Euclidean Score-Based Generative Modeling with Permutation Semantics

Gaël Heck*
SATIE Laboratory UMR 8029
Université Paris-Saclay
Gif-sur-Yvette, France

Nassim Bourarach
CEREMADE, CNRS
Université Paris-Dauphine, PSL
Paris, France

Sylvie Le Hégarat-Mascle
SATIE Laboratory UMR 8029
Université Paris-Saclay
Gif-sur-Yvette, France

Nicolas Lermé
SATIE Laboratory UMR 8029
Université Paris-Saclay
Gif-sur-Yvette, France

Abstract

Generative modeling over permutations is challenging because permutations are discrete structured objects, whereas many powerful generative samplers, including score-based diffusion models, are formulated in continuous spaces. Existing diffusion-based approaches typically bridge this mismatch either by relaxing permutations into continuous permutation surrogates or by designing discrete diffusion processes with specialized transition kernels. We propose a geometric alternative based on the coordinate ordering of centered vectors. Sorting a centered vector maps it to a permutation, and the sorting operation partitions the space into permutation-indexed regions whose facet adjacencies correspond to adjacent swaps between permutations. In this view, the learned score controls how probability mass is transported across sorting regions, thereby inducing transitions between neighboring permutations through their shared boundaries. The resulting sampler remains close to standard score-based diffusion: train in the centered Euclidean space, run the reverse diffusion, and sort once at the end. Experiments across a variety of standard benchmark tasks such as jigsaw puzzle reconstruction (structured prediction), sorting 4-digit MNIST numbers (ordering), the traveling salesman problem (combinatorial optimization), and preference-based ranking show competitive performance with state-of-the-art methods while remaining computationally efficient, especially compared with discrete generative methods.

1 Introduction

Many learning problems have permutations as their natural output space, including ranking, sorting, assignment and combinatorial optimization. In these problems, the target is not a vector of independent categorical variables but a globally constrained ordering. A generative model over permutations must satisfy two requirements at once: it must output exact discrete objects while remaining expressive enough to represent multimodal laws over orderings.

Existing approaches address this tension in two main ways. First, continuous relaxation methods embed permutations into differentiable surrogate spaces, such as the Birkhoff polytope, Sinkhorn-normalized matrices, or smooth sorting and ranking operators [1, 6, 7, 9,

*Corresponding author: gael.heck@universite-paris-saclay.fr

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

10, 12, 26, 29]. These methods are effective when the goal is to optimize through a sorting or matching operation. However, they do not directly specify the final hard permutation law: a decoding or projection step is still required after the continuous relaxation. Second, discrete and factorized generative methods define the stochastic process directly on finite permutation representations, preserving validity by construction [5, 8, 31, 34, 36]. This avoids a relaxation gap, but moves away from the Euclidean machinery that makes modern score-based diffusion simple and scalable: Gaussian noising, denoising score matching, and standard reverse-time numerical solvers [16, 32, 33]. In particular, recent generative approaches often rely on quadratic-complexity permutation representations.

This paper asks whether this separation is necessary. We show that it is not: once the decoder is fixed to be `argsort`, a Euclidean probability flow induces exact dynamics on the decoded law over S_n. The map `argsort` partitions the centered hyperplane into braid chambers, one chamber per permutation, and adjacent chambers meet along facets corresponding to adjacent transpositions [37]. Hence, when a Euclidean density evolves in the centered space, the decoded law on permutations evolves by probability flux across these sorting facets.

This analysis is the main theoretical contribution of the paper. It shows that standard Euclidean Variance–Preserving (VP) score-based generative dynamics induce well-defined dynamics on permutation laws once observed through the argsort partition of a centered Euclidean space. The second contribution is the generative methodology that follows from this analysis. We lift clean permutations to vertices of the centered permutahedron, apply Variance-Preserving Gaussian noising in the centered hyperplane, train a conditional Euclidean score model, and generate samples by running a standard reverse diffusion process followed by a single final hard argsort. The sampler therefore remains continuous throughout inference and operates in a linear-dimensional representation, rather than relying on quadratic-complexity permutation representations or explicit discrete transition mechanisms. This makes inference efficient and allows the method to inherit the mature toolkit of continuous diffusion models while still producing exact permutations. Our contributions are as follows.

1. **A theory of VP score dynamics through sorting geometry.** We show that Euclidean probability dynamics, when decoded by argsort, evolve permutation probabilities through fluxes across sorting facets. For VP diffusion, we characterize the induced score-controlled adjacent-swap currents and derive stability bounds for the final decoded law.
2. **A practical linear-dimensional sampler with empirical validation.** Building on this theory, we introduce a VP diffusion sampler on the centered hyperplane that uses standard Euclidean score learning, reverse-time sampling, and a single final argsort decode. We evaluate it on jigsaw reconstruction, digit sorting, TSP, and preference-based re-ranking against relaxation-based, discrete diffusion, and factorized permutation models.

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 introduces the geometric and probabilistic background. Section 4 presents the `argsort`-decoded VP diffusion sampler. Section 5 analyzes the permutation laws induced by argsort-decoded Euclidean VP diffusion. Section 6 presents the empirical evaluation.

2 Related Work

Probabilistic models of ranking laws. Probabilistic ranking models have long defined tractable laws directly on S_n, from choice-based Plackett–Luce models [24, 28] and distance-based Mallows models [25] to repeated-insertion families [11, 23]. Recent neural work extends this factorized view through bijective representations such as Lehmer codes, Fisher–Yates draws, and insertion vectors [31]. These models provide explicit parameterizations of distributions over the symmetric group S_n.

