AIAA SciTech Forum
January 3-7, 2022, San Diego, CA & Virtual
AIAA SCITECH 2022 Forum

10.2514/6.2022-2209

A Hybrid of Learning-based and Heuristic Methods for Robot Path Planning

Jaejeong (Jane) Shin*
University of Florida, Gainesville, FL, 32603

Choong Hee (Frank) Kim† and Silvia Ferrari‡
Cornell University, Ithaca, NY, 14850

Generalizations of traveling salesman problem (TSP) are often found in many robotics applications where mobile robots with onboard sensors are deployed for a given sensing objective, such as detection or classification. When the sensor field-of-view is continuous and bounded, the vehicle routing problems can be approximately formulated as close enough TSP (CETSP), which is a special case of traveling salesman problem with neighborhoods (TSPN) and searches for the shortest path to visit a set of given circles. While existing CETSP solutions, such as exact and heuristic approaches, have limitations on either computational intractability or poor solution quality depending on the problem scale, machine learning approaches have been getting attentions due to their expandability. This paper proposes a novel CETSP solution that is a hybrid of learning-based and heuristic methods. The proposed method is tested in toy examples, resulting in a better solution compared to the current state-of-the-art heuristic TSP method and suggesting future works on a more expandable learning-based CETSP solution approach.

I. Introduction

In many autonomous sensing tasks, mobile robots are deployed with onboard sensors that have continuous and bounded field-of-view. The robots and sensors are often required to visit multiple regions so that the sensors can obtain measurements to meet a given sensing objective, such as detection and classification. In order to find the most efficient path to gather necessary information within a limited operation time, it is essential to find the shortest path that visits all the given regions. This problem is a generalization of traveling salesman problems (TSPs) referred to as TSPN with neighborhoods (TSPN). When the geometry of neighborhoods is a uniform circle, the problem is referred to as close enough TSP (CETSP). Since it is a generalization of TSP, an NP-hard problem, TSPN is also a combinatorial optimization problem that is even more complex than TSPs.

Due to the high complexity of TSPN, most of the research on TSPN is done with some assumptions on the neighborhoods, such as convexity, fatness, and disjoint neighborhoods. An exact algorithm was proposed in [1], where the TSPN was formulated as a mixed-integer nonlinear programming (MINLP) for convex polyhedral or ellipsoidal neighborhoods. The proposed global non-convex MINLP solver was shown to be computationally feasible up the 16 neighborhoods. Due to the high computational time complexity, a heuristic algorithm for the neighborhoods with arbitrary shapes that gives a reasonable computation time was proposed in [2] and tested with instances comprised of less than 17 neighborhoods. Several heuristic methods are developed in the applied math community with a focus on proving the approximation factors of their solution on TSPN: a constant-factor approximation for the neighborhoods with comparable diameters [3], a constant-factor approximation for convex, fat, and disjoint neighborhoods [4], and a $O(\log n)$-approximation for arbitrary neighborhoods [5], where n is the number of neighborhoods. Also, a couple of evolutionary algorithms were proposed to solve the TSPN such that the neighborhoods are disjoint circles [6, 7].

On the other hand, due to the recent advancement in the machine learning field and its ability and expandability, learning-based TSP solution approaches have been recently getting attention in the computer science community along with other combinatorial optimization problems [8]. In [9], the transformer method [10] is proposed so that the trained

*Assistant Professor, Department of Mechanical and Aerospace Engineering, Address/Mail Stop, and AIAA Member Grade (if any) for first author.

†Ph.D. Student, Department of Mechanical and Aerospace Engineering, Address/Mail Stop, and AIAA Member Grade (if any) for third author.

‡John xx Professor, Department of Mechanical and Aerospace Engineering, Address/Mail Stop, and AIAA Member Grade (if any) for second author.

structure can output a promising solution even when the input dimension is unknown. Based on this approach, in [9], the authors propose a TSP solution approach based on the attention mechanism and refine the solution with reinforcement learning. Once appropriately trained, the deep learning-based algorithm [9] outperforms the well-known state-of-the-art heuristic method [11] in the solution optimality. Another approach is to use the Graph Neural Network (GNN) to solve the structure of the TSP problem. In [12], the authors used the graph convolutional network to find the graph embedding to solve TSP problems. The recent rise of deep learning approaches in combinatorial optimization has revealed that the approach can provide more optimal solutions than the heuristics within a shorter inference time than exact methods. Although there are still limitations on the scalability and generalization of this framework to solve TSPN with wide range of the problem size, this approach suggests another practical view to solve mixed-integer nonlinear programming (MINLP) problems.

In this paper, we propose a novel TSPN solution approach that combines a learning-based TSP solver that adapts attention mechanism and reinforcement learning frameworks and a heuristic algorithm that approximates a CETSP solution based on the computed TSP solution. Since many real world examples require CETSP solutions when the circles may overlap, this paper considers test cases with various overlap ratio. As there has not been a learning-based CETSP solution approach based on our literature review, this paper first focuses on the cases when relatively small ($n < 20$) number of uniform circles are distributed over a region of interest. Based on the discussion in this paper, future work will focus on developing learning-based CETSP algorithms, i.e. without combining the heuristic method, that can solve CETSP problems even when the number of given circles is unknown.

II. Closed Enough Traveling Salesman Problem

This paper considers close enough TSP (CETSP), which corresponds to the traveling salesman problem with neighborhood (TSPN) on unit circles as shown in Fig. 1. This TSPN searches for the shortest tour that visits n given unit circles in Euclidean plane starting from and returning back to a given starting point $\mathbf{p}_0 \in \mathbb{R}^2$. For each circle $i = 1, ..., n$, the center position $\mathbf{x}_i \in \mathbb{R}^2$ is given such that each circle is defined by

$$C_i = \{\mathbf{x} \in \mathbb{R}^2 \mid \|\mathbf{x} - \mathbf{x}_i\| \leq r^2\} \tag{1}$$

where the radius is $r = 1$. The set of circle center points is defined by $X = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$.

