Variance-Preserving Stochastic Differential Equation Algorithm for the Traveling Salesman Problem

Yuchen Wang* and Hiroyuki Ebara†
*Graduate School of Science and Engineering, Kansai University, 3-3-35, Yamate-cho, Suita-shi, Osaka, 564-8680 Japan
Email: k595345@kansai-u.ac.jp
†Faculty of Engineering Science, Kansai University, 3-3-35, Yamate-cho, Suita-shi, Osaka, 564-8680 Japan
Email: ebara@kansai-u.ac.jp

***Abstract*—The traveling salesman problem (TSP) stands as a classic combinatorial optimization problem with widespread research interest and practical relevance. Despite its computational complexities, recent advancements in deep learning and stochastic differential equations (SDE) have introduced new methodologies to improve TSP solutions. In this study, we introduce the vertex-conditioned forward path generation (V-FPG) method, building upon the variational path sampling with differential equations (VPSDE) framework. V-FPG integrates urban vertex graphs with optimal path datas, utilizing SDEs and Gaussian noise to generate candidate paths. Backward optimization with a scoring function ensures clear guidance during path generation. Experimental results demonstrate the superior accuracy and robustness of our method across random point tests and the TSP-LIB dataset.**

***Index Terms*—Combinatorial Optimization, Deep Learning, Generative Models, Diffusion Models**

I. Introduction

The traveling salesman problem (TSP), a classic problem in combinatorial optimization, holds significant value in various practical applications. As illustrated in Fig. 1, given a set of cities and the distances (or costs) between them, the task of the salesman is to find the shortest path that allows him to depart from the starting city, visit each city exactly once, and return to the starting city, while minimizing the total length of the path. In particular, TSP whose vertices are on a 2-dimensional Euclidean plane and whose cost of each edge is represented by the Euclidean distance between two cities is called 2D Euclidean TSP.

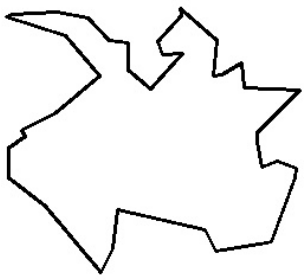

Fig. 1: Example of a solution for TSP.

While TSP finds widespread applications, its computational complexity exponentially rises with increasing problem size. This computational burden presents significant challenges in efficiently finding optimal or near-optimal solutions. Deep learning and generative models have demonstrated immense potential in addressing combinatorial optimization problems. Diffusion models, an emerging class of generative models, have achieved remarkable results in fields such as image synthesis and text generation. Unique in their ability to generate complex data distributions from simple noise distributions, they offer an innovative approach to tackle TSP.

To overcome the inefficiency challenges of solving large-scale TSP instances, this paper proposes a method that combines diffusion models with good edge distribution to solve TSP. We design a novel optimization framework based on diffusion models, leveraging the good edge distribution generated by the diffusion model to guide the TSP search process. This approach aims to improve the efficiency and quality of TSP solutions, providing a new perspective for solving TSP instances.

II. Prior Research

A. Traditional Algorithms for TSP

Since its inception, TSP has garnered extensive research attention. Traditional methods primarily focus on exact solving algorithms, such as branch and bound [1] which systematically explores the solution space by partitioning it into smaller subspaces. Another notable method is branch and cut [2], which combines the branch and bound framework with cutting planes to enhance solution quality.

Furthermore, various heuristic algorithms have been researched for solving TSP. Such as genetic algorithms [3], simulated annealing [4], tabu search [5], and the Lin-Kernighan algorithm [6]. These algorithms face challenges such as high computational complexity and inefficiency for large-scale problems.

Genetic algorithms [3] simulate the process of natural selection to evolve a population of potential solutions towards better solutions over successive generations . Simulated annealing [4] mimics the process of annealing in metallurgy, gradually decreasing the temperature to explore the solution space and escape local optima. Tabu employs memory structures to prevent revisiting previously explored solutions, guiding the search towards promising regions of the solution space . The Lin-Kernighan algorithm [6] is an iterative improvement

979-8-3503-5469-0/24/$31.00 ©2024 IEEE

heuristic that seeks to iteratively improve upon an initial solution by performing local search operations.

While these methods are not guaranteed to find the optimal solution, they offer efficient approximations for solving large-scale TSP instances. Each algorithm has its own strengths and limitations, contributing to the diverse landscape of TSP solving techniques.

