provide good results for the large-scale network. Moreover, the results show that some algorithm such as Gap-based algorithm, and Projection method are dominated by other algorithms for all scenarios.

The combination of Probabilistic approach (without step size) and Keep solution initialization is not necessarily always the best in terms of quality and speed, but it is the one that is most likely to obtain the best solution in all scenarios and can be considered as the most robust alternative. If the focus is only on the quality of the solution, more than one configuration can be used in most of the cases (Table 9). However, the computation time is very important, particularly for the large-scale network. For instance, in the large-scale and oversaturated scenario, Probabilistic algorithm cannot provide a better solution than Initialization MSA algorithm, but it is more than two days faster than Initialization MSA algorithm in computation time (Table 6).

About the two new methods for step size, they provide a better solution with a combination of different swap formulas than the classic method in the literature. The algorithms based on MSA are improved by the Reset step size method. Besides, the Smart method improves the algorithm based on the gap function and projection method. The new step size methods

TABLE 9 Best algorithms and settings with respect to the network size and loading

Network	Saturation level	Best algorithm	Initialization		Step size methods			Ranking for		Best compromise
			AoN	Keep	Initial	Reset	Smart	Quality	Speed	
5by5	US	MSAR		✓	✓	✓	✓	1	3	
		Prob.		✓				1	2	
		PI		✓	✓			1	1	✓
	S	MSA		✓		✓		2	1	✓
		Prob.		✓				1	3	
	OS	MSA		✓				4	1	✓
		GBN		✓			✓	1	4	
		BGB		✓		✓		1	6	
Ring city	US	MSAR		✓		✓		1	2	
		Prob.		✓				1	3	
		GBP		✓		✓		1	1	✓
	S	Prob.		✓				1	1	✓
	OS	MSAR		✓		✓		2	1	✓
		Prob.		✓				1	5	
Lyon 6V	US	Prob.		✓				1	2	
		SSP		✓		✓		1	1	✓
	S	Prob.		✓				1	4	
		SSP		✓			✓	2	1	✓
	OS	Prob.		✓				9	1	✓
		IMSA		✓			✓	1	4	

speed up the convergence of all algorithms, especially in the large-scale (Table 8).

For future work, the authors are looking for meta-heuristic algorithms for DTA problem in order to speed up the optimization process with parallel computation. In addition, designing the framework to predetermine the computation budget based on network size, topology, and saturation level is an interesting topic.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees for their thoughtful comments, which helped improve both the exposition and technical quality of the paper.

This project is supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (Grant agreement No 646592 - MAGnUM project).

References

Allen, T. T. (2011). *Introduction to discrete event simulation and agent-based modeling: Voting systems, health care, military, and manufacturing*. New York: Springer.

Ameli, M., Lebacque, J.-P., & Leclercq, L. (2017). Multi-attribute, multi-class, trip-based, multi-modal traffic network equilibrium model. In *Traffic and granular flow'17* (pp. 455–464). Springer.

Ameli, M., Lebacque, J.-P., & Leclercq, L. (2018). Day-to-day multimodal dynamic traffic assignment: Impacts of the learning process in case of non-unique solutions. *7th International Symposium on Dynamic Traffic Assignment (DTA)*.

Ballen, T. T. (2011). *Introduction to discrete event simulation and agent-based modeling: Voting systems, health care, military, and manufacturing* (2nd ed.). New York: Springer.

Beckmann, M., McGuire, C. B., & Winsten, C. B. (1956). *Studies in the economics of transportation* (Tech. Rep.).

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.

Blanchard, G., & Loubere, R. (2015). *High-order conservative remapping with a posteriori MOOD stabilization on polygonal meshes.* Retrieved from: `http://www.emn.fr/z-info/choco-solver/`.

Bliemer, M. C., Raadsen, M. P., Brederode, L. J., Bell, M. G., Wismans, L. J., & Smith, M. J. (2017). Genetics of traffic assignment models for strategic transport planning. *Transport reviews*, *37*(1), 56–78.

Boggs, R., Bozman, J., & Perry, R. (2002). *Reducing downtime and business loss: Adressing business risk with effective technology* (Tech. Rep. No. Technical report 91-18). Framingham, MA: International Data Corporation (IDC).

Braess, D., Nagurney, A., & Wakolbinger, T. (2005). On a paradox of traffic planning. *Transportation science*, *39*(4), 446–450.

Chevallier, E., & Leclercq, L. (2009a). Do microscopic merging models reproduce the observed priority sharing ratio in congestion? *Transportation Research Part C: Emerging Technologies*, *17*(3), 328–336.

Chevallier, E., & Leclercq, L. (2009b). Microscopic dual-regime model for single-lane roundabouts. *Journal of Transportation Engineering*, *135*(6), 386–394.