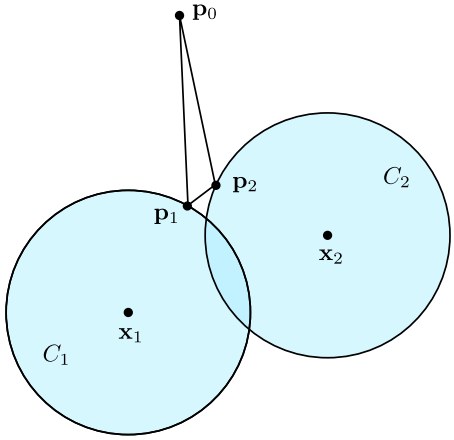

Figure 1 An example problem and notations of TSPN on unit circles

Since the starting position is given or set in many robotics application, the starting point is assumed to be given in this dissertation. This starting point can be considered as another neighborhood, which is a circle with zero radius. A hitting pointset, or a set of waypoints, is defined by a subset $P \subset \bigcup_{i=1}^{n} C_i$ such that $P \cap C_i \neq \emptyset$ for $i = 1, ..., n$ [13] as shown in Fig. 1. A hitting point set is defined by $P = \{\mathbf{p}_1, ..., \mathbf{p}_n\}$, where $\mathbf{p}_i \in C_i$ for $i = 1, ..., n$. The sequence of visit is described as a permutation $E = (\epsilon_1, ..., \epsilon_n)$ such that $1 \leq \epsilon_i \leq n$ and $\epsilon_i \neq \epsilon_j$ for any two $\epsilon_i, \epsilon_j \in \{1, ..., n\}$. Then, the

shortest tour is computed by finding P and E that minimize the length of the tour

$$L(P,E) = \sum_{i=1}^{n} \|\mathbf{p}_{\epsilon_i} - \mathbf{p}_{\epsilon_{i-1}}\| + \|\mathbf{p}_{\epsilon_0} - \mathbf{p}_{\epsilon_n}\| \tag{2}$$

III. CETSP Solution: Learning-based and Geometric Heuristic Methods

In order to overcome the intractability of exact methods for TSPN due to the intensive computational complexity [1], we propose a learning based approach to overcome the computation time and provide a better quality of solution compared to other heuristic methods. First, we propose a modification to the attention mechanism and deep reinforcement learning based TSP solver, which is proposed in [9], to compute a TSP solution for the circle centers. Then, the CETSP solution is computed using the TSP solution and a geometric approach that is inspired by *Fagnano's problem* to provide a qualified heuristic solution.

A. Attention Mechanism for TSP

Attention mechanism is the key idea of Transformer architecture [10] which outperforms the previous deep learning architecture such as CNN, LSTM, and RNN in natural language processing and computer vision domain. Self-attention, also known as intra-attention, is an attention mechanism relating different positions of a single sequence in order to compute a representation of the same sequence. Although there exist related work on solving TSP using attention based encoder-decoder framework [14] , here we modify the attention mechanism based structure for solving TSP problem. Before getting into each detail structure, TSP can be defined as following mathematical equations. For Euclidean 2 dimensional TSP that we are trying to solve for the path planning algorithm, problem instance s is defined as a graph with n nodes, where node $i \in \{1, 2, \cdots, n\}$ is represented by its position $\mathbf{x}_i \in \mathbb{R}^2$. The solution for the TSP can be defined as a tuple of integers $\mathbf{E} = (\epsilon_1, \epsilon_2, \cdots, \epsilon_n)$ as a permutation of the nodes. Each $\epsilon_i \in \{1, 2, \cdots, n\}$ and $\epsilon_t \neq \epsilon_{t'} \forall t \neq t'$. For given parameter θ, our goal is to find the probability of the next tour ϵ given the problem instance s. This can be formulated and factorized into the following equation using conditional probability.[9]

$$p_\theta(\epsilon|s) = \prod_{t=1}^{n} p_\theta(\epsilon_t|s, \epsilon_{1:t-1}) \tag{3}$$

B. Encoder

The overall encoder network structure follows the Transformer architecture introduced in [10] with several modification for TSP. The first layer of the encoder structure is the node embedding layer. From the input features $\mathbf{x}_i$ which has d_x dimension, the layer computes the d_h dimensional node embeddings through a learned parameters W_x and $\mathbf{b}^x$. After the linear embedding layer, extracted node embeddings($\mathbf{h}_i^{(0)} = W^x\mathbf{x}_i + \mathbf{b}^x$) are fed into the attention layer which was first introduced in [10]. We denote $\mathbf{h}_i^{(l)}$ the embedding of node i produced by layer $l \in \{1, 2, \cdots, N\}$. The mean of all node embedding values at the final layer $\bar{\mathbf{h}}^{(N)} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{h}_i^{(N)}$ is noted as the graph embedding $\mathbf{h}_i^{(N)}$. Both the node embeddings at the final layer $\mathbf{h}_i^{(N)}$ and graph embedding $\bar{\mathbf{h}}^{(N)}$ are used in decoder as contextual information.

1. Attention layer

The attention layer has two sublayers of a multi-head attention (MHA) and a node-wise fully connected feed-forward (FF) layer. The structure is first introduced in the Transformer architecture[10]. Each sublayer adds a skip-connection[15] and batch normalization (BN)[16] for better training.

$$\hat{\mathbf{h}}_i = \mathrm{BN}^l(\mathbf{h}_i^{l-1} + \mathrm{MHA}_i^l(\mathbf{h}_i^{(l-1)}, \cdots, \mathbf{h}_i^{(l-1)})) \tag{4}$$

$$\mathbf{h}_i^{(l)} = \mathrm{BN}^l(\hat{\mathbf{h}}_i + \mathrm{FF}^l(\hat{\mathbf{h}}_i)) \tag{5}$$

2. Attention mechanism

Attention mechanism introduced in [10] can be also interpreted as a message passing algorithm between nodes in a graph.[9] In high-level understanding, extracted graph embedding and node embedding value can be used to

compute the attention score that represents the probability of each node to be the next node for TSP solution given the current node. Formally, we define d_k, d_v for dimensions and define key, query, value as $\mathbf{k}_i \in \mathbb{R}^{d_k}, \mathbf{v}_i \in \mathbb{R}^{d_v}, \mathbf{q}_i \in \mathbb{R}^{d_k}$. Then, query, key, and value vector of each node $\mathbf{h}_i$ can be represented by projecting with learned parameters $W^Q \in \mathbb{R}^{(d_k \times d_h)}, W^K \in \mathbb{R}^{(d_k \times d_h)}, W^V \in \mathbb{R}^{(d_v \times d_h)}$.