B. Machine Learning for TSP

In recent years, with the development of machine learning and deep learning technologies, new solving methods have gained attention. Convolutional neural networks (CNNs) [7] and reinforcement learning [8], successful in image processing and pattern recognition, have also been applied to TSP solving. By transforming TSP into an image form, CNNs can learn specific patterns in images and generate corresponding solutions. However, these methods exhibit limited performance when dealing with complex and large-scale instances. Generative adversarial networks (GANs) [9], known for their prowess in image generation tasks, have been tried for generating TSP solutions [10]. While GANs generate new samples by learning data distributions, they may face challenges in generating feasible or suboptimal solutions for TSP. Transformers, celebrated for their success in natural language processing, have also been explored for TSP [11]. Transformers capture dependencies in sequence data through self-attention mechanisms, but their performance and efficiency in tackling combinatorial optimization problems still require enhancement. Pix2Pix [12] an image-to-image translation model primarily used for tasks like image restoration and colorization [13], has been explored in the context of TSP. In this framework, the generator network transforms images from one domain to another, while the discriminator network distinguishes between real and generated images. Through paired examples of input and output images, Pix2Pix effectively learns to translate images while preserving essential features.Tang et al. [14] proposed a method based on conditional generative adversarial networks for solving TSP. Despite the new solving approaches offered by machine learning methods, challenges persist in practical applications. Traditional machine learning methods often require extensive labeled data and substantial computational resources. This not only increases the complexity of the problem but also limits the interpretability of these models, making it difficult to explain the process of generating solutions to users. Furthermore, these methods may prove inadequate when handling TSP instances of varying scales and structures.

C. Diffusion model for TSP

To overcome these limitations, diffusion models have garnered attention, especially in generating complex high-dimensional data like images, sounds, and text. They originate from simple noise distributions and progressively approximate target data distributions, facilitating data generation [15]. In the field of image generation, diffusion models not only produce high-quality, clear images but also capture rich details. Given the limitations of traditional machine learning methods and the strengths of diffusion models, we propose employing the variance preserving stochastic differential equation (VPSDE) method for solving TSP. This method aims to enhance the accuracy of TSP solutions while maintaining a balance between model interpretability and efficiency.

The VPSDE method combines the strengths of stochastic differential equations and diffusion models, effectively handling TSP of different scales and structures while maintaining solution quality and stability [16]. Compared to GANs, diffusion models typically offer more stable and straightforward training. They do not require adversarial training between a generator and discriminator, reducing the likelihood of mode collapse during training. Due to the absence of adversarial training, diffusion models are generally less susceptible to mode collapse and can generate more diverse and rich solutions. They typically generate high-quality solutions with fewer training steps, enhancing efficiency in solving TSP instances.

The VPSDE method not only effectively handles large-scale TSP but also offers good interpretability and explainability. The model's generation process is stepwise, with each step improving the sample by introducing a small amount of noise and enhancing the edge distribution [17]. This stepwise improvement aids the model in more effectively capturing data distribution characteristics, allowing users to better understand how the model generates solutions and thereby producing higher-quality TSP solutions. Compared to discrete training processes, stochastic differential equations (SDE) offer a smoother and more continuous generation method, contributing to the model's ability to produce more continuous and natural solutions. The VPSDE method exhibits robustness, tolerating noise and uncertainty well. In summary, the VPSDE method offer a new perspective for solving the TSP, with high application value and research potential.

III. Proposed Method

Miki et al. proposed a CNN-based method for solving the planar TSP [18]. They represent TSP vertices and paths as images, learning the relationship between vertex images and optimal path images through CNN models. In their method, as illustrated in Fig. 2, the concept of a good edge distribution is crucial, representing the likelihood of each edge being selected as the optimal path. This lays the foundation for our subsequent optimization methods. In the following sections, we combine the VPSDE method with the concept of a good edge distribution to further optimize the TSP solution. Inspired by Miki et al.'s good edge distribution method, we further introduce the VPSDE as our main model, aiming to optimize the good edge distributions of our model further. To train and optimize the VPSDE model, we design a specific loss function.

A. Good-Edge Value (GEV)

The good-edge value is an approximate measure of whether an edge (i, j) is included in the optimal path. Edges with higher GEVs are more likely to be part of the optimal path. Unlike traditional shortest edge-first methods, the GEV-greedy

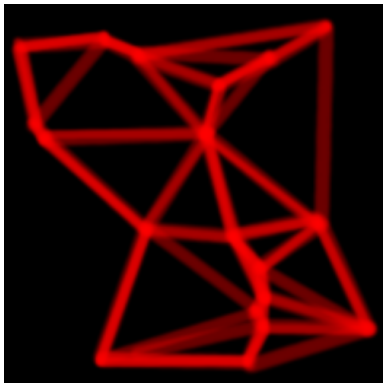

Fig. 2: Good Edge Distribution

method constructs solutions by prioritizing edges with high GEVs.

