Weight Uncertainty in Transformer Network for the Traveling Salesman Problem

Jie Zhao[1]*, Biwei Xie[2,1,✉] and Xingquan Li[3,1,✉]
[1]Peng Cheng Laboratory, Shenzhen, China
[2]Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[3]School of Mathematics and Statistics, Minnan Normal University, Zhangzhou, China
Email: *zhaoj@pcl.ac.cn, ✉xiebiwei@ict.ac.cn, ✉fzulxq@gmail.com

2023 International Symposium of Electronics Design Automation (ISEDA) | 979-8-3503-0451-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/ISEDA59274.2023.10218491

***Abstract*—Traveling Salesman Problem (TSP) and its variations have many applications, especially for the case of circuit design where the ordering is crucial for the final design quality. Recently the deep learning (DL) based solvers were found successful in solving TSP, and have been applied to solve practical circuit design problems. However as the problem size increases, the gap between the optimal solution and the one predicted by DL based solver increases. For practical applications where the following decision making strongly relies on the solver quality, knowing the model uncertainly is very important. In the present work, a generalized spike and slab distribution (gSaS) was introduced to model the weight uncertainty for the attention based neural network TSP solver. Instead of minimizing the variational free energy or the expected lower bound on the marginal likelihood in the traditional variational inference, we directly minimize the loss function of the original model with respected to the hyper parameters of gSaS. Numerical experiments indicate that our proposed method can predict reasonable model uncertainty without sacrificing any test accuracy.**

***Index Terms*—Uncertainty quantification, generalized spike and slab distribution, transformer network, traveling salesman problem, variational inference**

I. Introduction

Traveling Salesman Problem (TSP) and its variations have numerous applications, including job sequencing, computer wiring, crystallography, vehicle routing, circuit design, etc. Given the coordinates of some points, the goal of the TSP problem is to find a shortest path that visits each point exactly once and returns to the starting point. In circuit design, a direct application of the TSP is in the drilling problem for printed circuit boards (PCBs). To connect a conductor on one layer with a conductor on another layer, or to position the pins of integrated circuits, holes have to be drilled through the board. The objective is to minimize the travel time for the machine head to drill holes [1]. The precedence constrained TSP was also used to determine the component placement sequence on the PCB [2]. In physical design of integrated circuits, global and detailed routing are critical stages involving the determination of the interconnected paths of each net on a circuit while satisfying certain constraints. Recently, Reinforcement Learning (RL) based TSP solver was used to construct Rectilinear Steiner Minimum Tree which is extensively used in global routing [3]. Attention based encoder-decoder model for TSP was also used to generate the routing sequence in the detailed routing stage [4], [5].

A. TSP solvers

The exact algorithms can find an optimal solution but they are limited to extremely small problem size in practice, since the TSP is an NP-hard problem. Heuristic approaches provide a TSP solution with a much shorter computational time compared to exact solvers, but can not guarantee optimality. Considerable effects have been devoted to improve the solution quality and reduce the computational time. Many useful connections between statistical mechanics and a wide variety of combinatorial problems, including TSP have also been discussed [6]. For example, the physical system analogy phase transition in TSP has been identified which can be useful to identify and generate hard to solve instances [7], [8], the renormalization group which is a very powerful tool in physics for treating systems with many length scales has been proved extremely powerful in treating TSP [9]. Recently, many works are devoted to build TSP solvers based on Deep Learning (DL). The first neutral network designed to solve TSP was build from analog circuit [10]. Using modern DL, the Pointer Network which combines recurrent neural networks and attention mechanism was proposed as a general model that could solve TSP [11]. The model is trained in a supervised way with approximate TSP solutions. Motivated by this pioneer work, several improvement and generation of this model were performed [12], [13]. Graph neural network is also broadly used to solve graph-based problems, including TSP [14]–[17]. The permutation-invariant Pooling Network was designed to solve multiple TSP in Ref. [18].

Recently, the Transformer architecture originally developed for natural language processing [19] was proved very successful in various field including solving the combinatorial TSP [20]. For example, in Ref. [20], close to optimal results can be obtained for TSP problem with small size. Training is carried out with reinforce and a deterministic baseline, and the Transformer encoder and decoder is adopted. The model outperform the Pointer network, and proved very successful not only in standard TSP but also in their variants including Vehicle Routing Problem, Orienteering Problem, Prize Collecting TSP. Promising results can also be obtained with the Transformer encoder and decode sequentially with softmax modules together with a standard 2-Opt heuristic refinement [21]. In Refs. [22], [23], the authors use the

Transformer based network to learn to improve the heuristics, the performance and generalizability are further improved.

