Accepted for publication in the IEEE International Conference on Intelligent Transportation (ITSC 2021).

# Calibrating Dynamic Traffic Assignment Models by Parallel Search using Active-CMA-ES

Jaebak Hwang[1] and Sungahn Ko[1] and Tsz-Chiu Au[1]

***Abstract*— The widespread deployment of inductive-loop traffic detectors allows us to obtain a massive amount of traffic data in real time. A key step of utilizing the data is to use the data to fit a traffic model. Simultaneous perturbation stochastic approximation (SPSA) and its variants are popular techniques for the calibration of dynamic traffic assignment (DTA) models by searching for Origin-Destination (OD) matrices that fit the data. However, the performance of SPSA cannot scale with modern multi-core CPU architectures due to its sequential nature. This paper proposes the use of active covariance matrix adaptation evolution strategy (Active-CMA-ES) for optimizing OD matrices in microscopic traffic simulation. CMA-ES is a blackbox optimization algorithm that was found highly effective in many domains. According to our case study in Ulsan, South Korea, Active-CMA-ES outperforms Restart-WSPSA, one of the best SPSA algorithms, in terms of the calibration error and the running time. Moreover, the running time of Active-CMA-ES decreases as the number of parallel simulation processes increases.**

## I. INTRODUCTION

Nowadays, we can obtain many traffic data in real time from inductive loop sensors installed on roads. Data-driven approaches for analyzing traffic flows could potentially be more accurate than traditional macroscopic traffic flow modeling. The key step of utilizing the data is to use the data to fit a traffic model such as a microscopic traffic simulation model. Dynamic traffic assignment (DTA) models have been used to compute the equilibrium traffic flow in a traffic simulation. For example, PTV Vissim[1] uses a module called Dynamic Assignment to iteratively compute the traffic flow in equilibrium given a map and an OD matrix. To calibrate a DTA model, a calibration algorithm adjusts the OD matrix until the equilibrium traffic flow matches the collected data.

Simultaneous perturbation stochastic approximation (SPSA) is a popular approach for calibrating DTA models. Lu Lu et al. [1] proposed weighted SPSA (W-SPSA) for estimating OD matrices in dynamic settings in which the OD matrix is assumed to be non-static, and the DTA estimation algorithm has to estimate several OD matrices in a row for consecutive time intervals. Huan Yang et al. [2] presented a modified SPSA algorithm called Restart-SPSA, which improves SPSA by a restart strategy for dynamic origin-destination estimation. Both approaches showed that SPSA could be quite effective in calibrating DTA models. However, they are too slow for some applications in which we need to estimate the traffic flow as quickly as possible. Since both methods use gradient descent in which the search steps must be executed in sequential order, it is difficult to reduce their running times.

In this paper, we propose to replace SPSA by active covariance matrix adaptation evolution strategy (Active-CMA-ES) [3] in calibrating DTA models. CMA-ES, proposed by Hansen and Ostermeier, is an evolution strategy for blackbox optimization [4]. CMA-ES is highly effective in many application domains, including robotics [5]. Like the classical evolution strategies, CMA-ES evolves a population of solutions in parallel, making it suitable for parallel processing in modern multi-core CPU architectures. Some previous works have already used evolution algorithms or genetic algorithms for calibrating traffic simulation [6], [7]. However, their work address slightly different calibration problems and their focus are not on achieving parallel search. Moreover, CMA-ES could outperform these classical evolution strategies, as demonstrated in other application domains. Active-CMA-ES is a modified version of CMA-ES with new update rules for covariance matrices, making it more suitable for noisy, non-smooth, non-continuous, and non-convex blackbox optimization problems, including our DTA calibration problem.

This paper is organized as follows. After presenting the related work in Sec. II, we define the DTA calibration problem in Sec. III and describe Restart-WSPSA and Active-CMA-ES in Sec. IV. Then we present the results of experiments that evaluate Active-CMA-ES using the traffic data collected by inductive loop sensors in the downtown of Ulsan, a city in South Korea, in Sec. V. Lastly, we conclude in Sec. VI.

