IDEQ: an improved diffusion model for the TSP

Mickael Basson, Philippe Preux

IDEQ: an improved diffusion model for the TSP

Mickael Basson , Philippe Preux

RESEARCH REPORT
N° 9558
July 2024
Project-Team Scool

ISSN 0249-6399 ISRN INRIA/RR--9558--FR+ENG

IDEQ: an improved diffusion model for the TSP

Mickael Basson* †, Philippe Preux‡*

Project-Team Scool

Research Report n° 9558 — July 2024 — 23 pages

Abstract: We investigate diffusion models to solve the Traveling Salesman Problem. Building on the recent DIFUSCO and T2TCO approaches, we propose IDEQ (constrained Inverse Diffusion and EQuivalence class-based retraining of diffusion models for combinatorial optimization). IDEQ improves the quality of the solutions by leveraging the constrained structure of the state space of the TSP. Another key component of IDEQ consists in replacing the last stages of DIFUSCO curriculum learning by considering a uniform distribution over the Hamiltonian tours whose orbits by the 2-opt operator converge to the optimal solution as the training objective. Our experiments show that IDEQ improves the state of the art for such neural network based techniques on synthetic instances. More importantly, our experiments show that IDEQ performs very well on the instances of the TSPlib, a reference benchmark in the TSP community: it closely matches the performance of the best heuristics, LKH3, being even able to obtain better solutions than LKH3 on 2 instances of the TSPlib defined on 1577 and 3795 cities. IDEQ obtains 0.3% optimality gap on TSP instances made of 500 cities, and 0.5% on TSP instances with 1000 cities. This sets a new SOTA for neural based methods solving the TSP. Moreover, IDEQ exhibits a lower variance and better scales-up with the number of cities with regards to DIFUSCO and T2TCO.

Key-words: Machine learning, diffusion model, traveling salesman problem

* Univ. Lille, Inria, CNRS, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France
† Employee of Lilly France, detached to work on this project as part of a partnership agreement between Lilly, INRIA, CNRS and University of Lille
‡ Université de Lille, France

RESEARCH CENTRE
Centre Inria de l'Université de Lille

Parc scientifique de la Haute-Borne
40 avenue Halley - Bât A - Park Plaza
59650 Villeneuve d'Ascq

IDEQ : un modèle de diffusion pour le problème du voyageur de commerce

Résumé : Nous étudions l'utilisation de modèles de diffusion pour résoudre le problème du voyageur de commerce. En s'appuyant sur DIFUSCO et T2TCO récemment publiés, nous proposons IDEQ (constrained Inverse Diffusion and EQuivalence class-based retraining of diffusion models for combinatorial optimization). IDEQ améliore la qualité des solutions trouvées pour le TSP en tirant partie de la structure de l'espace de recherche du TSP et sur la redefinition de l'objectif d'apprentissage qui remplace la solution optimale par une distribution uniforme sur les tours hamilotoniens dont l'orbite par 2-opt converge vers celle-ci. Nos expériences montrent que IDEQ obtient d'excellentes performances sur la bibliothèque de problème TSPlib, une référence pour la communauté étudiant le TSP: IDEQ atteint des performances proches de la meilleure heuristique (LHK3), parvenant à obtenir de meilleurs tours que LHK3 sur deux instances de la TSPlib de taille 1577 et 3795. IDEQ obtient un écart à l'optimalité de 0,3% sur des instances de 500 villes, et de 0,5% sur des instances de 1000 villes. Cela représente le nouvel état de l'art pour les méthodes neuronales pour la résolution du TSP. De plus, les tours construits par IDEQ ont une variance faible et IDEQ passe mieux à l'échelle lorsque le nombre de villes augmente que DIFUSCO et T2TCO.

Mots-clés : Apprentissage automatique, modèle de diffusion, problème du voyageur de commerce

Contents

1 Introduction

Recent years have seen a surge in machine learning models to solve combinatorial optimization (CO) problems. The field of combinatorial optimization is a historical field of research and application in computer science. After decades of progress, very efficient algorithms exist to provide exact solutions or approximate solutions to many CO problems. The Traveling Salesman Problem (TSP) stands as a prominent example of this fact: the TSP is very appealing as it is very simple to understand, it has a wide range of applications, and we are able to solve exactly rather large instances of this problem on a mere laptop (an instance defined over a few thousands cities can be solved within one hour), and we have approximate algorithms that are able to find tours that are very close to optimality (LKH3 [4] can solve instances of 40,000 cities in about one hour on a laptop but we have no guarantee that the result is optimal). These facts can not be forgotten when we try to propose alternate approaches to solve the TSP.

Because of its appeal, the TSP has also drawn the attention of researchers in deep neural networks in the recent years. If the first attempts had difficulties solving even small TSP instances of a dozen cities, progress has been made. In this paper, we build on these previous works and go a step further. In terms of experimental results, we provide the new state-of-the-art results on TSP instances of size up to a 7397 cities using a neural network approach. These results are competitive with the best heuristic approaches for the TSP in terms of the quality of the solution, and competitive with exact solvers in terms of computation time.

Our approach, IDEQ, relies on diffusion-based models [25, 5, 26, 20, 2], and particularly the recent DIFUSCO and T2TCO [27, 15]. Solving a supervised learning problem, IDEQ aims at learning an optimal tour for a given TSP instance. With regards to DIFUSCO and T2TCO, we improve the inference by leveraging the highly constrained structure of the solution space of the TSP. We also replace the later stage of the training curriculum of DIFUSCO by adding a simplified objective replacing the unique instance-conditioned ground truth solution by a uniform distribution of solutions.

Our presentation goes as follows: after a review of the related literature, we describe IDEQ, a new diffusion-based combinatorial optimization solver for the TSP. In applied combinatorial optimization, the criterion is the experimental performance. So we took a great care to investigate the experimental performance of IDEQ: in the case when the authors made their software program available, we carefully reviewed their code which led us to the conclusion that at least in one case, the implementation differs from the description made in the paper and that this unreported difference is key in the experimental performance. Following the CO community practices, we experimented IDEQ on the TSPlib which is a well-known benchmark on which we report unprecedented performance using a neural net based approach, competitive with regards to state-of-the-art (SOTA) non neural approaches. We report on these experimental performance, and perform a detailed analysis of the various components of IDEQ to disentangle the contribution of each.

