2016 International Symposium on Nonlinear Theory and Its Applications,
NOLTA2016, Yugawara, Japan, November 27th-30th, 2016

Speeding up of the Traffic Congestion Mitigation by Stochastic Optimization in Deep Learning

Shinnnosuke NAKAMURA†1 Takumi UEMURA†2 Gou KOUTAKI†3 Keiichi UCHIMURA†3

†1 Graduate School of Engineering, Sojo Uniersity
†2 Faculty of the Computer Information Science, Sojo University
4-22-1 Ikeda, Nishi-ku Kumamoto city, Kumamoto Prefecture 860-0082, Japan
†3 Graduate School of Science and Technology, Kumamoto University
2-39-1 Kurokami, Chuo-ku Kumamoto city, Kumamoto Prefecture 860-8555, Japan
Email: g1511M03@m.sojo-u.ac.jp, t_uemura@cis.sojo-u.ac.jp,
koutaki@cs.kumamoto-u.ac.jp, uchimura@cs.kumamoto-u.ac.jp

Abstract: In recent years, many researchers have become interested in methods for mitigating traffic congestion by optimizing traffic signal parameters. To mitigate traffic congestion over a widespread area, a method using an advanced genetic algorithm and a traffic simulator has been proposed (Nishihara, T., et al., "The Verification with Real-World Road Network on Optimization of Traffic Signal Parameters using Multi-Element Genetic Algorithms", ITS World Congress, 2012). However, this method consumes considerable time when simulating traffic flow. This paper proposes a method that reduces the processing time of the simulator by using a neural network.

1. Introduction

In recent years, traffic congestion has caused several economic and environmental issues. The Ministry of Land, Infrastructure, and Transportation (MLIT) in Japan has estimated the economic loss caused by congestion to be 12 trillion yen [1]. In addition, it causes a loss of 30 h per year for each person. Moreover, it generates environmental pollution because of the greenhouse gases that are emitted from idling cars. Therefore, the mitigation of traffic congestion has become significant in recent years.

Nishihara et al. [2] addressed this problem by proposing a method that uses both a multiple-element genetic algorithm (ME-GA) and a traffic simulator. However, this method is slow when simulating a road network such as the one shown in Figure 1.

To shorten the processing time, we propose a method that replaces the simulator with a pre-trained neural network (NN). With this replacement, the prediction accuracy has a large effect on mitigation performance. Therefore, we propose a machine learning method that uses deep learning to train the NN.

2. Previous methods

The GreenWave method is an optimization method for traffic signals that is operated in actual service [3, 4]. GreenWave optimizes the traffic signals so as not to stop cars in a certain road section. However, it causes traffic congestion at the interface of this road section.

Xu et al. proposed the GreenSwirl method, which improves on GreenWave. GreenSwirl applies GreenWave to a loop road, and uses a path finding algorithm to allow cars to travel through the loop road optimally. GreenSwirl achieves better mitigation of traffic congestion than GreenWave [5]. However, the design of the loop road and the settings of the road signals are made manually, and thus, this method faces several problems when applied to widespread areas.

Nishihara et al. proposed a method that is constructed from a genetic algorithm (GA) and an evaluation to optimize the traffic signal parameters over a widespread area. Figure 2 shows the processing flow in Nishihara et al.'s method [2]. This method uses the optimization algorithm of multidimensional parameters to optimize the traffic signal parameters over a widespread area. Nishihara et al. used ME-GA to optimize multidimensional parameters in order to obtain generations with better genes. This method comprises the following five steps.

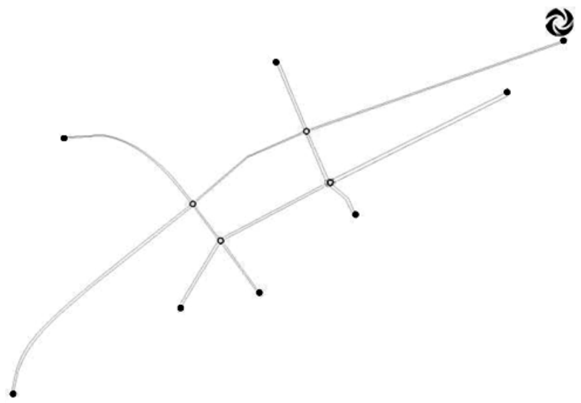
Figure 1 Road network modeled on the Ooe-Toroku area in Kumamoto city (Reproduced from [2]).

