Fast Data-Centric Optimization of Nonlinear Dynamic Flows on Network System Suited for Big-Data and Extreme Computing

Wataru Sakurai, Tsuyoshi Ichimura, Kohei Fujita, and Lalith Wijerathne
Earthquake Research Institute and Department of Civil Engineering, The University of Tokyo, Tokyo, Japan
Email: {sakurai-wa, ichimura, fujita, lalith}@eri.u-tokyo.ac.jp

Muneo Hori
Research Institute for Value-Added-Information Generation, Japan Agency for Marine-Earth Science and Technology, Yokohama, Japan
Email: horimune@jamstec.go.jp

***Abstract*—The network optimization problem, which is an optimization problem for a discrete system as a graph structure consisting of nodes and links, has been applied to several fields. Network optimization problems may be more difficult depending on the network scale and problem setting. In contrast, Big-Data & Extreme Computing (BDEC) is a method to process a large amount of data, such as observation data acquired by technologies such as 5G and IoT, by pouring them into abundant computing resources using high-speed cloud computing. In recent years, BDEC environments have become more common. Based on these considerations, this study aims to develop a network optimization method that combines AI and simulations to obtain a fast solution that is effective in BDEC and also to conduct basic studies on the effectiveness of the method by applying it to actual network optimization problems. Therefore, the method was applied to the inverse estimation problem, confirming its effectiveness. For the OD traffic estimation, heuristic solutions at high speed were obtained. Future developments comprise the inclusion of a reliable forward analysis to enhance the accuracy of the system reproduction, improvements in the sampling method and use of hyperparameter tuning method.**

***Index Terms*—optimization, deep learning, data-centric supercomputing, high-performance computing**

I. Introduction

Mathematical optimization problems, which require parameters to minimize or maximize an objective function such as cost formulated in a system described by a mathematical model, are important to support decision making in various situations in the real world. Particularly, the network optimization problem, which is an optimization problem for a discrete system as a graph structure consisting of nodes and links, has been applied to various fields. In fields such as transportation and logistics [1], optimization problems are solved by considering the spatial location as a graph structure and defining an objective function which can be expressed mathematically in the system as in the optimization problem of maximizing sales by the location of stores [2]. Network optimization has been applied to a wide range of fields other than transportation, such as echo state networks [3] and the configuration of communication paths in wireless networks [4]. The inverse estimation problem, in which the state parameters representing the internal state are estimated from the observed data, can also be solved as an optimization problem. For this, the error between the observed data and the estimated value of the observed data calculated by the estimated input parameters is taken as the objective function and minimized. An example of an inverse estimation problem that can be solved as a network optimization problem is the OD inverse estimation [5], which is the problem of estimating the traffic volume for each origin and destination in the field of transportation. Furthermore, for optimization problems related to real phenomena, such as the phenomenon of traffic flow into a network, a more realistic solution can be obtained by coupling with simulation [6]. Network optimization problems may be more difficult depending on the scale of the network and the problem setting. Examples include cases in which the phenomenon being treated exhibits strong nonlinearity or a time delay occurs in the system. Therefore, the problem may become one of the so-called NP-hard class problems, such as the traveling salesman problem, which cannot be solved in a realistic time due to the explosive computational complexity. In contrast, Big-Data & Extreme Computing (BDEC) [7] is a method to process large amounts of data, such as observation data acquired by technologies such as 5G and IoT, by pouring them into abundant computing resources using high-speed cloud computing. In recent years, the BDEC environment is becoming increasingly popular. This environment is highly effective when a large amount of data patterns is assigned to a large number of computational nodes on a computer. The computational nodes process the data independently without communication, which is highly compatible with some data science fields such as machine

Manuscript received July 22, 2021; revised December 27, 2021.

doi: 10.12720/jait.13.2.186-191

learning; thus, it is expected to solve various problems using a data-centric approach. As the field of data science attracts more attention, there is a growing need to construct analysis methods that can effectively use BDEC to obtain solutions at high speed. Based on these considerations, this study aims to develop a network optimization method that combines AI and simulation to obtain a fast solution, which is effective in BDEC, and also to conduct basic studies on the effectiveness of the method by applying it to actual network optimization problems.

