DOI: 10.1111/mice.12577

ORIGINAL ARTICLE

COMPUTER-AIDED CIVIL AND INFRASTRUCTURE ENGINEERING WILEY

Simulation-based dynamic traffic assignment: Meta-heuristic solution methods with parallel computing

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

[1]Univ. Gustave Eiffel, Paris, France

[2]Univ. Gustave Eiffel, ENTPE, Lyon, France

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

Funding information
European Research Council, Grant/Award Number: 646592

Abstract

The aim of this study is to solve the large-scale dynamic traffic assignment (DTA) model using a simulation-based framework, which is computationally a challenging problem. Many studies have been performed on developing an efficient algorithm to solve DTA. Most of the existing algorithms are based on path-swapping descent direction methods. From the computational standpoint, the main drawback of these methods is that they cannot be parallelized. This is because the existing algorithms need to know the results of the last iteration to determine the next best path flow for the next iteration. Thus, their performance depends on the single initial or intermediate solution, which means they exploit a solution that satisfies the equilibrium conditions more than explore the solution space for the optimal solution. More specifically, the goal of this study is to overcome the drawbacks of serial algorithms by using meta-heuristic algorithms known to be parallelizable and that have never been applied to the simulation-based DTA problem. This study proposes two new solution methods: a new extension of the simulated annealing and an adapted genetic algorithm. With parallel simulation, the algorithm runs more simulations in comparison with existing methods, but the algorithm explores the solution space better and therefore obtains better solutions in terms of closeness to the optimal solution and computation time compared to classical methods.

1 | INTRODUCTION

Simulation-based dynamic traffic assignment (DTA) is an effective tool for analyzing transportation systems for both operational and planning purposes (Schreiter, Wageningen-Kessels, Yuan, Van Lint, & Hoogendoorn, 2012). The aim of simulation-based DTA is to calculate the dynamic path flow distribution for all the origin–destination (OD) pairs in the traffic network depending on one equilibrium rule (Yang, Balakrishna, Morgan, & Slavin, 2017). The well-known equilibrium rule is user equilibrium (UE), wherein all users experience minimum travel time (Wardrop, 1952). The simulation-based DTA contains two procedures: (i) a simulation-based dynamic network loading (DNL) model to calculate experienced path travel times by considering the traffic dynamics for a given path flow pattern; and (ii) an algorithm for finding the UE solution (Marcotte & Nguyen, 1998). Here, we focus on the second procedure.

The mathematical foundations of the DTA problem are recalled in Friesz and Han (2019). Finding an equilibrium through simulation (when no closed-form analytical solution is available) typically involves a solution scheme that relies on an iterative procedure. Much research has shown that the DTA problem can be represented as a fixed-point problem (Y. Wang, Szeto, Han, & Friesz, 2018). To solve the fixed-point problem, a path-swapping descent direction method is used to

reassign a fraction of the users at each step (Sheffi, 1985). The reassignment process is monitored to check whether the solution is improved or not. In other words, the algorithm reassigns the share of the users who have chosen a nonoptimal path to a more efficient alternative at each iteration of the equilibrium calculation (Levin, Boyles, & Nezamuddin, 2014). The foundation of all iterative algorithms is based on starting from the initial solution and updating the path flow distribution for iteration i using the following formula (Friesz, 2010):

$$z^i = (1 - \beta_i)z^{i-1} + \beta_i f[z^{i-1}] \quad (1)$$

where z^i is the path flow distribution of the iteration i, $f[z^i]$ is the descent direction, and β_i is the step (descent) size of iteration i for the fixed-point algorithm. At each iteration, Equation 1 reassigns the users to move the traffic network toward the UE. Based on the review papers of Szeto and Lo (2006) and Y. Wang et al. (2018), almost all of the solution methods for DTA models used Equation 1 to find the network equilibrium. Many works can be found in the literature dedicated to finding the best β_i and $f[z^i]$ analytically or heuristically to improve the efficiency of the algorithms (see, e.g., Akamatsu, 2001; Alisoltani, Leclercq, Zargayouna, & Krug, 2019; Bar-Gera, 2002; Drissi-Kaïtouni & Hameda-Benchekroun, 1992; Dial, 2006; Galligari & Sciandrone, 2017; Gentile, 2016; Mehrabipour, Hajibabai, & Hajbabaie, 2019; Nguyen & Dupuis, 1984; Perederieieva, Ehrgott, Raith, & Wang, 2015; Seshadri & Srinivasan, 2017; Raadsen, Bliemer, & Bell, 2019; Xie, Nie, & Liu, 2018). Generally, first, we find the cheapest path (e.g., using Dijkstra's algorithm) at the beginning of each iteration according to the current travel times and shift a portion of the demand to the (newly) shortest path, which is called reassignment process. The optimal portion of demand to shift can be determined analytically when the cost function is link additive and strictly concave (Sancho, Ibáñez Marí, & Bugeda, 2015). Although in practice, reassignment is often based on heuristics. The reason to resort to heuristics is based on the fact that the simulator needs to know the path flow distribution to predict the travel time accurately whereas the DTA process requires this information to estimate the path flow distribution (Bekhor, Toledo, & Reznikova, 2009). However, it is not possible to guarantee that fixed point algorithms converge toward the optimal solution (Ben-Akiva, Gao, Wei, & Wen, 2012) and there is no exact method for determining the step size (β_i) (Levin, Pool, Owens, Juri, & Waller, 2014; Szeto & Lo, 2005). All the algorithms based on Equation 1 mainly have two drawbacks when equilibrium is sought for large-scale networks:

1. The calculation should be done sequentially: the algorithms need to know the last iteration results to determine the next best path flow for the next iteration. Therefore, all the steps are in series because we need information (travel time) from the last simulation run.
2. The reassignment decision is taken only at the OD level and independently at each step: the algorithms do not consider the effect of shared links between OD path sets in the reassignment process. Intersections between OD flows are only taken into account when running the simulation to derive travel time.

To highlight these drawbacks, let us consider the method of successive average (MSA) by Robbins and Monro (1951). The MSA algorithm is the classical method to solve the traffic assignment problem (Nagel & Flötteröd, 2016) and widely used in theory and application for DTA problems (Mounce & Carey, 2015). The $f[z^{i-1}]$ of the MSA algorithm in Equation 1 is extracted from the auxiliary path assignments obtained by the all-or-nothing procedure (y^i), that is, everyone is placed on the shortest path and $\beta_i = \frac{1}{i+1}$. Equation 2 presents the swapping formula of MSA algorithm.

$$z^i = \left(\frac{i}{i+1}\right)z^{i-1} + \frac{1}{i+1}y^i \quad (2)$$

Consequently, in iteration i, the MSA algorithm tries to improve the path flow distribution by swapping a fraction $\frac{1}{i+1}$ of users to the shortest path(s) from each nonshortest path. Then, one simulation is launched based on the updated path flow distribution. We have to wait until the simulation run is finished to know the new link travel times and adjust the path flow distribution to be tested in the next iteration accordingly. This is the serial process of the MSA algorithm, which limits the solution space exploration and computational process.

Moreover, the MSA algorithm performs the reassignment process for each OD independently without considering that some OD pairs are connected because they share certain links and nodes. For instance, if we have a set of shared links between two OD pairs that are heavily congested, the algorithm will reduce the flow of the paths containing these shared links for all the OD pairs whereas reducing only the flow of a few OD pairs would have been sufficient. Therefore, we may trigger a high compensation of heavily congested paths. The MSA algorithm not only makes no provision to take into account the correlations between the OD assignment and the travel time, but also there is no accurate definition for $f[z^{i-1}]$ to consider this effect (Flötteröd, 2018). In addition, the performance of all algorithms based on Equation 1 depends on the choice of the initial solution, which can slow down the algorithm or limits the exploration of the solution space. Particularly in large scale, there is a risk of being stuck at nonoptimal solutions. Note that there is no robust approach to choose the initial solution (Ameli, Lebacque, & Leclercq, 2017). Therefore, using stochastic algorithms, for example, meta-heuristics, can overcome this drawback.

All the works in the literature have the aforementioned limitations and perform the calculation in series. In this article, we explore a completely new area for overcoming the drawbacks of serial algorithms using meta-heuristic algorithms. Meta-heuristic algorithms are known to be parallelizable (Fonseca & Fleming, 1995). Traffic simulation, particularly micro-simulation, can be viewed as a complex system for which meta-heuristic algorithms are expected to be well-adapted because they are stochastic methods designed to search the solution space of complex and computationally costly problems (Yun & Park, 2006). We can better explore the solution space and will also run several simulations in parallel for certain path flow assumptions and take the decision on what the next exploration of the solution space should be. This overcomes not only the first drawback but also makes the algorithm capable of starting the optimization with different starting points at the same time. Note that the parallelization approach can also allow us to use distributed computation to improve the solver computation time (CT). Moreover, a new layer of optimization is added to the algorithm to take into account the correlations between OD pairs through shared links. Note that parallelization has been well investigated in other procedures of the problem, for example, traffic simulation (Barceló et al., 1998; Rickert & Nagel, 2001), calibration (Jiang, 2004; Lin, Valsaraj, & Waller, 2011), and shortest path calculation (Attanasi, Silvestri, Meschini, & Gentile, 2015; Idri, Oukarfi, Boulmakoul, & Zeitouni, 2017). However, in this study, we focus on applying and validating parallelization to UE path flow calculation process for large-scale simulation-based DTA problems.