Step 1: Set the actual road network and road signal parameters in the traffic flow simulator.
Step 2: Generate the initial individuals in ME-GA. ME-GA is initialized from the traffic congestion evaluation values derived from the traffic flow simulator.
Step 3: Derive better road signal parameters by calculating new generations in ME-GA.
Step 4: Derive the traffic congestion evaluation values of each road signal parameter from the traffic flow simulator.
Step 5: Apply each traffic congestion evaluation value to the ME-GA and iterate steps 3 to 5 until a certain number of generations has been calculated.

This method iterates the operation of the traffic flow simulator to optimize the traffic signal parameters; therefore, the processing time of the traffic flow simulator significantly affects the total time of the system. Nishihara et al.'s method consumes 19 hours to optimize the traffic signal parameters in the simple road network shown in Figure 1. The processing time of the traffic flow simulator needs to be shortened in order for this method to be operated in actual service.

3. Proposed method

3.1. Outline

In this paper, to shorten the processing time, we propose a traffic signal optimization method that replaces the traffic simulator with a pre-trained NN, in order to predict with high accuracy using deep learning.

The learning methods of deep learning can be classified into the following four types: autoencoder (AE) [6], restricted Boltzmann machine [7, 8], convolutional NN [9-11], and recurrent NN [12, 13]. Among them, we choose AE because it is the most versatile and has the same structure as that of a general NN. In addition, we use a stack denoising AE (SDA) because it is generally regarded as the method that obtains the best performance for AE.

We use a stochastic optimization approach, called Adam, to optimize the network parameters for deep learning. In recent years, stochastic optimization has become a major method in deep learning that accelerates learning convergence and reduces generalization error. Adam is regarded as learning more quickly than other conventional parameter optimization methods such as AdaGrad or RMSProp [14].

Figure 3 shows the process flow of the proposed method, which comprises two processes. The first is the learning process for predicting the output value of the traffic flow simulator. This process samples the learning data from the traffic flow simulator and extracts the features for the NN from the data via unsupervised learning using SDA. After feature extraction, the NN

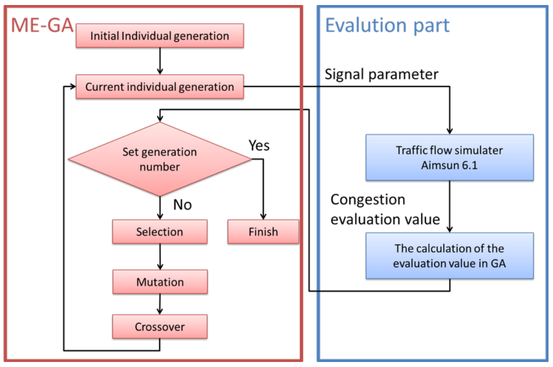
ME-GA
Initial Individual generation
Current individual generation
Set generation number
Yes
No
Selection
Finish
Mutation
Crossover
Evalution part
Signal parameter
Traffic flow simulater Aimsun 6.1
Congestion evaluation value
The calculation of the evaluation value in GA

Figure 2 Process flow of Nishihara et al.'s method

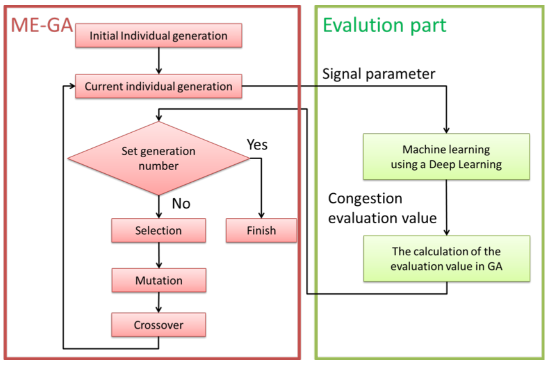
ME-GA
Initial Individual generation
Current individual generation
Set generation number
Yes
No
Selection
Finish
Mutation
Crossover
Evalution part
Signal parameter
Machine learning using a Deep Learning
Congestion evaluation value
The calculation of the evaluation value in GA

Figure 3 Process flow of the proposed method

predicts the output value via supervised learning using the back propagation method. The second process optimizes the traffic signal parameters via a trained NN and ME-GA. ME-GA better predicts the traffic signal parameters by using the congestion evaluation values that are outputted from the NN. The traffic signal parameters are optimized by iterating the optimization process.

3.2. Properties of NN

Supervised learning is performed after feature extraction by deep learning using the back propagation method. In this study, the training data are normalized as a real number from [0, 1]. We use a sigmoid function for the activation function, as shown in the following equation.

$$sigmoid(z) = \frac{1}{1+e^{\alpha z}} \qquad (1)$$

Here, Z is the output value of the element and α is the gain.