2 Background and related works

2.1 About the TSP

Let us consider the basic, and usual definition of the Traveling Salesman Problem. An instance of the TSP is defined in the Euclidean plane by a set of N cities. N is also known as the "size" of the instance. We will denote TSP-N a TSP instance of size N. The goal is to find a shortest tour that visits at least once each of the N cities. An instance may be defined either by the location of the cities, or by a distance matrix between the cities. There exists many variations of this definition in Euclidean and non Euclidean spaces. In this paper, TSP refers to this particular family of planar Euclidean instances. The TSP is a well-known example of a NP-hard problem: the time/memory

requirements to solve an instance of size N grow in $O(\exp(N))$ so that the size of the instances that can be solved within a reasonable amount of time is very limited. For instance, today, the state-of-the-art exact solver Concorde solves an instance of the TSP of size $N = 10^3$ in about 10 minutes on a laptop. For larger instances, one has to use heuristics that have no formal guarantee to find an optimal tour. The quality of a tour may be measured by its "optimality gap", defined by $\frac{\text{length of the best found tour} - \text{optimal tour length}}{\text{optimal tour length}}$: it is a non-negative real equal to 0 only for an optimal tour. The optimality gap does not depend on the size of the instance. There is no way to assess a priori the difficulty of an instance: we know for sure that its size says nothing about it: for any N, it is obvious to design an instance which is trivial to solve.

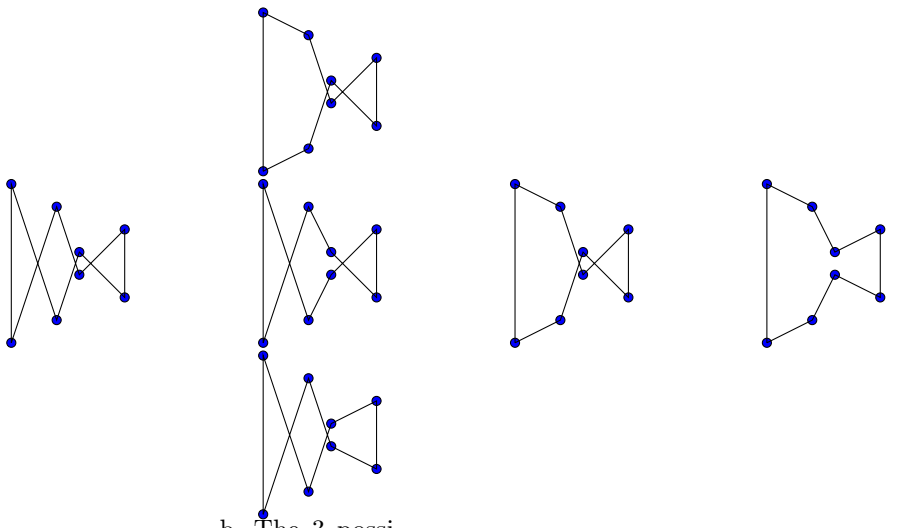
For some instance, there may be more than 1 tour that have the shortest length. An optimal tour goes once and only once per city: it is a Hamiltonian tour. Consequently, the search space may be reduced to the set of Hamiltonian tours. Each such tour is a potential "solution".

Given a Hamiltonian tour, there is a very simple heuristic that can decrease its length (when applicable) known as "2-opt" (see Fig. 1). Given a Hamiltonian tour (Fig. 1.a), a "2-change" consists in removing a pair of edges of the tour and reconstructing an other tour. In 2D, when the edges of the pair cross each other, applying a 2-change uncrosses the edges (Fig. 1.b): this is guaranteed to decrease the length of the tour. 2-opt considers all pairs of crossing edges and disentangles the pair that most decreases the length of the resulting tour (Fig. 1.c). 2-opt is both very simple and rather effective in optimizing a tour. In practice, it is customary to pipeline a first heuristic that produces a Hamiltonian tour, and then post-process it by iterating 2-opt to obtain a disentangled tour. There exists other such local optimizations, like the Lin-Kernighan operator (lk-opt) [16] and its more recent version LKH3 [4] that are more complex in terms of algorithm, though very efficient and effective in practice for such a post-processing. For a (still) relevant survey, we refer the interested reader to [10]. In this paper, we will restrict ourselves to the use of the 2-opt since it is simple, fast, and yet efficient.

2.2 Denoising diffusion models

Denoising diffusion models have been initially introduced by [25] and further expanded by various authors [5, 26, 20, 2]. The aim is to sample a complex data distribution. The idea is to start from an easy to sample distribution and then going through a set of iterative Markovian transformations which mimics the reverse of a diffusion process, to sample from more and more complex distributions which converge toward the target complex distribution. The training of these models is done in a supervised manner.

These family of generative models have now become SOTA for (conditioned) image and video generation [23, 22, 6, 24]. They have also achieved great performance in areas like audio synthesis [12, 33] or molecule generation [30, 29, 8]. They have also shown their usefulness for image or text generation [1], and



a. A Hamiltonian tour.

b. The 3 possible 2-change's that uncross one pair of crossing edges of the Hamiltonian tour shown in a.

c. If we apply 2-opt again twice on the tour shown in b., we obtain a local optimum (for 2-opt).

d. The local optimum reached after 3 2-opt's on the tour shown in a.

Figure 1: Illustration of 2-change and 2-opt operators on a Hamiltonian tour.

combinatorial optimization [27]. They can be defined in continuous or discrete time or space. Most of denoising diffusion models are using continuous state space but discrete models, initially introduced by [25] and further expanded by [1, 7].

2.3 Diffusion models for combinatorial optimization

A solution to a TSP can be represented by an $N \times N$ adjacency matrix $A = (a_{i,j})$ where $a_{i,j} = 1$ if there exists a link between city i and city j, 0 otherwise. A is symmetrical. One may soften this definition and define $a_{i,j}$ as the probability that there exits a link between i and j. This produces a heatmap which may be seen a stochastic adjacency matrix. A diffusion model can be used to sample from this heatmap starting from an easy to sample, all off-diagonal terms being equal to some value p.