$$\mathbf{q}_i = W^Q \mathbf{h}_i, \quad \mathbf{k}_i = W^K \mathbf{h}_i, \quad \mathbf{v}_i = W^V \mathbf{h}_i \tag{6}$$

The compatibility $u_{ij} \in \mathbb{R}$ of $\mathbf{q}_i, \mathbf{k}_j$ of node i and node $j$as the dot-product[10].

$$u_{ij} = \begin{cases} \frac{\mathbf{q}_i^T \mathbf{k}_j}{\sqrt{d_k}} & \text{if } i \text{ adjacent to } j \\ -\infty & \text{otherwise} \end{cases} \tag{7}$$

With this compatibility, we can compute the attention weights $a_{ij} \in [0, 1]$ using a softmax and create a vector $\mathbf{h}'_i$ that is received by node i with messages $\mathbf{v}_j$

$$a_{ij} = \frac{e^{u_{ij}}}{\sum_{j'} e^{u_{ij'}}} \tag{8}$$

$$\mathbf{h}'_i = \sum_j a_{ij} \mathbf{v}_j \tag{9}$$

3. Multi-head attention

Instead of having a single attention mechanism, multi-head attention improves the result by allowing nodes to receive different types of messages from different neighbors. The equation in attention mechanism was computed $M = 8$ times for creating multiple heads. The result vectors $\mathbf{h}'_i$ is divided into multiple heads $\mathbf{h}'_{im}$ for $m \in \{1, 2, \cdots, M\}$ and then projected with different parameters $W^O_m \in \mathbb{R}^{(d_h \times d_v)}$ and the final multi-head attention value for node i is created.

$$d_k = d_v = \frac{d_h}{M} = 16 \tag{10}$$

$$\mathrm{MHA}_i(\mathbf{h}_1, \cdots, \mathbf{h}_n) = \sum_{m=1}^{M} W^O_m \mathbf{h}'_{im} \tag{11}$$

4. Feed-forward layer

The feed forward layer is node-wise projections using a hidden sublayer with learned weights $W^{\mathrm{ff},l}$, $W^{\mathrm{ff},l-1}$, $\mathbf{b}^{\mathrm{ff},0}$, $\mathbf{b}^{\mathrm{ff},l}$ and dimension size of $d_{\mathrm{ff}} = 512$ and ReLu activation:

$$\mathrm{FF}(\hat{\mathbf{h}}_i) = W^{\mathrm{ff},l} \cdot \mathrm{ReLu}(W^{\mathrm{ff},l-1} \hat{\mathbf{h}}_i + \mathbf{b}^{\mathrm{ff},l-1}) + \mathbf{b}^{\mathrm{ff},l} \tag{12}$$

5. Batch normalization

The batch normalization used in this network also has affine transformation with parameters $\mathbf{w}^{bn}, \mathbf{b}^{bn}$

$$\mathrm{BN}(\mathbf{h}_i) = \mathbf{w}^{bn} \otimes \bar{\mathrm{BN}}(\mathbf{h}_i) + \mathbf{b}^{bn} \tag{13}$$

where $\otimes$ represent for element-wise product and $\bar{\mathrm{BN}}$ refers to batch normalization without affine transformation [16].

C. Decoder

Based on the contextual information $(\mathbf{h}_i^{(N)}, \bar{\mathbf{h}}^{(N)})$ extracted from the encoder, decoding happens sequentially. For each timestep $t \in \{1, 2, \cdots, n\}$, the node output ϵ_t is generated from the decoder based on the node and graph embedding from the encoder and the previous outputs. With computed graph embedding $\bar{\mathbf{h}}^{(N)}$ and node embedding $\mathbf{h}_i^{(N)}$, we can construct context vector $\mathbf{h}_{(c)}^{(N)}$ with the node embedding of the first node and the last node of the current sequence and overall graph embedding vector. $\mathbf{v}^1, \mathbf{v}^f$ is the input placeholder for the context vector at the first timestep.

$$\mathbf{h}_{(c)}^{(N)} = \begin{cases} [\bar{\mathbf{h}}^{(N)}, \mathbf{h}_{\epsilon_{t-1}}^{(N)}, \mathbf{h}_{\epsilon^{(N)}}] & t > 1 \\ [\bar{\mathbf{h}}^{(N)}, \mathbf{v}^1, \mathbf{v}^f] & t = 1 \end{cases} \tag{14}$$

The computed context vector is used to compute the probability of each node to be the next node by attention mechanism[9]. The new context node embedding $\mathbf{h}_{(c)}^{(N+1)}$ is created using the multi-head attention mechanism described in equation (11).

$$\mathbf{q}_{(c)}^{(N)} = W^Q \mathbf{h}_{(c)}, \quad \mathbf{k}_i^{(N)} = W^K \mathbf{h}_i, \quad \mathbf{v}_i^{(N)} = W^V \mathbf{h}_i \tag{15}$$

$$d_k = \frac{d_h}{M} \tag{16}$$

$$u_{(c)}j = \begin{cases} \frac{\mathbf{q}_{(c)}^{(N)T} \mathbf{k}_j}{\sqrt{d_k}} & \text{if } \; j \neq \epsilon_{t'} \;\; \forall t' < t \\ -\infty & \text{otherwise.} \end{cases} \tag{17}$$

Then, the new context node embedding $\mathbf{h}_{(c)}^{(N+1)}$ is created from the equation (8) to (11) as we did in the encoder part. However, there are no skip-connections, batch normalization and the feed-forward sublayer in the decoding part for the maximal efficiency as described in [9]. The result vector is similar to the *glimpse* described by [8].