The meta-heuristic algorithms are mainly applied to mathematical models that are very complex in nature and quite difficult to solve. They are usually employed in approaches where first-order derivatives are challenging to obtain, and the result may depend on the selection of an initial point (Stathopoulos & Tsekeris, 2004). This is certainly the case for large-scale traffic assignment problems, where each simulation to calculate the objective function(s) is costly. It means calculating first-order derivatives are costly as well. Therefore, with a large-scale simulation-based framework, we use the simulator as a black-box to calculate the objective function, making it necessary to run trials for optimization, for example, the MSA algorithm is a trial-and-error process with the descent method. To the best of our knowledge, no study in the literature has yet applied a meta-heuristic algorithm directly to find the UE for simulation-based DTA models. This may be because it is difficult to handle the variables which in this case are path flows.

Meta-heuristic algorithms can be classified into two categories: single solution and population based (Talbi, 2009). The single solution methods start with an initial solution and apply a process to improve the candidate solution to achieve the best solution by following a trajectory in the solutions space. The second class is population based; the purpose of the methods of this class is to improve a set of solutions (population) by applying a specific process. This study proposes two new solution methods based on two categories of meta-heuristic algorithms.

The meta-heuristic algorithms are different in searching approaches. For instance, in the single-solution category, some of them are inspired by nature (Siddique & Adeli, 2015b), for example, Water Drop Algorithm (Siddique & Adeli, 2014b) and Bacteria-Foraging Algorithm (J. Wang, Zhong, Adeli, Wang, & Liu, 2018), and some of them are inspired by physics (Siddique & Adeli, 2016b), for example, Spiral Dynamics Algorithm (Siddique & Adeli, 2014a) and Gravitational Search Algorithm (Siddique & Adeli, 2016a). Here, we employ the Simulated Annealing (SA) algorithm from the single-solution category inspired by physics. SA is well-known for its efficient application in different engineering fields and is often used when the solution space is discrete (Siddique & Adeli, 2016c). The major advantage of SA over other methods is the ability to avoid becoming trapped in local optima (Dekkers & Aarts, 1991). The SA algorithm uses a random search that accepts not only changes that improve the current solution but sometimes some may not to better explore the solution space (Busetti, 2003). In this study, a new extension of the SA method is designed and applied to the DTA problem.

In population-based algorithms category, there are different searching approaches, for example, Harmony search (Siddique & Adeli, 2015a) and genetic algorithm (GA) Holland (1992). Here, we choose GA because it is robust and dealt successfully with a wide range of difficult problems (Busetti, 2007). The main advantage of GA is the ability for global searching and exploration of the solution space. Also, it can combine with the local search method to increase exploitation. An adaptive GA is developed, in this study, to solve the trip-based network equilibrium problem.

Both meta-heuristic algorithms are generally developed to solve traffic assignments with parallel computation to consider more than one path flow distribution per iteration. It is possible that with a stochastic approach, more simulations must be run to explore the solution space compared to descent methods. However, with parallel simulation, the algorithm can counterbalance the number of simulation runs by finally reaching the optimum more quickly. Also, at the end of the optimization process, it is expected to achieve better solutions in terms of quality and closeness to the optimal solution because the algorithm explores the solution space more completely and more efficiently. Note that this study is focused on simulation-based DTA, and the design of meta-heuristic algorithms does not refer to analytical DTA models.

The next section, Section 2, presents a discussion on the mathematical conditions for UE solutions. It also presents the two indicators we use to assess algorithmic performance. The simulation-based framework and two meta-heuristic

algorithms are presented in Section 3. The simulator and the numerical experiments are presented in Section 4. The results obtained are discussed in Section 5. Finally, we present concluding remarks and introduce the future directions of the work in Section 6.

2 | PROBLEM STATEMENT

DNL is the combination of DTA with a traffic simulator that calculates network states and travel times (Yu, Ma, & Zhang, 2008). In other words, DTA models depend on a network performance module, which is called DNL. The DNL operator usually is not available in closed form because it severely complicates equilibrium calculation (Ngoduy, 2011; Song, Han, Wang, Friesz, & Del Castillo, 2017). Depending on the kind of simulator used to perform the network loading and determine travel times, the demand from origins to destinations can either be expressed as continuous flow or units of vehicles. The flow-based approach usually corresponds to dynamic macroscopic models, whereas the trip-based approach is widely implemented in microscopic models (Ramadurai & Ukkusuri, 2011). The latter approach is certainly more realistic for reproducing traffic flows but it is also more challenging when deriving UE because OD flow should always correspond to integer numbers during the convergence process (Jordan, Foytik, Collins, & Robinson, 2017). The trip-based approach is used in this study to address the real large-scale DTA problem. In this section, we present the conditions of dynamic UE for the DTA model and the indicators used to determine the proximity of solutions to UE in the simulation-based framework.

2.1 | Dynamic UE condition

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 for each OD pair. The period of interest (planning horizon) of duration H is discretized into a set of small time intervals indexed by τ. In an interval of τ, the traffic conditions are assumed constant for the DTA, that is, travel times are averaged at the path level over each time interval. Therefore, all the time-dependent variables of the model are indexed by τ. To simplify the equations, we present the model for each departure time interval of τ. In Table 1, we introduce the notations of all the symbols and variables used in this article.

According to the definition, we have:

$$\hat{TT}_p = \frac{\sum_{tr \in Tr_p} TT_{tr,p}}{\pi_p}; \quad \forall p \in P_w \tag{3}$$

$$\hat{TT}^*_w = \frac{\sum_{tr \in Tr_{p^*}} TT_{tr,p^*}}{\pi_{p^*}}; \quad \forall p^* \in P^*_w \tag{4}$$

TABLE 1 Nomenclature used in this article

W	OD pairs, subset of origin × destination nodes, $W \subset N \times N$
P_w	Set of paths for w
P^*_w	Set of shortest (i.e., minimum travel time) paths for w
w	Index of origin–destination (OD) pair, $w \in W$
D_w	Total demand for w pair
Tr_w	List of trips that travel for w
Tr_p	List of trips that travel for w on path p, $Tr_p \subset Tr_w$
p	Index of path, $p \in P_w$
p^*	Index of shortest path, $p^* \in P^*_w$
tr	Index of trip, $tr \in Tr_w$
π_w	Cardinality of a set Tr_w: number of users traveling for w
π_p	Cardinality of a set Tr_p: number of users on path p
$TT_{tr,p}$	Experienced travel time of trip tr on path p
TT^*_w	Minimum experienced travel time for w
$\hat{TT}_p$	Mean travel time of trips on path p
$\hat{TT}^*_w$	Mean travel time of trips on shortest path(s) of OD pair w

The network UE conditions with predefined travel demand and the users' departure times are (Peeta & Ziliaskopoulos, 2001):

$$\begin{cases} \hat{TT}_p - \hat{TT}^*_w \geq 0; & \forall w \in W,\ p \in P_w \\ \pi_p \left(\hat{TT}_p - \hat{TT}^*_w \right) = 0; & \forall w \in W,\ p \in P_w \\ \pi_p \geq 0; & \forall p \in P_w \end{cases} \tag{5}$$

Lu, Mahmassani, and Zhou (2009) reformulated the problem as a nonlinear problem to minimize the gap function. The gap function is defined as the gap between the average path travel time and the average shortest path travel time. Therefore, the solution to this fixed-point problem is equivalent to finding the solution to the following variational inequality:

$$\sum_{w \in W} \sum_{p \in P_w} TT_{w,p}{}^* \left[\pi_w - \pi_{p^*} \right] \geq 0 \tag{6}$$

where π_{p^*} is the optimal number of trips on path p and $\pi_w, \pi_{p^*} \in \mathcal{H}$ satisfy the equilibrium. $\mathcal{H}$ denotes the flow constraints based on D_w.

As mentioned before, finding the optimal solution for a large-scale DTA problem is hard to achieve, so indicators are required to measure the distance between the solutions and the optimal UE.

2.2 | Convergence quality

According to Equation 5, the UE situation is equivalent to the network state where there is no delay for travelers when compared to the minimum possible travel time for each OD pair. Using this characterization, we introduce a first quality

indicator for solutions, the average delay of each user (Janson, 1991):

$$AGap = \frac{\sum_{w \in W} \sum_{p \in P(w)} \sum_{tr \in Tr_p} \left(TT_{tr,p} - TT^*_{tr,w} \right)}{\sum_{w \in W} \pi_w} \quad (7)$$

The *AGap* is zero for the perfect UE path flow distribution, so any optimization algorithm should minimize *AGap*. Recall that trip-based DNL attempts to assign particle-discretized time-dependent origin/destination flows in a dynamic network equilibrium framework (Jayakrishnan & Rindt, 1999). The absolute UE situation means all trips perceive no delay, which is almost impossible to be reached in the real world. Therefore, the UE solution in the trip-based approach means each traveler perceives minimum possible delay with respect to the network dynamics and demand profile (Sbayti, Lu, & Mahmassani, 2007). By this definition, the convergence means the process of minimizing users delay.

A second indicator is Violation. The Violation indicator is calculated based on the definition of "user in violation" and "OD in violation." If the gap between user experienced travel time and shortest path travel time is bigger than the proportion ϵ of the shortest path travel time, the user is in violation. Then, the OD violation for an OD w is defined when there is more than a ratio ϵ' of the users on w in violation. Finally, the Violation indicator is the share of ODs, which are in violation. The perfect UE means *Violation*= 0 with $\epsilon = \epsilon' = 0$. In this study, the values of ϵ and ϵ' are fixed at 10% based on the study of Ameli, Lebacque, and Leclercq (2020) to evaluate the quality of the solution from a perspective different from *AGap*. Please note that the Violation indicator is not used in the convergence test. We calculate it afterward to assess the performance of algorithms.