Diffusion models to predict the instance-conditionned adjacency matrix of graph-based combinatorial optimization problems have been introduced by DIFUSCO [27]. In DIFUSCO, a denoising diffusion process is used to predict the heatmap of the solution for a given TSP instance starting with $p = 1/2$. Inference time is rather short. The training is done in a supervised manner on large synthetic datasets. One such dataset is made of thousands of random instances of a certain size N labeled with their optimal (or best known) tour. Training requires significant computational resources to create the dataset (we need to compute the optimal tour of each instance of the dataset) and then to train the network. DIFUSCO is trained by curriculum learning: the network is first trained on TSP-100 random 2D instances. Then, this first checkpoint (the weights of the network) is trained on TSP-500 instances. Then, the TSP-500 checkpoint is trained on TSP-1000 instances. We call "TSP-N checkpoint" the checkpoint of the model trained up to (and including) the TSP-N instances of the curriculum learning. The first training stage of the curriculum (the TSP-100 checkpoint) is the bottleneck in terms of computational needs, requiring an order of magnitude greater training steps than the subsequent trainings. It is noteworthy that the size of instances a checkpoint is trained on does not constrain the size of instances that can be solved: as we will see in the experiments, with a TSP-1000 checkpoint, we predict instances which size ranges from 100 to 7397. DIFUSCO can be used with either continuous, discrete, or discretrized diffusion. Then, T2TCO [15] have introduced a search procedure during the inference to improve the quality of the solution. Published results are impressive. However, while carefully inspecting and testing their code, we have found that their "gradient based search" is not responsible for the improvement. Indeed, this search is based on sequential steps of partial diffusion and subsequent inverse diffusion, but replacing these steps by the original, non guided, inverse diffusion steps does not affect the results. This approach generates an initial solution which can be improved by a search procedure such as 2-opt, or MCTS, or by some sampling procedure.

DISCO [31] is another diffusion-based combinatorial optimization solver that uses an analytically solvable diffusion to speed-up inference. Similar to our work,

they leverage the constrained structure of the solution space but unlike IDEQ, they do so by using a residual diffusion [17] to restrict the sampling space to a more contrained domain.

IDEQ builds further on the discrete state space version of DIFUSCO and T2TCO.

2.4 Other constructive neural solvers for combinatorial optimization

To avoid the costly generation of the supervised training dataset, other approaches leverage unsupervised learning or reinforcement learning. UTSP [18] uses scattering attention GNN [19] to generate a soft indicator matrix that can be simply transformed into a heatmap. The network parameters are optimized through gradient descent on the tour length augmented with a carefully chosen penalty terms. The tour is then refined with extensive Monte-Carlo tree search. They achieve results similar to DIFUSCO using lower training time.

The majority of the neural combinatorial optimization solvers rely on reinforcement learning. These can be either autoregressive (incremental construction of the solution) or generating the full solution in one-shot. The former category includes BQNCO [3]], POMO [14] and SymNCO [11] while the latter category includes DIMES [21]. DIMES employs a meta-learning framework and a continuous parameterization of the solution space which enables a stable REINFORCE-based training. In BQ-NCO the state space of the MDP associated with the problem is simplified by bisimulation quotienting. The policy network architecture is based on a transformer [28] or a perceiver [9] architecture. POMO builds on the attention-based model of [13] but leverages symmetries by data augmentation and stabilizes REINFORCE training by using multiple initiations. SymNCO also leverages symmetries by introducing regularization in REINFORCE and by learning invariant representation for pre-identified symmetries. SymNCO and POMO do not scale well beyond 200 cities for the TSP. ICAM [32] builds upon POMO, modifying the attention mechanism and the reinforcement learning training scheme to significantly improve performance on larger TSP instances (up to 10^3 cities).

3 Methods

This section describes the two main ingredients of IDEQ and then specifies IDEQ itself.

3.1 Background on discrete diffusion models for combinatorial optimization

Given a random variable $\mathbf{x}_0$ drawn from a certain distribution $p_{complex}$, a variational diffusion model is a latent variable model that progressively adds noise

to $\mathbf{x}_0$ following a Markovian forward process, denoted as $q(\mathbf{x_t}|\mathbf{x_{t-1}})$ which generates a sequence $\mathbf{x_{1:T}}$ recursively defined as $q(\mathbf{x_{1:T}}|\mathbf{x_0}) = \prod_{t=1}^{T} q(\mathbf{x_t}|\mathbf{x_{t-1}})$ such that $q(\mathbf{x_T}) \sim p_{simple}$ where p_{simple} is an easy to sample from distribution such as a Gaussian for continuous state diffusion, or a multinomial for discrete state diffusion. The reverse process, called the backward process, $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ is approximated by a Markovian process parameterized by a neural network: $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$. The training is performed by searching for θ that minimizes the variational upper bound of the negative log-likelihood:

$$\begin{aligned} L_{vb} = \mathbb{E}_{q(\mathbf{x_0})}[D_{KL}[q(\mathbf{x_T}|\mathbf{x_0})||p(\mathbf{x_T})] \\ + \sum_{t=2}^{T} \mathbb{E}_{q(\mathbf{x_t}|\mathbf{x_0})}[D_{KL}[q(\mathbf{x_{t-1}}|\mathbf{x_t}, \mathbf{x_0})||p_\theta(\mathbf{x_{t-1}}|\mathbf{x_t})]] \\ - \mathbb{E}_{q(\mathbf{x_1}|\mathbf{x_0})}[log p_0(\mathbf{x_0}|\mathbf{x_1})]] \end{aligned} \quad (1)$$

In the case of a discrete diffusion, the equilibrium distribution p_{simple} is induced by the Markov transition matrix $\mathbf{Q}$ associated with the forward transition kernel. Let us denote x the one-hot encoded version of $\mathbf{x}$, we have:

$$\begin{aligned} q(x_t|x_{t-1}) = Cat(x_t : p = x_{t-1}\mathbf{Q_t}) \\ q(x_t|x_0) = Cat(x_t; p = x_0 \prod_{i=1}^{T} \mathbf{Q_i}) = Cat(x_t; p = x_0 \bar{\mathbf{Q}}_\mathbf{t}) \\ q(x_{t-1}|x_t x_0) = Cat(x_{t-1}; p = \frac{x_t {\mathbf{Q_t}}^T \odot x_0 \bar{\mathbf{Q}}_\mathbf{t-1}}{x_0 \bar{\mathbf{Q}}_\mathbf{t} x_t^T}), \end{aligned}$$

where $Cat(x; p)$ denotes the categorical distribution with probability vector p.