1. Compute probabilities for the next node

The output probabilities $p_\theta(\epsilon_t | s, \epsilon_{1:t-1})$ can be computed by the final sublayer of single attention mechanism. The layer computes the attention score, which represents for the probability of each node i to be the next node given previous nodes $\epsilon_{1:t-1}$. Similar to the equation (7) in attention mechanism, we compute the compatibility but now clip the result using the *tanh* function and constant C=10.

$$u_{(c)j} = \begin{cases} C \cdot tanh(\frac{\mathbf{q}_i^T \mathbf{k}_j}{\sqrt{d_k}}) & \text{if } \; j \neq \epsilon_{t'}, \;\; \forall t' < t \\ -\infty & \text{otherwise} \end{cases} \tag{18}$$

The compatibility value of each node is unnormalized log-probabilities (logits) and compute we can $\mathbf{p}$ using a softmax with the compatibilities.

$$p_i = p_\theta(\epsilon_t = i | s, \epsilon_{1:t-1}) = \frac{e^{u_{(c)i}}}{\sum_j e^{u_{(c)j}}} \tag{19}$$

D. REINFORCE Algorithm for Improving the Encoder-Decoder Structure

After the probability distribution $p_\theta(\epsilon|s)$ was generated from the encoder-decoder structured network, the overall network is trained using the reinforcement learning algorithm . The loss function is defined as $\mathcal{L}(\theta|s) = \mathbb{E}_{p_\theta(\epsilon|s)}[L(\epsilon)]$ where $L(\epsilon)$ denotes for the tour length of TSP defined in (2). The loss function $\mathcal{L}$ is optimized using the REINFORCE algorithm. The gradient descent of the algorithm is defined as follow with baseline rewards $b(s)$.

$$\nabla \mathcal{L}(\theta|s) = \mathbb{E}_{p_\theta(\epsilon|s)}[(L(\epsilon) - b(s)) \nabla log p_\theta(\epsilon|s)] \tag{20}$$

From the above equation, choosing good baseline b(s) reduces the time of learning by reducing its gradient variance. The baseline is updated by $M \longleftarrow \beta M + (1-\beta) L(\epsilon)$ in subsequent iterations. Instead, we are trying to use the baseline method with LKH3[11] , a state-of-the art heuristic solver for solving TSP problem. This makes the difference from the supervised learning approaches [14, 17, 18]. For the optimizer, we used Adam optimizer [19] which is widely used optimization algorithm for machine learning.

Algorithm 1 REINFORCE

Input : number of epochs H, steps per epoch T, batch size B, significance α
Initialize : $\theta, \theta^{BL} \longleftarrow \theta$
for epoch = $1, 2, \cdots H$ **do**
  for step = $1, 2, \ldots, T$ **do**
    $s_i \longleftarrow$ RandomInstance() $\forall i \in \{1, \cdots, B\}$
    $\epsilon_i \longleftarrow$ SampleRollout(s_i, p_θ) $\forall i \in \{1, \cdots, B\}$
    $\epsilon_i^{BL} \longleftarrow$ GreedyRollout$(s_i, p_{\theta^{BL}})$ $\forall i \in \{1, \cdots, B\}$
    $\nabla \mathcal{L} \longleftarrow \sum_{i=1}^{B} (L(\epsilon_i) - L(\epsilon_i^{BL})) \nabla_\theta log p_\theta(\epsilon_i)$
    $\theta \longleftarrow$ Adam$(\theta, \nabla \mathcal{L})$
  end for
  if OneSidedPairedTTest$(p_\theta, p_{\theta^{BL}}) < \alpha$ **then**
    $\theta^{BL} \longleftarrow \theta$
  end if
end for

E. 2-Opt for Improving TSP Solutions in Larger Scale TSPs

Although the previous work of attention based TSP [9] is the current state of the art TSP algorithm in learning-based approaches, it has a huge limitation of dependency on the size of problem during training. For example, if you train your algorithm with problem size when $N = 10$, it outperforms heuristic methods for N=10 but the performance drops when the size of the problem gets bigger such as N=50 and N=100. In order to resolve the issue for dealing with larger problem, we first trained the algorithm for N=10 and try to look how it works for N=50 or N=100. Our observation shows that the algorithm misses the local order of nodes for the larger problem. In order to alleviate the issue, we applied 2-opt algorithm [20], one of the most popular heuristic methods in TSP solution, to the end of our solution.

F. Virtual Force (VF) Method for Solving CETSP from TSP Solutions

After solution for TSP is generated from the learning-based algorithm, solution for CETSP is generated with its upper bound with the TSP solution. Although there already exist heuristic methods to approximate TSP solution to CETSP solution such as [21]. Our method is inspired from *Fagnano's Problem* which is to find the inscribed triangle of minimum perimeter within an acute triangle. This is a simple set TSP problem that each vertex of the inscribed triangle is selected within a side of the outer triangle. The solution to the Fagnano's Problem is related to the *Lami's Theorem*.

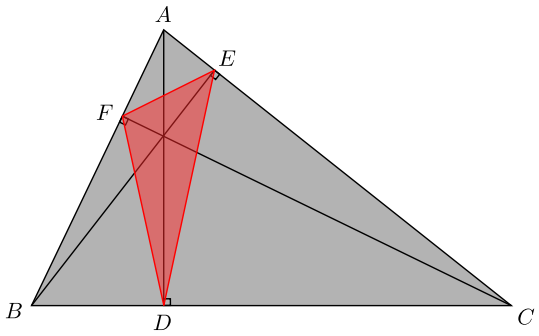

Figure 2 Fagnano's problem : For a given acute triangle ΔABC, the solution for the problem is the orthic triangle ΔDEF. This problem is the one of the earliest forms of set TSP as we can see that each vertex is chosen from each side of the acute triangle.

Theorem 1 *For three coplanar, concurrent and non-collinear vectors $\vec{A}, \vec{B}, \vec{C}$ applied to an object and keep static*

equilibrium and $\alpha, \beta,$ and γ are the angles directly opposite to the vectors, the following equations holds.

$$\frac{||\vec{A}||_2}{sin\alpha} = \frac{||\vec{B}||_2}{sin\beta} = \frac{||\vec{C}||_2}{sin\gamma} \tag{21}$$

The *Lami's Theorem* was inspired from static equilibrium of forces in Physics. If you put a rubber band around vertices of the inscribed triangle, position of each vertex will be determined in order to minimize the elastic energy of the rubber band where the derivative of the total length becomes zero. By using the *Lami's Theorem* based on the *Hooke's Law*, the inscribed triangle with minimum perimeter is actually the orthic triangle with vertices at the base points of the altitudes of the given triangle. By adopting the idea of the theorem, we devised an iterative heuristic methods to iterative find $\mathbf{x}_i$ around the original TSP solution $\mathbf{c}_i$ using virtual force vectors which are the position vector between the current point $\mathbf{x}_i$ and adjacent points $\mathbf{x}_{i-1}, \mathbf{x}_{i+1}$. For each iteration, force around point $\mathbf{x}_i$ is computed with its adjacent point $\mathbf{x}_{i-1}$ and $\mathbf{x}_{i+1}$. Vector summation of the force is treated as the gradient of the update for point $\mathbf{x}_i$. For the stepsize of the algorithm, we put $\frac{radius}{totaliteration}$ in order constrain the maximum move of the point should not go beyond the radius from the initial center of the neighborhood region.