3 | METHODOLOGY

Determining the UE path flow distribution requires two main steps: (i) identifying the time-dependent feasible paths between ODs and (ii) finding the optimal path flow with respect to demand and network dynamics. The first step refers to solving a time-dependent shortest path algorithm, which is a computationally expensive process in a large-scale network (Srinivasan, Riazi, Norris, Das, & Bhowmick, 2018). This study aims to address the second step. Therefore, we choose a framework from the literature which expresses the solution algorithm in two loops: the outer loop to find the shortest path and the inner loop to find the optimal path flow distribution (Lu et al., 2009). The main advantage of this framework is that it attempts to find the UE path flow distribution with a minimum number of running a time-dependent shortest path algorithm. The authors performed a comprehensive study on

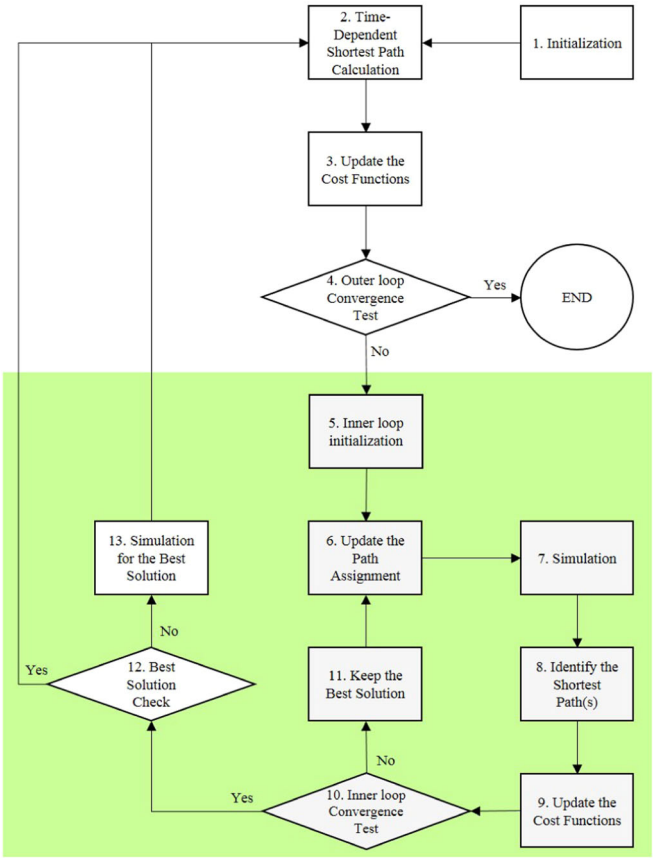


FIGURE 1 Solution algorithm for trip-based dynamic network equilibrium

this framework and proposed an extension to improve the two loops framework (Ameli et al., 2020). Figure 1 presents the optimization framework of this study. Please note that shortest path means a path with minimum travel time. Here, we focus on the inner loop, wherein the reassignment process is embedded. The green box in Figure 1 presents the classical inner loop structure. For the details of outer loop steps (Steps 1 to 4), readers can refer to Ameli et al. (2020).

The inner loop process starts with a single initial solution, which is generated in Step 5. Note that in each outer loop iteration, a fixed path set is provided by the outer loop for each OD pair that makes the subproblem for inner loop iterations. The optimization algorithm updates the path assignment based on the current state of the network (Step 6). The reassignment process is executed based on the swapping algorithm (Equation 1 in the classic approach), and a new path flow pattern is generated to be sent to the simulator to calculate the experienced travel time. Then, one simulation runs in Step 7, and the shortest path or paths is/are identified from the fixed path set, which comes from the outer loop, based on the simulation results in Step 8. The solution quality indicators are calculated in Step 9. Step 10 checks the convergence. The inner loop converges if the *AGap* is unchanged or if the maximum number of iterations is reached. If the process has not converged, Step

11 keeps track of the best solution obtained by current inner loop iterations; otherwise, the final solution is checked by Step 12 to ensure that it is the best solution based on $AGap$. If the last iteration solution is the best, the inner loop is finished; otherwise, we run one more simulation for the best solution from Step 11. As mentioned before, the optimization process is executed in series. The meta-heuristic algorithms redesign the inner loop structure.

Before presenting the meta-heuristic algorithms, we present the three most common swapping algorithms in the literature for a large-scale DTA problem. They are considered in this study as benchmarks for demonstrating the efficiency of the new meta-heuristic algorithms. The swapping algorithm is embedded in Step 6. The first algorithm is MSA (introduced in Section 1 and Equation 2), which is the most common algorithm used in the literature (Foytik, Jordan, & Robinson, 2017). The second method is an extension of the MSA algorithm by Sbayti et al. (2007), called MSA ranking (MSAR). The MSAR algorithm ranks the users by the experienced travel time then swaps a quantity $\frac{1}{i+1}D_w$ of travelers with the longest experienced travel time to the shortest(s) path. The design of the third method is based on the expected travel time reduction. This algorithm is called gap-based algorithm in this study. Lu et al. (2009) showed numerically that the gap-based algorithm obtains a better solution for UE than the MSA algorithm for the large-scale network. The gap-based algorithm uses the same formula as Equation 1 and same $f[z^{i-1}]$ as MSA algorithm with $\beta_i = \rho^i \frac{\hat{TT}_p - \hat{TT}^*_w}{\hat{TT}_p}$, where ρ^i is the step size of the gap-based algorithm.

$$z^i = \left(1 - \rho^i \frac{\hat{TT}_p - \hat{TT}^*_w}{\hat{TT}_p}\right) z^{i-1} + \left(\rho^i \frac{\hat{TT}_p - \hat{TT}^*_w}{\hat{TT}_p}\right) y^i \quad (8)$$

$$\rho^i = \begin{cases} \frac{i}{i+1}; & \text{if} \quad i = 0 \\ 1; & \text{o.w.} \end{cases} \quad (9)$$

In addition, the gap-based algorithm uses the Bernoulli trial for each user based on β_i to decide to swap or not according to the result of the trial (Verbas, Mahmassani, & Hyland, 2015). We will now present the implementation of the meta-heuristic algorithms.

3.1 | SA method

The SA algorithm is inspired by annealing in metallurgy. The basic SA algorithm is presented in Kirkpatrick, Gelatt, and Vecchi (1983). This study redesigns and adapts the classic SA to a simulation-based traffic assignment. Figure 2 presents the SA algorithm of this study.

The algorithm starts with an initial solution generated randomly, that is, users choose their path from the OD path set randomly. For solution S, the total gap $TGap(s)$ between the users' travel time and the shortest path travel time (Equation 10) is considered as the energy of the solution. A set of neighbor solution is generated with respect to the current one based on the temperature (T) of the current iteration. A neighbor solution is defined as a candidate to replace the current solution. The current phase of the iteration depends on the temperature of the process. Inspired by the physics of matter, this study distinguishes three different methods to generate a neighbor solution, gas, liquid, and solid; these methods represent the states of matter in nature. When the temperature is high ($T > \alpha$ where α denotes the boiling temperature), the *gas method* is applied. When running the SA algorithm, by decreasing the temperature, the algorithm enters the liquid phase ($\alpha > T > \alpha'$ where α' denotes the melting temperature) and then the *liquid method* is applied. When the temperature is quite low ($T < \alpha'$), the *solid method* is applied.

$$TGap(s) = \sum_{w \in W} \sum_{p \in P_w} \sum_{tr \in Tr_p} (TT_{tr,p} - TT^*_{tr,w}) \quad (10)$$

In the gas phase, we explore the solution space without limitation of any step size (β_i). Therefore, the candidates for the neighbor solution correspond to a random path flow distribution with respect to the demand value for each OD pair (feasible OD-assignment). In other words, all users choose randomly their path from the path set (determined by outer loop). The *gas method* generates one solution as a neighbor. In the liquid phase, we target the exploration of the solutions space randomly and also apply step size methods. First, we apply a randomizing process on the current solution and obtain the first neighbor solution. Then, we optimize it by applying the MSA to obtain the second solution and the gap-based method to obtain the third solution. The *liquid method* generates three candidates. In the solid phase, we execute the same process as in the liquid phase but without randomization. This means the two solutions are generated based on the current solution (Figure 2).

Afterwards, the algorithm runs parallel simulations (a maximum of three simulations) to update the network based on new different path flows obtained from the previous step. For a new solution or solutions s', the total gap $TGap(s')$ is calculated and corresponds to the energy of the solution (E) compared to the current solution s.