The backward transition is parameterized by a neural network which predicts $p(\tilde{x_0}) = NN_\theta(x_t, t)$. The conditioning on the instance of the combinatorial optimization problem is implicitly done by the neural network which is a GNN (Graph neural network) taking as input the instance so that we can write $p(\tilde{x_0}|I) = NN_\theta(x_t, t|I)$.

3.2 Key ingredient 1: leveraging the constrained structure of the solution space to improve inference-time solution generation

In the TSP, the true probability distribution $x_0(x_t, t|I)$ lies on a very constrained manifold of optimal Hamiltonian tours. In the original implementation of DIFUSCO, such a constraint is not enforced, the network has to learn it, leading to sub-optimal convergence of the solution to possibly non-Hamiltonian adjacency matrices.

In order to circumvent this problem, we propose to modify the parameterization of the backward process by applying a Hamiltonian tour reconstruction

operator H to the predicted $\tilde{x_0}$ to enforce the Hamiltonian constraint. This tour reconstruction operator is the one used at the end of the backward process in DIFUSCO and T2TCO to reconstruct the tour from the generated heatmap.

Similarly, to guide the backward process towards an optimal tour, we apply the 2-opt. Let us denote R_2 the 2-opt operator. As $H \circ \tilde{x_0}$ is now a Hamiltonian tour, we can further refine our estimate of $\tilde{x_0}$ as follows: $\hat{x_0}(x_t, t, I) = R_2 \circ H \circ NN_\theta(x_t, t, I)$.

There is no gradient flow through the R_2 operator and the H operator involves deterministic affine transformation of $NN_\theta(x_t, t, I)$ which leaves the optimal θ in equation 1 unchanged thus avoiding the need to retrain the entire diffusion model.

3.3 Key ingredient 2: leveraging equivalence class over target distribution to fine-tune diffusion checkpoints

If x_0 is an optimal tour then $R_2 x_0 = R_2^n x_0 = x_0$ where R_2^n denotes n successive applications of 2-opt: an optimal (locally or globally) tour is a fixed point of 2-opt. Let us further denote by $\mathcal{R}_2$ the following equivalence relation: $x \mathcal{R}_2 y$ if $\exists n \in \mathbb{N}$ such that either $x = R_2^n y$ or $y = R_2^n x$. We propose to replace the training objective of the diffusion model to be the uniform distribution over the equivalence class of x_0 for the relation $\mathcal{R}_2$ instead of a Dirac mass located on x_0 for a given instance I. This procedure increases the dimensionality of the set supporting the target distribution which we hypothesize will drive the diffusion process towards a more robust distribution. The initial distribution is supported by a single point from the new distribution. This procedure allows us to leverage the curriculum learning of DIFUSCO where the most resource-consuming step is the training of the TSP-100 checkpoint. Each of the subsequent trainings is initialized with the previous checkpoint and it requires approximately an order of magnitude less training steps making these subsequent curriculum learning steps a very good fit for this fine-tuning. This requires to sample solutions from the equivalence class of $x_0(I)$ by the equivalence relation $\mathcal{R}_2$ for any instance I. This procedure is expensive in terms of computations due to the local minima that can be generated when naively inverting the 2-change several times. To circumvent this problem while keeping the computational time low, we use only 2 applications of 2-change. This creates a set of diverse members of the equivalence class which cardinality is several orders of magnitude above the sampling steps needed for the later stages of the curriculum where we use it. while creating $\mathcal{O}(N^{-1})$ local minimal which is at least one order of magnitude lower than the number of epoches for TSP instance ≥ 500 as considered here.

3.4 IDEQ

Building on these two ingredients, let us specify the training and the inference phases of IDEQ, sketched in Algorithm 1.

- **Training phase:**

  - we train an IDEQ TSP-500 checkpoint, by initializing the network with DIFUSCO original TSP-100 checkpoint, and using their TSP-500 training set. The training objective is modified as follows: instead of associating the optimal tour, we associate a tour T obtained by applying 2 consecutive randomly sampled 2-change on the optimal tour (key ingredient 2). This produces the IDEQ TSP-500 checkpoint.
  - Likewise for the TSP-1000 checkpoint: we initialize the network with the previously trained IDEQ TSP-500 checkpoint and we use the DIFUSCO TSP-1000 training set with the updated objective (key ingredient 1). This produces the IDEQ TSP-1000 checkpoint.

- **Inference phase:** we follow the inference steps of T2TCO: the original DIFUSCO diffusion (from $t = T$ to $t = 0$ followed by 3 partial forward/backward diffusion steps (from $t = 0$ to $t = \alpha T$ and vice versa, $0 < \alpha < 1$) but with the updated diffusion steps detailed in algorithm 1.

Algorithm 1 IDEQ. H denotes the Hamiltonian reconstruction tour (the one used by DIFUSCO and T2TCO), and R is the 2-opt operator.

Require: The inference schedule (i.e. a sequence of decreasing timesteps $\mathcal{T} = \{t_T, ...t_0 = 0\}$)
Require: a (trained) neural network NN_θ (IDEQ checkpoint)
Require: I is the instance to solve (defined by the coordinates of the cities).
  $x_T \leftarrow$ a one-hot encoded adjacency list randomly sampled from $Cat(x_T, p = 1/2)$
  for t in $\mathcal{T} \backslash t_0$ **do**
    $\tilde{x_0} = R \circ H \circ NN_\theta(x_t, t|I)$
    if $t - 1 \neq t_0$ **then**
      Sample $x_{t-1} \sim Cat(x_{t-1}, p = \frac{x_t \mathbf{Q_t}^T \odot \tilde{x_0} \bar{\mathbf{Q}}_{\mathbf{t-1}}}{\tilde{x_0} \bar{\mathbf{Q}}_\mathbf{t} x_t^T})$
    end if
  end for

4 Experimental study

In computational combinatorial optimization, experimental results are key. Henceforth, in this section, we present our experimental results. The most important result is that IDEQ reaches SOTA performance on the TSPlib benchmark, achieving significantly better results than the previous SOTA DIFUSCO and T2TCO, and getting close to those of LKH3, even on TSP instances with several thousands of cities. We also show that IDEQ performance does not degrade very much with N. Overall, these results set unprecedented performance for a neural network approach on the TSP.