## II. RELATED WORK

One of the early works on the calibration of traffic models was based on genetic algorithms (GA) [7], [8]. However, as the size of OD matrices increases, GA requires a large population per generation and a larger number of iterations to converge to a solution. Thus, these early attempts were later overshadowed by gradient-descent methods, notably simultaneous perturbation stochastic approximation (SPSA) [9]–[11]. SPSA is a derivative-free optimization algorithm that estimates gradients by sampling, making it suitable for the calibration of traffic models which have no obvious derivative functions. When compared with GA, SPSA requires less computation time per iteration and can converge much faster. Huan Yang et al. [2] presented Restart-SPSA, which applies a restart strategy to improve the performance of SPSA. Lu

[1]Department of Computer Science and Engineering, Ulsan National Institute of Science and Technology (UNIST), South Korea. {milk,sako,chiu}@unist.ac.kr

[1]https://www.ptvgroup.com/en/solutions/products/ptv-vissim

Lu et al. [1] proposed Weighted SPSA (W-SPSA), which utilizes the correlation between OD matrices and the ground truth data to generate a weight matrix that gives more weight to certain elements in the OD matrix during gradient descent.

One drawback of SPSA is that there is no obvious way to parallelize its search process. In this paper, we consider covariance matrix adaptation evolution strategy (CMA-ES) [4], a famous blackbox optimization algorithm that has been quite successful in hundreds of applications [12]. Like GA, CMA-ES maintains a pool of solution candidates in each generation, and the evaluation of the fitness of the candidates can be done in parallel due to the lack of dependency among the candidates. Although CMA-ES takes more time to evolve one generation when compared with SPSA, CMA-ES is much easier than SPSA in escaping from local minima since the gradient are estimated by multiple sample points. Bojan Kostic et al. [13] compared SPSA, CMA-ES, and Nelder-Mead's Simplex algorithm (NMSIM) [14], and showed that both SPSA and CMA-ES are much faster than NMSIM when calibrating a macroscopic simulator based on traffic equilibrium for first-order dynamic traffic simulation models [15], [16]. In this paper, we calibrate a microscopic traffic simulator by Active-CMA-ES [3], an enhanced version of CMA-ES with faster convergence by using negative results to update the covariance matrix.

## III. THE DTA CALIBRATION PROBLEM

We define the calibration problem for DTA models as in [2]. The calibration process is comprised of a sequence of iterations, each of which aims to compute an estimated OD matrix that minimizes an objective function as defined below. Suppose there are $M$ lane groups with inductive loop sensors in a traffic network (e.g., the sensors in Fig. 1). Let $y_i(\tau)$ be the number of vehicles detected by the inductive loop sensors in the lane group $i$ in the time interval $\tau$. Then $Y(\tau) = [y_1(\tau), y_2(\tau), \ldots, y_M(\tau)]^T$ is the ground truth for the calibration. Suppose there are $N$ OD pairs in the OD matrix. Let $X_k(\tau) = [x_1(\tau), x_2(\tau), \ldots, x_N(\tau)]^T$ be the estimated OD matrix in the $k$ iteration, where $x_j^k(\tau)$ is the traffic demand of the $j$th OD pair in $\tau$. Let $\hat{Y}_k(\tau) = [\hat{y}_1^k(\tau), \hat{y}_2^k(\tau), \ldots, \hat{y}_M^k(\tau)]^T$ be a vector of the estimated numbers of vehicles in the lane groups in time interval $\tau$ according to $X_k(\tau)$. The objective function $Z(\cdot)$ in the $k$ iteration is:

$$\min_{X_{k+1}(\tau)\geq 0} Z(X_{k+1}(\tau)) = \omega_1 E(X_{k+1}(\tau), X_k(\tau)) + \omega_2 E(\hat{Y}_{k+1}(\tau), Y(\tau)) \quad (1)$$

where $E(V_1, V_2) = ||V_1 - V_2||$ is the root mean square error between any two vectors $V_1$ and $V_2$, and $\omega_1 + \omega_2 = 1$ for $\omega_1, \omega_2 \geq 0$. Eq. 1 is a weighted sum of two terms. The first term is used to reduce the change of the estimated OD matrix after each iteration, and the second term is used to minimize the error between the estimated traffic flows of each lane group and the ground truth. Typically, $\omega_1$ is much smaller than $\omega_2$ so that the accuracy of the estimated traffic flows has a higher weight. In our experiments, $\omega_1 = 0.1$ and $\omega_2 = 0.9$.

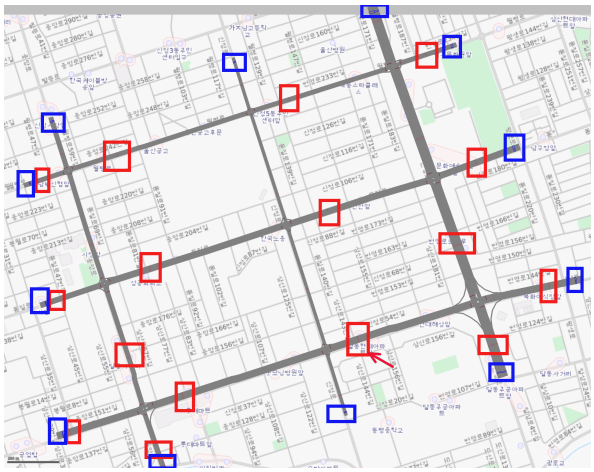


Fig. 1: The map of a region in the downtown of Ulsan in South Korea. The grey roads are the road segments that traffic were simulated in PTV Vissim in our experiments. The blue boxes are the entries and the exits to the region. The red boxes indicate the physical locations of the inductive loop sensors. All but one red box contain two inductive loop sensors, one for each direction of the road. The red box pointed by an arrow contains one inductive loop sensor only.