However, in practical applications known the optimal solution is far from enough. For example, in the detailed routing stage the best routing sequence with the lowest cost (tour length) not necessarily produce the best final routing solutions. Known the model uncertainly will give us a selection rule to choose a less optimal but good routing sequence. In addition, although the neural network based approaches for TSP perform closely to classical solvers when trained on small sizes, their generalization of the learnt policy to larger instances is still problematic [24], [25] and the performance of the DL based solver decrease rapidly as the problem size increases even if we train them with the same problem size. Thus, the estimate of the model uncertainties are more and more important, especially when the following decision making process is strongly rely on the output from the DL based solver.

B. Uncertainty quantification for neural networks

Bayesian models offer a mathematically grounded framework to quantify model uncertainty. Typically it is relies on Markov chain Monte Carlo methods (MCMC) to sample parameter space and obtain posterior predictions from the product of the likelihood function and prior distribution without any assumption on the posterior distribution of parameters. However, MCMC is computational too heavy and limited to models with very small parameter size. In practice for neural networks, variational inference is usually adopted. Rather than using sampling, the variational inference is to use optimization. First we posit a family of approximate distributions of the latent variables, then we try to find the member of that family that minimizes the variational free energy or the expected lower bound on the marginal likelihood [26]. The choice of approximate posterior distribution is one of the core problems in variational inference. The original derivation relied on the use of a Gaussian approximating distribution with a diagonal covariance matrix [27], [28]. In Ref. [29], the authors generalized the approximating distribution and proposed a scale mixture priors combined with a diagonal Gaussian posterior. Since these distributions can not capture the posterior correlations between parameters, several attempts devoted to overcome this problem. The matrix variate Gaussian distribution was introduced to capture the correlations between specific parameters [30], [31]. Normalizing flows in which a simple initial density is transformed into a more complex one by applying a sequence of invertible transformations until a desired level of complexity is attained was also introduced to improve the approximate posterior distributions [32]–[35]. In Ref. [36], the authors proved that dropout training in deep neural networks can be viewed as approximate Bayesian inference in deep Gaussian processes, so it is an alternative way to model uncertainty in DL.

C. Our contributions

In the present work, we focused on modeling the network uncertainty for DL TSP solver based on transformer architecture. We propose a model called gSaS transformer which can be viewed as a cheap way to realize the Bayesian inference. The main contributions in this work are summarized as follows:

- We proposed to model the weight uncertainty by a generalized spike and slab distribution (gSaS), where the hyperparameters of the gSaS distribution are learnt from training data. Thus the search space is composed of all candidate networks with different assignments of weight values that accomplish the computation task. We search for an optimal random network ensemble contains subnetworks of the original full network, which further allows for capturing uncertainty in the hypothesis space.
- Since the entropy has no analytic form for gSaS distribution, the traditional variational free energy or the expected lower bound on the marginal likelihood for hyperparameters optimization is not valid. Here we directly minimize the loss of the original model $\mathcal{L}(\theta|\boldsymbol{s}) = \mathbb{E}_{p_\theta(\boldsymbol{\pi}|\boldsymbol{s})}\left[L(\boldsymbol{\pi})\right]$. Thus we significantly simplified the loss calculation compared with the traditional variational inference.
- A mean field method combined with Monte Carlo sampling is adopted to do the forward propagation for the network. Unlike the standard back-propagation for each particular weight, a generalized back-propagation at the weight distribution level is used to minimize the objective function.

The proposed method only need a minimum modification of the existing model and without sacrificing test accuracy. We compare the model prediction and model uncertainty obtained from our method and the one from dropout for different TSP problem size.

The model is developed in Sec. II. Section III contains the numerical results obtained from our method. In Sec. IV we summarize the results and present a brief outlook for future studies.

II. The Method

A. Transformer for TSP

We start from the attention model for TSP proposed in Ref. [20]. A problem instance s can be defined as a graph with n nodes, where node $i \in \{1, \cdots, n\}$ is represented by its coordinate $\boldsymbol{x}_i$ and the graph is fully connected. The solution $\pi = (\pi_1, \cdots, \pi_n)$ is a permutation of the nodes, so $\pi_i \in \{1, \cdots, n\}$ and $\pi_i \neq \pi_{i'}$, $\forall t \neq t'$. The attention based encoder-decoder model defines a stochastic policy $p(\pi|s)$ for selecting a solution π given a problem instance s. It is parameterized by $\boldsymbol{\theta}$ as

$$p_{\boldsymbol{\theta}}(\pi|s) = \prod_{i=1}^{n} p_{\boldsymbol{\theta}}(\pi_i|s, \pi_{1:i-1}). \tag{1}$$