Permutation geometry and continuous relaxations. The geometry behind hard sorting is classical: the braid arrangement partitions the centered hyperplane into one chamber per permutation, and the centered permutahedron has this braid fan as its normal fan [37]. Classical approaches relax permutations through the Birkhoff polytope and Sinkhorn operators [6, 9, 26], while smooth sorting methods relax the sorting or ranking map itself [10, 12, 29]. Closest to ours in geometric spirit, [7] formulate differentiable sorting and

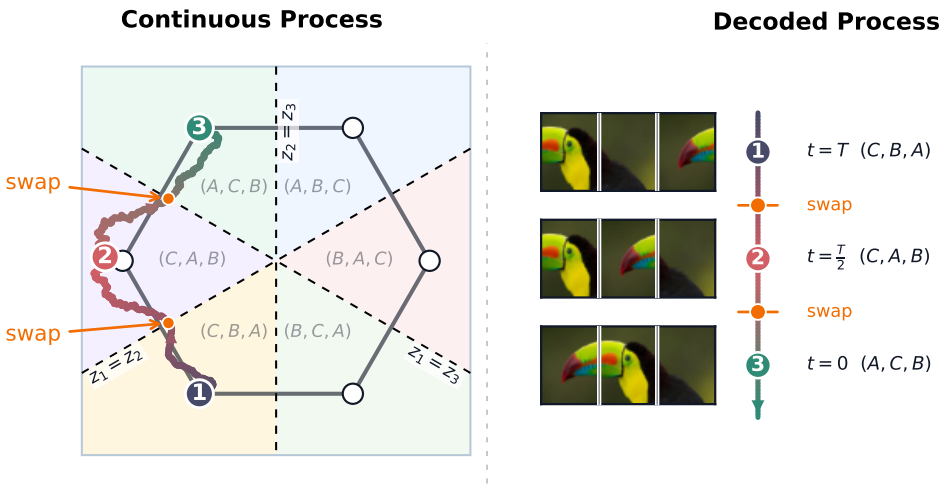


Figure 1: Illustration of the continuous Euclidean process decoded into a permutation.

ranking as regularized optimization over the permutahedron. More recent variants refine these relaxations through differentiable sorting networks and error-free sorting operators [19, 27]. These methods mainly provide differentiable interfaces to sorting.

Generative models. Score-based generative models learn reverse noising dynamics through marginal scores [16, 32, 33]. Their theoretical basis is stochastic time reversal for diffusion processes [3]. Discrete diffusion and continuous-time Markov chain formulations provide general frameworks for categorical variables [5, 8, 34]. For permutations, SymmetricDiffusers [36] is the first solution to construct a diffusion directly on S_n using riffle-shuffle random walks for forward noising and generalized Plackett–Luce distributions for reverse transitions. Soft-Rank [14] extends discrete permutation diffusion by replacing shuffle-based forward noising with reflected diffusion in continuous soft-rank space, while retaining a Plackett–Luce-style reverse denoiser. These two methods are the closest to ours, as they also formulate diffusion-based generative models for permutations. More broadly, generative modeling is not limited to diffusion, and recent optimal-transport-inspired alternatives such as flow matching have also emerged [2, 22].

3 Preliminaries

Global notation. We work with permutations of n items, $n \geq 2$, $[n] = \{1, \ldots, n\}$, and S_n is the symmetric group and we use the convention that $\mathrm{argsort}(z) = \sigma$ means

$$\mathrm{argsort}(z) = \sigma \quad \Longleftrightarrow \quad z_{\sigma(1)} > \cdots > z_{\sigma(n)}.$$

The tie set is the union of the braid hyperplanes $\{z_i = z_j\}$, which has Lebesgue measure zero. Since adding a constant to every coordinate does not change the ordering, all dynamics are placed on the centered hyperplane

$$H_n := \{z \in \mathbb{R}^n : \mathbf{1}^\top z = 0\},$$

equipped with the Euclidean structure inherited from $\mathbb{R}^n$. All differential operators ∇, div, and Δ are taken intrinsically on H. The notation below summarizes the objects that recur in the sequel. The definitions and standard background can be found in [3, 7, 33].

3.1 Permutations, braid chambers, and the centered permutahedron

The map $\mathrm{argsort} : H_n \to S_n$ is piecewise constant. Its decision regions are the braid chambers

$$C_\sigma = \{z \in H_n : \; z_{\sigma(1)} > \cdots > z_{\sigma(n)}\}, \qquad \sigma \in S_n,$$

namely the maximal cones of the hyperplane arrangement $z_i = z_j$. This chamber decomposition is illustrated in Figure 1. The facet adjacencies of these chambers encode adjacent swaps. If $s_k = (k, k+1)$, then C_σ and $C_{\sigma s_k}$ share the facet

$$F_{\sigma,k} := \overline{C_\sigma} \cap \overline{C_{\sigma s_k}}, \qquad n_{\sigma,k} := \frac{e_{\sigma(k+1)} - e_{\sigma(k)}}{\sqrt{2}},$$

where $n_{\sigma,k}$ points from C_σ to $C_{\sigma s_k}$ and $(e_i)_{i=1}^n$ is the canonical basis of $\mathbb{R}^n$. Thus the adjacent-swap graph on S_n is the facet-adjacency graph of the hard-sorting decision regions. The same geometry is encoded by the centered permutahedron. Let

$$\bar{r} = \left(\frac{n-1}{2}, \frac{n-3}{2}, \ldots, -\frac{n-1}{2}\right) \in H_n, \qquad v_\sigma := P_\sigma \bar{r}, \qquad \Pi_n^\circ := \operatorname{conv}\{v_\sigma : \sigma \in S_n\} \subset H_n.$$

Here $\bar{r}$ is the canonical centered rank vector, v_σ is the vertex encoding of the exact permutation σ, and Π_n° collects all such encodings. The normal cone of Π_n° at v_σ, namely the set of directions for which v_σ maximizes a linear objective, is exactly $\overline{C}_\sigma$. Thus braid chambers and permutahedral vertices are dual descriptions of hard sorting. This optimization view is exploited by differentiable sorting methods via regularized optimization over Π_n° [7]; here, Π_n° is used only to encode exact permutations and describe the geometry of the hard decoder.

3.2 Euclidean score diffusion on the centered hyperplane

Score-based diffusion is usually formulated on $\mathbb{R}^d$. Since H is a Euclidean vector space, we use the same definitions intrinsically on H. We write I_H for the identity covariance on H, and $\mathcal{N}(\mu, \Sigma)$ for Gaussian laws supported on H. The forward process smooths an initial law u_0 on H into increasingly Gaussian marginals u_t; for large T, u_T is close to $\mathcal{N}(0, I_H)$. Generation reverses this construction: start from the Gaussian reference law and follow the reverse dynamics back toward u_0.