The last step consists in making a decision on accepting one of the best new solutions based on $TGap$ compared to the current solution of the algorithm. The acceptance decision is made by the Bernoulli trial.

$$AP_s = P\left(S'_{\text{accepted}} = 1\right) = e^{\frac{-\nabla E}{T}} \quad (11)$$

where AP_s denotes the probability of accepting solution s', S'_{accepted} denotes the binary decision variable and $\nabla E = TGap(s') - TGap(s)$. If the solution s' is accepted, the

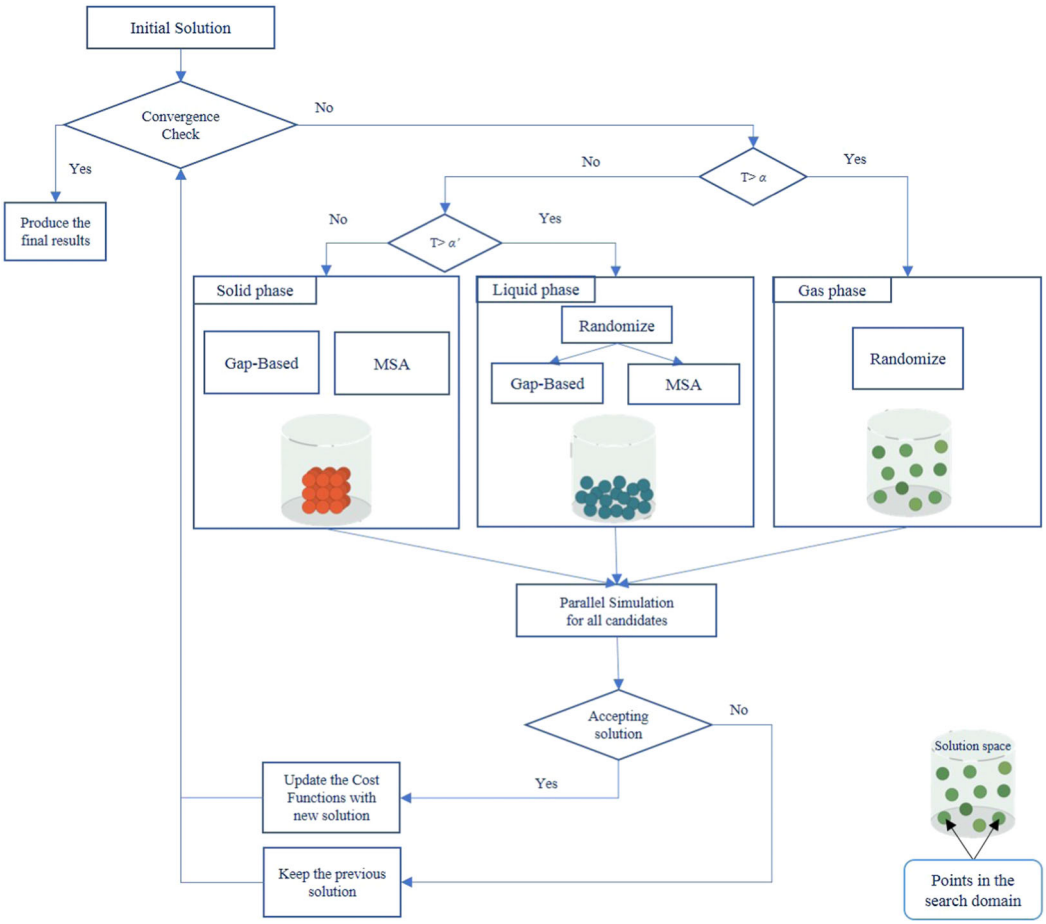


FIGURE 2 SA solution algorithm flowchart

current solution s is replaced by s'. At the end of each iteration, the temperature is decreased by the following formula:

$$T = \frac{T_0}{ln(i+1)} \tag{12}$$

where T_0 denotes the initial temperature. It should be remembered that i denotes the iteration index. In this study, the starting temperature is set to $T_0 = 3000$ based on the recent study of Franzin and Stützle (2019) about different components of the SA algorithm. The quality of the solution is evaluated in the convergence check step, which is similar to Step 10 in Figure 1.

The SA algorithm considers more than one solution per iteration in the liquid and solid phases. In addition, the exploration and exploitation degrees are changed based on T. The algorithm explores more in the gas phase and liquid phase when the temperature is high and exploits more in the solid phase when the T is low.

3.2 | GA

The GA was first proposed by Holland (1992); it is inspired by natural genetic variation and natural selection; selection, crossover, and mutation are the main operators of this approach. In the GA, we use the terms chromosomes and genes to refer to the different segments of an individual. In our implementation for the GA, we consider our solution and the $TGap$ as an individual (DNA) with a fitness value, our ODs assignment as chromosomes, and the path flows as the genes. Figure 3 illustrates the solution structure for GA. A gene is identified by a path (p) and contains the path flow value (π_p). A chromosome is the OD assignment, identified by w, which contains the genes of all the corresponding paths ($p \in P_w$).

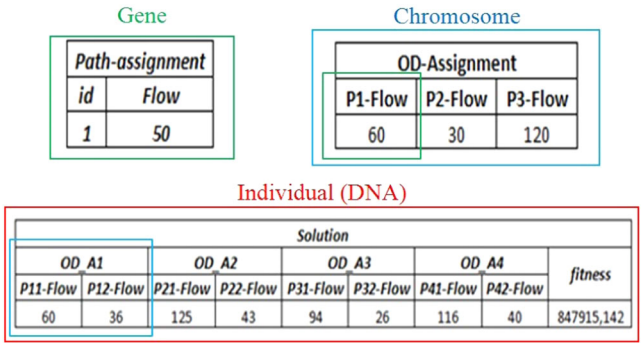


FIGURE 3 Solution structure in the GA case

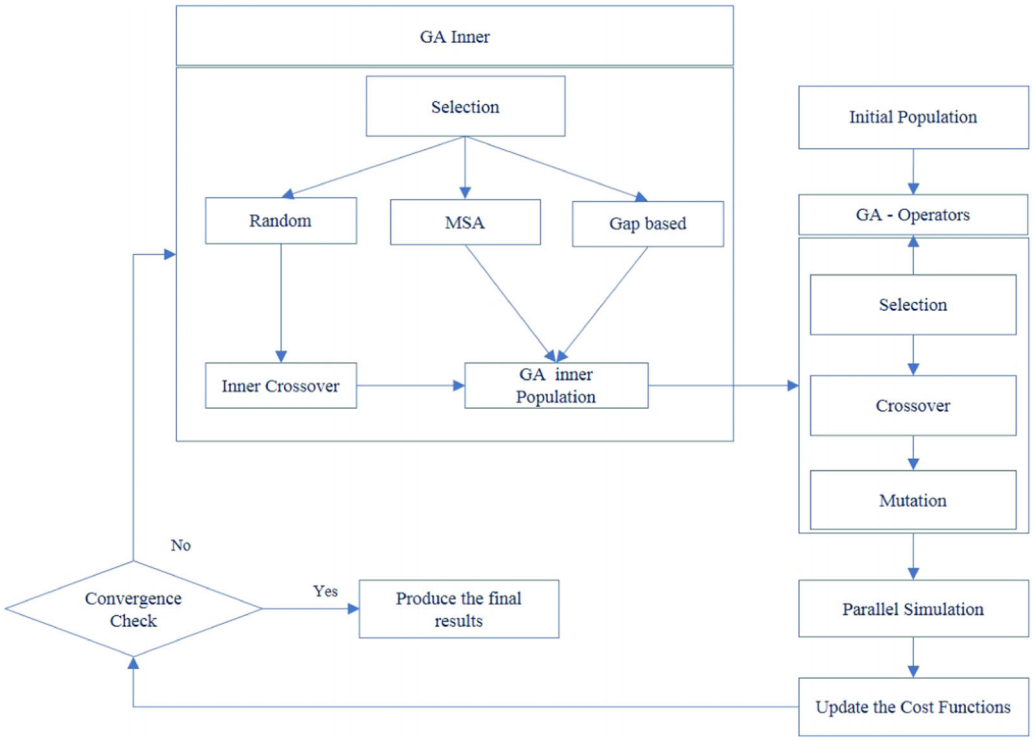


FIGURE 4 GA solution algorithm flowchart

A DNA is the full set of chromosomes that constitutes one individual solution. Finally, the set of individuals constitutes a population. Based on this definition, we can apply the GA operators to the DTA solution.

Figure 4 presents the application of GA to the DTA problem. The GA process starts by generating the initial population (initial set of individuals). In this study, we generate a random population. To avoid the GA algorithm from getting trapped in local optimum, this study designs a two-layered GA process to search the solution space by changing the path flows in the inner GA and take into account the correlations between OD pairs by considering a different combination of OD assignment in GA operators. In other words, the classical fixed-point algorithms plus a random method is applied in the inner GA, and the GA operators in one upper level generate different combinations of OD assignments to improve the population. All new solutions generated by GA process are evaluated by parallel simulations and added to the population to improve the population in each iteration. The steps of GA applied to traffic assignment are as follows:

- Selection: we use a random selection based on the crossover rate (Cr) and population size (PS) to compute the number of selected solutions for the crossover process:

$$SS = PS \times Cr \tag{13}$$

- Crossover: we apply a nonuniform crossover by using a bit-vector mask method (Maini, Mehrotra, Mohan, & Ranka, 1994). We select two different solutions (parents) from the set of selected solutions; we apply the crossover between each pair of solutions. As a result, we will have new solutions; the two new solutions will have a part of each parent.
- Mutation: we apply the mutation operator for a set of selected solutions; by replacing one OD assignment (chromosome) of the solution by another chromosome from another solution, this operator aims to increase the quality of the worst solution. The possibility of the mutation for one chromosome is calculated based on the quality of the chromosome:

$$MP_w = \frac{TGap_w(s)}{TGap(s)} \tag{14}$$

where $TGap_w(s)$ denotes the total gap of the OD assignment w in solution s.