Algorithm 2 Virtual Force Approximation

Input : number of iterations M, radius r, TSP solution $\mathbf{c} = \{c_1, \cdots, c_n\}, c_i \in \mathbb{R}^2$
Initialize : $\mathbf{x}^0 \longleftarrow \mathbf{c} \quad (\mathbf{x}^0 \in \mathbb{R}^{(n\times 2)})$
for $iter = 1, 2, \cdots M$ **do**
    Initialize : $gradF \in \mathbb{R}^{(n\times 2)}$
    for index $= 1, 2, \ldots, n$ **do**
        prev $= \mathrm{mod}(i-1, n)$, next $= \mathrm{mod}(i+1, n)$
        $\mathbf{f}_i = \mathbf{x}_{prev}^{iter-1} - \mathbf{x}_i^{iter-1} + \mathbf{x}_{next}^{iter-1} - \mathbf{x}_i^{iter-1}$
        $\mathbf{f}_i = \frac{\mathbf{f}_i}{||\mathbf{f}_i||_2} * \frac{r}{M}$ $\quad || \cdot ||$: L2-norm
        $gradF[i,:] \longleftarrow \mathbf{f}_i$
    end for
    $\mathbf{x}^{iter} \longleftarrow \mathbf{x}^{iter-1} + gradF$
end for
return $\mathbf{x}^M$

IV. Simulation Experiments

A. TSP Performance Comparison

The proposed method, which is named by *AttentionRL* in this paper, is first compared to the state-of-the-art TSP heuristic solution approach, LKH3 [11], and the exact TSP solution approach, Concorde [22]. The proposed AttentionRL method is first trained with small-scale TSPs, specifically $N = 10$ where N is the number of regions to visit, as that range of scale is found in many robotic path planning applications. In Fig. 3, the algorithm is trained for 1000 epochs. The size of the training dataset is 1000 and the size of testing dataset is 200. The loss function of the algorithm is set as the total tour of TSP problem. As shown in Fig. 3, after about 400 epochs passed, the proposed deep learning based algorithm shows a smaller optimality gap of the TSP solution compared to that of LKH3.

The TSP solution performance is compared for 1000 instances of randomly generated TSPs ($N = 10$), and the results are shown in Table 1. While the exact method results in the most optimal TSP solution, the proposed AttentionRL method outperforms the heuristic LKH3 method. Moreover, the simulation results show that the proposed learning-based method can improve the TSP solution while requiring a short inference time compare to the computation time of the exact method (Table 2). In order to compare runtime, *pyconcorde* code is used for implementing the exact method, and *elkai* module is used for implementing the heuristic LKH3 module. The runtime of each TSP solver is measured by excluding the pre-processing time for a fair comparison.

B. CETSP Performance Comparison and Overlap Ratio

CETSP solutions were generated by combining the AttentionRL method with the proposed VF approximation. In order to compare the algorithm with heuristic method [11], VF approximation algorithm is also combined with the

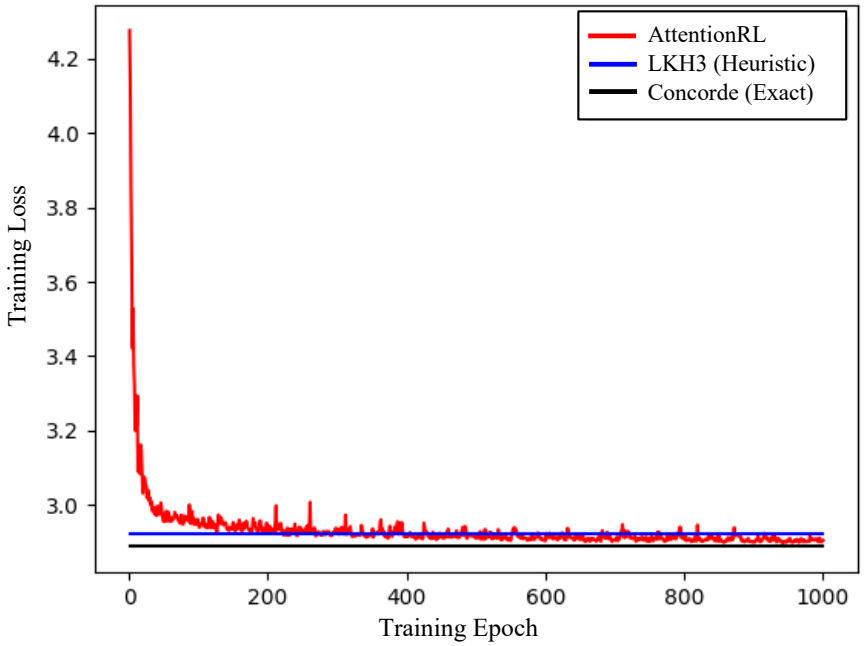

Figure 3 Training result for AttentionRL using TSPs size of $N = 10$.

Algorithm	AttentionRL (**Proposed**)	LKH3 (Heuristic)	Concorde (Exact)
Average TSP tour length [std]	2.887 [± 0.345]	2.903 [±0.348]	2.870 [±0.343]

Table 1 Average of TSP Solutions for 1000 instances

Algorithm	AttentionRL (**Proposed**)	LKH3 (Heuristic)	Concorde (Exact)
Runtime (sec)	0.026	0.0239	0.311

Table 2 Runtime for solving 100 TSP instances for each algorithm. Considering its application to path planning, we measured time after loading the pre-trained weight for Attention RL.

heuristic method to generate CETSP solutions. As CETSP algorithm performance depends on how much the given circles overlap, the proposed method and heuristic method are compared in wide range of overlap ratio, which is denoted by R in this paper. For comparison, a relative length, denoted by η, is defined by dividing the computed CETSP tour length by the exact TSP tour length, i.e.,

$$\eta = \frac{L(P,E)}{L(X,E)} \tag{22}$$

where the notation L is defined in (2), P is a hitting pointset, X is a set of circle center points, and E is the sequence of visit as defined in Section II.