Di, X., He, X., Guo, X., & Liu, H. X. (2014). Braess paradox under the boundedly rational user equilibria. *Transportation Research Part B: Methodological*, *67*, 86–108.

Di, X., & Liu, H. X. (2016). Boundedly rational route choice behavior: A review of models and methodologies. *Transportation Research Part B: Methodological*, *85*, 142–179.

Dial, R. B. (2006). A path-based user-equilibrium traffic assignment algorithm that obviates path storage and enumeration. *Transportation Research Part B: Methodological*, *40*(10), 917–936.

Ehrgott, M., Wang, J. Y., & Watling, D. P. (2015). On multi-objective stochastic user equilibrium. *Transportation Research Part B: Methodological*, *81*, 704–717.

Elbaum, S., Malishevsky, A. G., & Rothermel, G. (2002). Test case prioritization: A family of empirical studies. *IEEE Transactions on Software Engineering*, *28*(2), 159–182.

Friesz, T. L., & Han, K. (2019). The mathematical foundations of dynamic user equilibrium. *Transportation research part B: methodological*, *126*, 309–328.

Friesz, T. L., Kim, T., Kwon, C., & Rigdon, M. A. (2011). Approximate network loading and dual-time-scale dynamic user equilibrium. *Transportation Research Part B: Methodological*, *45*(1), 176–207.

Gentile, G. (2009). Linear user cost equilibrium: a new algorithm for traffic assignment. *submitted to Transportation Research B*, *101*.

Halat, H., Zockaie, A., Mahmassani, H. S., Xu, X., & Verbas, O. (2016). Dynamic network equilibrium for daily activity-trip chains of heterogeneous travelers: application to large-scale networks. *Transportation*, *43*(6), 1041–1059.

Halpern, B. (1967). Fixed points of nonexpanding maps. *Bulletin of the American Mathematical Society*, *73*(6), 957–961.

Han, K., Eve, G., & Friesz, T. (2018). Computing dynamic user equilibria on large-scale networks: From theory to software implementation. *arXiv preprint arXiv:1810.00777*.

Han, K., Friesz, T. L., Szeto, W., & Liu, H. (2015). Elastic demand dynamic network user equilibrium: Formulation, existence and computation. *Transportation Research Part B: Methodological*, *81*, 183–209.

Han, K., Szeto, W., & Friesz, T. L. (2015). Formulation, existence, and computation of boundedly rational dynamic user equilibrium with fixed or endogenous user tolerance. *Transportation Research Part B: Methodological*, *79*, 16–49.

Han, L., Ukkusuri, S., & Doan, K. (2011). Complementarity formulations for the cell transmission model based dynamic user equilibrium with departure time choice, elastic demand and user heterogeneity. *Transportation Research Part B: Methodological*, *45*(10), 1749–1767.

Huang, H.-J., & Lam, W. H. (2002). Modeling and solving the dynamic user equilibrium route and departure time choice problem in network with queues. *Transportation Research Part B: Methodological*, *36*(3), 253–273.

Iryo, T. (2015). Investigating factors for existence of multiple equilibria in dynamic traffic network. *Networks and Spatial Economics*, *15*(3), 599–616.

Iryo, T., & Smith, M. J. (2017). On the uniqueness of equilibrated dynamic traffic flow patterns in unidirectional networks. *Transportation research procedia*, *23*, 283–302.

Jafari, E., Pandey, V., & Boyles, S. D. (2017). A decomposition approach to the static traffic assignment problem. *Transportation Research Part B: Methodological*, *105*, 270–296.

Janson, B. N. (1991). Dynamic traffic assignment for urban road networks. *Transportation Research Part B: Methodological*, *25*(2-3), 143–161.

Jayakrishnan, R., & Rindt, C. R. (1999). Distributed computing and simulation in a traffic research test bed. *Computer-Aided Civil and Infrastructure Engineering*, *14*(6), 429–443.

Jayakrishnan, R., Tsai, W. T., Prashker, J. N., & Rajadhyaksha, S. (1994). *A faster path-based algorithm for traffic assignment* (Tech. Rep.).

Jeihani, M. (2007). A review of dynamic traffic assignment computer packages. In *Journal of the transportation research forum* (Vol. 46, pp. 34–46).

Jiang, N., & Xie, C. (2014). Computing and analyzing mixed equilibrium network flows with gasoline and electric vehicles. *Computer-Aided Civil and Infrastructure Engineering*, *29*(8), 626–641.

Kaufman, D. E., Smith, R. L., & Wunderlich, K. E. (1998). User-equilibrium properties of fixed points in dynamic traffic assignment. *Transportation Research Part C: Emerging Technologies*, *6*(1-2), 1–16.