Let u_t be the marginal density of Z_t and define the score $\mathfrak{s}_t(z) := \nabla \log u_t(z)$. With

$$\Gamma(t) := \int_0^t \beta(r)\, \mathrm{d}r, \qquad \beta(t) = \beta_{\min} + \frac{t}{T}\big(\beta_{\max} - \beta_{\min}\big), \qquad \alpha_t := e^{-\Gamma(t)/2}, \qquad \eta_t^2 := 1 - \alpha_t^2,$$

the forward VP process and its reverse are

$$\begin{cases} \mathrm{d}Z_t = -\dfrac{1}{2}\beta(t) Z_t\, \mathrm{d}t + \sqrt{\beta(t)}\, \mathrm{d}B_t, & Z_0 \sim u_0, \quad t : 0 \to T, \\ \mathrm{d}Y_t = \left(\dfrac{1}{2}\beta(T-t) Y_t + \beta(T-t)\mathfrak{s}_{T-t}(Y_t)\right)\mathrm{d}t + \sqrt{\beta(T-t)}\, \mathrm{d}\bar{B}_t, & Y_0 \sim u_T, \quad t : 0 \to T, \end{cases}$$

where $Y_t = Z_{T-t}$ in probability. The forward transition is known and easy to compute as $Z_t \mid Z_0 = x \sim \mathcal{N}(\alpha_t x, \eta_t^2 I_H)$. The strength of diffusion is that the reverse process is also a diffusion process and easy to compute as long as you know the score function. Since this function is unknown when Z_0 is not, these methods rely on learning the Stein score $\mathfrak{s}_t$. [3, 33].

For $X \sim u_0$ and $Z_t = \alpha_t X + \eta_t \varepsilon$, with $\varepsilon \sim \mathcal{N}(0, I_H)$, the conditional score is

$$\nabla_z \log p_{t|0}(Z_t \mid X) = -\frac{Z_t - \alpha_t X}{\eta_t^2}.$$

We train s_θ by denoising score matching:

$$\mathcal{L}(\theta) = \mathbb{E}_{t,X,\varepsilon}\left[\lambda(t)\left\| s_\theta(Z_t, t) + \frac{Z_t - \alpha_t X}{\eta_t^2}\right\|_H^2\right].$$

At the marginal level, the VP SDE satisfies the conservative Fokker–Planck form

$$\partial_t u_t + \operatorname{div} J_t^{\rightarrow} = 0, \qquad J_t^{\rightarrow} = -\frac{\beta(t)}{2}(z\, u_t + \nabla u_t), \qquad J_t^{\leftarrow} = -J_t^{\rightarrow} = \frac{\beta(t)}{2}(z\, u_t + \nabla u_t). \quad (1)$$

The reverse evolution is therefore $-\partial_t u_t + \operatorname{div} J_t^{\leftarrow} = 0$.

4 Permutation Diffusion Model

Problem definition. We consider tasks where n objects admit a task-dependent target order, and the goal is to model the corresponding permutation. An instance is described by a context

$$c = (x_1, \dots, x_m),$$

where x_i is a feature representation of object i, such as an image patch for jigsaw reconstruction, a digit image for sorting, a city coordinate for TSP, or a candidate-item representation for re-ranking. We model the target as a conditional law

$$p_0^c(\sigma) = \mathbb{P}(\sigma \mid c), \qquad p_0^c \in \Delta(S_n).$$

Here $\Delta(S_n)$ is the probability simplex over permutations. It is distinct from the centered permutahedron $\Pi_n^\circ \subset H_n$, whose vertices v_σ provide a Euclidean encoding of exact permutations. For a fixed context, we omit the superscript c when no confusion is possible.

Lifted data law. We lift the conditional law p_0^c to an atomic measure on H_n:

$$\mu_0^c = \sum_{\sigma \in S_n} p_0^c(\sigma)\, \delta_{v_\sigma}.$$

Thus the clean data distribution at time 0 is supported on exact permutahedral vertices. The Euclidean state is introduced only after this lift.

VP noising and score learning. Using the VP transition $Z_t = \alpha_t X + \eta_t \varepsilon$, a training pair (c, σ) is noised as

$$Z_t = \alpha_t v_\sigma + \eta_t \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, I_{H_n}).$$

Equivalently, for $t > 0$, the noised conditional density is

$$u_t^c(z) = \sum_{\sigma \in S_n} p_0^c(\sigma) \mathcal{N}\big(z; \alpha_t v_\sigma, \eta_t^2 I_{H_n}\big)\,.$$

We learn the conditional score $\mathfrak{s}_t^c(z) = \nabla_z \log u_t^c(z)$ with a centered network $s_\theta(z, t, c)$, using the denoising score-matching loss with $X = v_\sigma$:

$$\mathcal{L}_{\text{perm}}(\theta) = \mathbb{E}_{t,c,\sigma,\varepsilon}\left[\lambda(t) \left\| s_\theta(Z_t, t, c) + \frac{Z_t - \alpha_t v_\sigma}{\eta_t^2} \right\|_{H_n}^2\right].$$

Sampling and decoding. At generation time, we sample from the Gaussian reference law and integrate the learned reverse VP dynamics conditioned on c:

$$\widehat{Z}_T \sim \mathcal{N}(0, I_{H_n}), \qquad \widehat{Z}_T \longrightarrow \widehat{Z}_0.$$

with the learned score $s_\theta(\cdot, t, c)$, and decode once:

$$\widehat{\sigma} = \operatorname{argsort}(\widehat{Z}_0).$$

If $\widehat{u}_0^c$ denotes the terminal Euclidean law produced by the learned reverse dynamics, the generated permutation law is

$$\widehat{p}_0^c = \operatorname{argsort}_{\#}\, \widehat{u}_0^c.$$

Thus the sampler is continuous during inference, but its output is an exact permutation. The next section proves that this hard-decoded law has an exact braid-fan flux interpretation.