II. METHOD

In this section, the proposed method is explained using a general network inverse estimation problem as an example with reference to a previous study [8].

A. Problem

The problem of determining the number of resources for each path of the resource inflow is considered to minimize the cost of resources passing through the links when resources flow into and out of the network. In this case, the objective function is the sum of the costs of all links in the network. Moreover, the optimization problem is to determine the resource inflow amount for each resource inflow path minimizing the sum of costs. This optimization problem can be used for the inverse estimation problem of unobservable parameters using actual observation data. Specifically, the optimization problem is used to minimize the cost of each link at each time, which is defined as the error between "the time series of the resource at each link assumed when the resource flows with a certain allocation" and "the time series of each resource actually observed in the network.

B. Proposed Method

In the BDEC environment, a large amount of data can be input into the computer at once and processed. In this study, a method that is effective in this environment is proposed. A surrogate model is introduced, and the processing of multiple input patterns is parallelized. The basic approach is to reduce the computational cost by reducing the sequential processing, and also the complexity of the forward analysis by reducing the computation time through parallelization and introduction of the surrogate model. The optimization problem is used to define and minimize the function Err, which represents the error of the link reference solution in the observed network. Thus, Err is defined by the following equation;

$$Err = \Sigma_{l=1}^{n_{obs}} Err_l^{local} \tag{1}$$

$$\text{where} \quad Err_l^{local} = \sqrt{\frac{\Sigma_{i_t=1}^{n_t}(y_{l,i_t}^{ref} - y_{l,i_t})^2}{\Sigma_{i_t=1}^{n_t}(y_{l,i_t}^{ref})^2}}$$

where y_{l,i_t}^{ref} and y_{l,i_t} represent the observed and simulated values computed by the forward analysis of the resources at the i_t-th time step ($i_t = 1 - n_t$) and l-th observation link ($l = 1 - n_{obs}$), which can be computed using the time history x_{i_t} of the input resources as time-series data. The input parameters reducing the Err defined in the present study are assumed to be the appropriate parameters for the observed data and search for the parameter x that minimizes Err. In this method, a surrogate model, which is not a regression model of Err but a classification model based on the value of Err, is developed using a Deep Neural Network (DNN). This surrogate model estimates the class according to the error magnitude from the input, not the error itself, and can proceed with the evaluation using a model simpler than the regression model. This model is expected to provide a fast and heuristic solution. The $n_1(\gg n_0)$ parameters were input to the DNN constructed as described, and the n_0 input parameters were extracted, those with small $\Sigma_{l=1}^{n_{obs}} L_l$ values in which each L_l value is lower than the standard value. The input parameter that gives the smallest Err value computed by forward analysis can be selected as the solution x^{best} that fits the observed data. Therefore, the DNN constructed as a surrogate model can select a solution candidate from a large number of solution candidates. This significantly reduces the number of forward analyses to be performed, allowing the evaluation of a large amount of data with low computational costs. The algorithm summarizing the described above is demonstrated below.

TABLE I. AI-BASED OPTIMIZATION METHODS SUITED FOR THE ENVIRONMENT

Set up $datasetA$	generate n_0 samples based on x^{base} (past actual values), perform forward analysis, and compute y_{l,i_t}
Compute Err	compute Err from y_{l,i_t}^{ref}, y_{l,i_t}
Construct DNNs	compute L_l (= 0 − 9) for each observation link according to the value of Err
Generate samples	generate $n_1(\gg n_0)$ samples based on x^{base}
Evaluate by DNN Set up $datasetB$	extract n_0 samples whose total evaluation value is small and each L_l is less than the standard value among the samples input to DNNs from $n_1(\gg n_0)$ samples and perform forward analysis
Update x^{best}	select the parameter with the smallest Err in $datasetB$ as x^{best}
Update x^{base}	update the value of x^{base} to x^{best}

III. APPLICATION EXAMPLE

In this chapter, the effectiveness of the proposed method is validated by application to OD inverse estimation [5].