The GEV of an edge (v_{ij}) is represented by equation (1).

$$v_{ij} = \frac{1}{1+\tau_{ij}} \cdot \frac{\sum_{x=1}^{Sx}\sum_{y=1}^{Sy}\hat{f}(x,y)\cdot l_{ij}(x,y)}{\sum_{x=1}^{Sx}\sum_{y=1}^{Sy} l_{ij}(x,y)} \tag{1}$$

Here, τ_{ij} is the crossover penalty term used to reduce noise caused by overlapping with other edges. $l_{ij}(x,y)$ is an indicator function representing whether edge (i,j) passes through pixel (x,y).

- **EV-greedy method:** This method adopts a short edge priority approach, prioritizing edges with high GEVs to construct solutions. It selects edges based on their calculated GEVs estimated using a trained edge model that predicts the likelihood of each edge being included in the optimal path.
- **EV-2opt Method:** The traditional 2-opt method selects path combinations with shorter lengths by swapping two edges. In contrast, the EV-2opt method selects paths based on the high or low GEVs, leaning towards selecting edges with larger GEVs as neighboring solutions. The EV-2opt method is a local search algorithm that performs basic operations by swapping two edges to attempt to reduce path length. Unlike the traditional 2-opt method, the EV-2opt method selects edges for swapping based on GEVs, leaning towards swapping edges with higher GEVs.
- **Combining EV-greedy and EV-2opt Optimization Methods:** Neither the EV-greedy nor the EV-2opt methods guarantee that selected edges will not cross. To further improve solution accuracy, after obtaining solutions using EV-greedy or EV-2opt, we incorporate the traditional 2-opt exploration method, which we refer to as EV-greedy+2opt and EV-2opt+2opt methods.

B. Vertex-Conditioned Forward Path Generation (V-FPG)

VPSDE leverages the dynamic behavior of stochastic simulation systems to capture the random characteristics of data. The model consists of a drift term, describing the average change of a stochastic process, and a diffusion term, describing the strength and distribution of randomness. Its primary focus is maintaining variance characteristics to generate high-quality, detailed data such as city vertex images in TSP.

To learn the good edge distribution, we train the VPSDE model to approximate the real data distribution and ensure the generated good edge distribution image is as close as possible to the optimal path image. Our core training objective during this process is to minimize the loss function, which quantifies the difference between the model's predictions and the real data. Therefore, we propose the **Vertex-Conditioned Forward Path Generation (V-FPG)** method.

We employ V-FPG to model the mapping between the good edge distribution and the optimal path. During the training of the V-FPG model, our focus extends beyond the consistency between the good edge distribution image generated by the model and the optimal path image. We also emphasize the accurate capture of complex relationships in the city vertex image. To integrate information from the city vertex and the optimal path image, where the latter represents the shortest or optimal path between cities, we set clear optimization constraints for the model. The city vertex image serves as a condition, guiding path generation to ensure that the generated paths meet specific optimization criteria. This comprehensive approach ensures the model receives clear guidance when generating consideration ensures the model gains clear guidance when generating paths.

The core still uses SDE to describe system dynamics and approximates these dynamics through deep learning. SDE combines differential equations with stochastic processes, where randomness is introduced by noise terms. Given the GEV distribution function $f(x,y)$, the stochastic differential equation is:

$$dX_t = \alpha(t)\cdot b(X_t)dt + \alpha(t)\cdot\sigma(X_t)dW_t \tag{2}$$

Here, X_t represents the state at time t, $\alpha(t)$ is a time-dependent noise coefficient, $b(X_t)$ and $\sigma(X_t)$ are deterministic functions corresponding to drift and diffusion terms, respectively, and dW_t is the differential of the Brownian motion. $\alpha(t)$ is a stochastic process used to adjust drift and diffusion terms, enabling the model to adapt to data dynamics. In the forward process, the V-FPG model integrates information from the urban vertex image to generate candidate paths using SDE and Gaussian noise. In the reverse process, the model uses the optimal path image as constraints or criteria to optimize these candidate paths. Compared to traditional SDE and VPSDE methods solving image generation problems, V-FPG balances information between these two domains to generate paths that satisfy both the conditions of the urban vertex image and the constraints of the optimal path. This balance enables V-FPG to maintain rationality and stability in path generation, thereby avoiding unnecessary randomness. Such characteristics are particularly crucial for solving complex routing problems like TSP.