5 Variance–Preserving Diffusion Dynamics under argsort Decoding

The previous section defined the sampler; we now analyze its induced permutation dynamics. Running reverse VP diffusion in the centered Euclidean space H and decoding once by argsort already imposes a permutation structure: argsort partitions H into sorting chambers

$(C_\sigma)_{\sigma\in S_n}$, whose codimension-one adjacencies are exactly adjacent swaps. Whenever the Euclidean law has density u_t, its hard-decoded law is

$$p_t = \operatorname{argsort}_{\#} u_t \in \Delta(S_n), \qquad p_t(\sigma) = \int_{C_\sigma} u_t(z)\ \mathrm{d}z.$$

Thus the diffusion evolves in H, but the observable law after decoding evolves on S_n. The centered permutahedron is used to encode clean permutations and to define the associated rank geometry. The section is organized around four questions.

- **How does continuous probability flow become permutation dynamics?** We show that any probability law evolving in Euclidean space by mass-preserving flow induces a well-defined evolution of the decoded law on S_n through the argsort partition. Conversely, any smooth adjacent-swap evolution on S_n can be represented by such a Euclidean flow.
- **What geometric object is identified by the VP score?** We show that the VP score recovers a posterior barycenter over exact permutahedral vertices. Thus, the learned score acts as a soft decoder in the permutahedron rather than as a generic vector field.
- **What controls boundary crossings in reverse VP dynamics?** We show that, in reverse VP dynamics, the score is the only deterministic term driving crossings between sorting chambers. It therefore controls the decoded law's adjacent-swap transport.
- **Does score matching control the hard-decoded law?** We show that small excess population denoising score-matching loss controls the reverse endpoint law by Girsanov, data processing then transfers this guarantee to the final argsort distribution despite the decoder being discontinuous.

5.1 Euclidean flow induces adjacent-swap dynamics

Diffusion marginals naturally conserve probability mass. Equivalently, their densities satisfy the Fokker–Planck equation (1), $\partial_t u_t + \operatorname{div} J_t = 0$, where J_t is the probability current. We show that integrating this conservation law over sorting chambers induces adjacent-swap dynamics on S_n, and conversely that any smooth adjacent-swap current admits a Euclidean chamber-flow lift.

Theorem 5.1 (Chamber fluxes and adjacent-swap dynamics)**.** *Let (u_t, J_t) be a regular density-current pair on H, with sufficient decay, satisfying $\partial_t u_t + \operatorname{div} J_t = 0$. Set $p_t = \operatorname{argsort}_{\#} u_t$ and, for $\tau = \sigma s_k$,*

$$j_t(\sigma,\tau) = \int_{F_{\sigma,k}} J_t(z)\cdot n_{\sigma,k}\ \mathrm{d}S, \qquad j_t(\tau,\sigma) = -j_t(\sigma,\tau).$$

where $\mathrm{d}S$ denotes the surface measure on the facet $F_{\sigma,k}$. Then

$$\dot p_t(\sigma) = -\sum_{\tau\sim\sigma} j_t(\sigma,\tau).$$

Conversely, any smooth $p_t \in \Delta(S_n)$ and antisymmetric adjacent-swap current j_t satisfying the above equation admit a smooth compactly supported Euclidean lift (u_t, J_t) satisfying $\partial_t u_t + \operatorname{div} J_t = 0$, such that

$$\operatorname{argsort}_{\#} u_t = p_t, \qquad \int_{F_{\sigma,k}} J_t(z)\cdot n_{\sigma,k}\ \mathrm{d}S = j_t(\sigma,\sigma s_k).$$

5.2 VP score as a smoothed permutahedral decoder

The chamber-flow result above applies to any Euclidean current. It does not yet explain why the score learned by the sampler has permutation content. This is where the permutahedral lift matters: clean data are exact vertices v_σ, and VP noising turns a noisy point z into a posterior distribution over these vertices. After an affine change of variables, the score returns the posterior barycenter of this distribution.

This barycenter is conveniently expressed through the weighted log-sum-exp potential

$$\kappa_t = \frac{\alpha_t}{\eta_t^2}, \qquad A_{t,c}(z) = \frac{1}{\kappa_t}\log\sum_{\sigma\in S_n} p_0^c(\sigma)\exp\big(\kappa_t\,\langle z, v_\sigma\rangle\big).$$

Proposition 5.2 (VP score as smoothed permutahedral decoding)**.** *The posterior barycenter of the clean vertex is*

$$m_t(z,c) := \mathbb{E}[V \mid Z_t = z, c] = \nabla_z A_{t,c}(z) \in \Pi_n^\circ,$$

and the VP score satisfies

$$\mathfrak{s}_t(z,c) = \frac{\alpha_t m_t(z,c) - z}{\eta_t^2}, \qquad \textit{hence} \qquad m_t(z,c) = \frac{z + \eta_t^2 \mathfrak{s}_t(z,c)}{\alpha_t}.$$

The posterior weights are $\mathbb{P}(V = v_\sigma \mid Z_t = z, c) \propto p_0^c(\sigma) \exp\left(\kappa_t \langle z, v_\sigma \rangle\right)$.

Thus $\nabla A_{t,c}$ is a soft permutahedral decoder: it returns a posterior barycenter in Π_n°, converging at low noise to $v_{\mathrm{argsort}(z)}$ away from ties. Hence the learned score encodes a posterior mean over exact permutation vertices, not a relaxed permutation variable.

5.3 Reverse VP score determines adjacent-swap currents

The previous subsection showed that the VP score encodes a posterior permutahedral denoiser. We now show that it also controls how decoded mass moves during reverse sampling. Applying the chamber-flow theorem to the reverse VP current in Eq.(1) yields the adjacent-swap currents of the decoded permutation law. On every sorting facet $F_{\sigma,k}$, the non-score drift has no normal component since, for $z \in F_{\sigma,k}$,

$$z \cdot n_{\sigma,k} = 0.$$

Thus boundary crossings between adjacent sorting chambers are driven by the normal component of the score.