In the original model, the encoder computes initial d_h-dimensional node embeddings $h_i^{(0)} = W^x \boldsymbol{x}_i + b^x$, where W^x and b^x are parameters for the linear projection and $\boldsymbol{x}_i$ is two dimensional input node coordinates. The embeddings

are updated using N attention layers, each consisting of two sublayers. The node embeddings produced by layer l

$$h_i^{(l)} = \mathrm{BN}^{(\mathrm{l})}\left[\hat{\mathrm{h}}_{\mathrm{i}} + \mathrm{FF}^{(\mathrm{l})}(\hat{\mathrm{h}}_{\mathrm{i}})\right], \tag{2}$$

$$\hat{h}_i = \mathrm{BN}^{(\mathrm{l})}\left[\mathrm{h}_{\mathrm{i}}^{(\mathrm{l}-1)} + \mathrm{MHA}_{\mathrm{i}}^{(\mathrm{l})}(\mathrm{h}_1^{(\mathrm{l}-1)}, \cdots \mathrm{h}_{\mathrm{n}}^{(\mathrm{l}-1)})\right]. \tag{3}$$

The layer index l indicates that the layers do not share parameters. BN denote the batch normalization, MHA refers to the multi-head attention layer

$$\mathrm{MHA}_{\mathrm{i}}(\mathrm{h}_1, \cdots \mathrm{h}_{\mathrm{n}}) = \sum_{\mathrm{m}=1}^{\mathrm{M}} \mathrm{W}_{\mathrm{m}}^{\mathrm{O}} \mathrm{h}'_{\mathrm{im}}, \tag{4}$$

$$h'_{im} = \sum_j a_{ij} \boldsymbol{v}_j, \quad a_{ij} = \frac{e^{u_{ij}}}{\sum_{j'} e^{u_{ij'}}}, \tag{5}$$

$$u_{ij} = \begin{cases} \frac{\boldsymbol{q}_i^{\mathrm{T}} \boldsymbol{k}_j}{\sqrt{d_k}}, & \text{if i adjacent to j} \\ -\infty, & \text{otherwise,} \end{cases} \tag{6}$$

where $\boldsymbol{q}_i$, $\boldsymbol{k}_i$, and $\boldsymbol{v}_i$ are called query, key and value respectively, which are obtained by projecting the embedding h_i. d_k is the query/key dimensionality.

$$\boldsymbol{q}_i = W^Q \boldsymbol{h}_i, \quad \boldsymbol{k}_i = W^K \boldsymbol{h}_i, \quad \boldsymbol{v}_i = W^V \boldsymbol{h}_i. \tag{7}$$

W^Q, W^K, W^V, and W_m^O are training parameters. The expression for the node-wise fully connected feed-forward layer reads

$$\mathrm{FF}(\hat{h}_i) = W^{F,1} \cdot \mathrm{ReLu}(W^{F,0} \hat{h}_i + b^{F,0}) + b^{F,1}. \tag{8}$$

Decoding happens sequentially, and at time step $t \in \{1, \cdots, n\}$, the decoder outputs the node π_i based on the embeddings from the encoder and the output $\pi_{i'}$ generated at time $t' < t$. The final output probability

$$p_{\boldsymbol{\theta}}(\pi_t = i | \boldsymbol{s}, \boldsymbol{\pi}_{1:t-1}) = \frac{e^{u_{(c)i}}}{\sum_j e^{u_{(c)j}}}, \tag{9}$$

where $u_{(c)i}$ are compatibilities calculated as

$$u_{(c)j} = \begin{cases} C \cdot \frac{\boldsymbol{q}_c^{\mathrm{T}} \boldsymbol{k}_j}{\sqrt{d_k}}, & \text{if} j \neq \pi_{t'}, \forall t' < t \\ -\infty, & \text{otherwise,} \end{cases} \tag{10}$$

where $\boldsymbol{q}_c$ and $\boldsymbol{k}_j$ are query from the context node and key for each node from the final multi-head attention layer of the decoder. The loss of the model is defined as $\mathcal{L}(\theta|\boldsymbol{s}) = \mathbb{E}_{p_\theta(\boldsymbol{\pi}|\boldsymbol{s})}[L(\boldsymbol{\pi})]$, where $L(\boldsymbol{\pi})$ is the tour length. The network parameters are optimized by gradient descent using reinforce gradient estimator with baseline $b(\boldsymbol{s})$

$$\nabla \mathcal{L}(\theta|\boldsymbol{s}) = \mathbb{E}_{p_\theta(\boldsymbol{\pi}|\boldsymbol{s})}\left\{[L(\boldsymbol{\pi}) - b(\boldsymbol{s})] \nabla \log p_\theta(\boldsymbol{\pi}|\boldsymbol{s})\right\}. \tag{11}$$

The greedy rollout baseline is adopted in our present work. Details of the model can be found in Ref. [20].