4.1 Experimental setup

We used the same hyperparameters as in T2TCO. For model retraining we used a batch-size of 8, using the published DIFUSCO checkpoint. Running times and optimality gaps of POMO, LKH3 and ICAM are taken from [32], while for DISCO and BQNCO, these were taken from [31, 3]. As our experiments showed that the variance of the results was high, we increased the size of the test set to 2048 random instances. As most of the other publications report the running time on test sets made of 128 instances, we scaled (/16) our running time to 128 to ensure fair comparison (running time is linear in the size of the test set). Following previous publications, the inference for DIFUSCO, T2TCO and IDEQ is done with pytorch based code with a batch size of 1. The experiments based on neural networks were run on a cluster of 4 Nvidia A100 40GB with AMD EPYC 7513 CPU. To ensure fair comparison, we re-ran DIFUSCO and T2TCO on the same hardware. Concorde is a highly optimized C code. We run it on a 2x Intel Xeon Platinium 8358 32-core CPU servers (hence a total of 64 physical cores with 128 threads). The 2-opt algorithm as benchmark heuristics in table 1 is initiated from a random tour and run on the same hardware as Concorde.

4.2 Experimental methodology

First, we compare IDEQ to other approaches on 2D Euclidean instances made of N cities drawn uniformly at random, provided by the authors of DIFUSCO. This gives a rough first insight into the performance of IDEQ versus the others.

However, real TSP instances are not random in this way; instances are structured because they correspond to real-world problems. It is well-known in combinatorial optimization that the performance on uniformly random instances does not say much about the performance on structured instances. With this in mind, we run IDEQ and other approaches on the TSPlib, a highly renown benchmark in the TSP community. On the TSPlib instances, we compare the optimality gap of the solutions found by the different approaches. We also compare the variance of the optimality gap obtained when repeatedly solving a given instance and we consider how the optimality gap varies with the size of the instances.

Finally, to get a better understanding on how IDEQ works, we perform an ablation study.

Results are reported in the next 2 sections.

4.3 Experimental results

Table 1 reports on the first experiments in which we compare the performance of the different approaches on random instances. IDEQ clearly demonstrates smaller optimality gaps as well as a better scaling behavior when going from 500 cities instances to 1000 cities instances.

On the TSPlib, we used the TSP-1000 checkpoint on all instances with size ranging from 100 to 10^4 cities. Table 2 shows both the consistency of IDEQ performance and its ability to generalize beyond the training distribution. Instances up to 1,000 cities have been solved by Concorde, and instances larger than 1000 have been solved by LKH3 (default settings). In 2 cases (instances 'fl3795' and 'fl1577'), we obtained a tour which outperformed LKH3 (with default settings). To the best of our knowledge, this is the first time a neural method is reported to outperform LKH3.

Finally we explored the variability of the results, exploring both the sampling driven variability for a given instances and the optimality gap on different instances. Indeed, for a given instance, the denoising process can start from various noisy graphs with no guarantee to converge to the same solution. We thus looked at 32 random initial tours for 64 different TSP-1000 instances and looked at the distribution of the predicted solutions and associated optimality gaps. Results are shown in figures 2. The standard deviations of the optimality gap obtained on 2048 instances of the TSP-1000 are shown in table 3. IDEQ exhibits a smaller variability for the length of the inferred tour, both within and between instances, which is consistent with the reduced search space and the more robust objective in the fine-tuning.

		TSP 500	
Method	Type	OG.	running time
Concorde (*)	Exact	0.0 % (ref.)	2.6 mn
LKH3	Heuristic	0.0 %	0.2 mn
2-opt	Heuristic	13 %	0.6 s
ICAM	RL	1.6 %	1 s
BQNCO	RL	1.18 %	55 s
DISCO	SL + G + 2-opt	1.87 %	0.25 m
DIFUSCO (*)	SL + G + 2-opt	1.6 %	45 s
T2TCO (*)	SL + G + 2-opt	0.89 %	0.4 mn
IDEQ (*)	SL + G + 2-opt	**0.41 %**	1.3 mn
POMO x8	RL + S	22.2 %	1.1 mn
ICAM aug. x8	RL	0.77 %	38 s
BQNCO x16	RL +BS	0.55 %	15 mn
DISCO x16	SL +S +2opt	1.1 %	3.9 mn
DIFUSCO (*) x4	SL + S + 2-opt	0.93 %	2.6 mn
T2TCO (*) x4	SL + S + 2-opt	0.67 %	1.1 mn
IDEQ (*) x4	SL +S + 2-opt	**0.29 %**	3.7 mn
		TSP 1000	
Method	Type	OG.	running time
Concorde (*)	Exact	0.0 % (ref.)	1.2 h
LKH3	Heuristic	0.0 %	0.4 mn
2-opt	Heuristic	13 %	13 s
ICAM	RL	2.9 %	2 s
BQNCO	RL	2.29 %	2 mn
DISCO	SL + G + 2-opt	2.29 %	1.1 mn
DIFUSCO (*)	SL + G + 2-opt	2.0 %	48 s
T2TCO (*)	SL + G + 2-opt	1.37 %	1.6 mn
IDEQ (*)	SL + G + 2-opt	**0.63 %**	5.1 mn
POMO x8	RL + S	40.6 %	8.5 mn
ICAM aug. x8	RL	1.6 %	3.8 mn
BQNCO x16	RL +BS	1.38 %	38 mn
DISCO x16	SL +S +2opt	1.9 %	9.6 mn
DIFUSCO (*) x4	SL + S + 2-opt	1.46 %	3.3 mn
T2TCO (*) x4	SL + S + 2-opt	1.21 %	3.4 mn
IDEQ (*) x4	SL +S + 2-opt	**0.49 %**	13.6 mn

Table 1: Comparison of averaged optimality gaps (OG) and running times on large scale 2D uniformly distributed Euclidean TSP instances. RL = Reinforcement learning, SL = supervised learning, G = Greedy decoding, aug. = data augmentation, S = sampling, BS = beam search. For both sampling and beam search, we use the notation 'xN' to indicate a sampling size of N or a beam search of size N. The results marked with an asterisk (*) are obtained by running ourselves the code, others originate from the publications. 'ref.' indicates the reference method used to calculate the optimality gap. Information about the variance of optimality gaps is available in figure 2 and table 3.

