Applying Deep Learning and Reinforcement Learning to Traveling Salesman Problem

Shoma Miki
GS of Science and Engineering
Kansai University
Suita, Osaka, Japan
k154911@kansai-u.ac.jp

Daisuke Yamamoto
GS of Science and Engineering
Kansai University
Suita, Osaka, Japan

Hiroyuki Ebara
GS of Science and Engineering
Kansai University
Suita, Osaka, Japan
ebara@kansai-u.ac.jp

***Abstract*—In this paper, we focus on the traveling salesman problem (TSP), which is one of typical combinatorial optimization problems, and propose algorithms applying deep learning and reinforcement learning. This method is marked by learning the optimal tour as an image using a convolutional neural network, and acquires the Good-Edge Distribution which is the map of edges that could be included in the optimal tour. And it performs neighborhood search by using Good-Edge Value: evaluations of each edge calculated from the distribution. In addition, there are cases where it is not possible to obtain an optimal solution such as large scale instances or other combinatorial optimization problems, so learning by using the best solution instead of the optimal solution is important. Therefore, we also consider learning methods using reinforcement learning. We conduct experiments to examine the performance of these methods, and verify the effectiveness of improving quality of solutions.**

***Index Terms*—Combinatorial Optimization Problem, Traveling Salesman Problem, Deep Learning, Reinforcement Learning, Convolutional Neural Network**

I. Introduction

Combinatorial optimization is one of fundamental problems in computer science. And many applications in various fields such as transportation, communication, manufacturing, infrastructure planning, are expected. One of typical combinatorial optimization problems is the traveling salesman problem (TSP): given a graph, finding the tour that have minimum total edge cost and visit all vertices only once. Especially the problem whose vertex is on the 2-dimensional plane and the cost of the edge is defined as the Euclidean distance between vertices is called 2D Euclidean TSP.

Solving methods for optimization problems are roughly divided according to the accuracy of solution as exact algorithms and heuristics. Exact algorithms find the optimal solution by enumeration or the branch-and-bound, etc., but these are hard to obtain the optimal in practical time against large scale instances. Meanwhile, heuristics do not guarantee the accuracy of the solution theoretically, but it can obtain solutions fast, for example, genetic algorithms [1]. From these backgrounds, development of fast and high-quality heuristics are important.

In recently years, techniques using machine learning have been studied vigorously, and show high performance at many problems. Deep learning makes it possible to extract features of the problem and acquire high-quality approximation with reducing calculation time by spending a long time for learning in advance. From this, it can be expected that using deep learning to a combinatorial optimization problem leads to obtain a faster algorithm to find better solution.

Image processing is a domain that deep learning is actively applied to. The Convolutional Neural Network (CNN) is a neural network consisted of layers that perform convolution operation, and shows high capacity in image recognition and other problems that deal with images. For example, there are primary works for generating image using Deep Convolutional Generative Adversarial Network (DCGAN) [2], [3].

As deep learning is applied to various problems, the importance of reinforcement learning is getting higher. At reinforcement learning, rewards are given for actions of an agent, and the agent is trained to maximize expected rewards. Since this framework does not need the teacher, reinforcement learning is applicable to problems that is hard to conduct supervised learning due to non-obviousness of their best action. In the method of applying deep learning to computer Go [4], the state of the board is input in CNNs to approximate the evaluation value for each position. And the strength is improved by reinforcement learning using the history of games made with self-playing.

There are several works trying to apply deep learning to combinatorial optimization problems including TSP [5]–[7]. In the method using the graph structure of the problem [7], by connecting the layers of the network according to the graph and propagating their outputs between adjacent vertices to extract features efficiently. However, this method seems difficult to extract high-quality features for some problems like TSP whose all vertices are adjacent to each other, then it is necessary to tweak such as limiting the neighborhood of propagation. On the other hand, the learning method using non-optimal solutions instead of optimal is important for combinatorial optimization problems, because it may be difficult to obtain an optimal from the aspect of calculating cost. With respect to this, research by Bello, et al. [6] proposed a learning method for the Pointer Networks [5] using reinforcement learning that does not require the optimal solution as a teacher and gives the objective function of a solution as a reward.