The error value E in the output layer is derived from the mean square error, as shown in following equation.

$$E = \frac{1}{2}\sum_k (T_k - O_k)^2 \qquad (2)$$

Here, T_k shows the average value of the training data and O_k shows the k th sample value of the output.

Table 1 shows the Adam parameter optimization procedure. Parameter θ is the parameter to be determined, *f* is the objective function, g is a gradient derived from *f*, and t is the number of iterations for learning. In addition, the recommended parameters for Adam are shown at the top of Table 1.

4. Experiment

We carried out two experiments to confirm the effectiveness of the proposed method. Experiment 1 compares the congestion mitigation performances of the proposed method, Nishihara et al.'s method, and the actual measured traffic signal parameters. Experiment 2 evaluates the calculation speed of the proposed method and Nishihara et al.'s method.

4.1. Experimental setup

Table 2 shows the specifications of the computer used for each method.

4.1.1. Properties of the traffic flow simulator

The traffic flow simulator used in the evaluation is Aimsun 6.1. The road network in these experiments is the same as that used to evaluate Nishihara et al.'s method (shown in Figure 1) [2]. In addition, the traffic volume and agent of each vehicle in Aimsun 6.1 are set the same as those for Nishihara et al.'s method.

The traffic signal parameters consist of Cycle, Split, and Offset signals. These are set for each intersection.

Aimsun 6.1 outputs WaitOut, Inside, and GoneOut data. WaitOut is the number of vehicles that cannot enter the road network because of traffic jams and are located outside of the road network to be optimized. Further, Inside is the number of vehicles in the road network to be optimized. Finally, GoneOut is the number of vehicles that have left the road network to be optimized.

4.1.2. Properties of the NN

In the deep learning for these experiments, the NN pre-learns using SDA before it is trained using error back propagation. The NN does not include an output layer because of the unsupervised learning in pre-learning.

We construct an NN for every congestion evaluation value because the congestion evaluation values do not affect each other.

Mini-batch learning is used as the learning method. To consider versatility, each number of iterations for learning is set to minimize the differences between generalization and training errors in the learning process.

4.1.3. Properties of ME-GA

The ME-GA parameters are set such that the number of generations is 500 and the population is 300. The other settings are the same as those in Nishihara et al.'s method. (The number of generations and population of Nishihara et al.'s method are 100 and 75, respectively). The evaluation value F for ME-GA is derived from the following two equations.

$$F = exp\left(\frac{V_{wo}}{C_w}\right) + exp\left(\frac{V_{in}}{C_i}\right) + exp\left(\frac{C}{C_d}\right) \tag{3}$$

$$C = \frac{t_{delay}}{TTD} \tag{4}$$

Here, V_{wo} is WaitOut and V_{in} is Inside. Further, t_{delay} (DelayTime) is the average difference between the actual running time and the ideal running time; the total travel distance (*TTD)* indicates the sum of the travel distance of all vehicles in the simulation; and C_w, C_i, and C_d are weights for the congestion evaluation value, where C_w = 100, C_i = 500, and C_d = 500.

Table 1 Algorithm of Adam

Adam : α = 0.001, β_1 = 0.9, β_2 = 0.999, ε = 10^{-8}, $\beta_1{}^t$ and $\beta_2{}^t$ are *t*th powers of β_1 and β_2, respectively.

```
$m_0 \leftarrow 0$
$v_0 \leftarrow 0$
$t \leftarrow 0$
$while$ learn number $do$
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$
    $\hat{m}_t \leftarrow m_t \,/\, (1 - \beta_1^t)$
    $\hat{v}_t \leftarrow v_t \,/\, (1 - \beta_2^t)$
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t/(\sqrt{\hat{v}_t} + \varepsilon)$
$end\ while$
```

4.2. Experimental results

Table 4 shows the results of the first experiment. Here, low values for WaitOut and Inside indicate that the traffic situation is good. A large value for GoneOut also indicates that the traffic situation is good.

The results of this experiment confirm that the traffic congestion resulting from Nishihara et al.'s method and the proposed method are equal. The results also show that the proposed method performs worse than Nishihara et al.'s method when the number of generations and population are set to be the same as those in Nishihara et al.'s method. Therefore, the method for predicting the output values of the traffic flow simulator would probably be improved by reconsidering the samples used for pre-training.

Table 5 shows the results of Experiment 2. The processing time in the proposed method is about 15 minutes, while that in Nishihara et al.'s method is about 19 hours. Therefore our proposed method achieves a processing time that is 99% shorter than that in Nishihara's method.