Krug, J., Burianne, A., & Leclercq, L. (2017). *Reconstituting demand patterns of the city of lyon by using multiple gis data sources* (Tech. Rep.). University of Lyon, ENTPE, LICIT.

Kuwahara, M., & Akamatsu, T. (2001). Dynamic user optimal assignment with physical queues for a many-to-many od pattern. *Transportation Research Part B: Methodological*, *35*(5), 461–479.

Laval, J. A., & Leclercq, L. (2008). Microscopic modeling of the relaxation phenomenon using a macroscopic lane-changing model. *Transportation Research Part B: Methodological*, *42*(6), 511–522.

Leclercq, L. (2007a). Bounded acceleration close to fixed and moving bottlenecks. *Transportation Research Part B: Methodological*, *41*(3), 309–319.

Leclercq, L. (2007b). Hybrid approaches to the solutions of the âĂIJlighthill–whitham–richardsâĂİ model. *Transportation Research Part B: Methodological*, *41*(7), 701–709.

Leclercq, L., & Laval, J. A. (2009). A multiclass car-following rule based on the lwr model. In *Traffic and granular flowâĂŹ07* (pp. 151–160). Springer.

Leclercq, L., Laval, J. A., & Chevallier, E. (2007). The lagrangian coordinates and what it means for first order traffic flow models. In *Transportation and traffic theory 2007. papers selected for presentation at isttt17.*

Leclercq, L., Verchier, A., Krug, J., & Menendez, M. (2016). Investigating the performances of the method of successive averages for determining dynamic user equilibrium and system optimum in manhattan networks. In *Dta2016, 6th international symposium on dynamic traffic assignment* (pp. 1–p).

Levin, M. W., Pool, M., Owens, T., Juri, N. R., & Waller, S. T. (2015). Improving the convergence of simulation-based dynamic traffic assignment methodologies. *Networks and Spatial Economics*, *15*(3), 655–676.

Lin, D.-Y., Valsaraj, V., & Waller, S. T. (2011). A dantzig-wolfe decomposition-based heuristic for off-line capacity calibration of dynamic traffic assignment. *Computer-Aided Civil and Infrastructure Engineering*, *26*(1), 1–15.

Lu, C.-C., Mahmassani, H. S., & Zhou, X. (2009). Equivalent gap function-based reformulation and solution algorithm for the dynamic user equilibrium problem. *Transportation Research Part B: Methodological*, *43*(3), 345–364.

Mahmassani, H. S. (2001). Dynamic network traffic assignment and simulation methodology for advanced system management applications. *Networks and spatial economics*, *1*(3), 267–292.

Marcotte, P. (1985). A new algorithm for solving variational inequalities with application to the traffic assignment problem. *Mathematical Programming*, *33*(3), 339–351.

Mehrabipour, M., Hajibabai, L., & Hajbabaie, A. (2019). A decomposition scheme for parallelization of system optimal dynamic traffic assignment on urban networks with multiple origins and destinations. *Computer-Aided Civil and Infrastructure Engineering*.

Miaou, S.-P., Summers, M. S., & Lieu, H. C. (1999). Laboratory evaluation of real-time dynamic traffic assignment systems. *Computer-Aided Civil and Infrastructure Engineering*, *14*(4), 281–298.

Mounce, R., & Carey, M. (2011). Route swapping in dynamic traffic networks. *Transportation Research Part B: Methodological*, *45*(1), 102–111.

Mounce, R., & Carey, M. (2014). On the convergence of the method of successive averages for calculating equilibrium in traffic networks. *Transportation science*, *49*(3), 535–542.

Nagel, K., & Flötteröd, G. (2012). Agent-based traffic assignment: Going from trips to behavioural travelers. In *Travel behaviour research in an evolving world* (pp. 261–294).

Ng, M., & Waller, S. T. (2012). A dynamic route choice model considering uncertain capacities. *Computer-Aided Civil and Infrastructure Engineering*, *27*(4), 231–243.

Nie, Y. M. (2010). A class of bush-based algorithms for the traffic assignment problem. *Transportation Research Part B: Methodological*, *44*(1), 73–89.

Pas, E. I., & Principio, S. L. (1997). Braess' paradox: Some new insights. *Transportation Research Part B: Methodological*, *31*(3), 265–276.

Patriksson, M. (2015). *The traffic assignment problem: models and methods*. Courier Dover Publications.

Peeta, S., & Mahmassani, H. S. (1995). System optimal and user equilibrium time-dependent traffic assignment in congested networks. *Annals of Operations Research*, *60*(1), 81–113.