- Parallel simulation: all the new solutions obtained from the previous steps are simulated in parallel to calculate the fitness function, which is the $TGap$ in this study.
- Replacement: after applying the different GA operators and parallel simulation, the size of the evaluated population set is increased. To keep it as a fixed value (PS), we apply the selection operator, and we keep the best solutions as a replacement strategy.
- Convergence check: the algorithm converges when the maximum number of iterations is reached, or when the

algorithm tends to stagnation. To check the stagnation, we use a "stagnation factor (SF)" indicator: when SF tends to zero, our process tends to stagnation, which means the quality of the population solution has not changed. The stagnation factor is presented as follows:

$$SF = 1 - \frac{TGap^{i}_{\text{Max}}}{TGap^{(i-1)}_{\text{Max}}} \tag{15}$$

where $TGap^{i}_{\text{Max}}$ denotes the total gap of the worst quality DNA (high $TGap$) in the population of iteration i. If the algorithm does not converge, the iteration index is increased, and the inner layer of GA is applied to search the solution space by gene modifications inside the chromosomes of the selected solutions.

- GA Inner: as mentioned before, the initial population, selection, crossover, mutation, and replacement are the basics of GA. In this study, we extend the GA by adding a new operator, called "GA Inner." The purpose of this operator is to create diversity in the current population (diversity inside the OD assignment). This approach is applied using three different methods: the MSA, the gap-based methods, and the adaptive random method.

  – Adaptive random method: the foundation of this method is based on the GA. We consider the selection and crossover operators for this method.

    (a) *Selection*: select a set of solutions from the main population. The worst solutions that have a large $TGap$ are selected because the aim of this selection is to increase the quality of the population by improving the solutions. The number of selected solutions are determined by Equation 13.

    (b) *Inner Crossover*: We consider the OD assignment (Chromosome) as an individual and the flow of each path (Gen) as a chromosome to apply the crossover in the same way as in the previous layer of the optimizer. By applying the crossover, we risk having a nonfeasible OD assignment with respect to the demand constraint.To solve this problem, we use the following process to keep only the feasible solutions:

    Step 1: put zero for the flow of the worst path (wp), which has the maximum travel time.

    Step 2: apply the crossover on the other paths of the current OD.

    Step 3: compute R, the difference between total flow of the current chromosome and the demand level of OD pair w:

$$R = \sum_{i \neq wp} \pi_i - D_w \tag{16}$$

If $R \leq 0$ then we put the rest of the flow on the worst path ($X_{wo} = R$); otherwise we reject this chromosome because it makes the generated solution infeasible.

  – The MSA assignment method and gap-based method (introduced in Section 3) are also applied to the chromosomes of the selected solutions in GA inner.

- The new solutions are injected into the main population and the algorithm iterates while GA converges.

Note that based on the early study of Srinivas and Patnaik (1994), we set the crossover rate set to $Cr = 0.5$, and the mutation rate is fixed to 0.4. The PS is set to 10 individuals in this study. Note that in both algorithms, the feasibility of the new path flow distribution is guaranteed, and each solution is evaluated by simulation in parallel process independently.

4 | NUMERICAL EXPERIMENTS

In this section, we present the dynamic simulator and the case study, which corresponds to a large-scale traffic network of this study.

4.1 | Dynamic simulator

In this work, we use Symuvia (Symuvia is an open source simulator: https://github.com/Ifsttar/Open-SymuVia) as a trip-based simulator for calculating the necessary variables, in particular, the experienced travel times of all travelers. Symuvia is a microscopic simulator based on the Lagrangian resolution of the LWR (Lighthill Whitham Richards) model (Leclercq, Chevallier, & Laval, 2008), which is the conservation law with respect to traffic density. We set the simulation time-step to 1 second and collect the link traffic information (travel times) every 1 minute. The users' routes are determined by the DTA model and the rolling horizon technique (Mahmassani, 2001) which determine the path flow distribution based on a prediction period of 20 minutes and an assignment period of 15 minutes (Mahmassani, 1998).

4.2 | Case study

For the large-scale test case, the real network of Lyon 6e + Villeurbanne (Figure 5) is considered. This network has 1,883 nodes, 3,383 links, 94 origins, 227 destinations.

The network is loaded with 47,341 travelers of all ODs with given departure times to represent the two morning peak hours of the network between 7:30 to 9:30. The demand profile comes from the study of Krug, Burianne, and Leclercq (2019). Figure 6a presents the demand profile of the numerical experiment. To show a quick and synthetic

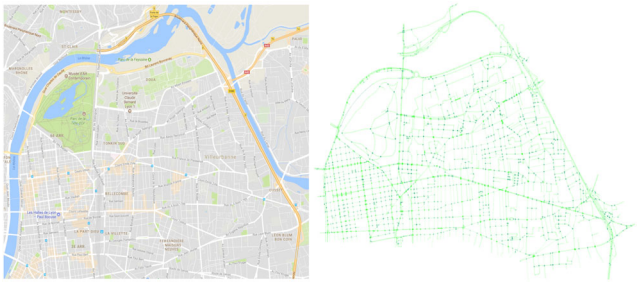
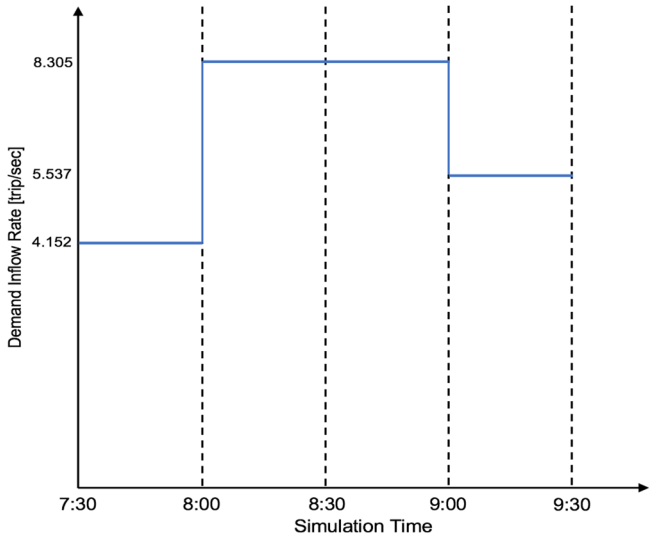
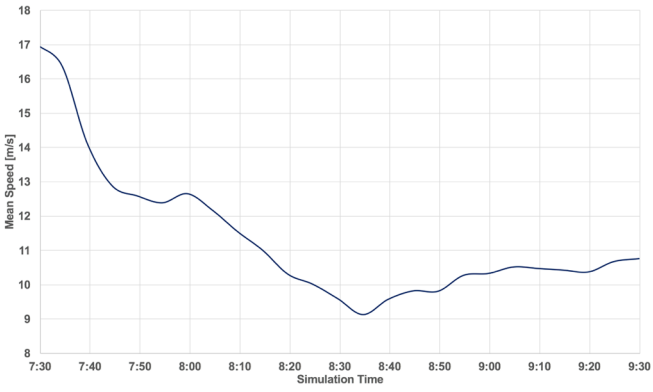


FIGURE 5 Lyon 6e + Villeurbanne: Mapping data ©Google 2019 and the traffic network used by Symuvia



(a) The demand profile



(b) The evolution of the mean speed

FIGURE 6 The demand scenario of Lyon 6e + Villeurbanne

overview of the network state, we plot time and evolution of the mean speed over the full network in Figure 6b. First, the mean speed curve decreases from 16.9 m/s and the traffic states remain undersaturated when demand is light, in this case from 7:30 to 8:10. Afterwards, travel production, which is equivalent to the total travel distance for a given period of time, stabilizes whereas the accumulation (or total travel time) continues to increase. Therefore, the decrease of the mean speed slowdown. This corresponds to the saturation level occurring from 8:10 to 8:35. The increase in mean speed shows that the network starts to exit the saturation level at the end of the simulation period and slowly returns to the undersaturation level from 8:35 to 9:30.

5 | RESULTS

In this section, the results of the implementation and application of SA and GA to the simulation-based DTA problem for the large-scale test case are presented in view to comparing them with the MSA, MSAR, and gap-based algorithms. The MSA algorithm is chosen as a reference algorithm. MSAR and gap-based are chosen as the best algorithms for large-scale networks based on several studies in the literature such as Sbayti et al. (2007), Lu et al. (2009), Levin, Pool et al. (2014), Verbas, Mahmassani, and Hyland (2016), and our comprehensive benchmark on simulation-based DTA solution methods (Ameli et al., 2020). All the experiments are first initiated by the first outer loop with the all-or-nothing assignment algorithm (see Step 1 in Figure 1). To compare the performance of the algorithms, we use the same outer loop component with different methods in the inner loop.

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 restrict the computational cost. Each classical algorithm is run for 10 outer loops. This means that users have to finally choose from a minimum of 11 paths (the first path comes from Step 1) for each OD pair. For SA and GA, to avoid blocking the random methods with a small number of paths per OD, we start the first outer loop with the K-shortest path algorithm ($K = 3$) and continue in the same way as the other methods with one shortest path per outer loop iteration. This means the maximum number of outer loops for meta-heuristic algorithms is 8 to keep the total number of minimum paths at 11. In addition, the CT of the time-dependant k-shortest path algorithm at each assignment period is much higher than a single shortest path discovery process in the considered large-scale network. The extra CT for meta-heuristic algorithms is approximately equivalent to having two less outer loop iterations than classical algorithms in our test case. The inner loop runs for a maximum of 20 iterations ($i_{max} = 20$) for all the algorithms. With a 15-minute assignment interval, we will then have a maximum of 1,600 simulations for the classical methods. With the meta-heuristic algorithms, the maximum can be exceeded as some simulations will run in parallel.