Table 2 Computer specifications for each method

Method	CPU	Memory	OS
Nishihara et al.'s	Intel i7 870 2.93 GHz	8 GB	Windows 7 Professional 64 bit
Proposed	Intel i5 4690K 3.9 GHz	8 GB	Ubuntu 15.04 64 bit

Table 3 Iteration times needed to derive each evaluation in machine learning

Evaluation	Iteration time†
WaitOut	26
Inside	20
TTD	60
DelayTime	20

† This is the iteration times with mini-batch learning

Table 4 Evaluation values for each method

Method	WaitOut	Inside	GoneOut
Nishihara et al.'s	324	1003	4111
Proposed	326	960	4293
Actual measured	859	1122	3569

Table 5 Processing times for each method

Method	Processing time
Nishihara et al.'s	19 hour
Proposed	15 minutes

5. Conclusion

In this study, to shorten the processing time while maintaining accuracy, we proposed a traffic signal parameter optimization method that uses a pre-trained NN instead of a traffic flow simulator.

The experimental results show that we achieved a similar level of mitigation for traffic congestion and a 99% shorter processing time than Nishihara et al.'s method. However, our method performed worse than theirs when the numbers of generations and individuals were set to the same values as in their method. In addition, the proposed method could optimize the traffic signal parameters in 15 min. However, VICS updates information every 5 min generally [15]. Therefore, the processing time of the system should be shortened to less than 5 min in order to be practical for actual service.

In future work, we plan to apply parallel processing to the calculation of individual evaluations in ME-GA or increase the sampling data in deep learning.

References

[1] Ministry of the Land, Infrastructure and Transportation in Japan, "Performance Management of Road Administration in Japan", http://www.mlit.go.jp/road/ir/ir-perform/h18/07.pdf

[2] Nishihara, T., Wijaya, I.G.P.S., Matsumoto, S., Koutaki, G., Uchimura, K., Sugitani H., and Ishigaki, S., "The Verification with Real-World Road Network on Optimization of Traffic Signal Parameters using Multi-Element Genetic Algorithms", ITS World Congress 2012, AP-00144, 2012.

[3] Warberg, A., Larsen, J., and Jrgensen, R. "Green Wave Trrafic Optimization", A Survey, Informatics and Mathematical Modeling, 2008.

[4] Saki, M. and Nagatani, T., "Transition and Saturation of Traffic Flow Controlled by Traffic Lights, Physica A", Statistical Mechanics and Its Applications, Vol.325, Issue 3-4, pp.531-546, 2003.

[5] Jiaxing X., Akira K., Naoki S., and Minoru I., "GreenSwirl: A Combination Method of Traffic Signal Control and Route Guidance for Reducing Traffic Congestion", Journal of Information Processing, Vol.57, No. 1, pp.66-78, 2016.

[6] Vincent, P., Larochelle, H., Lajoie, L., Bengio, Y., and Manzagol, P.-A., "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion", Journal of Machine Learning Research, Vol. 11, pp. 3371-3408, 2010.

[7] Hinton, G. E., "A Practical Guide to Training Restricted Boltzmann Machines", Technical Report, 2010.

[8] Hinton, G. E., "Training Products of Experts by Minimizing Contrastive Divergence", Neural Computation, Vol. 14, No. 8, pp. 1771-1800, 2002.

[9] Hubel, D. H. and Wiesel, T. N., "Receptive Fields, Binocular Interactions, and Functional Architecture in the Cat's Visual Cortex", Journal of Physiology, Vol. 160, pp. 106-154, 1962.

[10] Fukushima, K. and Miyake, S., "Neocognitron: A New Algorithm for Pattern Recognition Tolerant of Deformations and Shifts in Position", Pattern Recognition, Vol. 12, pp. 455-469, 1982.

[11] Lecun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D., "Backpropagation Applied to handwritten Zip Code Recognition", Neural Computation, Vol. 1, No. 4, pp. 541-551, 1989.

[12] Mkoloc, T., Karafiat, M., Cernocky, J., and Khudanpur, S., "Recurrent Neural Network Base Language Model", In Proc, Interspeech, 2010.

[13] Murphy, K. P. Machine Learning. "A Probabilistic Perspective", MIT Press, 2012.

[14] Diederik, P. K. and Jimmy, L. B., "Adam: A Method for Stochastic Optimization", Proceedings of International Conference on Learning Representations 2015, arXiv:1412.6980v8, 2015.

[15] Vehicle Information and Communication System Center, "VICS", http://www.vics.or.jp/index1.html.