A. Definition of the Considered Problem

In this study, the problem of inverse OD estimation [5], which is used to estimate the breakdown of the time history traffic volume flowing into the network from each origin by destination based on the time history traffic observed in the network, is considered. Real-time OD inverse estimation can be used in combination with traffic simulation to predict future traffic volumes in a network and is expected to be applied to real-time decision making on measures against traffic congestion, such as road pricing. Several methods have been proposed for inverse OD estimation (e.g., [9], such as Kalman filter [10],

particle filter [11] combined with simulation, machine learning [12]). The proposed method is based on the premise that the BDEC environment can be used with a large number of computational nodes in parallel. Moreover, it aims to achieve an ultra-fast network optimization by developing a new process for network minimization of the difference between observed data and simulations, which is suitable for parallel processing while also incorporating DNNs; this is a departure from previous research.

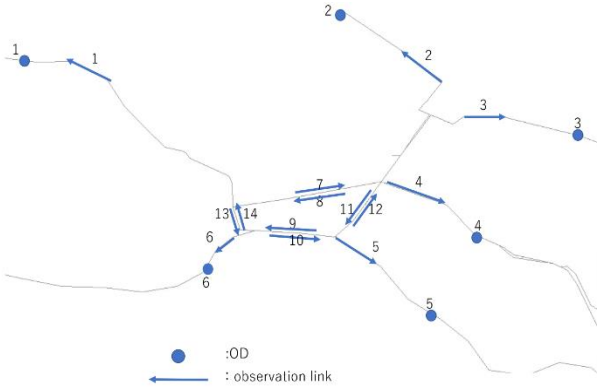

Figure 1. Observation links and OD in the target network.

Figure 1 shows the target network used in this study, which imitates a major trunk line in a city near Tokyo, where tourism congestion is a problem [13]. The network consists of 59 nodes and 240 links, in which each link is divided into upper and lower lines, achieving a directed graph network. For simplicity, the situations in which a vehicle OD is inside the network or a vehicle entering the network OD at the same location as its origin are not considered. Six origin/destination nodes are considered, and also the vehicles entering the network at any of these six nodes as origin and exiting the network at the other five nodes as destination. The aim of this study is to estimate the time history traffic volume of each OD pair that matches the observed data using network optimization. Thus, a problem in which time history traffic volume is observed at 14 observation links is considered.

B. Results

The optimization was performed according to the flow described in Table I, and the estimation of x_{i,j,i_t}^{ref} was performed from y_{l,i_t}^{ref}. Initially, n_0 = 1000 OD traffic volumes x_{i,j,i_t} were generated by adding random numbers of uniform distribution (-x_{i,j,i_t}^{base}/2, x_{i,j,i_t}^{base}/2) at each time to the past actual values x_{i,j,i_t}^{base}. The total number of agents entering the network at each inflow point $\Sigma_{j=1,i\neq j}^{6} x_{i,j,i_t}$ was adjusted; thus, no changes are observed at each time to be consistent with past results. Forward analysis was performed on each x_{i,j,i_t} in $datasetA$ and y_{l,i_t} was obtained at each observation point. The Err obtained by y_{l,i_t} for each x_{i,j,i_t} is shown in Fig. 2. Because the number of parameters is 90 ($i = 1-6, j = 1-6, i \neq j, i_t = 1-3$) and each parameter exhibits a wide range of motion, a good solution cannot be obtained with approximately 1000 samples. In fact, the x_{i,j,i_t} for the case in which Err displayed the minimum value, shown in Fig. 3, was considerably different from the reference solution (Fig. 4). This is because when looking at the Err_l^{local} (Table II), although the value of Err was small, the estimation performance of x_{i,j,i_t} was degraded due to the large errors in some Err_l^{local}. The Err_l^{local} for all x_{i,j,i_t} in $datasetA$ did not generate uniformly small candidates, indicating that good solutions cannot be selected using 1000 samples. The quality of x_{i,j,i_t} in $datasetA$ was evaluated by the error shown in (1) and the results are shown in Fig. 2.