Overlap Ratio (R)	Relative Length (η) [std]	
	AttentionRL+VF (**Proposed**)	LKH3+VF (Heuristic)
0.01	**0.974** [±0.020]	0.979 [±0.020]
0.05	**0.865** [±0.024]	0.869 [±0.023]
0.1	**0.754** [±0.033]	0.757 [± 0.032]
0.15	**0.655** [±0.040]	0.659 [± 0.040]
0.2	**0.560** [±0.048]	0.564 [± 0.048]
0.3	**0.373** [±0.063]	0.378 [±0.064]
0.4	**0.199** [±0.070]	0.204 [±0.071]
0.5	**0.065** [±0.047]	0.069 [±0.049]

Table 3 CETSP solution for various overlap ratio

The comparison is summarized in Table 3, where the performance is compared using the average value computed for 1000 randomly generated instances in each overlap ratio value. The result shows that the proposed method outperforms the heuristic method for all the given overlap ratios. As the overlap ratio increases, the CETSP tour length computed by the proposed method decreases the original TSP more effectively compared to the heuristic method. Several representative examples are shown in Fig. 4, Fig. 5, and Fig. 6 to illustrate the behaviors of the CETSP solutions computed by the proposed and heuristic methods in various overlap ratios.

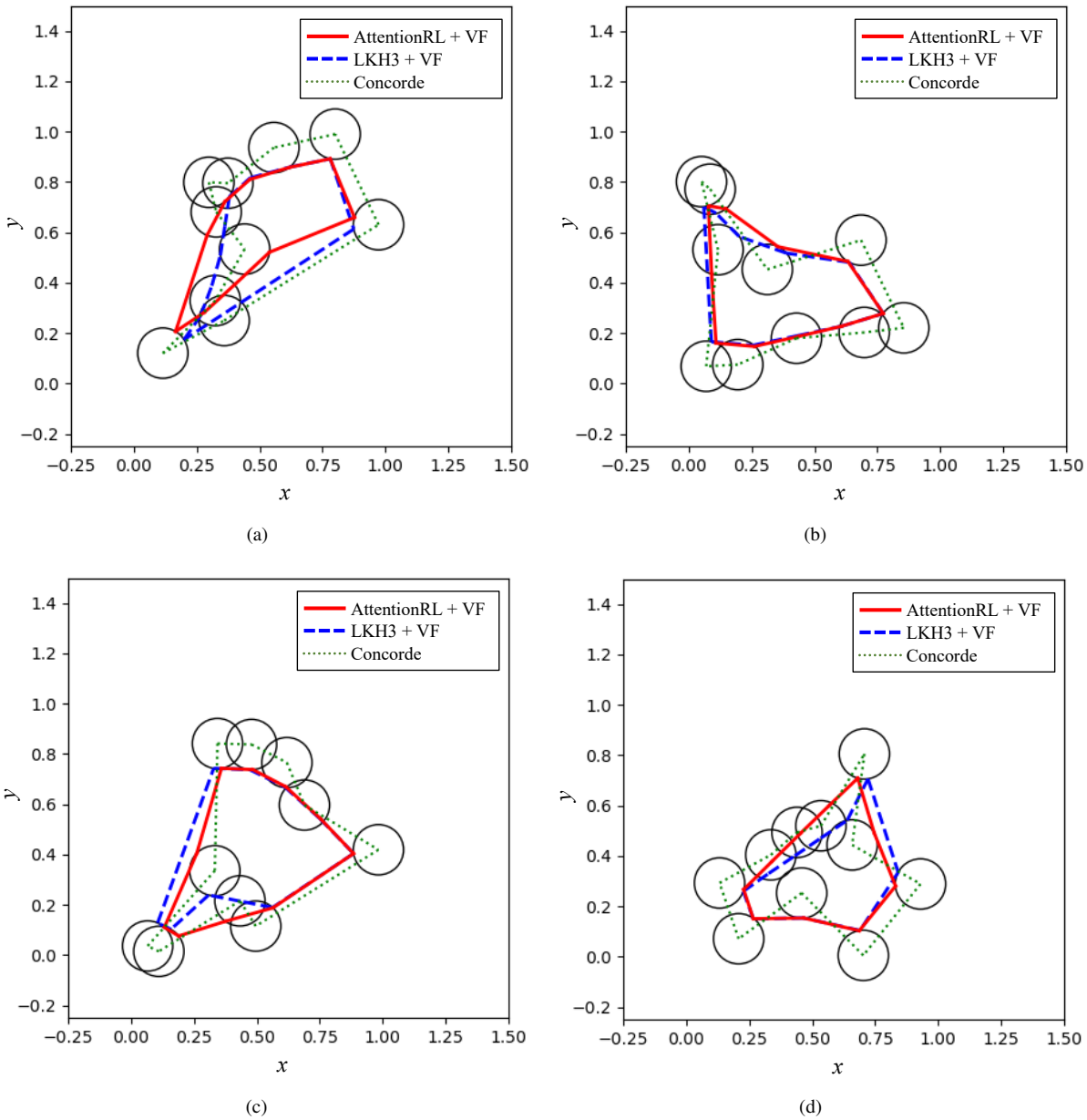

Figure 4 Four examples for CETSP comparison: proposed method (AttentionRL+VF) and heuristic method (LKH+VF) when the overlap ratio $R = 0.1$