## IV. THE CALIBRATION ALGORITHMS

In this section, we describe two calibration algorithms: Restart-WSPSA and Active-CMA-ES.

### A. Restart-SPSA

SPSA is a gradient descent algorithm that starts from an initial point of the parameter vector and then updates the parameter by using gradient approximation until the error converges to zero. Let $\hat{X}_k$ be the estimated vector of parameters in the $k$th iteration. The iterative process of the SPSA algorithm is based on the following equation:

$$\hat{X}_{k+1} = \hat{X}_k - a_k \hat{g}_k(\hat{X}_k)$$

where $\hat{g}_k(\hat{X}_k)$ is the estimated gradient column vector at $\hat{X}_k$, and $a_k$ is a small positive number as the gain factor of $\hat{g}_k(\hat{X}_k)$. In other words, $a_k$ is the main factor of the step size per iteration. In each iteration, the gradient is estimated by:

$$\hat{g}_k(\hat{X}_k) = \frac{(Z(\hat{X}_k + c_k\Delta_k) - Z(\hat{X}_k - c_k\Delta_k))}{2c_k} \begin{bmatrix} \Delta_{k1}^{-1} \\ \Delta_{k2}^{-1} \\ \vdots \\ \Delta_{kN}^{-1} \end{bmatrix} \quad (2)$$

where 1) $\Delta_k$ is a $N \times 1$ vector those elements are generated by the Bernoulli distribution with the values of $\pm 1$, 2) $N$ is the dimension of $\hat{X}$, and 3) $c_k$ is a positive number called the gain factor of $\Delta_k$. Both $a_k$ and $c_k$ get smaller as $k$ increases according to the following equations:

$$a_k = a/(A + k + 1)^\alpha \quad (3)$$

$$c_k = c/(k + 1)^\gamma \quad (4)$$

where $a$, $c$, $\alpha$, $\gamma$, and $A$ are parameters that are tuned manually. According to [10], we can pick $\alpha = 0.602$ and $\gamma = 0.101$. If $a$ is too large, the oscillation of the estimated parameter vector occurs, and the algorithm cannot converge. If $a$ is too small, the algorithm will converge slowly and may easily be trapped at local minima.

**Restart Strategy**: Restart-SPSA is an extension to SPSA with the following restart strategy. First, Restart-SPSA runs SPSA with a large value of $a$ for a given number of iterations. Let $\hat{X}_k$ be the best parameter in the execution of SPSA. Then Restart-SPSA reruns SPSA from scratch with 1) a smaller value of $a$ and 2) $\hat{X}_k$ as the initial search point. Let $\hat{X}'_k$ be the best parameter in the second execution of SPSA. Then Restart-SPSA returns $\hat{X}'_k$ as the solution. Typically, Restart-SPSA can find better solutions than SPSA.

### *B. Restart-WSPSA*

Weighted-SPSA (W-SPSA) was introduced in [1]. The idea is to modify the estimated gradient in Eq. 2 by a weight matrix $W$ such that the $i$th element in the gradient becomes larger (or smaller) if the $i$th element in $\hat{X}_k$ has a stronger (or weaker) influence to the convergence of $\hat{X}$. More precisely, W-SPSA modifies Eq. 2 by computing the $i$th element in $\hat{g}_k(\hat{X}_k)$ using the following equation instead:

$$\hat{g}_{ki}(\hat{X}_k) = \frac{\acute{Z}(\hat{X}_k + c_k\delta_k) - \acute{Z}(\hat{X}_k - c_k\delta_k)}{2c_k\delta_{ki}} W_i, \tag{5}$$

where $\acute{Z}(\theta)$ and $W_i$ are computed by the following equations:

$$\acute{Z}(X_k) = \begin{bmatrix} w_1((Y(X_k) - \hat{Y}(X_k)) \bigotimes ((Y(X_k) - \hat{Y}(X_k)) \\ w_2((X_k - X_{k-1}) \bigotimes (X_k - X_{k-1})) \end{bmatrix} \tag{6}$$