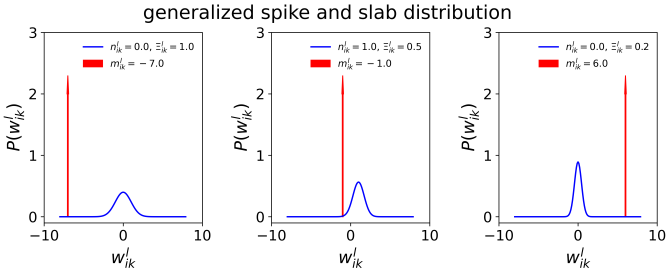

Fig. 1. Schematic plot of the generalized spike and slab (gSaS) distributions.

B. gSaS Transformer

The general approach to uncertainty quantification in machine learning is based on the ensemble method, which is a technique that makes predictions using multiple machine learning models and outputs a weighted average as the final prediction [37]. The original model in Ref. [20] itself provide the possibility to estimate the uncertainty from the decode process where we can sample the action at each step from the final output probability $p_\theta(\boldsymbol{\pi}|\boldsymbol{s})$. To capture the uncertainty from the model parameters, we propose to model the weight by a generalized spike and slab (SaS) distribution

$$\begin{aligned} P(w_{ik}^l) =& \pi_{ik}^l \delta(w_{ik}^l - m_{ik}^l) \\ &+ (1 - \pi_{ik}^l) \mathcal{N}(w_{ik}^l | n_{ik}^l, \Xi_{ik}^l), \end{aligned} \tag{12}$$

where the discrete probability mass at m_{ik}^l defines the spike, and the slab is characterized by a Gaussian distribution with mean n_{ik}^l and variance Ξ_{ik}^l over a continuous domain. The schematic plot of the gSaS distributions are shown in Fig. 1. We use a mean field method to learn the generalized SaS parameters $\theta_{ik}^l \equiv (m_{ik}^l, n_{ik}^l, \Xi_{ik}^l, \pi_{ik}^l)$ for all layers. The first and second moments of the weights w_{ik}^l are given by

$$\mu_{ik}^l = \mathbb{E}\left[w_{ik}^l\right] = \pi_{ik}^l m_{ik}^l + (1 - \pi_{ik}^l) n_{ik}^l, \tag{13}$$

$$\begin{aligned} \varrho_{ik}^l =& \mathbb{E}\left[(w_{ik}^l)^2\right] \\ =& \pi_{ik}^l (m_{ik}^l)^2 + (1 - \pi_{ik}^l)\left[(n_{ik}^l)^2 + \Xi_{ik}^l\right], \end{aligned} \tag{14}$$

The central-limit theorem implies that the pre-activation follows approximately a Gaussian distribution $\mathcal{N}\left[z_i | G_i^l, (\Delta_i^l)^2\right]$, where the mean and the variance are given by

$$G_i^l = \frac{1}{\sqrt{N_{l-1}}} \sum_k \mu_{ki}^{l-1} h_k^{l-1}, \tag{15}$$

$$(\Delta_i^l)^2 = \frac{1}{N_{l-1}} \sum_k \left[\varrho_{ki}^{l-1} - (\mu_{ki}^{l-1})^2\right] (h_k^{l-1})^2. \tag{16}$$

Then the feed-forward transformation of the input signal can be reparameterized by $z_i^l = G_i^l + \epsilon_i^l \Delta_i^l$. ϵ_i^l is a layer and component-dependent standard Gaussian random variable with zero mean and unit variance, and quenched for every single training mini-epoch and the same value is used in both forward and backward computations Learning of the hyperparameter θ_{ik}^l can be achieved by gradient descent of the objective function

$$\Delta \theta_{ik}^l = -\eta \frac{\partial \mathcal{L}}{\partial z_i^{l+1}} \frac{\partial z_i^{l+1}}{\partial \theta_{ik}^l}. \tag{17}$$

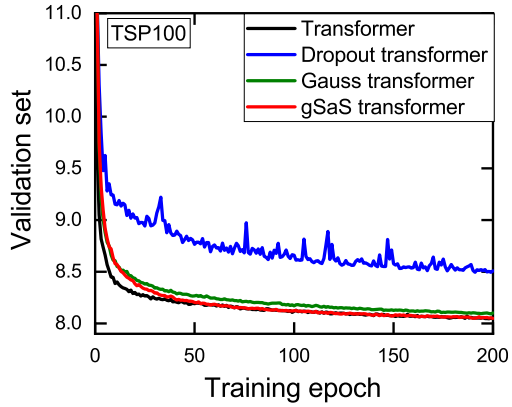

Fig. 2. Validation set average tour length as a function of the number of epochs for TSP of problem instances $n = 100$ nodes. The red line (gSaS transformer) denote the results obtained from our method with weight distribution of Eq. (12). The results from the original attention model in Ref. [20] is denoted as transformer. Dropout transformer refers to the results from the original attention model but with dropout layer included with dropout rate 0.2. The results obtained from our model but with Gaussian weight distribution $\mathcal{N}(w^l_{ik}|n^l_{ik}, \Xi^l_{ik})$ is also shown for comparison (Gauss transformer).