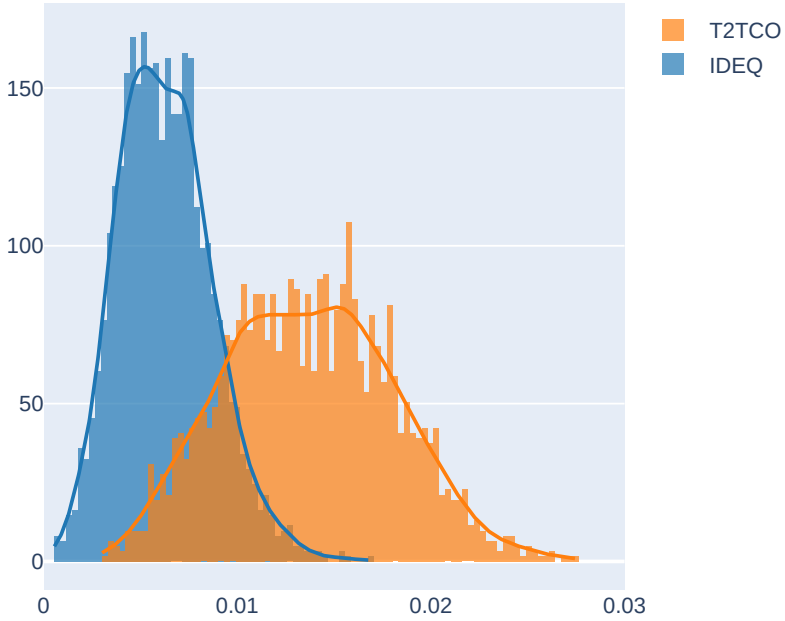


Figure 2: Distribution of optimality gaps measured on 32 repetitions of the first 64 TSP-1000 instances from the test set. IDEQ in blue and T2TCO in orange. Bars represent the empirical data ditribution and the line is the Gaussian kernel density estimate.

	TSPlib 100-10^4	
Method	OG.	running time
Concorde/LKH3	0. % (ref.)	2.3 mn
DIFUSCO	2.9 %	1.5 mn
T2TCO	1.9 %	4.5 mn
IDEQ	1.2 %	29 mn
DIFUSCO x4	1.9 %	5.3 mn
T2TCO x4	1.7 %	8.05 mn
IDEQ x4	0.98 %	1.2 h

Table 2: Comparison of optimality gaps (OG) and running times on TSPlib instances ranging from 100 to 10^4 cities.

Method	Mean std. dev.
T2TCO	0.13
T2TCO with IDEQ checkpoint	0.082
IDEQ with DIFUSCO checkpoint	0.075
IDEQ (with IDEQ checkpoint)	0.057

Table 3: Standard deviation of the optimality gap of the tours generated by 4 different approaches. With regards to T2TCO, IDEQ more than halves the standard deviation, while (see table 1) also more than halving the optimality gap.

4.4 Ablation studies

We conducted ablation studies on the two components of IDEQ: the re-training of the later stages of DIFUSCO curriculum learning and the updated estimator of $\hat{x_0}(x_t, t, I)$. The ablation of both brings back to T2TCO. Removal of the late stage re-training was explored by using the original DIFUSCO checkpoints and keeping the Hamiltonian tour reconstruction operator and the 2-opt at inference time. The effect of the late stage re-training alone was explored by using the IDEQ checkpoints in DIFUSCO and T2TCO.

Table 4 shows the results of these ablation studies. These results are consistent with the two effects adding up independently as we could anticipate based on the methodology.

As a side note, let us mention that only iterating 2-opt on a random Hamiltonian tour produces a tour which optimality gap is at least a few percents: hence, the diffusion model is really useful to obtain such small optimality gaps.

5 Conclusion and future work

Building on DIFUSCO and T2TCO, we have introduced IDEQ which leads to a significant improvement of the state-of-the-art neural diffusion-based solver for the Traveling Salesman Problem. The optimality gap is significantly reduced

	TSP 500	
Method	OG.	running time
IDEQ	**0.41 %**	3.7 mn
IDEQ with DIFUSCO checkpoint	0.65 %	1.3 mn
T2TCO with IDEQ checkpoint	0.6 %	0.4 mn
DIFUSCO with IDEQ checkpoint	1.3 %	0.7 mn
T2TCO	0.89 %	0.4 mn
	TSP 1000	
Method	OG.	running time
IDEQ	**0.63 %**	5.1 mn
IDEQ with DIFUSCO checkpoint	0.93 %	6.2 mn
T2TCO with IDEQ checkpoint	1.0 %	1.6 mn
DIFUSCO with IDEQ checkpoint	1.6 %	0.8 mn
T2TCO	1.37 %	1.6 mn

Table 4: Results of the ablation study of IDEQ. OG stands for "optimality gap".

on instances of up to thousands of cities. It is noticeable that the optimality gap obtained with IDEQ does not degrade as much as the methods it gets inspired of. It is also noticeable that the variability of the optimality gap of the tours found by IDEQ is smaller than the one of previous neural methods. These modifications rely on an improvement of the inference process, as well as a fine-tuning of the existing checkpoints. As such IDEQ does not require the expensive retraining of a diffusion model from scratch. IDEQ relies on a strong effort to understand previous approaches which led us to propose two original key ingredients. IDEQ may be applied to other combinatorial optimization problems for which a simple invertible transformation exists (like the 2-change for the TSP). This application to other combinatorial optimization problems is one of our lines of research in the future.