In this paper, we focus on 2D Euclidean TSP, and propose heuristics using CNN and its learning method. This algorithm

978-1-5386-4904-6/18/$31.00 ©2018 IEEE

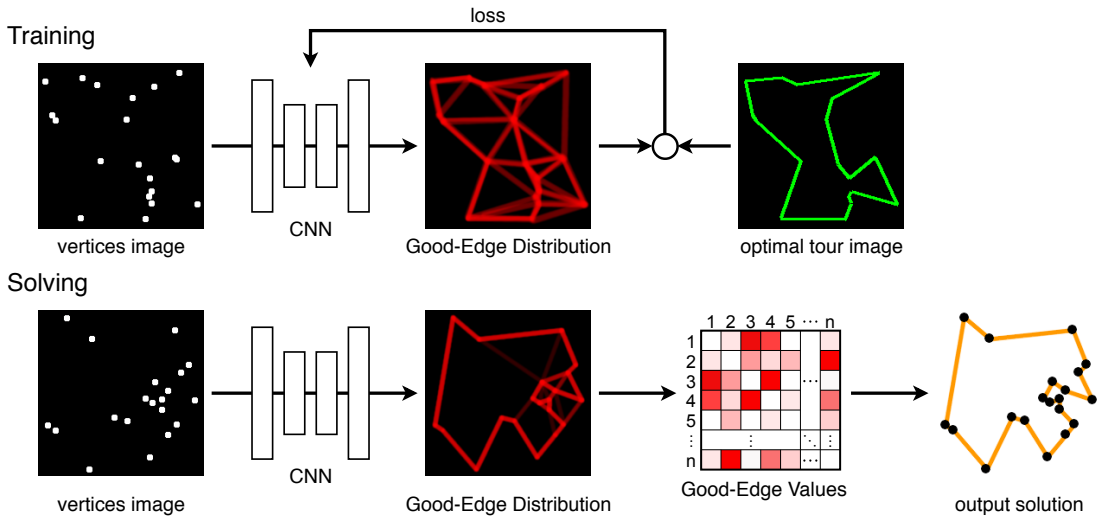


Fig. 1. Overview of supervised learning and solving with the Good-Edge Distribution.

treats a problem and its solution as images, calculates the Good-Edge Distribution approximating the image of the optimal tour by CNN, and chooses the edge according to this distribution to find the solution. Furthermore, we consider a method of applying reinforcement learning to the Good-Edge Distribution in order to learn without optimal solution. In our experiments, we verify the performance of these methods and discuss about their effectiveness.

II. Edge Evaluation with CNN

In 2D Euclidean TSP, each vertices is represented as 2-dimensional Euclidean coordinates, and the problem and the solution can be expressed as images by drawing points and lines geometrically. Given an instance of the problem as a pair of vertices and edges $G = (V, E)$ where $|V| = n$, we define a value on the vertices image $N(x, y)$ and the optimal solution image $t(x, y)$ on which are drawn all vertices and the optimal tour for each position $(x, y) \in \{1 \,..\, S_1\} \times \{1 \,..\, S_2\}$ of pixels. They have pixel values 0 as default and are drawn circles or lines with maximum value 1.

If there is a model which takes vertices image $N(x, y)$ as input and outputs the optimal tour image $t(x, y)$, we can construct an optimal tour by selecting edges which appear on its output image. Our proposed method approximates this model using a CNN that outputs the image as Good-Edge Distribution $p(x, y)$. The overview of the learning method of Good-Edge Distribution and solving algorithm are shown in Fig.1. The Good-Edge Distribution is learned with the objective of bringing its output $p(x, y)$ closer to the $t(x, y)$ as teacher. The loss function is defined as:

$$\text{loss} = \sum_{x=1}^{S_1} \sum_{y=1}^{S_2} (t(x, y) - p(x, y))^2 \tag{1}$$

for each instance of TSP, and parameters of CNN are updated to minimize this loss with gradient method.

When the Good-Edge Distribution is obtained, the Good-Edge Values v_{ij} are calculated as:

$$v_{ij} = \frac{1}{1 + \tau_{ij}} \cdot \frac{\sum_{x=1}^{S_1} \sum_{y=1}^{S_2} p(x, y)\, l_{ij}(x, y)}{\sum_{x=1}^{S_1} \sum_{y=1}^{S_2} l_{ij}(x, y)} \tag{2}$$

v_{ij} represents a likelihood that an edge (i, j) is included in the optimal tour, and is calculated as an average of $p(x, y)$ weighted by the image of line $l_{ij}(x, y)$.