η denotes the learning rate. Note that unlike the standard back-propagation (BP) for each particular weight, our learning protocol can be thought of as a generalized back-propagation (gBP) at the weight distribution level. The optimization of the variational free energy for deep neural networks is computationally challenging, especially for the SaS like prior whose entropy has no analytic form as well [38]. Here we directly minimize the loss of the original model $\mathcal{L}(\theta|\boldsymbol{s}) = \mathbb{E}_{p_\theta(\boldsymbol{\pi}|\boldsymbol{s})}[L(\boldsymbol{\pi})]$. During learning the spike mass π should be clipped as max[0, min(1, π)], and the variance Ξ should be clipped as max(0, Ξ).

III. Numerical Experiments

The aim of the present work is to qualify the model uncertainty with the minimum modification of the original model and without sacrificing test accuracy, thus for the model hyperparameters we closely following the work of Ref. [20], i.e., the initial node embedding $d_h = 128$, the MHA layers uses $M = 8$ heads with dimensionality $d_h/M = 16$, the FF sublayer has one hidden sublayer with dimension 512 and ReLu activation, three MHA layers are adopted for the encoder etc. We replace all of the weight parameters $\{W^x, W^Q, W^K, W^V, W^O_m, W^{F,0}, W^{F,1}, \cdots\}$ both in encoder and decoder with the distribution of Eq. (12). In the present calculation, we omitted all of the bias terms. We initialize parameters m^l_{ik}'s and π^l_{ik}'s as zeros, n^l_{ik}'s are initialized with stand norm distribution, and Ξ^l_{ik}'s are initialized with uniform distribution (0, 0.01). We train 200 epochs using training data generated on the fly. Every epoch we process 2500 batches of 512 instance. The node locations are sampled uniformly at random in the unit square. We use a constant learning rate $\eta = 8 \times 10^{-4}$.

In Fig. (2) we present the predicted average tour length as a function of the number of epochs for a validation set of size 10000. The problem size is $n = 100$ nodes. The red line (gSaS transformer) denote the results from our model, and the black line (transformer) denote the results obtained from the original model. The learning rate $\eta = 10^{-4}$ was adopted for the BP calculations. As we can see from Fig. (2), start from 50 epochs, our predicted average tour length agrees very well with the original model. For the purpose of uncertainty qualification, we should maintain the test accuracy of the original model as much as possible. In this sense our model is a good candidate for the purpose of model uncertainty qualification. For comparison we also show the results from the original model but with dropout layer included with dropout rate 0.2 (Dropout transformer). Dropout was usually used as a simple and effective way to estimate the uncertainty for neutral networks. However for the specific case of TSP discussed here, the including of the dropout layer can significantly degrade the validation accuracy. In Fig. (2), we also show the results obtained by replace the distribution of Eq. (12) with Gaussian distribution $\mathcal{N}(w^l_{ik}|n^l_{ik}, \Xi^l_{ik})$ (Gauss transformer). As shown in Fig. (2), the predicted average tour length is lager than the one with the generalized SaS distribution Eq. (12), but still much better than the one from dropout.

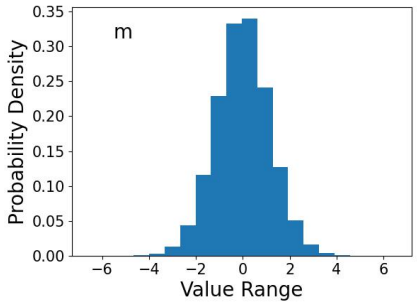

Fig. 3. The final distribution of the trained model parameter m^l_{ik}.

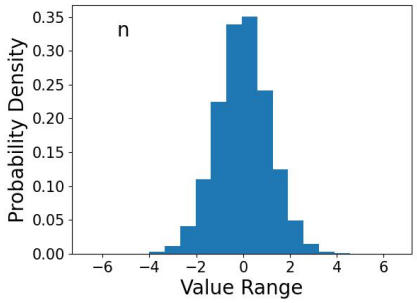

Fig. 4. The final distribution of the trained model parameter n^l_{ik}.

In Figs. (3) and (4) we show the distributions of the final trained parameters m^l_{ik}'s and n^l_{ik}'s respectively for the problem size of $n = 100$. Both distributions follow the similar Gaussian distribution which is a consequence of the central-limit theorem. In Fig. (5) we show the distribution of the trained parameters π^l_{ik}'s. An L-shaped distribution is observed, with the highest probability appears at the extreme $\pi = 1$, suggesting that the corresponding connection either very important and contribute less uncertainty ($m^l_{ik} \neq 0$)