Our approach is amenable to several improvements with other transformations or projections. The solution improvement step can also be further improved by using more expensive algorithms like beam search or Monte-Carlo Tree Search (MCTS).

Beyong engineering further IDEQ to obtain yet faster better tours, a more fundamental question remains regarding the role of the partial re-noising/denoising introduced in T2TCO at inference time. As previously mentioned, a careful investigation led us to realize that these steps need not be guided to guarantee an improvement of the tour, contrary to the exposition made in the T2TCO paper. The explanation for the observed improvement is unknown and surprising.

Now, from the CO community perspective, the idea of pipelining various algorithms is a common, often advocated one, to combine the benefits of each. In the case of the TSP, using 2-opt to post-process a tour is very common, the question being about the algorithm that provides the tour to 2-optimize. Various algorithms have been studied (simulated annealing, tabu search, evolutionary algorithms, ant colonies, *etc.*) and we may now add diffusion models to this list.

Acknowledgments

French Ministry of Higher Education and Research, Inria, Région Hauts-de-France CPER project CornelIA

References

[1] Jacob Austin et al. "Structured Denoising Diffusion Models in Discrete State-Spaces". In: *Proc. NeurIPS*. 2021, pp. 17981–17993.

[2] Prafulla Dhariwal and Alex Nichol. "Diffusion Models Beat GANs on Image Synthesis". In: *Proc. NeurIPS*. 2021, pp. 8780–8794.

[3] Darko Drakulic et al. "BQ-NCO: Bisimulation Quotienting for Efficient Neural Combinatorial Optimization". In: *Proc. NeurIPS*. 2023. URL: https://arxiv.org/abs/2301.03313.

[4] K. Helsgaun. *An Extension of the Lin-Kernighan-Helsgaun TSP Solver for Constrained Traveling Salesman and Vehicle Routing Problems*. Technical Report, Roskilde University. 2017.

[5] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising Diffusion Probabilistic Models". In: *Proc. NeurIPS*. 2020, pp. 6840–6851.

[6] Jonathan Ho et al. *Imagen Video: High Definition Video Generation with Diffusion Models*. 2022. arXiv: 2210.02303 [cs.CV]. URL: https://arxiv.org/abs/2210.02303.

[7] Emiel Hoogeboom et al. "Argmax Flows and Multinomial Diffusion: Learning Categorical Distributions". In: *Proc. NeurIPS*. 2021, pp. 12454–12465.

[8] Emiel Hoogeboom et al. "Equivariant Diffusion for Molecule Generation in 3D". In: *Proc. ICML*. 2022, pp. 8867–8887.

[9] Andrew Jaegle et al. "Perceiver: General Perception with Iterative Attention". In: *Proc. ICML*. 2021, pp. 4651–4664.

[10] D.S. Johnson and L.A. McGeoch. "The Traveling Salesman Problem: A Case Study in Local Optimization". In: *Local Search in Combinatorial Optimisation*. John Wiley and Sons Ltd, 1997, pp. 215–310.

[11] Minsu Kim, Junyoung Park, and Jinkyoo Park. "Sym-NCO: Leveraging Symmetricity for Neural Combinatorial Optimization". In: *Proc. NeurIPS*. 2022, pp. 1936–1949.

[12] Zhifeng Kong et al. "DiffWave: A Versatile Diffusion Model for Audio Synthesis". In: *Proc. ICLR*. 2021.

[13] Wouter Kool, Herke van Hoof, and Max Welling. "Attention, Learn to Solve Routing Problems!" In: *Proc. ICLR*. 2019. URL: https://openreview.net/forum?id=ByxBFsRqYm.

[14] Yeong-Dae Kwon et al. "POMO: Policy Optimization with Multiple Optima for Reinforcement Learning". In: *Proc. NeurIPS*. 2020, pp. 21188–21198.

[15] Yang Li et al. "T2T: From Distribution Learning in Training to Gradient Search in Testing for Combinatorial Optimization". In: *Proc. NeurIPS*. 2023.

[16] S. Lin and B. W. Kernighan. "An effective heuristic algorithm for the traveling salesman problem". In: *Operation Research* 21 (1973), pp. 498–516.

[17] Jiawei Liu et al. "Residual Denoising Diffusion Models". In: *Proc. CVPR*. 2024, pp. 2773–2783.

[18] Yimeng Min, Yiwei Bai, and Carla P. Gomes. "Unsupervised Learning for Solving the Travelling Salesman Problem". In: *Proc. NeurIPS*. 2023, pp. 47264–47278.

[19] Yimeng Min et al. "Can Hybrid Geometric Scattering Networks Help Solve the Maximum Clique Problem?" In: *Proc. NeurIPS*. 2022, pp. 22713–22724.

[20] Alex Nichol and Prafulla Dhariwal. "Improved Denoising Diffusion Probabilistic Models". In: *Proc. ICML*. 2021, pp. 8162–8171.

[21] Ruizhong Qiu, Zhiqing Sun, and Yiming Yang. "DIMES: A Differentiable Meta Solver for Combinatorial Optimization Problems". In: *Proc. NeurIPS*. 2022, pp. 25531–25546.

[22] Aditya Ramesh et al. "Zero-Shot Text-to-Image Generation". In: *Proc. ICML*. 2021, pp. 8821–8831.

[23] Robin Rombach et al. "High-Resolution Image Synthesis with Latent Diffusion Models". In: *Proc. CVPR*. 2022.

[24] Chitwan Saharia et al. "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding". In: *Proc. NeurIPS*. 2022, pp. 36479–36494.

[25] Jascha Sohl-Dickstein et al. "Deep Unsupervised Learning using Nonequilibrium Thermodynamics". In: *Proc. ICML*. Lille, France, 2015, pp. 2256–2265.

[26] Jiaming Song, Chenlin Meng, and Stefano Ermon. "Denoising Diffusion Implicit Models". In: *Proc. ICLR*. 2020.