Forward Process

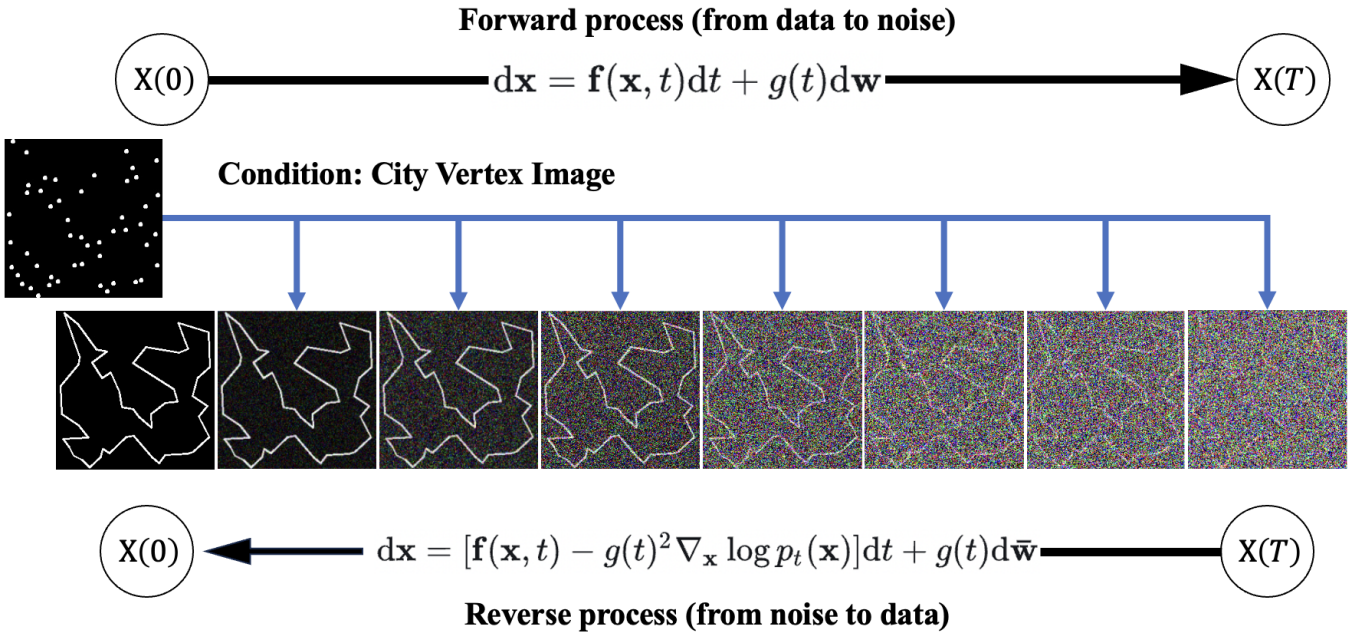

Fig. 3: V-FPG

Condition: Urban Vertex Image

In the forward process, the condition is the urban vertex image, which provides information about the order of cities the traveling salesman needs to visit. This condition, combined with the current sample x_t, jointly influences the sample generation.

Noise: Gaussian Noise z

During the forward process, Gaussian noise z introduces randomness and variability. It is combined with the current sample x_t to generate a new sample for the next time step. This randomness makes the generated samples non-deterministic, thereby providing the capability to explore potential solutions. The equation is represented as:

$$x_{t+1} = x_t + \text{drift} \times \Delta t + \text{diffusion} \times \sqrt{\Delta t} \times z \tag{3}$$

Here drift and diffusion are drift and diffusion terms determined by the Urban Vertex Image and the current sample x_t, Δt is the time step, and z is the Gaussian noise.

Reverse Process

In TSP, the core task of the reverse process is to optimize the generated paths to ensure they meet the specific constraints of TSP, such as visiting each city once, returning to the starting point, and minimizing the path length. This optimization process can be achieved by introducing a score function to find the optimal path. The score function, commonly used in probability density estimation and latent variable models to represent the model's derivative. In probability density estimation, the score function is the derivative of the log probability density function divided by the probability density function itself. In deep learning, it is often used in flow models to represent the gradient of the model's output. Mathematically, the score functionscore score is calculated based on time and the model's output:

$$\text{score} = \nabla_x \log p(x, t) \tag{4}$$

Here, $\log p(x, t)$ is the logarithm of the probability density given input x and time t. ∇_x denotes the gradient with respect to x.

Compared to traditional SDE and VPSDE methods, which typically focus on optimizing the quality of image generation, our approach concentrates on optimizing the quality and effectiveness of the TSP path, ensuring the rationality and shortest distance of the path.