TABLE I

THE EVALUATION SET AVERAGE TOUR LENGTH TOGETHER WITH THE COORRESPONDING MODEL UNCERTAINTY FOR TSP OF PROBLEM INSTANCES $n = 20, 50, 100, 150$, AND 200 NODES. GSAS TRANSFORMER DENOTE THE RESULTS OBTAINED FROM OUR METHOD WITH WEIGHT DISTRIBUTION OF EQ. (12). THE RESULTS FROM THE ORIGINAL ATTENTION MODEL IN REF. [20] ARE DENOTED AS TRANSFORMER. DROPOUT TRANSFORMER REFERS TO THE RESULTS FROM THE ORIGINAL ATTENTION MODEL BUT WITH DROPOUT LAYER INCLUDED WITH DROPOUT RATE 0.2. THE RESULTS OBTAINED FROM OUR MODEL BUT WITH GAUSSIAN WEIGHT DISTRIBUTION $\mathcal{N}(w_{ik}^l | n_{ik}^l, \Xi_{ik}^l)$ (GAUSS TRANSFORMER) ARE ALSO SHOWN FOR COMPARISON.

	Transformer	Dropout transformer	Gauss transformer	gSaS transformer
TSP20	3.8400 ± 0.0002	3.8788 ± 0.0006	3.8438 ± 0.0008	3.8405 ± 0.0003
TSP50	5.7823 ± 0.0007	6.0757 ± 0.0017	5.8177 ± 0.0042	5.7876 ± 0.0011
TSP100	8.0906 ± 0.0014	8.9700 ± 0.0029	8.1558 ± 0.0156	8.0886 ± 0.0024
TSP150	9.9594 ± 0.0022	11.3113 ± 0.0035	10.0240 ± 0.0350	9.9405 ± 0.0066
TSP200	11.5139 ± 0.0022	14.0281 ± 0.0050	11.8284 ± 0.0489	11.5425 ± 0.0116

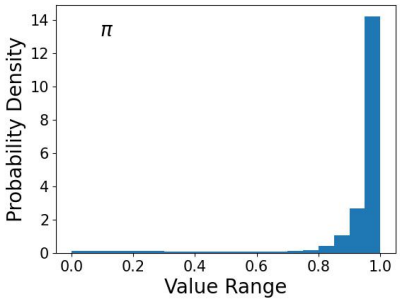

Fig. 5. The final distribution of the trained model parameter π_{ik}^l.

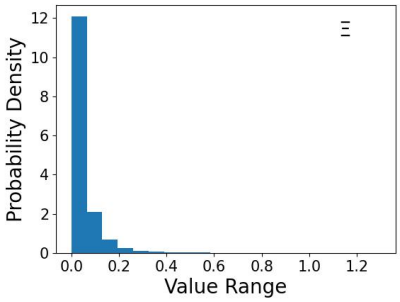

Fig. 6. The final distribution of the trained model parameter Ξ_{ik}^l.

or the corresponding connection is unimportant and thus can be pruned ($m_{ik}^l = 0$). The corresponding connection will contribute more to the model uncertainty as π decreases. Fig. (6) shows the distribution of the variance parameter Ξ. Still it shows an L-shape with the highest probability located at the $\Xi_{ik}^l = 0$ side. Note that if $\Xi_{ik}^l = 0$, the generalized SaS distribution reduces to a distribution with two delta peaks.

In Table I, we list the evaluated average tour length together with the corresponding predicted model uncertainty for an evaluation set of size 10000. transformer denotes the results from the original model with point estimates of neural network parameters. Dropout transformer refers to the results from the dropout out method with dropout rate set to 0.2. gSaS transformer denotes the results obtained from our model with the distribution of Eq. 12, while the one with pure Gaussian distribution is denoted as Gauss transformer. For the case of very small problem size of $n = 20$, the attention model can predict nearly optimal solutions thus gives a very small model uncertainty. All of the above tested method can obtain similar results. As the problem size increases, the performance of the attention model degrades and the derivatives between these methods increases. Without taken into account the uncertainty from the weight parameters, the model certainly can not predict reasonable model uncertainty. For example $n = 200$ have larger gap with the optimal solutions than the case of $n = 150$, that we should have lager model uncertainty, but the model uncertainty obtained from the decoding stage (transformer) gives the same model uncertainty. Dropout method predicted a much reasonable model uncertainty but unfortunately the predicted average tour lengths are lager than the one from the original attention model, which is consistent with the results for the validation set presented in Fig. (2). In contrast, our method with generalized SaS distribution can gives almost the same average tour length with the original attention model, but still predicted a reasonable model uncertainty. For the case of pure Gaussian distribution, the predicted model uncertainties are even larger than the one with generalized SaS distribution, with the cost of sacrificing a little bit of the test accuracy.