$$W_{i,m} = \frac{d_{i,j}}{\sum_{j=1}^{M} d_{i,j}}, \tag{7}$$

where $Y(X_k)$ is the observed traffic measurements with OD matrix $X_k$, $\hat{Y}(X_k)$ is the simulated measurements, $X_{k-1}$ is the prior value of parameters, $w_1$ and $w_2$ are constants, and $d_{i,j}$ is the increment or decrement of the $j$th observation when the $i$th element increases. In [1], the same weight matrix $W$ is used in all time intervals. However, in this paper, we generated different $W$ in each time interval, using the calibrated OD matrix in the previous time interval. We also apply the restart strategy to W-SPSA to form *Restart-WSPSA* as in [2].

### *C. Active-CMA-ES*

CMA-ES is a blackbox optimization algorithm based on evolution strategies. Active-CMA-ES is an enhanced version of CMA-ES [3]. CMA-ES generates $\lambda$ offspring based on mutation strength $\sigma$ and $n \times n$ covariance matrix C from the search point $\theta$, and updates $C$ using $n$-dimension search path vectors $p_\sigma$ and $p_c$. The search path vectors accumulate the information of high-ranking offsprings. Active-CMA-ES also uses low-ranking offsprings for updating the covariance matrix $C$ negatively. This negative update makes CMA-ES adaptation much faster. The search path vectors are initialized to zero, and the covariance matrix is initialized to a unity matrix. More precisely, Active-CMA-ES proceeds according to the following steps:

1) For $N$ number of parameters, $C = BD(BD)^T$ where $n \times n$ matrix $B$ is normalized eigenvectors of $C$, and $D$ is a diagonal $n \times n$ matrix. The diagonal elements are the square roots of the eigenvalues of $C$.
2) Generate an offspring $X$ based on the following equation:
$$X_i = \theta + \sigma BDz_i, \tag{8}$$
where $z_i$ is a mutation vector with $n$ elements, generated from a normal distribution, for $1 \le i \le \lambda$.
3) Calculate the loss $Z(X_i)$ and sort the results by $z_{k;\lambda}$, where $k;\lambda$ means the index of the $k$th ranked result in the population. Compute the average of $\mu$ rank mutation vectors:
$$E^z = \frac{1}{\mu} \sum_{k=1}^{\mu} z_{k;\lambda} \tag{9}$$
In this paper, we set $\mu = \lambda/2$.
4) Update the search point:
$$\theta = \theta + \sigma BDE^z$$
5) Update the search paths:
$$p_c = (1 - c_C)p_c + \sqrt{\mu c_C(2 - c_C)}BDE^z \tag{10}$$
$$p_\sigma = (1 - c_\sigma)p_\sigma + \sqrt{\mu c_\sigma(2 - c_\sigma)}BE^z \tag{11}$$
where $c_C = c_\sigma = \frac{4}{n+4}$.
6) Update the covariance matrix:
$$C = (1 - c_{cov})C + c_{cov}p_c p_C^T + \beta E, \tag{12}$$
where
$$E = BD\left(\frac{1}{\mu}\sum_{k=1}^{\mu} z_{k;\lambda}z_{k;\lambda}^T - \frac{1}{\mu}\sum_{k=\lambda-\mu+1}^{\lambda} z_{k;\lambda}z_{k;\lambda}^T\right)BD^T \tag{13}$$
and
$$c_{cov} = \frac{2}{(n+\sqrt{2})^2},$$
where $\beta$ is set according to the function type.
7) Update the mutation strength:
$$\sigma = \sigma exp(\frac{\|p_\sigma\| - X_n}{d_\sigma X_n}), \tag{14}$$
where $X_n = \sqrt{n}(1 - 1/(4n) + 1/(21n^2))$ is an approximation of the expectation value of the length of a random vector with a normal distribution and $d_\sigma = 1 + 1/c_\sigma$ is the damping factor.

In Eq. 13, the term $\frac{1}{\mu}\sum_{k=1}^{\mu} z_{k;\lambda}z_{k;\lambda}^T$ denotes the positive adaptation based on the high-ranking offsprings from rank 1 to $\mu$, and the term $-\frac{1}{\mu}\sum_{k=\lambda-\mu+1}^{\lambda} z_{k;\lambda}z_{k;\lambda}^T$ denotes the negative adaptation based on the low-ranking offsprings from $\lambda - \mu + 1$ to $\lambda$.

CMA-ES and Active-CMA-ES have many parameters, but most of them are set by some recommended equations. In most cases, we only need to tune $\sigma$. For more information about how to set the parameters, please refer to [17].