The intersective penalty τ_{ij} is an indication how much vertices exclusive of i, j overlap to edge (i, j) on the image. τ_{ij} is equal to 0 if the edge (i, j) does not overlap with other vertices $V \setminus \{i, j\}$, or k if the edge (i, j) exactly overlaps just k of vertices. An edge that shares some part of pixels with another edge whose Good-Edge Value is high might be evaluated higher unfairly. To suppress this issue, Good-Edge Value of the edge that intersects to other vertices is evaluated lower by using the penalty τ_{ij}.

III. Heuristics with Good-Edge Value

The Good-Edge Value is a likelihood each edge is included in the optimal tour, and it is expected that we can find solutions close to the optimal by selecting edges with high Good-Edge Values. So we propose some heuristics that use Good-Edge Values instead of distance: EV-greedy and EV-2opt.

One of greedy algorithms for TSP is a method of constructing a tour by repeatedly adding the shortest edge. The EV-greedy method shown as Algorithm 2 is based on greedy algorithm and selects edges with high Good-Edge Value.

The conventional 2-opt search is one of the neighbor search methods in TSP, and searches by selecting a shorter tour that is changed from the current tour by 2-opt swap of two edges. In the EV-2opt method, a neighbor solution having a larger total sum of Good-Edge Values of its tour is selected,

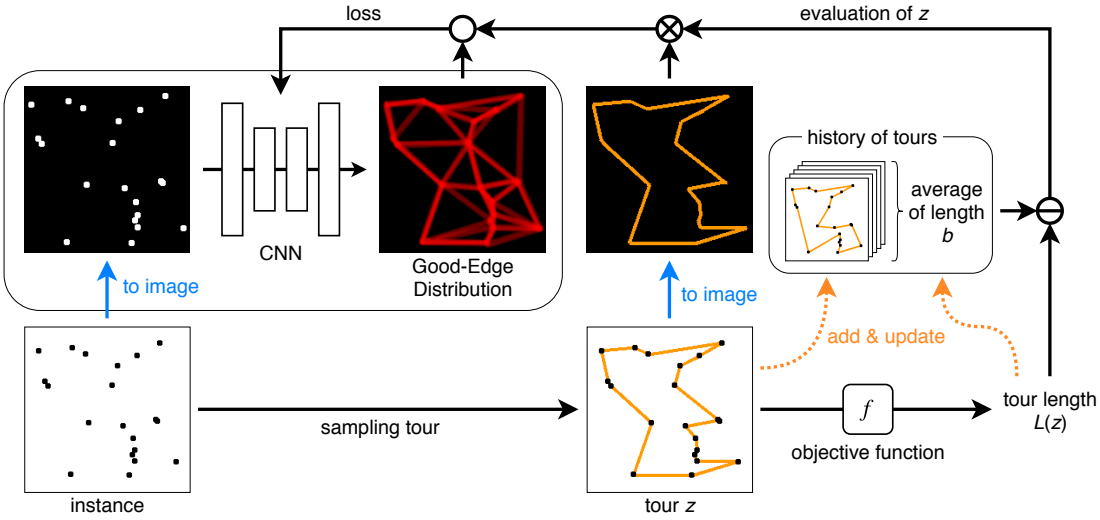


Fig. 2. Overview of reinforcement learning for the Good-Edge Distribution.

as Algorithm 2. There are the first admissible move strategy (FAMS) and the best admissible movement strategy (BAMS) as transition method of neighbor searches, and we use FAMS in the experiment.

EV-greedy and EV-2opt method cannot guarantee that edges of the solution do not intersect each other. So it is expected that applying the conventional 2-opt search using the distance after applying these proposed methods is valid to remove intersections from the tour and further improve accuracy. We call these improved methods as EV-greedy+2opt and EV-2opt+2opt.

IV. Reinforcement Learning for Good-Edge Distribution

It may be difficult to prepare a sufficient number of teacher data for large scale instances, so learning algorithms that do not require an optimal solution is necessary. So we also propose a method of applying reinforcement learning as a learning method of CNN to obtain the Good-Edge Distribution.