Theorem 5.3 (Reverse VP facet currents)**.** *Let u_t be a regular VP marginal on H, and set $p_t = \mathrm{argsort}_\# u_t$. Under reverse-time transport from T to 0,*

$$\dot{p}_t(\sigma) = \sum_{k=1}^{n-1} j_t^{\leftarrow}(\sigma, \sigma s_k), \qquad j_t^{\leftarrow}(\sigma, \sigma s_k) = \frac{\beta(t)}{2} \int_{F_{\sigma,k}} u_t(z) \left\langle \mathfrak{s}_t(z), n_{\sigma,k} \right\rangle \, \mathrm{d}S.$$

Thus adjacent-swap transport is driven by posterior rank gaps. The current from C_σ to $C_{\sigma s_k}$ is determined by the normal component of the posterior permutahedral barycenter on their shared facet. This links the training semantics of the score to the generative dynamics of the decoded law.

5.4 Score error controls the decoded law

We finally give a conditional, continuous-time, population-level guarantee: controlling the achieved Euclidean score error controls the hard-decoded permutation law, with no additional decoder-specific error term. Let u_t^c be the exact VP marginal and let $\widehat{u}_\varepsilon^{\theta,c}$ be the law obtained at time $\varepsilon > 0$ by the continuous-time learned reverse VP process initialized from $\mathcal{N}(0, I_{H_n})$. Define

$$\Delta_\theta(c) = \int_\varepsilon^T \beta(t) \, \mathbb{E}_{Z_t \sim u_t^c} \left[\| s_\theta(Z_t, t, c) - \mathfrak{s}_t^c(Z_t) \|_{H_n}^2 \right] \mathrm{d}t.$$

For the DSM loss weighted by $\lambda(t) = \beta(t)$, this quantity is exactly the population excess denoising score-matching loss.

Theorem 5.4 (Decoded-law stability)**.** *Assume the standard absolute-continuity and integrability conditions for Girsanov's theorem, and assume $u_T^c \ll \mathcal{N}(0, I_{H_n})$. Then*

$$\mathrm{TV}\big(\mathrm{argsort}_\# u_\varepsilon^c, \mathrm{argsort}_\# \widehat{u}_\varepsilon^{\theta,c}\big) \le \left[\frac{1}{2} \mathrm{KL}(u_T^c \| \mathcal{N}(0, I_{H_n})) + \frac{1}{4} \Delta_\theta(c) \right]^{1/2}.$$

Thus the discontinuity of argsort causes no extra difficulty: the endpoint Euclidean law is controlled first, and the bound is then pushed forward through argsort by data processing. If the terminal VP marginal is exactly Gaussian, then the terminal KL term vanishes.

6 Experimental Setup

We evaluate our model on the main permutation-generation benchmarks considered in the literature: jigsaw reconstruction, 4-digit MNIST sorting, Euclidean TSP, and MovieLens re-ranking. These tasks cover deterministic reconstruction, long-range ordering, combinatorial optimization, and multimodal ranking. We compare against relaxation-based methods, discrete permutation diffusions, and factorized permutation baselines. All results for generative models are reported as the mean of 5 runs.

Baselines. For continuous relaxations we include Error-free DiffSort [19] as it is the SOTA in this category. For discrete generative models we include categorical diffusion over positions, SymmetricDiffusers [36] and Soft-Rank Diffusion [14] denoisers. We also include Diffassemble [30] as it is a Euclidean diffusion model formulated as a translation retrieval task, which further show how the same generative process performs without the argsort decoding.

Metrics. For permutation tasks, we report three metrics. i) Exact accuracy $\mathbf{1}\{\widehat{\sigma} = \sigma\}$ measures whether the entire predicted permutation matches the target. ii) Position correctness $n^{-1}\sum_{i=1}^{n}\mathbf{1}\{\widehat{\sigma}(i) = \sigma(i)\}$ measures the fraction of items assigned to their correct positions. iii) Kendall-$\tau = (C - D)/\binom{n}{2}$ measures pairwise ordering quality, where C and D count item pairs whose relative order is respectively preserved or reversed. For MovieLens 32M, we report Normalized Discounted Cumulative Gain (NDCG@k), which measures how well the predicted top-k ranking agrees with the true user ratings. Scores are normalized by the ideal ranking, so the maximum value is 1.0 [17].

6.1 Network architecture and training

We follow the token-based Transformer [35] architecture of SymmetricDiffusers [36]. For each instance, we denote by $X(c, Z_t, t) \in \mathbb{R}^{n\times d}$ the token matrix obtained from the task-specific object encoder, the noisy state Z_t, and the sinusoidal time embedding. This sequence is processed by a Transformer encoder, yielding $H_\theta \in \mathbb{R}^{n\times d}$. A shared scalar head produces $\tilde{s}_\theta(Z_t, t, c) \in \mathbb{R}^n$, updating all coordinates jointly at each reverse step without autoregressive permutation construction. The attention backbone still costs $O(n^2)$: linear dimensionality refers to the diffusion state, not the overall computational complexity. Since the score must lie in H_n, we use the centered prediction $\widehat{s}_\theta(Z_t, t, c) = \tilde{s}_\theta(Z_t, t, c) - \frac{1}{n}\mathbf{1}\mathbf{1}^\top\tilde{s}_\theta(Z_t, t, c)$. Training uses the denoising score-matching loss of Section 4. At sampling time, we integrate the reverse VP dynamics using the Euler–Maruyama algorithm with $\widehat{s}_\theta$ and decode once by argsort. We refer to the supplementary for details about this architecture and for training and sampling algorithms. The descriptions of c and its dedicated encoding processes for the respective tasks are given in the corresponding paragraphs and fully described in the supplementary material.

For all baselines, we use the public source code when available and otherwise rely on our own implementation. All reimplemented baselines will be included in our code release. We reuse reference architectures and task-specific encoders where applicable and match parameter counts across methods. These controls limit, but do not fully isolate, architectural effects on empirical gains. When a paper proposes multiple solutions, we report only the best-performing one for figure readability. Additional hyperparameter and training details are provided in the supplementary material.