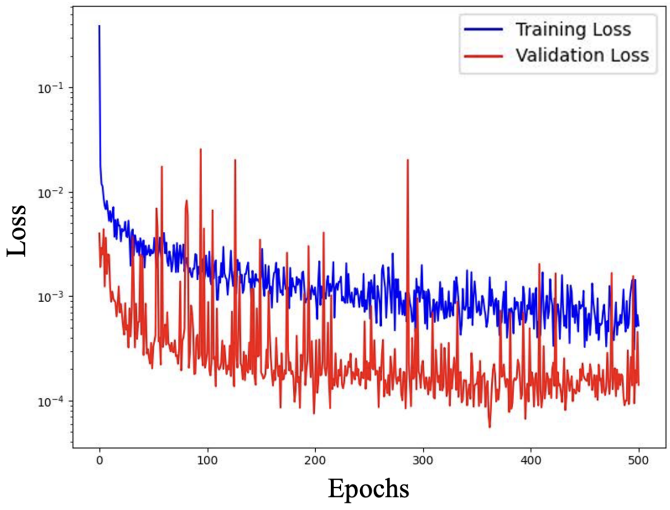

Fig. 4: Loss Function

C. Loss Function

Traditional Loss Function

In traditional loss function design, the primary objective is to ensure that the model accurately predicts the optimal path for TSP. To achieve this, mean squared error (MSE) is commonly used as a metric to measure the discrepancy between the predicted and true paths:

$$\mathcal{L} = \sum_{i=1}^{N} ||f(x_i, y_i) - \hat{f}(x_i, y_i)||^2 \tag{5}$$

Here, $f(x_i, y_i)$ represents the optimal path, while $\hat{f}(x_i, y_i)$ is the model's predicted GEV path. However, during the forward path generation, it is essential not only to consider the match between the predicted and true paths but also to incorporate the urban vertex image as a condition to guide the path generation. Such considerations imply that merely evaluating the discrepancy between predicted and true values may not fully capture the constraint conditions, potentially leading to reduced accuracy in constraint judgment by the loss function.

To ensure the model accurately predicts solutions to TSP, we propose a more comprehensive loss function that combines the discrepancy between model predictions and target data with the quality of the edge distribution:

$$\begin{aligned}\mathcal{L}(model_{xy}, model_{yx}, batch) =& \mathcal{L}_{score}(model_{xy}, x, y) \\ &+ \mathcal{L}_{score}(model_{yx}, y, x)\end{aligned} \tag{6}$$

Here $\mathcal{L}_{score}$ is the function evaluating the score function loss, defined as:

$$\begin{aligned}&\mathcal{L}_{score}(model, img, cond) = \\ &\text{reduce\_op}\left((\text{score} \cdot \text{std\_img}[:, None, None, None] + z)^2\right)\end{aligned} \tag{7}$$

Here, $score$ denotes the model's score for a given condition $cond$ and perturbed image perturbed_img. std_img represents the standard deviation of the input image, and z is noise. We use reduce_op for dimension reduction, which can be mean or sum, depending on the setting of the 'reduce_mean' parameter.

The design of the comprehensive loss function fully considers the role of the urban vertex image in path generation. This loss function not only quantifies the discrepancy between the model's path generation and the optimal path but also focuses on capturing information from the urban vertex image. It ensures consistency between the good edge distribution and the optimal path image while maximizing the utilization of information within the urban vertex image to improve the accuracy and efficiency of path generation. Achieving these goals allows the V-FPG model to leverage its advantages in complex routing problems.

Moreover, we evaluate the quality of the model's output good edge distribution through the score function loss $\mathcal{L}_{score}$. This ensures not only that the paths generated by the model meet the constraints of TSP but also enhances the accuracy and robustness of the paths. By considering both path accuracy and edge distribution quality, our comprehensive loss function provides a more holistic evaluation of model performance, thereby effectively optimizing the model and improving its performance in solving TSP.

IV. Experimental Evaluation

A. Computer Configuration

For our experiments, we utilized a computer with the specifications outlined in Table I. Python 3.8 served as the programming language, complemented by the PyTorch machine learning library.

TABLE I: Computer Configuration

Component	Description
CPU	13th Gen Intel (R) Core (TM) i9-13900KF CPU@3.6GHz
GPU	GeForce RTX 3090 Ti 24GB
Main Memory	64 GB
OS	Ubuntu 22.04

B. Experimental Setup

Our evaluation rigorously assessed the performance of diverse solutions across test problem instances. Evaluation metrics encompassed the total cost of the solution, average error rate of the solution, and computation time required for solving.