Overview of proposed reinforcement learning framework is illustrated in Fig.2. The objective function of the TSP and the accuracy of the solution are defined as the length of the tour, and the shorter the tour length is, the better the solution is. Therefore, we train the CNN so that the image is more likely to be output as the path length of tour on the image is shorter. In the learning process, solutions are randomly sampled for any instances of the test dataset, and the accuracy of the solution is evaluated by comparing with the average b of the path lengths of solutions generated in the past. If the tour length $L(z)$ of a sampled solution z is shorter than the average b, the CNN is trained so as to strengthen the output of that tour, and weaken it if its length longer. Following these rules, the CNN is trained by backpropagating the loss as shown in:

$$\text{loss} = \min\left(\beta, \frac{L(z)-b}{b}\sum_{x=1}^{S_1}\sum_{y=1}^{S_2} p(x,y)\,t(z,x,y)\right) \tag{3}$$

for an instance and its solution z, where $t(z,x,y)$ represents pixel values of the image that the tour z is drawn on. And β is the up-limit of clipping to reduce pressure that is trying to

Algorithm 1 EV-greedy

Calculate Good-Edge Value v_{ij} for $(i,j) \in E$
Initialize set of remained edges $E' \leftarrow E$
Initialize set of edges in solution $Z \leftarrow \emptyset$
while $|z| < n$ **do**
  $(i,j) \leftarrow \text{argmax}_{(i,j)\in E'} v_{ij}$
  Remove (i,j) from E'
  if number of edges connected to i,j is less than 2 for each, and (i,j) doesn't make tour Z close **then**
    Add (i,j) to Z
  end if
end while
return tour made by edges in Z

Algorithm 2 EV-2opt

Calculate Good-Edge Value v_{ij} for $(i,j) \in E$
Set initial tour z
loop
  Find a set of edges $(i,j),(k,h) \in z$ where $i \neq h \wedge j \neq k \wedge v_{ij} + v_{kh} < v_{ik} + v_{jh}$
  if not found $(i,j),(k,h)$ **then**
    break loop
  end if
  Update z with 2-opt to swap edges $(i,j),(k,h)$ to $(i,k),(j,h)$
end loop
return z

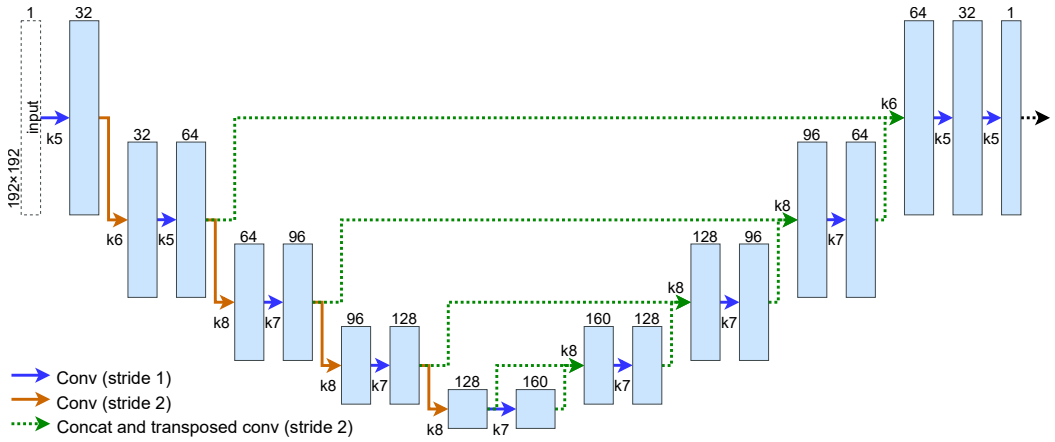


Fig. 3. Architecture of CNN obtaining Good-Edge Distribution.

weaken $p(x, y)$ caused by training with low-accuracy solution. In our experiment $\beta = 0.1$.

The learning is carried out by repeating this process: sampling solution(s) and updating the CNN in this way as one step. At the end of a step, sampled solutions are recorded in the past history, and the value of the average tour length b is updated by the exponential moving average, as shown as the dashed arrows in the Fig.2. e rarely obtained. Therefore, we also propose For the sampling solutions, EV-ε-greedy method is used in order to consider the performance of the CNN while learning. In this method, a random edge is chosen with the possibility ε at selecting the next edge during the EV-greedy method.