The aim of the experiment performed on the Lyon 6e + Villeurbanne network is to examine the convergence pattern and validate the solutions for a large-scale network case. Thus, the UE is calculated for the considered network five times with

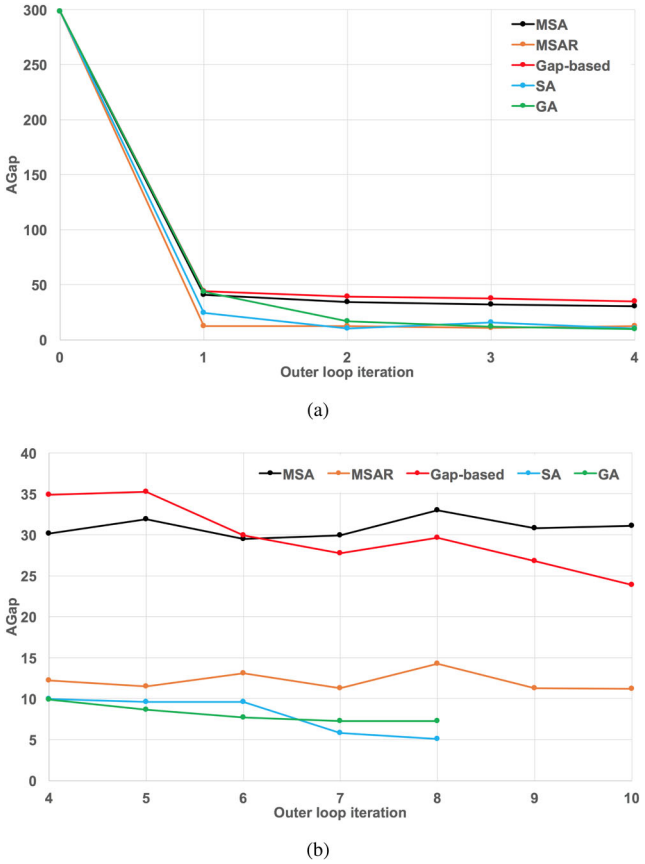


FIGURE 7 Convergence patterns of the outer loop iterations

the following algorithms: MSA, MSAR, gap-based, SA, and GA. The *AGap* indicator is used to evaluate the quality of the solution. Figure 7 presents the convergence pattern for the five algorithms. Figure 7a presents the convergence pattern of the first four outer loop iterations. Figure 7b shows the convergence patterns of all the algorithms from outer loop 4 to the end of the optimization process, and magnifies the difference between the convergence patterns of the algorithms. Note that each algorithm is terminated once the maximum number of outer loops is reached. Another possibility, which does not occur here, is that convergence is reached when the solution is not changed in two consecutive outer loops or no new shortest path is found for all the ODs.

In Figure 7, the convergence pattern and the final result of MSA and gap-based algorithms are close, but MSAR dominates both algorithms easily from the first outer loop. Note that the better performances of the MSAR have a cost. Each inner loop iteration takes longer because of the sorting of users by the experienced travel time for each OD. MSAR algorithm converges faster than SA and GA at the first outer loop, but it is dominated by both meta-heuristic algorithms after the fourth outer loop. Both meta-heuristic algorithms produce better convergence patterns than the classical algorithms. Note that in GA, the best DNA of the population is sent to the outer loop. Therefore, the convergence pattern of GA is always decreasing compared to the other methods.

The results for the performance indicators of all the algorithms are presented in Table 2. As expected, the numbers of total simulations for SA and GA are larger than those of the classical methods around 320%, despite the fact that the CTs are significantly lower than those of the classical methods because of the parallel simulation framework. Moreover, the solutions obtained by the meta-heuristic algorithms are significantly closer to the optimal UE than those of the classical methods. The *AGap* of the GA solution is better than that of the classical method, indeed it is 76% better than the MSA algorithm. The SA algorithm manages to reduce the UE *AGap* of the MSAR method by more than 54%, the MSAR method being the best classical method compared to the MSA and gap-based methods. The Violation indicator also shows that GA and SA work much better than the MSA and gap-based methods (reduction of 82%) and even better than the MSAR method as they reduce the violation by one third of its value (−6%). In addition, the meta-heuristic algorithms dominate the others regarding the percentage of incomplete trips. The incomplete trips denote the share of travelers who could not finish their trip by the end of the simulation in the final path flow distribution of each algorithm. A lower number of incomplete trips means a lower total travel time spent in the system over the simulation period. The SA algorithm finds the closest solution to UE (minimum *AGap*) in this study. Moreover, the final solution of SA has the best value for other quality indicators (Violation and Incomplete travelers) in Table 2.

To evaluate the performance of the algorithms, the CT should also be considered. The CTs of both meta-heuristic algorithms are better than those of the classical methods. In particular, it is significantly better than the MSA method (Table 2). Note that each iteration for the MSAR method takes longer than for the MSA method, but the MSAR method dominates the MSA method at the end as it requires fewer iterations than MSA. Note also that because of the network size, the DTA process over the full simulation period requires considerable computational resources, that is, about a week for the classical methods (MSA, MSAR, gap-based). Therefore, the computational improvement obtained by switching to the meta-heuristic methods is huge; 36 hours (a day and a half) for the SA algorithm and 67.5 hours (two and a half days) for the GA when compared to the MSA algorithm. The GA can take the most advantage of parallel computing as each DNA can be run as a separate thread.

From the application standpoint, every simulation usually needs one central processing unit (core) of the computer. The classical algorithms are run in series, so they use one core per iteration. The performance of SA is very good in terms of solution quality, but the potential for parallelization is limited. Only the different exploration methods running in parallel can be assigned to different threads. The SA generates a

TABLE 2 Solution quality and performance indicators (CT: computation time)

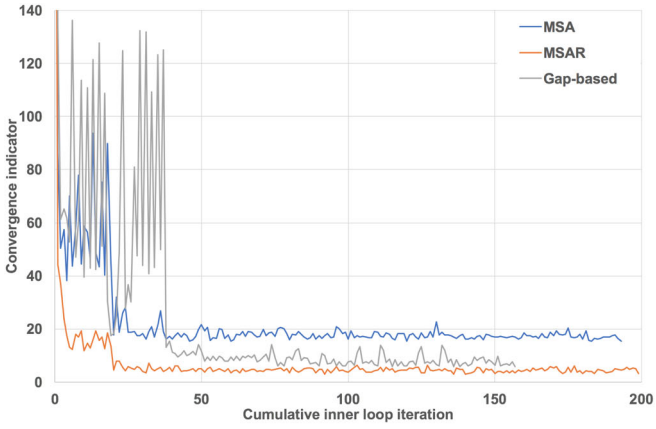
Indicator/ Method	Number of simulations	Incomplete travels (%)	Violation	Final *AGap*	CT (Hours)	Improvement to CT compared to MSA	MAX number of cores used
MSA	1,485	6,04%	35.34%	31.12	183.74	–	1
MSAR	1,145	5.48%	9.01%	11.19	175.41	4.54%	1
Gap-based	1,512	6.71%	22.70%	23.90	186.16	−6.13%	1
SA	2,317	5.07%	2.61%	5.05	147.86	14.85%	3
GA	9,242	5.26%	3.09%	7.27	116.10	21.48%	12

maximum of three new path flow patterns (in *liquid method*) per iteration, which must be simulated at once. Therefore, we use a maximum of three submethods and then three threads, and finally three cores for the simulation process. For the GA algorithm, according to PS, Cr, and the mutation rate, we will have a maximum of 20 new individuals (children) from the GA-operators and the GA Inner. We limit the number of cores to 12 because all the experiments are conducted on a 64-bit personal computer with 12 cores. If the number of new individuals is bigger than 12, the algorithm executes in two successive phases, the first 12 simulations in the first phase and the remaining individuals in the second phase.

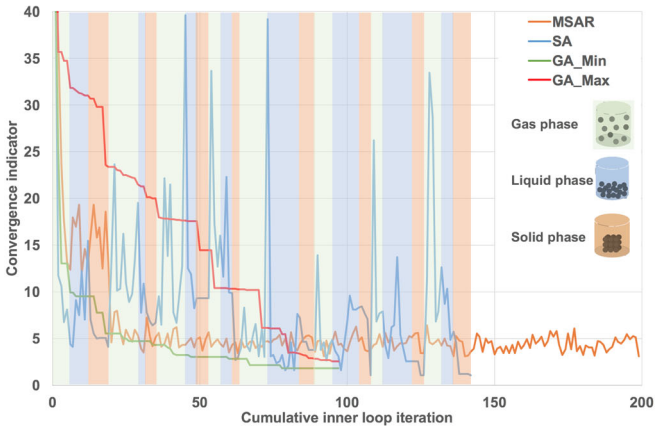
To analyze the behavior of the algorithms, the inner loop convergence patterns of the algorithms are presented in Figure 8. As we are looking for the closest solution to perfect UE ($AGap = 0$), the value of $AGap$ was not used as a stopping criterion. Before analyzing the convergence pattern, we explain why fluctuations exist in the convergence figures. First, when the outer loop index is changed, a new shortest path is added to the system, and the inner loop $AGap$ is calculated by considering the new shortest path, so $AGap$ increases for the first inner loop and then decreases after executing the inner loop iterations. The second reason is that the swapping algorithm does not necessarily improve the $AGap$ at every iteration. The outcome will be even worse when using trip-based approaches because their discrete nature makes them less stable. Thus, we expect more variations, especially when the step size is fixed as in classical methods.