6.2 Experiments

Jigsaw puzzles. Following the jigsaw setup of [30, 36], each image is split into a regular grid of $k \times k = n$ patches and randomly permuted. We use the CIFAR-10 [20] 2×2, 3×3, and 4×4. The input context is the shuffled patch tensor, and the target is the permutation that restores the original image. Each shuffled patch is encoded by a CNN into an item token, which is summed with embeddings of the current diffusion coordinate Z_t^i and the diffusion time before being processed by the transformer.

Performance analysis: We report benchmark performance in Figure 2. The results show strong reconstruction performance, with improvements in exact puzzle accuracy across the evaluated sizes.

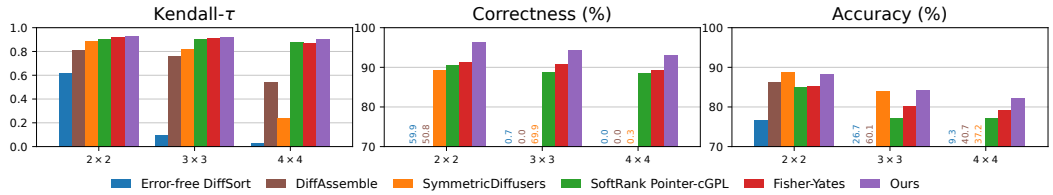


Figure 2: Jigsaw reconstruction results on CIFAR-10. We report Kendall-τ, exact puzzle accuracy, and position correctness across puzzle sizes.

4-digit MNIST sorting. We construct sequences of $N \in \{15, 32, 52, 100, 200\}$ images, where each item is a horizontally concatenated four-digit MNIST [21] number. The model observes the unordered set of image items and must output the permutation that orders them by their underlying numeric value. Each image is encoded in the same way as in the Jigsaw Puzzle benchmark. We evaluate both short sequences, where differentiable sorting networks are competitive, and long sequences, where errors in shuffle-based diffusion or relaxed sorting can accumulate over many positions.

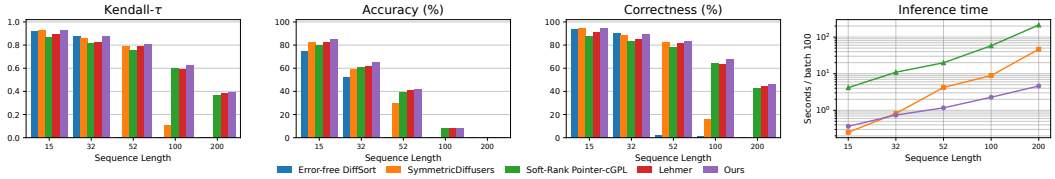


Figure 3: 4-digit MNIST sorting results across sequence lengths. We report Kendall-τ, exact sequence accuracy, and position correctness. Inference time, measured in seconds per batch, is reported against competitive generative methods.

Performance analysis: We report benchmark performance in Figure 3. Our method improves long-sequence position correctness and Kendall-τ, corresponding to more correctly placed items and fewer pairwise ordering errors. At $n = 200$, it is 46× faster than the strongest competing diffusion method.

Traveling Salesperson Problem. We use the Euclidean TSP-20 and TSP-50 benchmarks from [18], where nodes are sampled i.i.d. from $[0, 1]^2$ and optimal tours are generated by Concorde [4], following the setup used by [14, 36]. Each node coordinate is encoded with a sinusoidal coordinate encoder and projected to a node token, which is summed with the embedding of the current diffusion coordinate. Because a TSP tour is invariant to cyclic shifts and reversal, evaluation canonicalizes tours before computing permutation metrics, and optimization quality is reported by tour length and optimality gap.

	TSP-20				TSP-50			
Metric	Soft-Rank	SymmetricDiffusers	Lehmer Code	Ours	Soft-Rank	SymmetricDiffusers	Lehmer Code	Ours
Tour Length↓	3.848	3.850	3.861	3.850	5.789	5.732	5.921	5.712
Optimality Gap (%)↓	0.18	0.19	0.43	0.19	1.76	0.92	4.09	0.51

Table 1: TSP-20 and TSP-50 results compared with learning-based baselines.

Performance analysis: We report performance on this benchmark in Table 1. For TSP-20, all models achieve strong performance, with Soft-Rank Diffusion [14] performing best. TSP-50 is more discriminative: the factorized method [31] performs worst in both settings, while Soft-Rank Diffusion still maintains solid performance.

MovieLens 32M. We use MovieLens 32M [13] to construct user-specific candidate sets of $n \in \{10, 20, 50\}$ movies. For each user, the target permutation sorts the candidate movies by that user's ratings (e.g., 4 out of 5 stars). At test time, the model observes a prefix of this ranking and must sample a plausible completion. Unlike deterministic sorting tasks, this benchmark is genuinely multimodal: several completions may be consistent with the observed

preferences, ranking quality and calibration are more informative than exact permutation accuracy. The context representation follows the insertion-vector protocol of [31]. Each movie token encodes the movie identity together with the user-relative preference information available from the observed prefix. We then augment this context with the current coordinate Z_t^i, item-position information, and the diffusion time t. Full encoding details and results for $n < 50$ are given in Section A.4.

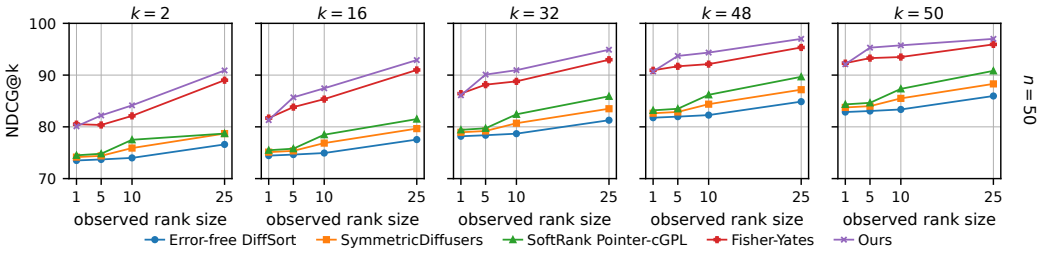


Figure 4: MovieLens 32M re-ranking results for $n = 50$.