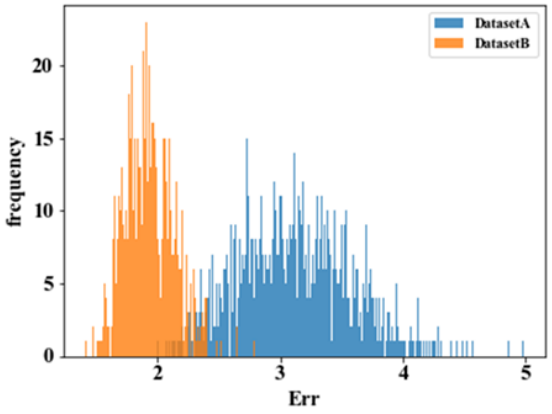

Figure 2. Comparison of the Err distribution for $datasetA$ and $datasetB$.

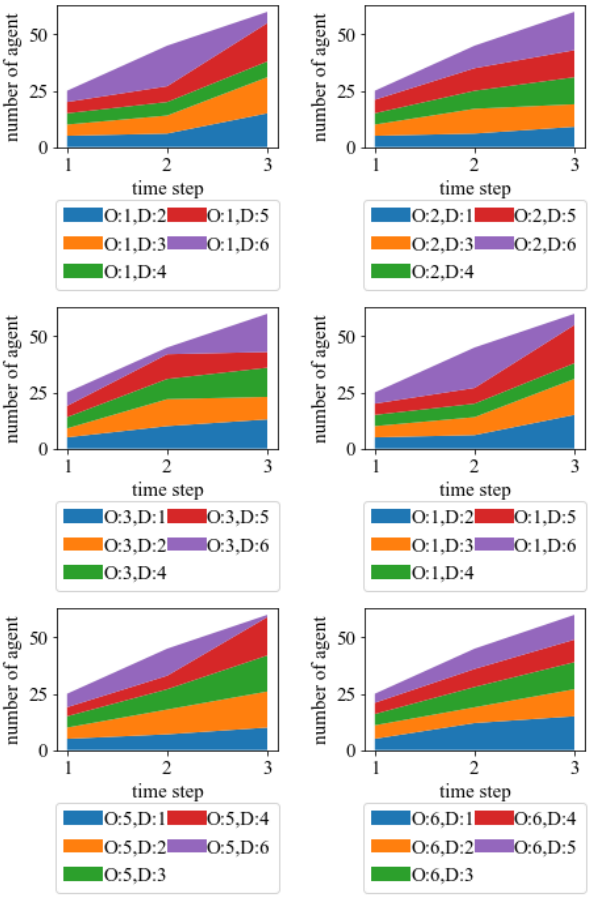

Figure 3. OD traffic volume in the optimum solution of random sampling.