## V. EXPERIMENTAL EVALUATION

This section presents a case study that compares the performance of Restart-WSPSA and Active-CMA-ES when calibrating the OD-matrix in a traffic model.

### A. Experimental Settings

Our case study focuses on a region in the downtown of Ulsan, a city in South Korea. As shown in Fig. 1, the 1.5 km $\times$ 1.1 km region has 9 intersections and 24 road segments. All intersections are managed by traffic signals. We obtained the traffic signals data from Korean National Police Agency[2]. We used PTV Vissim (version 11), a widely used microscopic traffic simulator, with the Dynamic Assignment module for DTA calculation. There are 31 inductive loop sensors in the data. We collected the inductive loop data (i.e., the number of vehicles passing the inductive loops) of the four 15-minute time intervals from 7:00AM to 8:00AM on December 12, 2015 from the Road Traffic Authority of Ulsan[3].

The OD matrix consists of $12 \times 12$ entries since there are 12 entries and 12 exits in the region. The calibration algorithms have to tune $12 \times 11$ parameters in the OD matrix. In the first time interval, the initial value of each element in the OD matrix is half of the maximum value of each element, which is determined manually based on historical traffic records. In the other time interval, the initial OD matrix is the calibrated OD matrix in the previous time interval.

The experiments were conducted on a computer with an Intel Core i7 8700K CPU with 32GB RAM. The CPU has six cores, and thus it can run six traffic simulations in parallel. Nonetheless, PTV Vissim can utilize at most four cores in parallel due to license issues. Both Restart-WSPSA and Active-CMA-ES were implemented in Python 3, and they start and control the simulations in PTV Vissim by the COM interface. We used pycma[4] to implement Active-CMA-ES. The population size of Active-CMA-ES is 16.

To increase the performance of Restart-WSPSA, we optimized the W-matrix in each time interval by using the calibrated OD matrix in the previous time interval. According to [10], we set $\alpha = 0.602$ and $\gamma = 0.101$ in WSPSA. In addition, we restarted WSPSA every 150 generations. We also estimated the extra loss measurement for each iteration by an additional simulation and then rejected the iteration if the extra loss measurement is larger than the previous iteration loss measurement by 20%. According to [11], the rejection can speed up the convergence of Restart-WSPSA. Hence, each iteration in Restart-WSPSA has three simulations, two for calculating the gradient for SPSA and one for loss measurement. Note that the third simulation must be run after the first two simulations.

### B. Calibration Errors vs. the Number of Simulations

First, we looked at the RMSE of the traffic volume on the road segments with inductive loop sensors. In our experiment, we gave both Restart-WSPSA and Active-CMA-ES a budget of 900 simulations they can run using PTV Vissim. In each simulation, PTV Vissim used the Dynamic Assignment module to iteratively estimate the traffic flow given an OD matrix until it reaches an equilibrium. Then we calculated the RMSE based on the traffic flow at the equilibrium and the ground truth. We kept tracking the change of RMSE as the algorithms used more and more simulations to refine the OD matrix. The results are shown in Figures 2(a), 2(c), 2(e), and 2(g). Although Restart-WSPSA ran two simulations in parallel and Active-CMA-ES ran four simulations in parallel, these figures report the running time as if the simulations were run sequentially one by one.

According to the four graphs on the left of Fig. 2, Active-CMA-ES had a higher RMSE at the beginning. But Active-CMA-ES's performance increased rapidly and eventually outperformed Restart-WSPSA in all time intervals. The reason for the poor initial performance is that Active-CMA-ES requires 16 simulations for each generation while Restart-WSPSA requires 3 simulations for each iteration. Hence, Active-CMA-ES needs more simulations to make progress initially. However, Active-CMA-ES eventually outperformed Restart-WSPSA since Active-CMA-ES is more capable of escaping the local minima.

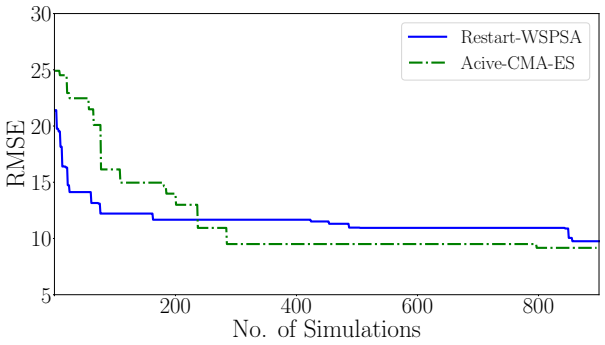
Table I summaries the RMSE for each interval of Restart-WSPSA and Active-CMA-ES after running 900 simulations. According to the table, the performance gap between Restart-WSPSA and Active-CMA-ES increases as the time interval increases. Active-CMA-ES's superior performance is more noticeable in the higher time intervals because the error in a time interval can carry over to the next time interval.