Performance analysis: Results are reported in Figure 4. Although [31] naturally suits MovieLens because ratings define partial rankings, these rankings also have a direct geometric form in our framework: each comparison $i \succ j$ corresponds to the half-space $z_i > z_j$ in H_n. Our method achieves competitive NDCG, suggesting the Euclidean score dynamics can learn preference completions without an explicit ranking code.

7 Conclusion

Euclidean diffusion and exact permutation generation are often viewed as requiring either a relaxation or a custom discrete process. We show that this is not intrinsic. The main limitations are scalability and architecture: although performance at large n improves over existing methods, it remains limited for practical use, and our linear-dimensional diffusion state is paired with quadratic-cost attention layers. Future work could develop near-linear architectures exploiting braid-fan or permutahedral structure, and richer continuous representations that retain Euclidean diffusion benefits while improving empirical performance.

Acknowledgments and Disclosure of Funding

Funding. This work was supported by the French National Research Agency (ANR) through the ARTEAK project (grant ANR-21-CE23-0001-01).

Competing interests. The authors declare no competing interests.

References

[1] R. P. Adams and R. S. Zemel. Ranking via sinkhorn propagation, 2011.

[2] M. S. Albergo and E. Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *International Conference on Learning Representations*, 2023.

[3] B. D. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

[4] D. L. Applegate, R. E. Bixby, V. Chvatál, and W. J. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2006. ISBN 9780691129938.

[5] J. Austin, D. D. Johnson, J. Ho, D. Tarlow, and R. van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, volume 34, pages 17981–17993, 2021.

[6] G. Birkhoff. Tres observaciones sobre el algebra lineal. *Univ. Nac. Tucumán. Revista A*, 5:147–151, 1946.

[7] M. Blondel, O. Teboul, Q. Berthet, and J. Djolonga. Fast differentiable sorting and ranking. In *International Conference on Machine Learning*, 2020.

[8] A. Campbell, J. Benton, V. De Bortoli, T. Rainforth, G. Deligiannidis, and A. Doucet. A continuous time framework for discrete denoising models. In *Advances in Neural Information Processing Systems*, volume 35, pages 28266–28279, 2022.

[9] M. Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, volume 26, 2013.

[10] M. Cuturi, O. Teboul, and J.-P. Vert. Differentiable ranking and sorting using optimal transport. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[11] J.-P. Doignon, A. Pekeč, and M. Regenwetter. The repeated insertion model for rankings: Missing link between two subset choice models. *Psychometrika*, 69(1):33–54, 2004. doi: 10.1007/BF02295838.

[12] A. Grover, E. Wang, A. Zweig, and S. Ermon. Stochastic optimization of sorting networks via continuous relaxations. In *International Conference on Learning Representations*, 2019.

[13] F. M. Harper and J. A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4), Dec. 2015. ISSN 2160-6455. doi: 10.1145/2827872.

[14] S. He, Y. Zhang, S. Zhang, and D. van Dijk. Learning permutation distributions via reflected diffusion on ranks, 2026.

[15] J. Heek, A. Levskaya, A. Oliver, M. Ritter, B. Rondepierre, A. Steiner, and M. van Zee. Flax: A neural network library and ecosystem for JAX, 2024.

[16] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851, 2020.

[17] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, Oct. 2002. ISSN 1046-8188. doi: 10.1145/582415.582418.

[18] C. K. Joshi, Q. Cappart, L.-M. Rousseau, and T. Laurent. Learning tsp requires rethinking generalization. In *International Conference on Principles and Practice of Constraint Programming*, 2021.

[19] J. Kim, J. Yoon, and M. Cho. Generalized neural sorting networks with error-free differentiable swap functions. In *International Conference on Learning Representations*, 2024.

[20] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[21] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

[22] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

[23] T. Lu and C. Boutilier. Effective sampling and learning for mallows models with pairwise-preference data. *Journal of Machine Learning Research*, 15(117):3963–4009, 2014.

[24] R. D. Luce. *Individual choice behavior: A theoretical analysis*. Wiley, 1959.

[25] C. L. Mallows. Non-null ranking models. i. *Biometrika*, 44(1/2):114–130, 1957.

[26] G. E. Mena, D. Belanger, S. Linderman, and J. Snoek. Learning latent permutations with gumbel-sinkhorn networks. In *International Conference on Learning Representations*, 2018.

[27] F. Petersen, C. Borgelt, H. Kuehne, and O. Deussen. Differentiable sorting networks for scalable sorting and ranking supervision. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8546–8555. PMLR, 18–24 Jul 2021.

[28] R. L. Plackett. The analysis of permutations. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 24(2):193–202, 1975.

[29] S. Prillo and J. Eisenschlos. Softsort: A continuous relaxation for the argsort operator. In *International Conference on Machine Learning*, 2020.

[30] G. Scarpellini, S. Fiorini, F. Giuliari, P. Morerio, and A. Del Bue. Diffassemble: A unified graph-diffusion model for 2d and 3d reassembly. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2024.

[31] D. Severo, B. Karrer, and N. Nolte. Learning distributions over permutations and rankings with factorized representations. In *International Conference on Learning Representations*, 2026.

[32] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

[33] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[34] H. Sun, L. Yu, B. Dai, D. Schuurmans, and H. Dai. Score-based continuous-time discrete diffusion models. In *International Conference on Learning Representations*, 2023.

[35] 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 I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[36] Y. Zhang, D. Yang, and R. Liao. Symmetricdiffusers: Learning discrete diffusion on finite symmetric groups. In *International Conference on Learning Representations*, 2025.

[37] G. M. Ziegler. *Lectures on Polytopes*, volume 152. Springer Science & Business Media, 1995.

A Technical Details

Common implementation. All experiments were implemented in JAX/Flax [15]. In every benchmark, the target permutation σ was first lifted to its centered permutahedral vertex $v_\sigma \in H_n$. The network output was always centered by subtracting its coordinate mean, so that the predicted score lies in H_n. At sampling time, we integrated the learned reverse VP dynamics and decoded only once, at the end, using argsort.