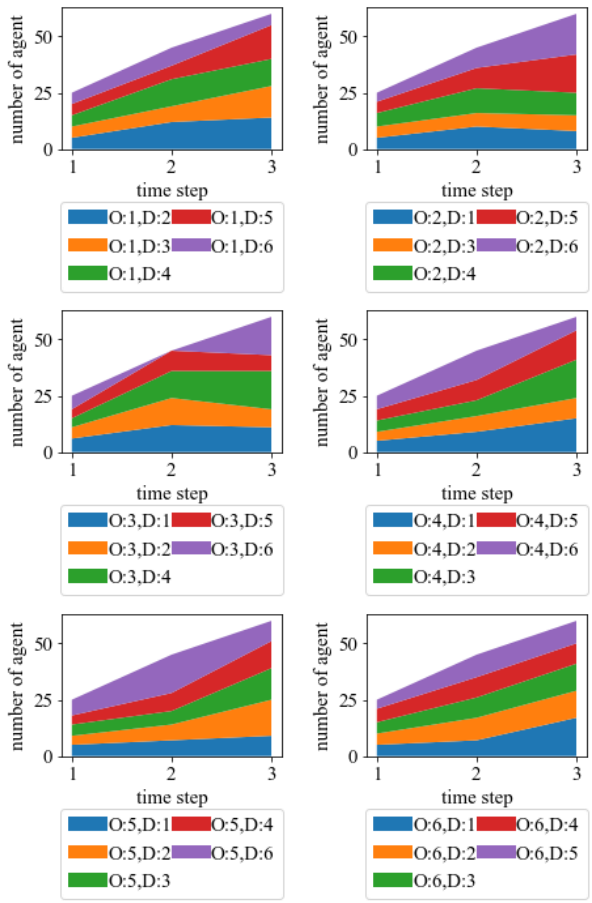

Figure 4. Reference OD traffic volume.

Although $datasetA$ was generated based on the actual values, the candidate solutions close to the optimal values were not generated with a sampling of 1000 owing to a large number of parameters and their large range of motion as described above.

Next, the results of this forward analysis were used to construct a DNN by DL. The DNN was divided into 10 intervals between the maximum and minimum values of Err_l^{local} computed at each observation point. In a decreasing Err_l^{local} value order, the samples in each interval were ranked $L_l(=0-9)$, and $n_0(=1000)$ datasets were created for training. The input of each DNN is 3D data and the hourly OD traffic is converted to 1D data x. The relationship between x and L_l was used for training. In the form of learning the relationship between x and L_l, a fully-connected DNN (the activation functions were ReLU (middle layer) and cross-entropy (output layer), and the number of units and layers were 60 and 4, respectively) was constructed for general class estimation by Pytorch. Adam was used as the optimizer (learning rate 0.001), L_1 regularization (coefficients $\lambda = 10^{-4}$) to reject parameters spatially uncorrelated with the errors of the observed links, 1000 epochs, and a batch size of 200. The correctness of the validation data for DNN_l constructed for each observation link with $n_{obs}(=14)$ is shown in Table III. In this table, in addition to the usual correct answer rate, the correct answer rate is described when the estimated value is allowed to be one level before or after the correct answer. Because the grade is a value that divides the actual computed error, it is sufficient to predict the approximate value rather than estimate it precisely. Therefore, the DNNs can be used if the accuracy when the estimated value is allowed to be ±1 from the correct answer is high. As shown in the table, the accuracy was high when the value ±1 from correct answer was also allowed; therefore, the optimization proceeded using the DNN_l.

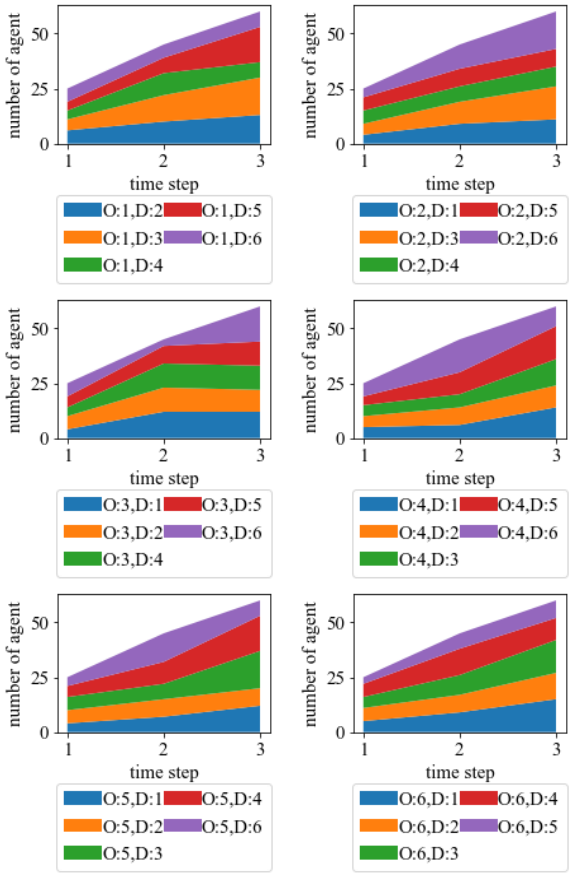

Figure 5. OD traffic volume in the optimum solution by the proposed method.