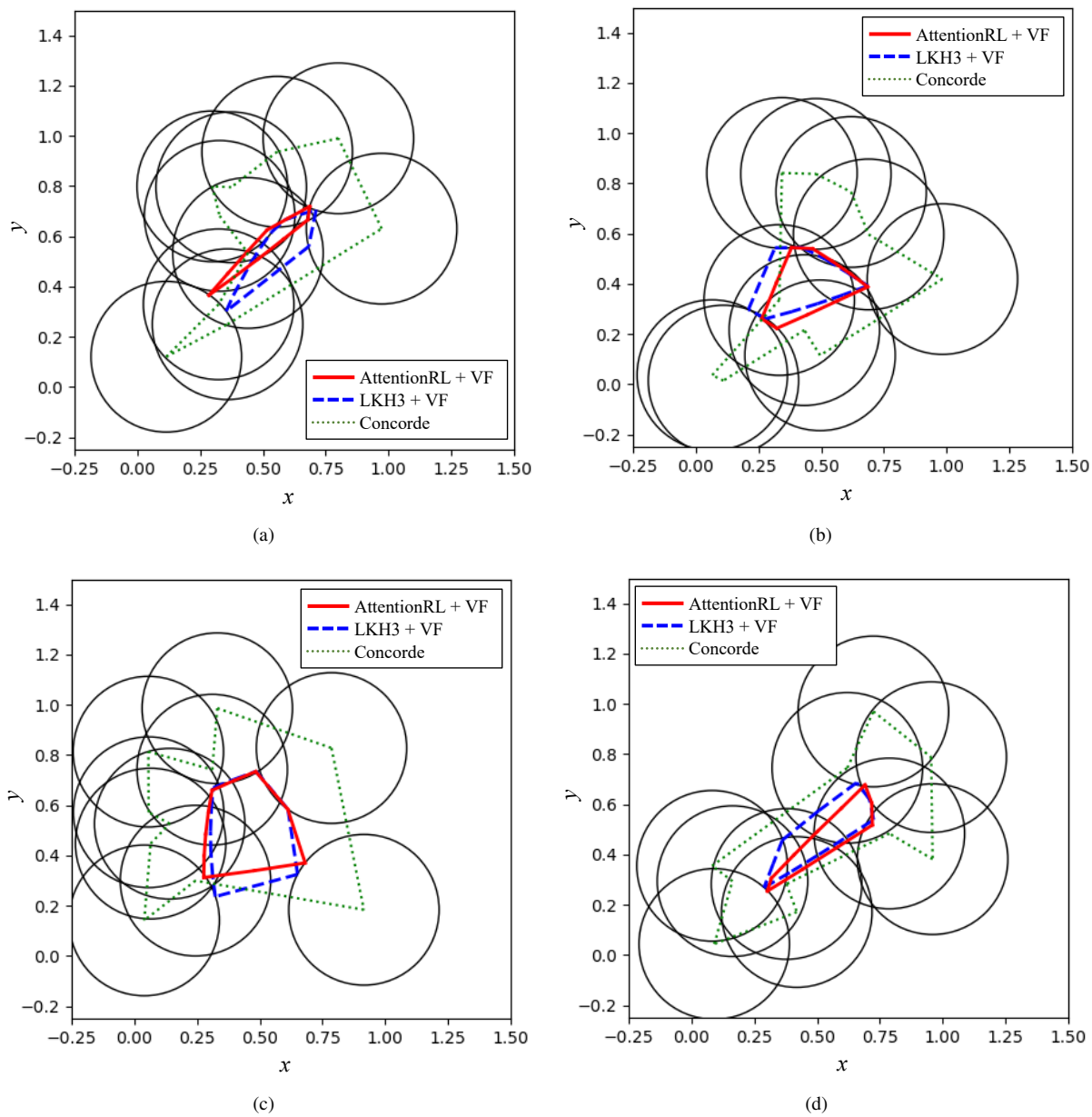

Figure 5 Four examples for CETSP comparison: proposed method (AttentionRL+VF) and heuristic method (LKH+VF) when the overlap ratio $R = 0.3$