Ramadurai, G., & Ukkusuri, S. (2011). B-dynamic: An efficient algorithm for dynamic user equilibrium assignment in activity-travel networks 1. *Computer-Aided Civil and Infrastructure Engineering*, *26*(4), 254–269.

Robbins, H., & Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, 400–407.

Rothermel, G., Harrold, M. J., Hirt, C. W., Amsden, A. A., & Cook, J. L. (1998). A safe efficient regression test selection technique. *ACM Transactions on Software Engineering and Methodology*, *6*(2), 173–210.

Sancho, E. C., Ibáñez Marí, G., & Bugeda, J. B. (2015). Applying projection-based methods to the asymmetric traffic assignment problem. *Computer-Aided Civil and Infrastructure Engineering*, *30*(2), 103–119.

Sbayti, H., Lu, C.-C., & Mahmassani, H. (2007). Efficient implementation of method of successive averages in simulation-based dynamic traffic assignment models for large-scale network applications. *Transportation Research Record: Journal of the Transportation Research Board*(2029), 22–30.

Schulz, A., & Doblhammer, G. (2012). Aktueller und zukünftiger Krankenbestand von Demenz in Deutschland auf basis der outinedaten der AOK. (Current and future number of people suffering from dementia in Germany based on routine data from the AOK.). In C. Gnster, J. Klose, & N. Schmacke (Eds.), *Versorgungs-report* (pp. 161–175). Piscataway, NJ: IEEE Press.

Sheffi, Y. (1985). Urban transportation networks.

Smith, M. (1993). A new dynamic traffic model and the existence and calculation of dynamic user equilibria on congested capacity-constrained road networks. *Transportation Research Part B: Methodological*, *27*(1), 49–63.

Szeto, W., & Lo, H. K. (2005). Non-equilibrium dynamic traffic assignment. In *Transportation and traffic theory. flow, dynamics and human interaction. 16th international symposium on transportation and traffic theoryuniversity of maryland.*

Szeto, W., & Wong, S. (2012). Dynamic traffic assignment: model classifications and recent advances in travel choice principles. *Central European Journal of Engineering*, *2*(1), 1–18.

Szeto, W. Y., & Lo, H. K. (2004). A cell-based simultaneous route and departure time choice model with elastic demand. *Transportation Research Part B: Methodological*, *38*(7), 593–612.

Taale, H., & Pel, A. (2015). Better convergence for dynamic traffic assignment methods. *Transportation Research Procedia*, *10*, 197–206.

Verbas, O., Mahmassani, H. S., & Hyland, M. F. (2015). Dynamic assignment-simulation methodology for multimodal urban transit networks. *Transportation Research Record: Journal of the Transportation Research Board*(2498), 64–74.

Verbas, O., Mahmassani, H. S., & Hyland, M. F. (2016). Gap-based transit assignment algorithm with vehicle capacity constraints: Simulation-based implementation and large-scale application. *Transportation Research Part B: Methodological*, *93*, 1–16.

Wang, Y., Szeto, W., Han, K., & Friesz, T. L. (2018). Dynamic traffic assignment: A review of the methodological advances for environmentally sustainable road transportation applications. *Transportation Research Part B: Methodological*.

Wardrop, J. G. (1952). Road paper. some theoretical aspects of road traffic research. *Proceedings of the institution of civil engineers*, *1*(3), 325–362.

Wie, B.-W., Tobin, R. L., & Carey, M. (2002). The existence, uniqueness and computation of an arc-based dynamic network user equilibrium formulation. *Transportation Research Part B: Methodological*, *36*(10), 897–918.

Xie, J., Nie, Y., & Liu, X. (2018). A greedy path-based algorithm for traffic assignment. *Transportation Research Record*, 0361198118774236.

Xu, H.-K. (2002). Iterative algorithms for nonlinear operators. *Journal of the London Mathematical Society*, *66*(1), 240–256.

Yang, X., Ban, X. J., & Ma, R. (2017). Mixed equilibria with common constraints on transportation networks. *Networks and Spatial Economics*, *17*(2), 547–579.

Yoo, S., & Harman, M. (2007). Pareto efficient multi-objective test case selection. In *Proceedings of the International Conference on Software Testing and Analysis* (pp. 140–150). New York, NY: ACM.

Ypma, T. J. (1995). Historical development of the newton–raphson method. *SIAM review*, *37*(4), 531–551.

Yu, N., Ma, J., & Zhang, H. M. (2008). A polymorphic dynamic network loading model. *Computer-Aided Civil and Infrastructure Engineering*, *23*(2), 86–103.

Zhang, X., Liu, W., & Waller, S. T. (2019). A network traffic assignment model for autonomous vehicles with parking choices. *Computer-Aided Civil and Infrastructure Engineering*.