As in $datasetA$, n_1 (= 10^9) OD traffic volume x_{i,j,i_t}, which was generated to be consistent with the past results, was input into DNN_l and $L_l$was calculated for each x_{i,j,i_t}. By evaluating $\Sigma_{l=1}^{n_{obs}} L_l$, x_{i,j,i_t} with small error for each link as a whole would be selected and also 1000 x_{i,j,i_t} with small $\Sigma_{l=1}^{n_{obs}} L_l$ and $L_l < 3$. The error Err of the entire network was evaluated using (1) with y_{l,i_t} obtained by forward analysis for each x_{i,j,i_t} in $datasetB$. Compared with the distribution of Err obtained by the initial random sampling of $datasetA$, x_{i,j,i_t} was selected as $datasetB$ because of the constraint of the solution space via DNN and the error with the observed data was small. The minimum value of Err in $datasetB$ is 1.42 and the x_{i,j,i_t} in this case (Fig. 5) is in good agreement with the reference solution (Fig. 4). The Err_l^{local} (Table II) exhibited a smaller value as Err, unlikely that of the best $datasetA$ solution, and the error was small for all Err_l^{local}, indicating that the estimation performance of x_{i,j,i_t} was

high. The quality of x_{i,j,i_t} in $datasetB$ was evaluated by the errors shown in (2).

$$Errinput = \Sigma_{i=1}^{6} Errinput_i^{local},$$

$$where\ Errinput_i^{local} = \sqrt{\frac{\Sigma_{i_t=1}^{3}\Sigma_{j=1,i\neq j}^{6}\left(x_{i,j}^{ref,i_t} - x_{i,j}^{i_t}\right)^2}{\Sigma_{i_t=1}^{3}\Sigma_{j=1,i\neq j}^{6}\left(x_{i,j}^{ref,i_t}\right)^2}} \quad (2)$$

TABLE II. VALUE OF Err_l^{local} BY RANDOM SAMPLING AND PROPOSED METHOD

l	$\boldsymbol{Err_l^{local}}$ (Random sampling)	Err_l^{local} (Proposed method)
1	0.238	0.049
2	0.065	0.08
3	0.196	0.057
4	0.094	0.019
5	0.17	0.07
6	0.17	0.014
7	0.04	0.054
8	0.04	0.064
9	0.11	0.096
10	0.073	0.051
11	0.42	0.11
12	0.19	0.33
13	0.11	0.19
14	0.091	0.095

TABLE III. DNN ACCURACY

DNN	Accuracy (%)	Accuracy (%, ±1 class tolerance)
1	60	99
2	40	90.5
3	49	88.5
4	65.5	97.5
5	40.5	84.5
6	41.5	92.5
7	68.5	93.5
8	63.5	98
9	41.5	88.5
10	48	93.5
11	62.5	98
12	50	92
13	50	92
14	49	91.5

Compared with $datasetA$, this optimization process via DNN, which is trained on the structure of the problem, allows us to set a solution candidate closer to the reference solution as $datasetB$. Compared to $datasetA$, $datasetB$ is closer to the reference solution by going through this optimization process via a DNN that has learned the problem structure.

However, in the BDEC environment, the number of DNNs does not affect the execution time of the optimization because the construction of DNNs and their evaluation can be performed in parallel. Therefore, similar to the method described in the previous chapter, the optimization can be completed within a few minutes, which is essentially the computation time of several forward analyses and the computation time of DNN alone. The optimization process with a similar setup was performed (1 iteration of the process is accomplished).

Conversely, optimization using simulated annealing [14] as a conventional method was attempted, and 1000 iterations (real-time for 1000 forward analyses) were used for the optimization. A comparison of $Errinput_i^{local}$ of x_{i,j,i_t} obtained by the proposed method, annealing method, and random sampling is shown in Table IV.