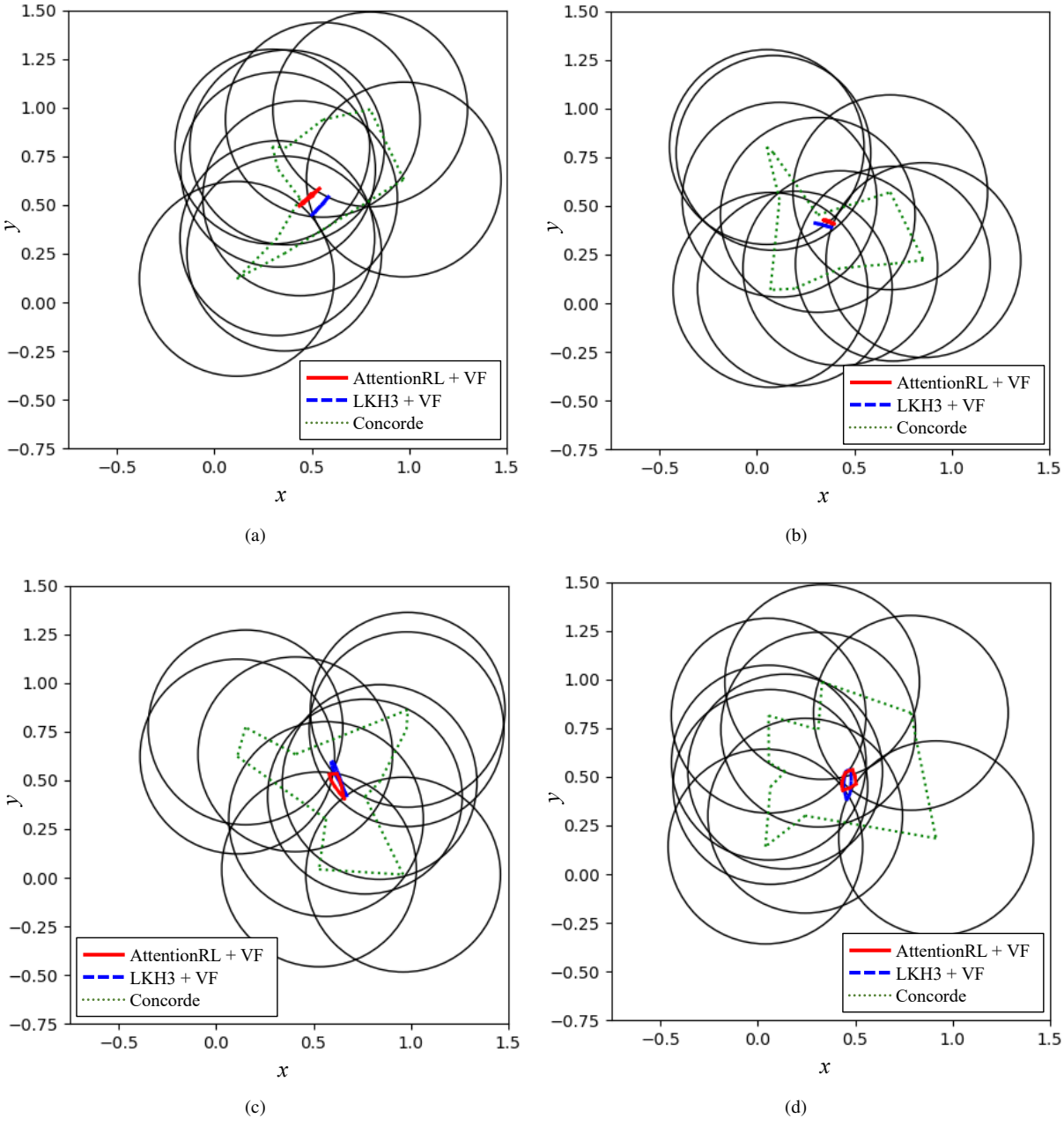

Figure 6 Four examples for CETSP comparison: proposed method (AttentionRL+VF) and heuristic method (LKH+VF) when the overlap ratio $R = 0.5$

V. Conclusion

In this work, we propose a deep learning-based CETSP solution approach, which is developed by combining a learning-based TSP solution approach, AttentionRL, and a physics-inspired approximation method, virtual force (VF) method. The presented method outperforms the state-of-the-art heuristic method [11] in both TSP and CETSP in terms of optimality of the computed tour length. The results show several promising possibilities of deep learning algorithms being applied in combinatorial optimization problems in a way to avoid the curse of dimensionality and provide solutions in real-time while resulting in more optimal solutions compared to the heuristic method. However, the performance of the proposed deep learning algorithm trained with small problems ($N = 10$) does not guarantee the same performance in bigger problems ($N > 50$). This limitation is the major challenge of deep learning approaches and should be addressed in the future work. Also, providing the proof of guarantee on the proposed deep learning algorithm remains as a future work in order to apply the method in real-world applications.

References

[1] Gentilini, I., Margot, F., and Shimada, K., "The travelling salesman problem with neighbourhoods: MINLP solution," *Optimization Methods and Software*, Vol. 28, No. 2, 2013, pp. 364–378.

[2] Alatartsev, S., Mersheeva, V., Augustine, M., and Ortmeier, F., "On optimizing a sequence of robotic tasks," *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2013, pp. 217–223.

[3] Dumitrescu, A., and Mitchell, J. S., "Approximation algorithms for TSP with neighborhoods in the plane," *Journal of Algorithms*, Vol. 48, No. 1, 2003, pp. 135–159.

[4] de Berg, M., Gudmundsson, J., Katz, M. J., Levcopoulos, C., Overmars, M. H., and van der Stappen, A. F., "TSP with neighborhoods of varying size," *Journal of Algorithms*, Vol. 57, No. 1, 2005, pp. 22–36.

[5] Elbassioni, K., Fishkin, A. V., and Sitters, R., "Approximation algorithms for the Euclidean traveling salesman problem with discrete and continuous neighborhoods," *International Journal of Computational Geometry & Applications*, Vol. 19, No. 02, 2009, pp. 173–193.

[6] Yuan, B., Orlowska, M., and Sadiq, S., "On the optimal robot routing problem in wireless sensor networks," *IEEE transactions on knowledge and data engineering*, Vol. 19, No. 9, 2007, pp. 1252–1261.

[7] Comarela, G., Gonçalves, K., Pappa, G. L., Almeida, J., and Almeida, V., "Robot routing in sparse wireless sensor networks with continuous ant colony optimization," *Proceedings of the 13th annual conference companion on Genetic and evolutionary computation*, 2011, pp. 599–606.

[8] Bello, I., Pham, H., Le, Q. V., Norouzi, M., and Bengio, S., "Neural Combinatorial Optimization with Reinforcement Learning," , 2017.

[9] Kool, W., van Hoof, H., and Welling, M., "Attention, Learn to Solve Routing Problems!" , 2019.

[10] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I., "Attention Is All You Need," *CoRR*, Vol. abs/1706.03762, 2017. URL `http://arxiv.org/abs/1706.03762`.

[11] Helsgaun, K., "An effective implementation of the Lin–Kernighan traveling salesman heuristic," *European Journal of Operational Research*, Vol. 126, No. 1, 2000, pp. 106–130. https://doi.org/https://doi.org/10.1016/S0377-2217(99)00284-2, URL https://www.sciencedirect.com/science/article/pii/S0377221799002842.

[12] Joshi, C. K., Laurent, T., and Bresson, X., "An Efficient Graph Convolutional Network Technique for the Travelling Salesman Problem," *CoRR*, Vol. abs/1906.01227, 2019. URL http://arxiv.org/abs/1906.01227.

[13] Elbassioni, K., Fishkin, A. V., Mustafa, N. H., and Sitters, R., "Approximation algorithms for Euclidean group TSP," *International Colloquium on Automata, Languages, and Programming*, Springer, 2005, pp. 1115–1126.

[14] Vinyals, O., Fortunato, M., and Jaitly, N., "Pointer Networks," , 2017.

[15] He, K., Zhang, X., Ren, S., and Sun, J., "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778. https://doi.org/10.1109/CVPR.2016.90.

[16] Ioffe, S., and Szegedy, C., "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," , 2015.

[17] Sutskever, I., Vinyals, O., and Le, Q. V., "Sequence to Sequence Learning with Neural Networks," , 2014.

[18] Hottung, A., Bhandari, B., and Tierney, K., "Learning a Latent Search Space for Routing Problems using Variational Autoencoders," *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=90JprVrJBO.

[19] Kingma, D. P., and Ba, J., "Adam: A Method for Stochastic Optimization," , 2017.

[20] Croes, G. A., "A Method for Solving Traveling-Salesman Problems," *Operations Research*, Vol. 6, No. 6, 1958, pp. 791–812. URL http://www.jstor.org/stable/167074.

[21] Antonescu, M., and Bîră, C., "Discrete Gravitational Search Algorithm (DGSA) applied for the Close-Enough Travelling Salesman Problem (TSP / CETSP)," *2019 International Semiconductor Conference (CAS)*, 2019, pp. 123–126. https://doi.org/10.1109/SMICND.2019.8923719.

[22] Applegate, D., Bixby, R., Chvatal, V., and Cook, W., "Concorde TSP solver," , 2006.