Figure 8a shows that MSA and gap-based algorithms are dominated by MSAR. The convergence pattern of MSA shows that by increasing the inner loop index, the flexibility of the method for exploring the solution space is decreased. The same scenario occurs for the gap-based method with one major difference, which is the high searching flexibility at the beginning of the process. This stems from the gap criterion in the swapping formula. However, decreasing the step size prevents the gap-based method from finding a better solution. The MSAR algorithm also suffers from the step size, but with ranking technique it is able to find a better solution than the other classical algorithms. We keep the MSAR in Figure 8b to compare it with meta-heuristic algorithms.

The SA algorithm has a regular variation behavior in each outer loop, which corresponds to the three optimization



(a)



(b)

FIGURE 8 Convergence patterns of the inner loop iterations

phases. The gas phase has no limitation for exploring the solution space; hence, we have high variations at the beginning of the outer loop. Then, the variation is decreased in the liquid phase and afterwards the SA algorithm looks for the local optimum in the solid phase. This approach finds the best UE solution in comparison with the other algorithms.

Two values represent the convergence pattern of the GA algorithm in Figure 8b. The lower value (GA_Min) is the best solution in the solution population of GA, and the other value (GA_Max) represents the worst individual of the current population. The interval between two $AGap$ values shows the range of the other individuals in the population. The initial

TABLE 3 Solution quality and performance indicators [CT: Computation time]

Indicator/ Method	Number of simulations	Incomplete travels (%)	Violation	Final *AGap*	CT [Hours]	Improvement to CT compared to MSA	MAX number of cores used
MSA	80	6.34%	36.78%	35.67	19.44	–	1
MSAR	80	5.94%	13.06%	18.59	20.05	−3.14%	1
Gap-based	80	7.59%	25.61%	24.02	19.69	−1.29%	1
SA	157	5.27%	3.17%	5.54	16.87	13.22%	3
GA	688	5.67%	5.04%	7.63	15.66	19.46%	12

population at the beginning of each outer loop is generated randomly, so the new shortest path attracts users before starting the inner loop iterations. Note that the outer loop finds the new shortest path and it has zero flow in the final solution of the previous outer loop. Thus, the newly generated solutions for the initial population decrease the variation of *AGap* compared to the other algorithms when starting a new outer loop. During the inner loop, the quality of the best solution never increases because the top 10 solutions are always kept. The range of population *AGap* is decreased until the process converges. In GA, with two layers of optimization, we consider a wide range of solutions at each iteration whereas the algorithm has less focus on finding the local optimum. The parallel simulation framework helps GA to converge faster than the other methods with minimum CT.

To evaluate the performance of the solution methods in the absence of the path discovery procedure, we fixed the path set for all ODs and rerun the optimization algorithms. In other words, the optimization is started from a predefined set of paths, which is also an option for simulation-based DTA (Han, Eve, & Friesz, 2019). It means there is no outer loop in the optimization process, and we directly iterate in the inner loop for each assignment period. All the paths that are discovered by the outer loop iterations in previous experiments are selected, and then we rerun the simulation with this setting. Table 3 presents the results of the solution algorithms with predefined paths. The meta-heuristic algorithms also dominate the classical algorithms starting from a predefined path set. The quality of the solution (*AGap*) is improved by SA, 70%, and GA, 59%, compared to the best classical method (MSAR). Similar to previous experiments, the number of simulations for meta-heuristic algorithms is more than classical methods. Still, we improve the CT, in addition to solution quality, significantly by using the parallel approach (Table 3).

We have demonstrated the efficiency of the meta-heuristic algorithms for the general problem. Note that the shortest path calculation procedure plays an important role in the optimization process. It is embedded in the outer loop in this framework. As mentioned before, there are many studies in the literature about improving the calculation of the shortest path in large-scale (see, e.g., Attanasi et al., 2015; Heywood et al., 2019; Hribar, Taylor, & Boyce, 2001; Zhang, Yang, Jia, Wang, & Chen, 2010). Those methods can be combined with our meta-heuristic algorithms for path flow calculation to improve the overall CT further.

6 | CONCLUSION

This article focused on finding the UE solution for a simulation-based DTA problem. This problem is computationally challenging for large-scale networks. Much research has been performed to propose efficient algorithms. Most of the works done have been based on fixed-point algorithms and iterate in series to improve the current solution. First, this study highlighted the drawbacks of serial algorithms. Second, for the first time, parallelized meta-heuristic approaches were applied to solve the network equilibrium problem. Two new meta-heuristic algorithms were proposed to overcome the disadvantages of the serial algorithms: the first derived from the SA framework and the second from that of GA.

The SA algorithm has three layers of optimization for searching the solution space. The algorithm starts in the gas phase and searched the feasible solution space without limitations. Then, the temperature of the algorithm is decreased and the search space is narrowed by shifting the optimization process from the gas phase to the liquid phase and then to the solid phase (Figure 2). In the GA framework, a new layer in the optimization process has been added to account for correlations between OD assignments (Figure 4). Moreover, GA considers a set of solutions instead of a single solution at every iteration. Both algorithms were implemented using parallel computing for calculating the UE in the DTA model. The new algorithms were applied to the real large-scale network of Lyon6e + Villeurbanne in a simulation-based DTA model for a time period of 2 hours. To compare the SA and GA with existing methods, we have considered three algorithms from the classical approach in the literature: the MSA algorithm, which is one of the commonest solution algorithms implemented in the field of DTA, and two recent extensions of the MSA algorithm for simulation-based DTA, namely, MSAR and gap-based algorithms.

The results show that meta-heuristic algorithms dominate classical methods. They provide wide coverage when exploring the solution space. Hence, they find a better solution in terms of closeness to the optimal UE solution. The SA has

provided the best solution, which was significantly better than the best solution obtained by the classical methods (MSAR). The parallel simulation framework helped the meta-heuristic algorithms to run more simulations compared to serial algorithms and speeded up the exploration process by more than 37%, meaning that we could obtain the solutions about 3 days in advance.

This study has introduced a new branch of optimization algorithms for simulation-based DTA models. Therefore, designing other types of meta-heuristic algorithms is certainly a very promising direction of research when attempting to overcome the curse of dimensionality related to large-scale DTA problems. Here, we introduced the first parallel computation framework to solve DTA problems. It is also interesting to investigate other effective parallel and distributed algorithms or computing platforms to improve the CT of large-scale problems. Integrating this approach with the path discovery and simulation steps can be an interesting research direction.

ACKNOWLEDGMENTS

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

Akamatsu, T. (2001). An efficient algorithm for dynamic traffic equilibrium assignment with queues. *Transportation Science*, *35*(4), 389–404.

Alisoltani, N., Leclercq, L., Zargayouna, M., & Krug, J. (2019). Optimal fleet management for real-time ride-sharing service considering network congestion. In *The 98th annual meeting of the Transportation Research Board*. Washington, DC.

Ameli, M., Lebacque, J.-P., & Leclercq, L. (2017). Multi-attribute, multi-class, trip-based, multi-modal traffic network equilibrium model. *The 12th Conference on Traffic and Granular Flow (TGF)*.

Ameli, M., Lebacque, J.-P., & Leclercq, L. (2020). Cross-comparison of convergence algorithms to solve trip-based dynamic traffic assignment problems. *Computer-Aided Civil and Infrastructure Engineering*, *35*(3), 219–240.

Attanasi, A., Silvestri, E., Meschini, P., & Gentile, G. (2015). Real world applications using parallel computing techniques in dynamic traffic assignment and shortest path search. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. IEEE, pp. 316–321.

Bar-Gera, H. (2002). Origin-based algorithm for the traffic assignment problem. *Transportation Science*, *36*(4), 398–417.

Barceló, J., Ferrer, J., García, D., Grau, R., Forian, M., Chabini, I., & Le Saux, E. (1998). Microscopic traffic simulation for ATT systems analysis. A parallel computing version. *25th Anniversary of CRT*, 1–16.

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.

Ben-Akiva, M. E., Gao, S., Wei, Z., & Wen, Y. (2012). A dynamic traffic assignment model for highly congested urban networks. *Transportation Research Part C: Emerging Technologies*, *24*, 62–82.

Busetti, F. (2003). Simulated annealing overview. Retrieved from www.geocities.com/francorbusetti/saweb.pdf, 4.

Busetti, F. (2007). Genetic Algorithms Overview. http://www.scribd.com/doc/396655/Genetic-Algorithm-Overview (30 August 2008)

Dekkers, A., & Aarts, E. (1991). Global optimization and simulated annealing. *Mathematical Programming*, *50*(1–3), 367–393.

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.

Drissi-Kaïtouni, O., & Hameda-Benchekroun, A. (1992). A dynamic traffic assignment model and a solution algorithm. *Transportation Science*, *26*(2), 119–128.

Flötteröd, G. (2018). DTA simulation with reduced number of iterations. In *Swedish national transport conference*. Göteborg, Sweden, October 16, 2018.

Fonseca, C. M., & Fleming, P. J. (1995). An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, *3*(1), 1–16.

Foytik, P., Jordan, C., & Robinson, R. M. (2017). Exploring simulation based dynamic traffic assignment with a large-scale microscopic traffic simulation model. In *Proceedings of the 50th Annual Simulation Symposium*, page 11. Society for Computer Simulation International.

Franzin, A., & Stützle, T. (2019). Revisiting simulated annealing: A component-based analysis. *Computers & Operations Research*, *104*, 191–206.

Friesz, T. L. (2010). *Dynamic optimization and differential games*, vol. 135, New York: Springer Science & Business Media.

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

Galligari, A., & Sciandrone, M. (2017). A convergent and fast path equilibration algorithm for the traffic assignment problem. *Optimization Methods and Software*, *33*(2), 354–371.