By simulated annealing without appropriate modifications, a solution with lower accuracy than the best solution in random sampling was obtained. Conversely, the proposed method could obtain heuristic solutions with better accuracy than random sampling using a simple surrogate model. Finally, the time for computation was evaluated. The forward analysis used in this evaluation is an agent analysis on a network, which tends to present non-uniform granularity and is difficult to ensure scalability for a large number of computation nodes. When using one Intel Xeon E5-2690 CPU, it takes nearly 24 seconds to compute one case. Furthermore, it takes approximately 41 seconds to build one DNN with one Nvidia V100 GPU. It takes almost 24 seconds to evaluate 14 DNNs for 10^5 samples on one Nvidia V100 GPU. Based on the above, the computation time of the proposed method can be evaluated on the BDEC environment (e.g., 256 Intel Xeon ® E5-2690 and 512 Nvidia V100 GPUs). The actual computation time was approximately 557 seconds in total: one time of forward analysis (24 seconds), DNN construction (41 seconds), DNN application (468 seconds), and one time of forward analysis (24 seconds). Conversely, because the annealing process was performed 1000 times sequentially, the computation time was 1000×24 seconds = 24000 seconds, indicating that the proposed method achieves fast processing using the BDEC environment as the concept shown in Fig. 6.

TABLE IV. VALUE OF $\text{Errinput}_i^{\text{local}}$ FOR THE BEST SOLUTION OBTAINED BY EACH METHOD

i	1	2	3	4	5	6
Random Sampling	0.40	0.27	0.52	0.18	0.30	0.27
Simulated Annealing	1.30	0.57	0.76	0.82	0.87	0.89
Proposed Method	0.17	0.14	0.20	0.18	0.40	0.18

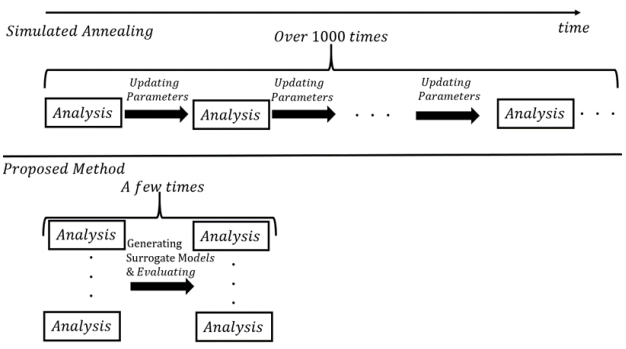

Figure 6. Concept of computation time for optimization using simulated annealing and the proposed method.

IV. CONCLUSION

In this study, as a basic evaluation for proposing a data-centric network optimization method based on BDEC, the method was applied to the OD inverse estimation problem, confirming its effectiveness. For the OD traffic estimation,

which is a problem with time delay and nonlinearity, a surrogate model using deep learning was used instead of forward analysis, and heuristic solutions at high speed were obtained. Future developments comprise the inclusion of a reliable forward analysis to enhance the accuracy of the system reproduction, in addition to improvements in the sampling method. For the former, a sophisticated sequential analysis (e.g., [15]) was considered. For the latter, the accuracy of DNNs was improved by effective sampling using the design of experiments, and more accurate solutions were achieved by inputting samples to DNNs based on the sampling results of training data. Furthermore, the use of hyperparameter tuning methods such as Bayesian optimization in the construction of surrogate models would result in faster and more accurate optimization.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

Wataru Sakurai conducted the simulations and analyzed the data; Wataru Sakurai, Tsuyoshi Ichimura, Kohei Fujita, Lalith Wijerathne and Muneo Hori developed the proposed method and wrote the paper; all authors approved the final version.

ACKNOWLEDGMENT

We acknowledge support from CART project and Japan Society for the Promotion of Science (18K18873). Road network data is provided by CART project.