Initially, we trained the model over 500 learning epochs and then tested it on 200 randomly selected points. To further validate the robustness and generality of the model, additional tests were conducted using the TSP-LIB dataset.

C. Experimental Procedures

- **Model Training:** Our model was trained over 500 learning epochs. Each epoch utilized, a set of randomly generated TSP instance as training data.
- **Testing Setup:** Following the 500 epochs of training, tests were conducted on 200 randomly generated TSP instances. Accuracy assessment was conducted for paths spanning 20 to 200 cities in each test instance.
- **TSP-LIB Dataset Testing:** To further validate the effectiveness and generality of our approach, tests were conducted on the TSP-LIB dataset, which encompasses TSP instances of varying scales.

D. Evaluation Metrics

- **Path Accuracy:** In each test instance, we calculated the similarity between the model-generated path and the optimal path. Path accuracy was quantified using the formula:

$$\text{Accuracy} = \frac{\text{Generated Path Length}}{\text{Optimal Path Length}} \times 100[\%] \tag{8}$$

Accuracy was computed across various scales (20-200 cities), and the mean and standard deviation were reported.

E. Results Analysis

- **Random Point Testing:** Path accuracy was computed across 200 randomly generated TSP instances. The results indicate a high level of accuracy achieved by our method on these instances.
- **TSP-LIB Dataset Testing:** On the TSP-LIB dataset, our method demonstrated commendable performance across instances of all scales, nearing optimal solutions.

TABLE II: Average total error rate for random problem instances (%)

Instance Set	2-opt	greedy	EV-2opt	EV-greedy	EV-2opt+2-opt	EV-greedy+2-opt
rand20-A	3.326	19.288	**0.000**	**0.000**	**0.000**	**0.000**
rand20-B	0.341	23.175	3.283	**0.000**	**0.000**	**0.000**
rand50-A	6.451	10.464	**0.000**	**0.000**	**0.000**	**0.000**
rand50-B	5.160	16.071	3.802	**0.000**	0.465	**0.000**
rand75-A	5.029	15.727	3.519	**0.000**	0.236	**0.000**
rand75-B	6.284	14.547	**0.000**	**0.000**	**0.000**	**0.000**
rand100-A	7.793	22.424	1.947	**0.000**	0.143	**0.000**
rand100-B	7.106	17.708	0.507	0.026	**0.000**	**0.000**
rand150-A	6.524	22.786	0.575	**0.000**	0.041	**0.000**
rand150-B	7.643	25.941	**0.000**	**0.000**	**0.000**	**0.000**
rand200-A	8.056	10.643	5.088	12.615	**1.287**	2.647
rand200-B	7.143	20.441	**0.000**	**0.000**	**0.000**	**0.000**

TABLE III: Comparison of Average Error Rates with Previous Studies Using TSPLIB (%)

Instance Set	2-opt	greedy	EV-2opt	EV-greedy	EV-2opt+2-opt	EV-greedy+2-opt	Miki's method
eil51	4.554	16.667	7.747	6.338	**1.408**	2.582	2.958
berlin52	7.855	31.941	0.264	16.362	**0.000**	8.539	**0.000**
st70	4.741	17.037	9.304	7.704	2.785	1.481	**0.452**
eil76	7.230	10.409	8.364	8.364	3.122	1.859	**0.037**
pr76	4.944	29.762	15.262	6.718	3.146	1.918	**0.265**
rat99	6.557	21.222	9.422	18.167	2.502	4.707	**1.936**
kroA100	6.442	13.688	18.099	11.094	2.213	6.367	**0.477**
kroB100	6.608	16.585	7.878	3.089	3.501	1.391	**0.660**
kroC100	6.484	11.133	7.038	4.357	1.207	2.111	**0.104**
kroD100	5.276	14.812	18.516	21.269	3.633	3.541	**0.986**
kroE100	6.328	12.588	8.462	7.672	4.211	**2.379**	2.696
rd100	6.783	16.928	5.965	3.502	2.220	1.466	**1.320**
Eil101	7.234	24.483	14.452	9.221	3.212	**1.272**	1.932
lin105	6.445	16.601	6.139	1.739	1.841	1.182	**0.807**
bier127	8.213	22.398	3.580	1.568	1.044	**0.730**	2.681
ch130	5.699	28.445	8.186	8.805	2.745	**1.718**	2.619
ch150	7.655	18.597	4.070	1.057	0.914	**0.628**	2.500
kroA150	6.495	20.238	4.770	4.464	**1.767**	2.500	2.672
d198	4.932	20.330	59.930	43.682	3.223	**2.972**	2.817
kroA200	6.095	17.659	11.335	10.804	4.183	**2.268**	2.747

Average Error Rate

Tables II and III display error rates of different methods on test problem instances. It is observed that solutions employing V-FPG methods, such as EV-2opt+2opt and EVgreedy+2opt, consistently exhibit lower error rates compared to those using traditional 2-opt and greedy methods. This suggests a significant improvement in solution accuracy attributed to the incorporation of superior edge distribution.