TABLE I: The RMSE for each interval of Restart-WSPSA and Active-CMA-ES after running 900 simulations.

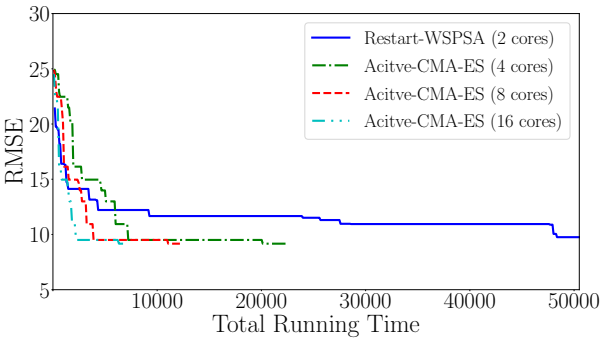
| Interval | 1st | 2nd | 3rd | 4th |
|---|---|---|---|---|
| Restart-WSPSA | 9.755 | 16.629 | 15.142 | 20.484 |
| Active-CMA-ES | 9.167 | 9.958 | 10.058 | 12.675 |

The advantage of Active-CMA-ES is prominent when more simulations can run in parallel. Therefore, we analyzed the data we collected in the experiment to estimate the total running time of the algorithms as the number of CPU cores used by the algorithms in parallel increases. We recorded the running time of every simulation in Active-CMA-ES. Then we estimated the total running time when there are more than 4 CPU cores by overlapping some of the simulations. Note that the running time of both Active-CMA-ES and Restart-WSPSA were dominated by the simulation times, and the running times of other computations were negligible. Hence, we can estimate the total running time by greedily allocating simulations in each generation to the CPU cores. We implemented a discrete event simulation to simulate the execution of Active-CMA-ES when the number of cores is 8 and 16. The results are shown in Figures 2(b), 2(d), 2(f), and 2(h). According to these figures, as the number of

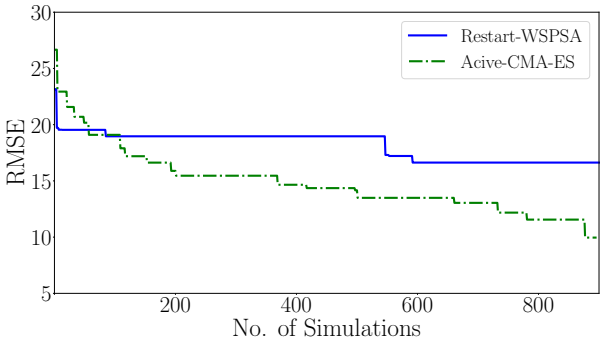
[2] https://www.police.go.kr/eng/main.do
[3] https://utrhub.its.ulsan.kr
[4] https://github.com/CMA-ES/pycma



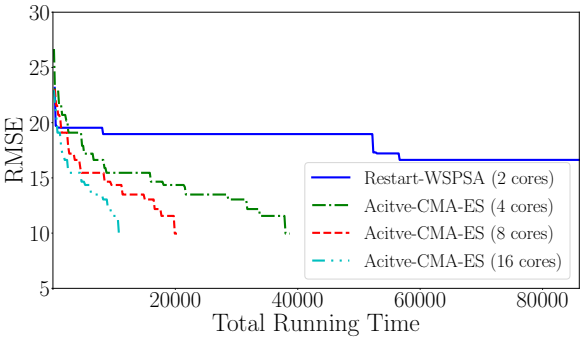

(a) RMSE vs. the number of simulations in the first time interval.



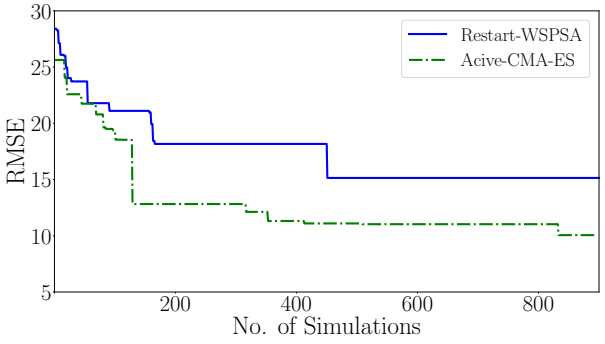

(b) RMSE vs. the total running time in the first time interval.