REFERENCES

[1] T. Yamada, T. Nakamura, T. Yokoyama, and E. Taniguchi, "A discrete optimisation model for investigating the design and vulnerability of transport network considering supply chain performance," *Journal of Japan Society of Civil Engineers, Series D3 (Infrastructure Planning and Management)*, vol. 68, no. 4, pp. 272-284, 2012.

[2] M. Sugiyama, Y. Honma, and Y. Munemasa, "Commerical equibrium distribution considering locational cost and facility by mathematical optimization method," *The Architectural Institute of Japan's Journal of Architecture and Planning*, vol. 83, no. 745, pp. 427-434, 2018.

[3] J. Liu, T. Sun, Y. Luo, S. Yang, Y. Cao, and J. Zhai, "Echo state network optimization using binary grey wolf algorithm," *Journal of Neurocomputing*, vol. 385, pp. 310-318, 2020.

[4] L. Liu, B. Yin, S. Zhang, X. Cao, and Y. Cheng, "Deep learning meets wireless network optimization: Identify critical links," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 1, pp. 167-180, 2020.

[5] Y. Shimakawa and S. Kashima, "A method structuring a road traffic data for traffic assignment," *Theory and Applications of GIS*, vol. 17, no. 2, pp. 69-75, 2009.

[6] M. S. Shishvan and J. Benndorf, "Simulation-based optimization approach for material dispatching in continuous mining systems," *European Journal of Operational Research*, vol. 275, no. 3, pp. 1108-1125, 2019.

[7] M. Asch, *et al.*, "Big data and extreme-scale computing: Pathways to convergence-toward a shaping strategy for a future software and data ecosystem for scientific inquiry," *The International Journal of High Performance Computing Applications*, vol. 32, no. 4, pp. 435-479, 2018.

[8] T. Ichimura, K. Fujita, T. Yamaguchi, M. Hori, L. Wijerathne, and N. Ueda, "Fast multi-step optimization with deep learning for data-centric supercomputing," in *Proc. HP3C*, 2020.

[9] K. Abe, H. Fujii, and S. Yoshimura, "Origin-destination estimation for microscopic traffic simulator considering intersection capacity," SIG-SAI, pp. 1-6, 2017.

[10] K. Yasui, K. Ikenoue, and H. Takeuchi, "A Kalman filter for quasi-dynamic o-d flow estimation/updating," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 11, pp. 3604-3612, 2018.

[11] T. Nakamura, Y. Sekimoto, T. Usui, and R. Shibasaki, "Estimation of people flow in an urban area using particle filter," *Journal of Japan Society of Civil Engineers, Series D3 (Infrastructure Planning and Management)*, vol. 69, no. 3, pp. 227-236, 2013.

[12] J. Ou, J. Lu, J. Xia, C. An, and Z. Lu, "Learn, assign, and search: Real-time estimation of dynamic origin-destination flows using machine learning algorithms," *IEEE Access*, vol. 7, pp. 26967-26983, 2019.

[13] Ministry of Land, Infrastructure, Transport and Tourism. (2019). Japan, on traffic conditions in Kamakura. [Online]. Available: https://www.ktr.mlit.go.jp/yokohama/06data/plan/kamakura/03/01.pdf

[14] M. M. Mafarja and S. Mirjalili, "Hybrid whale optimization algorithm with simulated annealing for feature selection," *Journal of Neurocomputing*, vol. 260, pp. 302-312, 2017.

[15] T. Morinuma, R. Raymond, T. Suzumura, R. Takahasi, T. Ide, T. Osogami, T. Imamichi, and H. Mizuta. (2012). IBM mega traffic simulator. [Online]. Available: http://domino.research.ibm.com/library/cyberdig.nsf/papers/7BE8B8BF8CAE073A85257B1900190ACE

Wataru Sakurai is a Master student in Department of Civil Engineering, the University of Tokyo, Japan. He received his B.Eng. degree in the area of Civil Engineering from the University of Tokyo, Japan in 2019. His current research of interest includes Civil Engineering, High Performance Computing, and also Computational Science.