In TSP, Cost typically refers to the total length or total cost of a tour, which is calculated as the sum of distances between consecutive vertices:

$$\text{Cost} = \sum_{i=1}^{n} \text{distance}(i, i+1) \tag{9}$$

Here, n is the number of vertices, and distance$(i, i+1)$ denotes the distance between vertices i and $i+1$. The error rate is computed using the formula:

$$\text{Error Rate} = \frac{\text{Cost} - \text{Optimal Cost}}{\text{Optimal Cost}} \tag{10}$$

Here, Optimal Cost refers to the cost of the optimal solution, typically representing the shortest tour length. The error rate quantifies describes the relative discrepancy between the computed and optimal solutions.

Computation Time

Tables IV and V present the average computation time required to solve each given problem instance. The time complexity for edge evaluations, involving operations like line drawing and multiplication, is $O(n^2)$. This complexity results in increased time as the number of vertices n grows. In our experiment, we observed that the greedy method, due to its fewer iterations, exhibits relatively shorter computation time compared to the 2-opt method. Notably, the 2-opt method requires the longest computation time among the methods evaluated. Furthermore, the EV-greedy+2-opt yielded the most optimal results with significantly reduced computation time compared to using 2-opt alone. However, as the vertex count increases in TSP instances, computation time scales proportionally at $O(n^2)$, potentially requiring more memory and processing power.

Comparison with Prior Research

Table II compares the performance of our method with prior CNN-based solutions [18] on TSPLIB problem instances. Our method consistently outperforms previous approaches on nearly all problem instances. Particularly noteworthy is its outstanding performance and significant improvements, especially on the TSP-LIB dataset, especially for larger scales.

Due to the structural constraints of CNNs, input-output image resolutions are typically set at $(S1, S2) = (192, 192)$, leading to interference between points. In contrast, our method

TABLE IV: Average total error rate for random problem instances (sec)

Instance Set	2-opt	greedy	EV-2opt	EV-greedy	EV-2opt+2-opt	EV-greedy+2-opt
20-A	0.00094	0.00019	0.00086	0.00012	0.00095	0.00020
20-B	0.00107	0.00021	0.00094	0.00011	0.00110	0.00019
50-A	0.02377	0.00050	0.01502	0.00027	0.01557	0.00082
50-B	0.02577	0.00056	0.01474	0.00027	0.01610	0.00081
75-A	0.09686	0.00097	0.04809	0.00045	0.05146	0.00169
75-B	0.10472	0.00152	0.05588	0.00048	0.05715	0.00176
100-A	0.24402	0.00291	0.13515	0.00068	0.13888	0.00291
100-B	0.29787	0.00205	0.14770	0.00070	0.15329	0.00418
150-A	0.95196	0.00437	0.43086	0.00126	0.43654	0.00647
150-B	0.97966	0.00809	0.49073	0.00127	0.49579	0.00630
200-A	2.70270	0.00440	1.16019	0.00683	1.25030	0.13283
200-B	2.53676	0.00893	1.15922	0.00200	1.16822	0.01097

TABLE V: Average total error rate for TSPlib instances (sec)

Instance Set	2-opt	greedy	EV-2opt	EV-greedy	EV-2opt+2-opt	EV-greedy+2-opt
eil51	0.02249	0.00058	0.01625	0.00027	0.01803	0.00124
berlin52	0.02326	0.00112	0.01777	0.00118	0.01934	0.00666
st70	0.07861	0.00098	0.04994	0.00050	0.05541	0.00520
pr76	0.11403	0.00208	0.04330	0.00060	0.05191	0.00459
eil76	0.09800	0.00119	0.05789	0.00049	0.06344	0.01143
rat99	0.28608	0.00258	0.11783	0.00123	0.13046	0.00608
kroA100	0.29779	0.00202	0.13964	0.00106	0.17269	0.01102
kroB100	0.27490	0.00184	0.14267	0.00067	0.15073	0.00487
kroC100	0.25047	0.00162	0.13769	0.00067	0.15705	0.00733
kroD100	0.28841	0.00235	0.15124	0.00169	0.17384	0.02047
kroE100	0.24842	0.00162	0.13769	0.00069	0.15128	0.01732
rd100	0.25608	0.00227	0.12425	0.00072	0.13330	0.00492
Eil101	0.23288	0.00262	0.16130	0.00241	0.18399	0.02154
lin105	0.33484	0.00296	0.18302	0.00075	0.19525	0.00571
bier127	0.59623	0.00672	0.28693	0.00127	0.32835	0.03213
ch130	0.64602	0.00594	0.28280	0.00410	0.30994	0.04618
ch150	0.99784	0.00389	0.40337	0.00126	0.41845	0.00639
kroA150	1.04023	0.00601	0.48051	0.00124	0.51589	0.02726
d198	2.81523	0.01405	1.03531	0.01220	1.81258	0.56066
kroA200	2.88469	0.00558	1.11072	0.00222	1.26465	0.18209