V. Experiments

In this section, we describe the method and results of computer experiments conducted to evaluate the performance of the proposed method. Using Python as the programming language of programs and TensorFlow as the framework of machine learning. Components of computers are shown in TABLE I, TABLE II.

As the dataset of 2D Euclidean TSP used for training, we generate 200,000 random instances of 20–100 vertices with coordinates according to the uniform random, and use the TSP solver Concorde [8] to find the optimal solution for supervised learning. The image resolution of teacher and input/output of the CNN is $(S_1, S_2) = (192, 192)$. And these images are drawn with the point of vertex with radius 1.5 and the line of edge with width 1, to express vertices image $N(x, y)$ and teacher tour image $t(x, y)$. For the test instances, we excerpt from random instances generated in the same way as training, and instances of dataset TSPLIB [9].

For the evaluation of the solution at experiments, the error ratio of the tour length to the optimal is used. The error ratio ε of the solution z whose tour length is $L(z)$ is defined as:

$$\varepsilon(z) = \frac{L(z) - L^*}{L^*} \times 100 \ (\%) \tag{4}$$

where the optimal tour length L^*, and the lower this value, the better.

TABLE I
Machine Environment for Supervised Learning.

Item	Content
CPU	Intel Core i7-6900K CPU @ 3.20GHz
GPU	NVIDIA GeForce GTX TITAN X
RAM	125.7 GiB
OS	CentOS 7

TABLE II
Machine Environment for Reinforcement Learning.

Item	Content
CPU	Intel Core i7-7800X CPU @ 3.50GHz
GPU	NVIDIA TITAN Xp
RAM	125.4 GiB
OS	CentOS 7

A. Learning and Solving with Supervised Learning

We train the CNN of Good-Edge Distribution by supervised learning using the instances of training dataset and show the performance of EV-greedy+2opt and EV-2opt+2opt.

For learning Good-Edge Distribution, we use CNN consisting of 14 layers of convolution layers and 4 layers transposed convolution layer as shown in Fig.3. In this figure, the number of channels at the top of a rectangle and the kernel size as number following k are indicated for each convolutional layer. This CNN has a structure similar to that of U-Net [2], [10], [11] which have skip connections from encoder to decoder layers. Leaky ReLU [12] with slope 0.2 is used for each layer except for the output layer that uses identity as activation function. The number of minibatch at training is 32, and Adam optimizer [13] is adopted as learning algorithm.

Since the performance of solution of EV-2opt+2opt depends on the initial solution as same as the conventional 2-opt method, the solution obtained by the distance-based greedy method is used as the initial solution. In the result for the instances of TSPLIB, we also compare the results of S2V-DQN (Structure2Vec Deep Q-Learning) [7].

(a) Optimal tour. (b) Good-Edge Distribution. (c) Initial by greedy ($\varepsilon = 16.979\%$). (d) EV-2opt ($\varepsilon = 2.996\%$). (e) EV-2opt+2opt ($\varepsilon = 0.721\%$).

Fig. 4. Output of the Good-Edge Distribution and solutions (rd100).

TABLE III
AVERAGE ERROR RATIOS FOR EACH ALGORITHM (TSPLIB).

Instances	**EV-greedy**	**EV-greedy+2opt**	**EV-2opt**	**EV-2opt+2opt**	greedy	2opt	S2V-DQN
eil51	6.338	**1.954**	11.162	2.958	24.648	5.070	3.052
berlin52	0.146	**0.000**	0.350	**0.000**	31.941	9.344	**0.000**
st70	**0.119**	0.415	1.489	0.452	11.111	5.963	3.111
eil76	**0.000**	**0.000**	0.372	0.037	8.736	6.952	4.833
pr76	7.766	1.319	6.648	2.188	36.370	3.750	**0.265**
rat99	6.639	**1.497**	5.925	1.936	18.910	7.605	5.698
kroA100	32.361	1.559	1.415	**0.477**	14.120	4.933	2.890
kroC100	11.302	1.588	4.170	**0.104**	12.270	6.255	1.566
rd100	17.649	2.027	2.996	**1.320**	16.979	6.414	3.148
eil101	10.016	**1.093**	5.469	1.932	24.483	6.677	4.769
lin105	2.031	0.852	5.709	**0.807**	16.601	5.639	4.479
bier127	11.912	3.022	8.553	**2.681**	19.493	6.478	2.785
ch130	14.599	3.279	9.814	3.205	18.216	7.117	**2.619**
kroA150	18.878	**2.418**	10.163	2.672	20.238	6.498	5.143
kroA200	12.982	4.342	7.388	**2.747**	17.659	6.396	5.438
Total Avg.	10.182	1.691	5.442	**1.568**	19.452	6.339	3.320