IV. CONCLUSION

A generalized spike and slab distribution (gSaS) was introduced to model the weight uncertainty for the attention based neural network Traveling Salesman Problem (TSP) solver. Instead of minimizing the variational free energy or the expected lower bound on the marginal likelihood in the traditional variational inference, we directly minimize the loss function of the original model with respected to the hyper parameters of gSaS. Numerical experiments on the case of $n = 20, 50, 100, 150$, and 200 indicate that our proposed method can predict reasonable model uncertainty without sacrificing any test accuracy. The method presented here is simple and only need a minimum modification of the origin model. Note that although our method have introduced four times of the model parameter compared with the original model (actually less than four times since we have omitted the bias term) at the training stage, the same amount of model parameter (less when taken into account the bias term) is included in the model at the test stage after sampling from the trained distribution. In addition, we can further compress the model by eliminate the parameters with $\pi \approx 1$ and $m \approx 0$

without loss the test accuracy. The extension of this method to other deep learning networks and to other problems is straightforward.

During training the mean field approximation is adopted where we assume the pre-activations follow approximately a Gaussian distribution. A direct evaluation of the connected four-point correlators of the pre-activations $\mathbb{E}[z^l_{i_1} z^l_{i_2} z^l_{i_3} z^l_{i_4}]\big|_{\text{connected}} \equiv \mathbb{E}[z^l_{i_1} z^l_{i_2} z^l_{i_3} z^l_{i_4}] - \mathbb{E}[z^l_{i_1} z^l_{i_2}]\mathbb{E}[z^l_{i_3} z^l_{i_4}] - \mathbb{E}[z^l_{i_1} z^l_{i_3}]\mathbb{E}[z^l_{i_2} z^l_{i_4}] - \mathbb{E}[z^l_{i_1} z^l_{i_4}]\mathbb{E}[z^l_{i_2} z^l_{i_3}] = G^l \neq 0$ indicate the mean field approximation is not strictly fulfilled. Indeed $G^l \to 0$ when the layer width $n_l \to \infty$ [39]. In practice for large layer width n_l, G_l is small we can go beyond mean-field approximation by introducing a nearly-Gaussian distribution. Such work is in progress.

ACKNOWLEDGMENT

This work is supported in part by the Major Key Project of PCL (No. PCL2021A08), the Natural Science Foundation of Fujian Province (No. 2020J05161).

REFERENCES

[1] M. Grötschel, M. Jünger, and G. Reinelt, *Optimal control of plotting and drilling machines: A case study*, ZOR Methods and Models of Operations Research Methods and Models of Operations Research **35**, 61 (1991).

[2] E. Duman and I. Or, *Precedence constrained TSP arising in printed circuit board assembly*, International Journal of Production Research **42**, 67 (2004).

[3] J. Liu, G. Chen, and E. F. Young, *REST: Constructing Rectilinear Steiner Minimum Tree via Reinforcement Learning*, in *2021 58th ACM/IEEE Design Automation Conference (DAC)* (IEEE, 2021).

[4] H. Liao, Q. Dong, X. Dong, W. Zhang, W. Zhang, W. Qi, E. Fallon, and L. B. Kara, *Attention Routing: track-assignment detailed routing using attention-based reinforcement learning*, arXiv:2004.09473 (2020a).

[5] H. Liao, Q. Dong, W. Qi, E. Fallon, and L. B. Kara, *Track-Assignment Detailed Routing Using Attention-based Policy Model With Supervision*, in *Proceedings of the 2020 ACM/IEEE Workshop on Machine Learning for CAD* (ACM, 2020) pp. 105–110.

[6] D. S. Dean, D. Lancaster, and S. N. Majumdar, *Statistical mechanics of combinatorial optimization problems with site disorder*, Physical Review E **72**, 026125 (2005).

[7] I. P. Genta and T. Walsh, *The TSP phase transition*, Artificial Intelligence **88**, 349 (1996).

[8] M. Cárdenas-Montes, *Creating hard-to-solve instances of travelling salesman problem*, Applied Soft Computing **71**, 268 (2018).

[9] J. Houdayer and O. C. Martin, *Renormalization for Discrete Optimization*, Physical Review Letters **83**, 1030 (1999).

[10] J. J. Hopfield and D. W. Tank, *"Neural" computation of decisions in optimization problems*, Biological Cybernetics **52**, 141 (1985).

[11] O. Vinyals, M. Fortunato, and N. Jaitly, *Pointer Networks*, in *Advances in Neural Information Processing Systems*, Vol. 28, (Curran Associates, Inc., 2015).

[12] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, *Neural Combinatorial Optimization with Reinforcement Learning*, arXiv:1611.09940 (2016).

[13] M. Nazari, A. Oroojlooy, L. Snyder, and M. Takac, *Reinforcement Learning for Solving the Vehicle Routing Problem*, in *Advances in Neural Information Processing Systems*, Vol. 32, (Curran Associates, Inc., 2018).

[14] H. Dai, E. B. Khalil, Y. Zhang, B. Dilkina, and L. Song, *Learning Combinatorial Optimization Algorithms over Graphs*, arXiv:1704.01665 (2017).