increases the input-output image resolution to $(S1, S2) = (256, 256)$. This elevation allows for larger problem scales and effectively reduces point interference, thereby enhancing solution accuracy.

V. Conclusion

This study introduces a novel method named Vertex-Conditioned Forward Path Generation (V-FPG), which is grounded in diffusion models and superior edge distributions. Our experimental results highlight the pronounced advantages of V-FPG, particularly in handling large-scale data. Compared to traditional approaches, V-FPG demonstrates higher accuracy and enhanced robustness across both random point tests and the TSP-LIB dataset.

Our research not only confirms the effectiveness of the V-FPG method, which integrates diffusion models and superior edge distributions, in solving TSP but also underscores its potential in managing large-scale instances.

In the future, we intend to expand the application of the V-FPG method to address larger and more complex combinatorial optimization problems. This endeavor will involve extensive research and optimization of the algorithm to accommodate higher-dimensional data and more intricate constraints. We are confident that with continuous effort and refinement, the V-FPG method will provide robust and efficient solutions for a broader range of application scenarios.

Acknowledgment

This research was partly supported by the Information and Communication Technology Research Group of ORDIST Kansai University.

References

[1] Applegate, D., Bixby, R., Chvátal, V., and Cook, W. (2006). The Traveling Salesman Problem: A Computational Study. Princeton University Press.

[2] Padberg, M.W., Rinaldi, G. (1991). A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems. SIAM Rev., 33, 60-100.

[3] Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley.

[4] Kirkpatrick, S., Gelatt Jr, C. D., Vecchi, M. P. (1983). Optimization by simulated annealing. science, 220(4598), 671-680.

[5] Glover, F. (1989). Tabu search—Part I. ORSA Journal on computing, 1(3), 190-206.

[6] Lin, S., Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling-salesman problem. Operations research, 21(2), 498-516.

[7] Khalil, E., Khalil, M., and Bello, I. (2017). Learning to Travel: Training Hierarchical LSTM Networks for Vehicle Navigation. In Proceedings of the 34th International Conference on Machine Learning.

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

[9] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... and Bengio, Y. (2014). Generative Adversarial Nets. In Advances in Neural Information Processing Systems (pp. 2672-2680).

[10] Nazari, M., et al. (2018). Generating Solutions to the Traveling Salesperson Problem with Generative Adversarial Networks. In Proceedings of the 35th International Conference on Machine Learning.

[11] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... and Polosukhin, I. (2017). Attention is All You Need. In Advances in Neural Information Processing Systems (pp. 5998-6008).

[12] Khalil, E., Babiloni, F., and El-Bouri, A. (2020). Transformer Networks for the Traveling Salesman Problem. In Proceedings of the 37th International Conference on Machine Learning.

[13] Isola, P., Zhu, J. Y., Zhou, T., and Efros, A. A. (2017). Image-to-Image Translation with Conditional Adversarial Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 1125-1134).

[14] Tang, Y., Gao, J., and Ma, J. (2019). Solving the Traveling Salesman Problem with Conditional Generative Adversarial Networks. In Proceedings of the 36th International Conference on Machine Learning.

[15] Socher, R., Bengio, Y., and Li, D. (2020). Deep Learning for Graphs. arXiv preprint arXiv:2005.03675.

[16] Chen, Z., Zhang, H., and Wang, J. (2021). Variance Preserving Stochastic Differential Equation for Combinatorial Optimization. In Proceedings of the AAAI Conference on Artificial Intelligence.

[17] Wang, L., Zhao, T., and Liu, H. (2022). Explaining Variance Preserving Stochastic Differential Equation for Traveling Salesman Problem. Journal of Artificial Intelligence Research.

[18] Miki, S., and Ebara, H.(2018) . Solving the Traveling Salesman Problem Using Deep Learning (In Japanese). Journal of Information Processing, vol 60. No.2 651-659 (Feb. 2019)