The best error ratio of algorithms is thickened for each instances.

Fig.4 shows example of Good-Edge Distribution obtained by supervised learning and output of solution obtained by the proposed algorithms. (a) of figures is the image of optimal tour of the given instance, and (b) is the Good-Edge Distribution output by CNN. (c)–(e) show a series of solutions during EV-2opt+2opt. By comparing the output of CNN (b) to (a) of Fig.4, we can see that the Good-Edge Distribution has many same edges as optimal tour, and it almost express the outline of the optimal. However, in places where density of vertices is high on image, the CNN tends to output large values even on non-optimal edges. Since this tendency may cause to decrease the accuracy for solving, it is necessary to improve the accuracy of CNN by using instances with a larger number of vertices when training, for example.

Averages of error ratio of solutions for the test instances that are excerpt 15 instances with 200 or less vertices from TSPLIB, are shown in TABLE III. As this result, the total average of the error ratio is EV-2opt+2opt the best, then EV-greedy+2opt, which are better than the greedy, 2opt, S2V-DQN. Even comparing for each instance, the average error ratio of EV-2opt+2opt became smaller except in some instances. From these results, we confirm that proposed algorithms can improve the accuracy of solution.

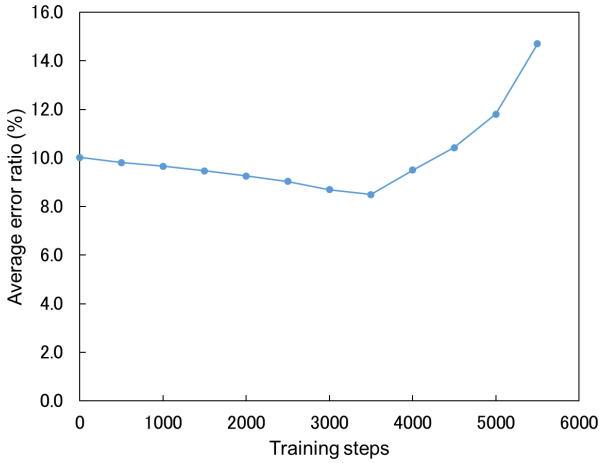
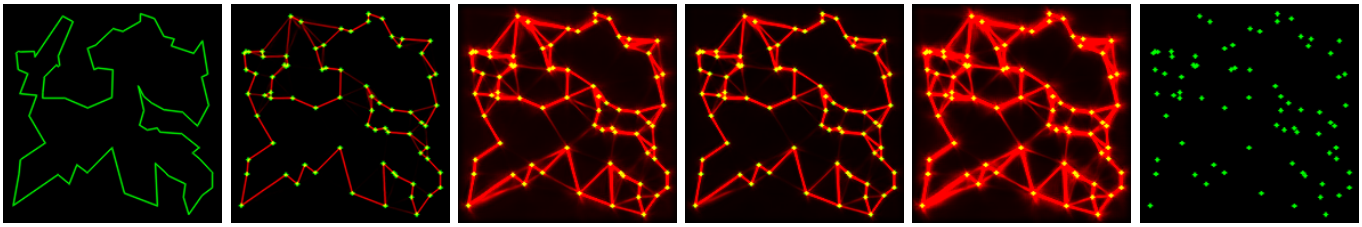


Fig. 5. Error ratio during reinforcement learning.

B. *Effect of Reinforcement Learning*

We develop the proposed method to learn the Good-Edge Distribution with reinforcement learning and show its availability. The reinforcement learning is carried out after supervised learning in order to save experiment time. The



(a) Optimal tour. (b) 0 step. (c) 2000 step. (d) 4000 step. (e) 6000 step. (f) 8000 step.