[27] Zhiqing Sun and Yiming Yang. "DIFUSCO: Graph-based Diffusion Solvers for Combinatorial Optimization". In: *Proc. NeurIPS*. 2023, pp. 3706–3731. URL: `https://openreview.net/forum?id=JV8Ff0lgVV`.

[28] Ashish Vaswani et al. "Attention Is All You Need". In: *Proc. NeurIPS*. 2017.

[29] Clement Vignac et al. "DiGress: Discrete Denoising diffusion for graph generation". In: *Proc. ICLR*. 2023.

[30] Minkai Xu et al. "GeoDiff: a Geometric Diffusion Model for Molecular Conformation Generation". In: *Proc. ICLR*. 2022.

[31] Kexiong Yu et al. *DISCO: Efficient Diffusion Solver for Large-Scale Combinatorial Optimization Problems*. 2024. arXiv: 2406.19705 [cs.AI]. URL: https://arxiv.org/abs/2406.19705.

[32] Changliang Zhou et al. *Instance-Conditioned Adaptation for Large-scale Generalization of Neural Combinatorial Optimization*. arXiv:2405.01906. 2024.

[33] Ge Zhu et al. *EDMSound: Spectrogram Based Diffusion Models for Efficient and High-Quality Audio Synthesis*. NeurIPS workshop: machine learning for audio. 2023.

A Detailed results on the instances of the TSPlib

For each instance of the TSPlib, we give:

- the name,
- the size (number of cities),
- the length of the tour obtained by Concorde for instances smaller than 1000 cities, LKH3 otherwise
- the length of the tour obtained by IDEQ,
- the optimality gap (in %).

name	size	Concorde/LKH3	IDEQ	OG (%)
kroA100.tsp	100	5.382	5.391	0.167
kroB100.tsp	100	5.622	5.677	0.972
kroC100.tsp	100	5.268	5.285	0.329
kroD100.tsp	100	5.396	5.531	2.497
kroE100.tsp	100	5.556	5.586	0.541
rd100.tsp	100	8.056	8.068	0.153
eil101.tsp	101	8.314	8.484	2.034
lin105.tsp	105	4.659	4.659	0.000
pr107.tsp	107	2.697	2.711	0.504
pr124.tsp	124	4.345	4.388	0.977
bier127.tsp	127	5.861	5.863	0.038
ch130.tsp	130	8.745	8.932	2.146
pr136.tsp	136	6.709	6.763	0.808
gr137.tsp	137	9.924	10.065	1.423
pr144.tsp	144	3.845	3.866	0.558
ch150.tsp	150	9.336	9.384	0.511
kroA150.tsp	150	6.678	6.729	0.759
kroB150.tsp	150	6.633	6.672	0.591
pr152.tsp	152	4.656	4.656	0.000
u159.tsp	159	4.950	4.950	0.000
rat195.tsp	195	7.911	7.975	0.798
d198.tsp	198	3.924	3.948	0.610
kroA200.tsp	200	7.426	7.448	0.303
kroB200.tsp	200	7.461	7.489	0.375
gr202.tsp	202	6.371	6.375	0.069
ts225.tsp	225	7.915	8.060	1.826
tsp225.tsp	225	6.175	6.181	0.096
pr226.tsp	226	4.908	5.079	3.492
gr229.tsp	229	9.174	9.315	1.538
gil262.tsp	262	24.099	25.129	4.272
pr264.tsp	264	4.417	4.424	0.172
a280.tsp	280	8.982	9.020	0.427
pr299.tsp	299	5.461	5.496	0.643
lin318.tsp	318	10.368	10.541	1.671

name	size	Concorde/LKH3	IDEQ	OG (%)
rd400.tsp	400	15.314	15.354	0.260
fl417.tsp	417	5.817	5.870	0.904
gr431.tsp	431	10.795	10.916	1.120
pr439.tsp	439	7.826	7.954	1.633
pcb442.tsp	442	13.364	13.447	0.619
d493.tsp	493	9.350	9.458	1.150
att532.tsp	532	10.080	10.120	0.388
ali535.tsp	535	11.336	11.492	1.379
u574.tsp	574	10.363	10.418	0.526
rat575.tsp	575	13.619	13.688	0.505
p654.tsp	654	5.915	6.139	3.786
d657.tsp	657	12.212	12.431	1.791
gr666.tsp	666	17.327	17.997	3.872
u724.tsp	724	11.945	12.049	0.871
rat783.tsp	783	15.247	15.272	0.169
dsj1000.tsp	1000	17.027	17.274	1.450
pr1002.tsp	1002	15.395	15.580	1.203
u1060.tsp	1060	10.531	10.646	1.088
vm1084.tsp	1084	12.653	12.860	1.635
pcb1173.tsp	1173	16.524	16.648	0.750
d1291.tsp	1291	12.946	13.565	4.776
rl1304.tsp	1304	13.289	13.450	1.205
rl1323.tsp	1323	14.156	14.393	1.679
nrw1379.tsp	1379	6.949	7.182	3.359
fl1400.tsp	1400	9.660	9.797	1.425
u1432.tsp	1432	17.872	18.407	2.989
fl1577.tsp	1577	11.110	10.980	-1.166
d1655.tsp	1655	16.915	17.326	2.431
vm1748.tsp	1748	17.698	17.935	1.343
u1817.tsp	1817	17.118	17.362	1.420
rl1889.tsp	1889	16.549	16.801	1.522
d2103.tsp	2103	18.536	18.813	1.497
u2152.tsp	2152	18.935	19.285	1.847
u2319.tsp	2319	27.298	27.856	2.044
pr2392.tsp	2392	23.808	24.242	1.824
pcb3038.tsp	3038	35.227	35.829	1.711
fl3795.tsp	3795	14.244	14.097	-1.030
fnl4461.tsp	4461	17.271	17.752	2.784
rl5915.tsp	5915	29.996	30.284	0.959
rl5934.tsp	5934	29.326	29.951	2.131
pla7397.tsp	7397	37.515	38.357	2.243

RESEARCH CENTRE
Centre Inria de l'Université de Lille

Parc scientifique de la Haute-Borne
40 avenue Halley - Bât A - Park Plaza
59650 Villeneuve d'Ascq

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399