[15] A. Nowak, S. Villar, A. S. Bandeira, and J. Bruna, *A Note on Learning Algorithms for Quadratic Assignment with Graph Neural Networks*, Proceedings of the 34th International Conference on Machine Learning (2017).

[16] Z. Xing and S. Tu, *A Graph Neural Network Assisted Monte Carlo Tree Search Approach to Traveling Salesman Problem*, IEEE Access **8**, 108418 (2020).

[17] C. K. Joshi, T. Laurent, and X. Bresson, *An Efficient Graph Convolutional Network Technique for the Travelling Salesman Problem*, arXiv:1906.01227 (2019).

[18] Y. Kaempfer and L. Wolf, *Learning the Multiple Traveling Salesmen Problem with Permutation Invariant Pooling Networks*, arXiv:1803.09621 (2018).

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, *Attention is All you Need*, in *Advances in Neural Information Processing Systems*, Vol. 30, (Curran Associates, Inc., 2017).

[20] W. Kool, H. van Hoof, and M. Welling, *Attention, Learn to Solve Routing Problems!*, arXiv:1803.08475 (2018).

[21] M. Deudon, P. Cournut, A. Lacoste, Y. Adulyasak, and L.-M. Rousseau, *Learning Heuristics for the TSP by Policy Gradient*, in *Integration of Constraint Programming, Artificial Intelligence, and Operations Research* (Springer International Publishing, 2018) pp. 170–181.

[22] Y. Wu, W. Song, Z. Cao, J. Zhang, and A. Lim, *Learning Improvement Heuristics for Solving the Travelling Salesman Problem*, arXiv:1912.05784 (2019).

[23] X. Bresson and T. Laurent, *The Transformer Network for the Traveling Salesman Problem*, arXiv:2103.03012 (2021).

[24] W. Ouyang, Y. Wang, S. Han, Z. Jin, and P. Weng, *Improving Generalization of Deep Reinforcement Learning-based TSP Solvers*, in *2021 IEEE Symposium Series on Computational Intelligence (SSCI)* (IEEE, 2021).

[25] C. K. Joshi, Q. Cappart, L.-M. Rousseau, and T. Laurent, *Learning the travelling salesperson problem requires rethinking generalization*, Constraints **27**, 70 (2022).

[26] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, *Variational Inference: A Review for Statisticians*, Journal of the American Statistical Association **112**, 859 (2017).

[27] G. E. Hinton and D. van Camp, *Keeping the neural networks simple by minimizing the description length of the weights*, in *Proceedings of the sixth annual conference on Computational learning theory - COLT '93* (ACM Press, 1993).

[28] A. Graves, *Practical Variational Inference for Neural Networks*, in *Advances in Neural Information Processing Systems 24 (NIPS 2011)* (2011).

[29] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, *Weight Uncertainty in Neural Networks*, in *Proceedings of the 32nd International Conference on Machine Learning* (2015).

[30] D. Barber and C. M. Bishop, *Ensemble Learning in Bayesian Neural Networks*, in *Neural Networks and Machine Learning* (1998) pp. 215–237.

[31] C. Louizos and M. Welling, *Structured and Efficient Variational Deep Learning with Matrix Gaussian Posteriors*, in *Proceedings of the 33rd International Conference on Machine Learning* (2016).

[32] D. J. Rezende and S. Mohamed, *Variational Inference with Normalizing Flows*, Proceedings of the 32nd International Conference on Machine Learning (2015).

[33] C. Louizos and M. Welling, *Multiplicative Normalizing Flows for Variational Bayesian Neural Networks*, Proceedings of the 34th International Conference on Machine Learning (2017).

[34] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling, *Improved Variational Inference with Inverse Autoregressive Flow*, in *30th Conference on Neural Information Processing Systems (NIPS 2016)* (2016).

[35] D. P. Kingma and M. Welling, *Auto-encoding variational Bayes*, in *In International Conference on Learning Representations* (2014).

[36] Y. Gal and Z. Ghahramani, *Dropout as a Bayesian approximation: representing model uncertainty in deep learning*, in *ICML16: Proceedings of the 33rd International Conference on International Conference on Machine Learning*, Vol. 48 (2016) pp. 1050–1059.

[37] G. Palmer, S. Du, A. Politowicz, J. P. Emory, X. Yang, A. Gautam, G. Gupta, Z. Li, R. Jacobs, and D. Morgan, *Calibration after bootstrap for accurate uncertainty quantification in regression models*, npj Computational Materials **8**, 115 (2022).

[38] C. Li and H. Huang, *Learning Credit Assignment*, Physical Review Letters **125**, 178301 (2020).

[39] D. A. Roberts, S. Yaida, and B. Hanin, *The Principles of Deep Learning Theory* (Cambridge University Press, 2022).