Fig. 6. Good-Edge Distribution during reinforcement learning for a random instance ($n = 76$).

same dataset of training instances as supervised learning is used, and only vertex coordinates of them are given as input. We used EV-ε-greedy algorithm to find a solution at the time of sampling during reinforcement learning, and EV-greedy without 2-opt improvement for testing. 11 of random instances and 15 of TSPLIB are used as test instances.

The transition of error ratio during reinforcement learning is shown as solid lines in Fig.5. The error ratio of the solution is decreasing from the beginning of reinforcement learning until 3500 steps, and this is seen as an improvement of accuracy by reinforcement learning. However, after 3500 steps the error ratio has keeps increasing, and the solution is getting worse on average (we terminated the learning when it seems no further improvement). Outputs of the Good-Edge Distribution of Fig.6 are strengthened overall as the learning advances, and in 8000 steps, all pixels of the output vanish to zero. When the output of Good-Edge Distribution disappears like this, it seems that the error ratio got worse because EV-greedy and EV-ε-greedy select edges with almost random. In order to solve the issue of learning collapse, it is considered necessary to adopt a stabilizing method such as experience replay, and redesign of learning algorithm is also conceivable.

VI. Conclusion

In this paper, we proposed algorithms using CNN for 2D Euclidean TSP and a method to apply reinforcement learning for this model, and conducted experiments to evaluate these performances. These algorithms use the CNN which approximates the image of optimal tour as Good-Edge Distribution, and find a solution using Good-Edge Values calculated from the distribution as an indication alternative to distances. Experiments with supervised learning confirmed that error ratios of solutions were improved in comparison with the conventional method. And we also examined the method applying reinforcement learning which does not require optimal solution as a teacher, and showed the possibility to improve the accuracy of solution by performing reinforcement learning after supervised learning in experiment.

For future work, improvement of the reinforcement learning method more stable is necessary, and it seems to demand revision of the algorithm. Also we consider that "more constructive" solving model which repeatedly uses the CNN during searching solutions, such as the greedy algorithm, is effective for improving the accuracy of approximation and propagating the progress of searching solution to the model efficiently. We believe that our proposed method is suitable for solving problems that have geometric spatial nature thanks to using CNN for extracting features as images instead of 1-dimensional vectors. And it could be applicable to other problems which input and output are expressed as images such as scheduling problems shown as Gantt charts.

Acknowledgment

This research was partly supported by JSPS KAKENHI Grant Number 18K11484, JSPS KAKENHI Grant Number 17K01309, the Kansai University Grant-in-Aid for Progress of Research in Graduate Courses, and Information and Communication Technology Research Group of ORDIST Kansai University.

References

[1] Y. Nagata and S. Kobayashi, "A powerful genetic algorithm using edge assembly crossover for the traveling salesman problem," *INFORMS Journal on Computing*, vol. 25, no. 2, pp. 346–363, 2013. [Online]. Available: https://doi.org/10.1287/ijoc.1120.0506

[2] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," In *CVPR 2017*, *arXiv:1611.07004*, 2016.

[3] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[4] D. Silver, A. Huang, C. J. Maddison *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–489, Jan 2016. [Online]. Available: http://dx.doi.org/10.1038/nature16961

[5] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," *arXiv preprint arXiv:1506.03134*, 2015.

[6] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," *arXiv preprint arXiv:1611.09940*, 2016.

[7] H. Dai, E. B. Khalil, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," *arXiv preprint arXiv:1704.01665*, 2017.

[8] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook, "Concorde tsp solver," http://www.math.uwaterloo.ca/tsp/concorde/, 2006, accessed 2018/2/22.

[9] G. Reinelt, "TSPLIB—A Traveling Salesman Problem Library," *ORSA Journal on Computing*, vol. 3, no. 4, pp. 376–384, Nov 1991. [Online]. Available: https://doi.org/10.1287/ijoc.3.4.376

[10] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *arXiv preprint arXiv:1505.04597*, 2015.

[11] T. Yonetsuji, "Paintschainer," https://github.com/pfnet/PaintsChainer, 2017, accessed 2018/2/22.

[12] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," In *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[13] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.