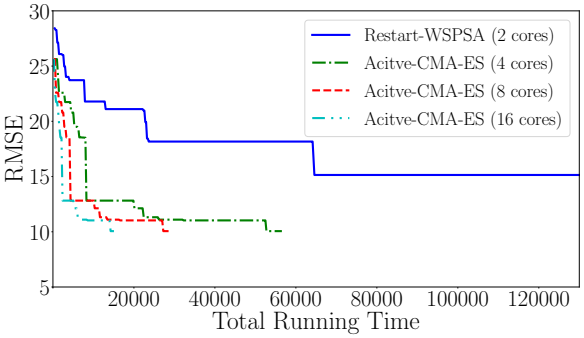

(c) RMSE vs. the number of simulations in the second time interval.



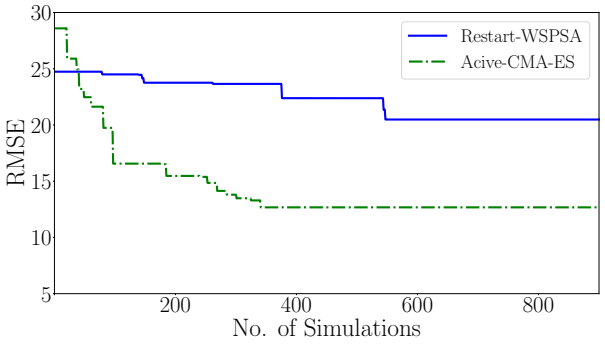

(d) RMSE vs. the total running time in the second time interval.



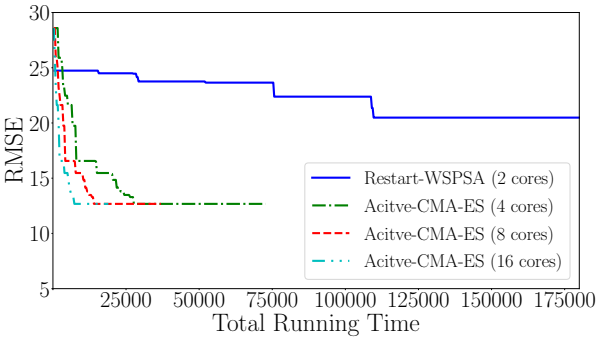

(e) RMSE vs. the number of simulations in the third time interval.




(f) RMSE vs. the total running time in the third time interval.




(g) RMSE vs. the number of simulations in the fourth time interval.




(h) RMSE vs. the total running time in the fourth time interval.

Fig. 2: The performance of Restart-WSPSA and Active-CMS-ES in each of the four time intervals when the traffic simulations ran sequentially and in parallel.

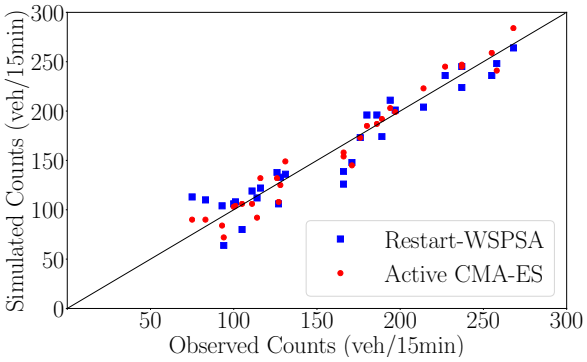



Fig. 3: The relationship between the vehicle counts collected by the inductive loop sensors in the simulation and the vehicle counts in the observed data collected by the inductive loop sensors in the fourth time interval.

CPU cores increases, the total running time of Active-CMA-ES drops tremendously, and the initial poor performance of Active-CMA-ES disappears. Therefore, Active-CMA-ES can be an effective parallel search algorithm for calibrating DTA models.

### C. Calibration Errors of Individual Roads

RMSE measures the error of the calibration of the entire model at once. However, RMSE cannot tell exactly how well the traffic flow at each inductive loop sensor readings is calibrated. Therefore, we examined the observed vehicle counts of the roads at each inductive loop sensor at every interval and compared them with the predicted values generated by Restart-WSPSA and Active-CMA-ES. Fig. 3 is a scatter plot of the vehicle counts of the roads at which the inductive loop sensors are located in the fourth time interval. There are 31 red dots and 31 blue dots. The x-axis is the observed vehicle count (i.e., the ground truth), and the y-axis is the predicted vehicle count (i.e., the simulated count) after calibration. Clearly, the observed counts of both Restart-WSPSA and Active-CMA-ES are the same. The difference between an observed count of a sensor and the corresponding simulated count is the calibration error of the traffic flow at the sensor.