Gentile, G. (2016). Solving a dynamic user equilibrium model based on splitting rates with gradient projection algorithms. *Transportation Research Part B: Methodological*, *92*, 120–147.

Han, K., Eve, G., & Friesz, T. L. (2019). Computing dynamic user equilibria on large-scale networks with software implementation. *Networks and Spatial Economics*, *19*(3), 869–902.

Heywood, P., Maddock, S., Bradley, R., Swain, D., Wright, I., Mawson, M., … Richmond, P. (2019). A data-parallel many-source shortest-path algorithm to accelerate macroscopic transport network assignment. *Transportation Research Part C: Emerging Technologies*, *104*, 332–347.

Holland, J. H. (1992). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. Cambridge, MA: MIT Press.

Hribar, M. R., Taylor, V. E., & Boyce, D. E. (2001). Implementing parallel shortest path for parallel transportation applications. *Parallel Computing*, *27*(12), 1537–1568.

Idri, A., Oukarfi, M., Boulmakoul, A., & Zeitouni, K. (2017). Design and implementation issues of a time-dependent shortest path algorithm for multimodal transportation network. In *TD-LSG@ PKDD/ECML*, 32–43.

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.

Jiang, H. (2004). *Parallel implementations of dynamic traffic assignment models and algorithms for dynamic shortest path problems*. PhD thesis, Massachusetts Institute of Technology.

Jordan, C., Foytik, P., Collins, A., & Robinson, R. M. (2017). Development of a future year large-scale microscopic traffic simulation model. In *TRB 2017, Transportation Research Board 96th Annual Meeting*.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*(4598), 671–680.

Krug, J., Burianne, A., & Leclercq, L. (2019). Reconstituting demand patterns of the city of Lyon by using multiple GIS data sources. *Technical report*, University of Lyon, ENTPE, LICIT, Lyon, France.

Leclercq, L., Chevallier, E., & Laval, J. (2008). The lagrangian coordinates applied to the LWR model. In T. Y. Hou & E. Tadmor (Eds.) *Hyperbolic problems: Theory, numerics, applications* (pp. 671–678). Berlin Heidelberg: Springer.

Levin, M. W., Boyles, S. D., & Nezamuddin (2014). Warm-starting dynamic traffic assignment with static solutions. *Transportmetrica B: Transport Dynamics*, *3*(2), 99–113.

Levin, M. W., Pool, M., Owens, T., Juri, N. R., & Waller, S. T. (2014). 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. (1998). Dynamic traffic simulation and assignment: Models, algorithms and application to ATIS/ATMS evaluation and operation. In M. Labbe, G. Laporte, K. Tanczos, & P. Toint (Eds.), *Operations research and decision aid methodologies in traffic and transportation management* (pp. 104–135). Berlin Heidelberg: Springer.

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

Maini, H., Mehrotra, K., Mohan, C., & Ranka, S. (1994). Knowledge-based nonuniform crossover. *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*.

Marcotte, P., & Nguyen, S. (Eds.). (1998). *Equilibrium and advanced transportation modelling*, Boston: Kluwer Academic Publishers.

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*, *34*, 915–931.

Mounce, R., & Carey, M. (2015). 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. (2016). Agent-based traffic assignment. In A. Horni, K. Nagel, & K. W. Axhausen (Eds.), *The multi-agent transport simulation MATSim* (pp. 315–326). London: Ubiquity Press.

Ngoduy, D. (2011). Kernel smoothing method applicable to the dynamic calibration of traffic flow models. *Computer-Aided Civil and Infrastructure Engineering*, *26*(6), 420–432.

Nguyen, S., & Dupuis, C. (1984). An efficient method for computing traffic equilibria in networks with asymmetric transportation costs. *Transportation Science*, *18*(2), 185–202.

Peeta, S., & Ziliaskopoulos, A. K. (2001). Foundations of dynamic traffic assignment: The past, the present and the future. *Networks and Spatial Economics*, *1*, 233–265.

Perederieieva, O., Ehrgott, M., Raith, A., & Wang, J. Y. (2015). Numerical stability of path-based algorithms for traffic assignment. *Optimization Methods and Software*, *31*(1), 53–67.

Raadsen, M. P., Bliemer, M. C., & Bell, M. G. (2019). A review of (dis)aggregation and decomposition methods in traffic assignment. In *TRB 2019, Transportation Research Board 98th Annual Meeting*.

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

Rickert, M., & Nagel, K. (2001). Dynamic traffic assignment on parallel computers in transims. *Future Generation Computer Systems*, *17*(5), 637–648.

Robbins, H., & Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, *22*, 400–407.

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. S. (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.

Schreiter, T., Wageningen-Kessels, V., Yuan, Y., Van Lint, J., & Hoogendoorn, S. (2012). Fastlane: Traffic flow modeling and multi-class dynamic traffic management. In *Trail-Beta Congress 2012, Mobility and Logistics-Science Meets Practice*, Rotterdam, The Netherlands. Citeseer.

Seshadri, R., & Srinivasan, K. K. (2017). Robust traffic assignment model: Formulation, solution algorithms and empirical application. *Journal of Intelligent Transportation Systems*, *21*(6), 507–524.

Sheffi, Y. (1985). *Urban transportation networks: Equilibrium analysis with mathematical programming methods* (chapters 10 and 11, pp. 262–308). Englewood Cliffs, NJ: Prentice Hall Inc.

Siddique, N., & Adeli, H. (2014a). Spiral dynamics algorithm. *International Journal on Artificial Intelligence Tools*, *23*(06), 1430001.

Siddique, N., & Adeli, H. (2014b). Water drop algorithms. *International Journal on Artificial Intelligence Tools*, *23*(06), 1430002.

Siddique, N., & Adeli, H. (2015a). Harmony search algorithm and its variants. *International Journal of Pattern Recognition and Artificial Intelligence*, *29*(08), 1539001.

Siddique, N., & Adeli, H. (2015b). Nature inspired computing: An overview and some future directions. *Cognitive Computation*, *7*(6), 706–714.

Siddique, N., & Adeli, H. (2016a). Gravitational search algorithm and its variants. *International Journal of Pattern Recognition and Artificial Intelligence*, *30*(08), 1639001.

Siddique, N., & Adeli, H. (2016b). Physics-based search and optimization: Inspirations from nature. *Expert Systems*, *33*(6), 607–623.

Siddique, N., & Adeli, H. (2016c). Simulated annealing, its variants and engineering applications. *International Journal on Artificial Intelligence Tools*, *25*(06), 1630001.

Song, W., Han, K., Wang, Y., Friesz, T., & Del Castillo, E. (2017). Statistical metamodeling of dynamic network loading. *Transportation Research Procedia*, *23*, 263–282.

Srinivas, M., & Patnaik, L. (1994). Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, *24*(4), 656–667.

Srinivasan, S., Riazi, S., Norris, B., Das, S. K., & Bhowmick, S. (2018). A shared-memory parallel algorithm for updating single-source shortest paths in large dynamic networks. In *2018 IEEE 25th International Conference on High Performance Computing (HiPC)*. IEEE.

Stathopoulos, A., & Tsekeris, T. (2004). Hybrid meta-heuristic algorithm for the simultaneous optimization of the o–d trip matrix estimation. *Computer-Aided Civil and Infrastructure Engineering*, *19*(6), 421–435.

Szeto, W., & Lo, H. K. (2005). Non-equilibrium dynamic traffic assignment. In H. S. Mahmassani (Ed.), *Transportation and traffic theory* (pp. 427–445). College Park: Elsevier.

Szeto, W. Y., & Lo, H. K. (2006). Dynamic traffic assignment: Properties and extensions. *Transportmetrica*, *2*(1), 31–52.

Talbi, E.-G. (2009). *Metaheuristics: From design to implementation*, Vol. 74, Hoboken: John Wiley & Sons.

Verbas, Ö., 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.

Verbas, İ. Ö., 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*(1), 64–74.

Wang, J., Zhong, D., Adeli, H., Wang, D., & Liu, M. (2018). Smart bacteria-foraging algorithm-based customized kernel support vector regression and enhanced probabilistic neural network for compaction quality assessment and control of earth-rock dam. *Expert Systems*, *35*(6), e12357.

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*, *111*, 370–394.

Wardrop, J. G. (1952). Some theoretical aspects of road traffic research. *Institution of Civil Engineering*, *1*, 325–362.

Xie, J., Nie, Y. M., & Liu, X. (2018). A greedy path-based algorithm for traffic assignment. *Transportation Research Record: Journal of the Transportation Research Board*, *2672*(48), 36–44.

Yang, Q., Balakrishna, R., Morgan, D., & Slavin, H. (2017). Large-scale, high-fidelity dynamic traffic assignment: Framework and real-world case studies. *Transportation Research Procedia*, *25*, 1290–1299.

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

Yun, I., & Park, B. (2006). Application of stochastic optimization method for an urban corridor. In *Proceedings of the 2006 Winter Simulation Conference*. IEEE.

Zhang, L., Yang, Z., Jia, H., Wang, B., & Chen, G. (2010). Test and implement of a parallel shortest path calculation system for traffic network. In *International Conference on Information Computing and Applications*, Springer, 282–288.

How to cite this article: Ameli M, Lebacque J-P, Leclercq L. Simulation-based dynamic traffic assignment: Meta-heuristic solution methods with parallel computing. *Comput Aided Civ Inf*. 2020;35:1047–1062. https://doi.org/10.1111/mice.12577