As can be seen, most of the calibration errors are quite small. There are a few outliners, but they appear to occur randomly on different roads. In general, there is a strong linear relationship between the observed counts and the simulated counts for both Restart-WSPSA and Active-CMA-ES. However, Active-CMA-ES performed slightly better as the red points are closer to the diagonal.

## VI. Summary and Future Work

To utilize real-time traffic data to calibrate a traffic model as quickly as possible, we propose to use Active-CMA-ES, which can utilize modern multi-core computer architecture to conduct parallel search. Our experiments showed that Active-CMA-ES outperforms Restart-WSPSA, the best SPSA algorithm in the literature. We have also demonstrated the scalability of Active-CMA-ES as the number of CPU cores increases. Although Active-CMA-ES outperforms the existing calibration algorithms, it has not yet achieved the *real-time* calibration of traffic models. If one can calibrate a traffic model in real time, some important applications such as real-time traffic rerouting for congestion prevention can be more effective. In the future, we intend to extend Active-CMA-ES for incremental calibration of traffic models over time.

## Acknowledgment

This work has been taken place at UNIST supported by NRF (2016R1D1A1B0101359816 and 2016M3C4A795263722).

## References

[1] L. Lu, Y. Xu, C. Antoniou, and M. Ben-Akiva, "An enhanced spsa algorithm for the calibration of dynamic traffic assignment models," *Transportation Research Part C: Emerging Technologies*, vol. 51, pp. 149–166, 2015.

[2] H. Yang, Y. Wang, and D. Wang, "Dynamic origin-destination estimation without historical origin-destination matrices for microscopic simulation platform in urban network," in *International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2994–2999.

[3] G. A. Jastrebski and D. V. Arnold, "Improving evolution strategies through active covariance matrix adaptation," in *IEEE International Conference on Evolutionary Computation*, 2006, pp. 2814–2821.

[4] N. Hansen and A. Ostermeier, "Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation," in *Proceedings of IEEE International Conference on Evolutionary Computation*, 1996, pp. 312–317.

[5] P. MacAlpine, S. Barrett, D. Urieli, V. Vu, and P. Stone, "Design and optimization of an omnidirectional humanoid walk: A winning approach at the RoboCup 2011 3D simulation competition," in *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI)*, July 2012.

[6] H. K. Lo and W. Y. Szeto, "A cell-based dynamic traffic assignment model: formulation and properties," *Mathematical and Computer Modelling*, vol. 35, no. 7-8, pp. 849–865, 2002.

[7] I. Yun and B. Park, "Estimation of dynamic origin destination matrix: A genetic algorithm approach," in *IEEE Intelligent Transportation Systems*, 2005, pp. 522–527.

[8] S. Baek, H. Kim, and Y. Lim, "Multiple-vehicle origin–destination matrix estimation from traffic counts using genetic algorithm," *Journal of Transportation Engineering*, vol. 130, no. 3, pp. 339–347, 2004.

[9] J. C. Spall *et al.*, "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation," *IEEE Transactions on Automatic Control*, vol. 37, no. 3, pp. 332–341, 1992.

[10] J. C. Spall, "Implementation of the simultaneous perturbation algorithm for stochastic optimization," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 34, no. 3, pp. 817–823, 1998.

[11] ——, "An overview of the simultaneous perturbation method for efficient optimization," *Johns Hopkins Apl Technical Digest*, vol. 19, no. 4, pp. 482–492, 1998.

[12] P. I. C. Ryan *et al.*, "References to cma-es applications," *Strategies*, vol. 4527, no. 467, 2007.

[13] B. Kostic, L. Meschini, and G. Gentile, "Calibration of the demand structure for dynamic traffic assignment using flow and speed data: exploiting the advantage of distributed computing in derivative-free optimization algorithms," *Transportation Research Procedia*, vol. 27, pp. 993–1000, 2017.

[14] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The Computer Journal*, vol. 7, no. 4, pp. 308–313, 1965.

[15] G. Gentile and L. Meschini, "Using dynamic assignment models for real-time traffic forecast on large urban networks," in *Proceedings of the 2nd International Conference on Models and Technologies for Intelligent Transportation Systems, Leuven, Belgium*, 2011.

[16] G. Gentile, L. Meschini, and N. Papola, "Spillback congestion in dynamic traffic assignment: a macroscopic flow model with time-varying bottlenecks," *Transportation Research Part B: Methodological*, vol. 41, no. 10, pp. 1114–1138, 2007.

[17] N. Hansen, S. D. Müller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es)," *Evolutionary Computation*, vol. 11, no. 1, pp. 1–18, 2003.