Unless otherwise specified, we used AdamW optimization, global gradient clipping at 1.0, EMA decay 0.999, and the VP schedule $\beta_{\min} = 0.1$, $\beta_{\max} = 20.0$, $t_\varepsilon = 10^{-3}$ and $T = 1$. Samples were generated with Euler–Maruyama integration. Across benchmarks, we kept the data construction, neural capacity, and evaluation protocol of the corresponding reference method whenever possible. All models are trained for 500 epochs. We used 4 Nvidia H100 80 Gb for our experiments.

A.1 TSP

For the TSP experiments, we followed the same architectural configuration as Soft-Rank Diffusion [14]. The input context consists of the 2D city coordinates. We encoded these coordinates with a sinusoidal coordinate encoder and used a Transformer score network with embedding dimension 256, feed-forward dimension 1024, 16 layers, 8 attention heads, and no dropout.

The TSP tour is invariant to cyclic shifts and reversal. To make the learning problem well-defined, we used the same canonicalization convention as the reference protocol: tours are represented in a fixed canonical order for supervision and are canonicalized again during evaluation. In the diffusion model, this means that the target canonical tour is lifted to v_σ, the reverse process evolves in H_n, and the final tour order is obtained by argsort. Thus we keep the TSP data and evaluation protocol unchanged, while replacing the Soft-Rank reverse denoiser by the Euclidean VP score model.

We used a learning rate 3×10^{-4}, weight decay 10^{-4}, and batch size 128 for TSP-20 and 64 for TSP-50. Sampling used 100 Euler–Maruyama reverse steps. This configuration has approximately 13.0M trainable parameters.

A.2 4-digit MNIST sorting

For 4-digit MNIST sorting, we followed the data generation and model scale of Soft-Rank Diffusion [14]. Each item is a horizontally concatenated four-digit MNIST image, and the target permutation sorts the items by their underlying integer value. Each image item was encoded independently by a two-layer CNN with 32 and 64 channels. The resulting item tokens were combined with the current diffusion coordinate Z_t^i, a sinusoidal time embedding, and positional information before being processed by the Transformer score network.

This adaptation keeps the benchmark exactly as a sorting task over unordered image items, but removes the soft-rank/reflected-diffusion state used by Soft-Rank Diffusion. The clean target order is instead represented by the centered permutahedral vertex v_σ, and the generated ordering is decoded by a final argsort.

For the short-sequence experiments, including $n = 32$, we used hidden dimension 512, time embedding dimension 256, 7 Transformer layers, 8 attention heads, dropout 0.1, and feed-forward dimension 2048. This model has approximately 30.25M trainable parameters. For the longer $n = 100$ and $n = 200$ settings, we kept the same width and depth but used feed-forward dimension 640, giving approximately 19.43M parameters. Training used learning rate 10^{-4}, weight decay 10^{-2}, and warmup of 51,600 steps.

A.3 CIFAR-10 jigsaw reconstruction

For CIFAR-10 jigsaw reconstruction, we followed the protocol used in the factorized permutation model of [31]. Each CIFAR-10 image is split into a $k \times k$ grid, the patches are randomly shuffled, and the target permutation is the inverse shuffle that restores the original image. We used the 2×2, 3×3, and 4×4 settings.

The adaptation to our model is direct: the shuffled patches form the context, while the restoration order is lifted to v_σ. Each patch was encoded by a two-layer CNN with 32 and 64 channels. The patch token, the current diffusion coordinate, and the time embedding were then passed to a Transformer score network. At inference, the model outputs centered utilities over the shuffled patches, and the restored order is obtained by sorting these utilities.

The Transformer used hidden dimension 128, time embedding dimension 128, feed-forward dimension 512, 7 layers, 8 heads, and dropout 0.1. We trained with batch size 128, learning rate 10^{-4}, and weight decay 10^{-2}. The model has approximately 1.63M parameters for 2×2 puzzles and 1.56M parameters for the 3×3 and 4×4 settings.

A.4 MovieLens 32M re-ranking

For MovieLens 32M re-ranking, we used the same candidate-set construction and insertion-prefix conditioning protocol as [31]. We kept the 1000 most-rated movies with at least 1000 ratings and used a 20% validation split. For each user, the target permutation is the ranking induced by held-out ratings. At test time, the model observes a prefix of this ranking and must complete the remaining order.

The context representation follows the insertion-prefix protocol of [31]. For each user, we first sample n rated movies from the selected MovieLens candidate pool and randomly permute their input order. Let m_i denote the movie at input index i, and let $\sigma \in S_n$ be the target ranking obtained by sorting these movies by the user's held-out rating, with random tie-breaking. We then convert this target ranking into its insertion-vector representation. If $r_\sigma(i)$ is the rank position of input item i in the target order, the insertion label of item i is

$$a_i = \left|\{j < i : \ r_\sigma(j) < r_\sigma(i)\}\right|, \qquad a_0 = 0.$$

Thus $a_i \in \{0, \dots, i\}$ records where item i would be inserted among the previously processed items.

At training and evaluation time, only a prefix of these insertion labels is revealed. For a prefix length r, item i receives the feature vector

$$\phi_i = \left[\text{onehot}(m_i), \ \mathbf{1}\{i < r\}, \ \mathbf{1}\{i < r\}\frac{a_i}{\max(i, 1)}\right].$$

The first component identifies the movie within the selected pool of 1000 movies, the second indicates whether the insertion label is observed, and the third gives the normalized insertion information when available and is set to zero otherwise. Hence the item feature dimension is 1002. In the original factorized model, these observed insertion labels define a partially observed factorized permutation code. In our model, the same conditional information is used only as context for the Transformer score network. The target permutation itself is still lifted to the centered permutahedral vertex v_σ, and generation proceeds by Euclidean reverse diffusion followed by a final argsort decode.

For $n = 10, 20, 50$, the observed prefix sizes were respectively $\{1, 5\}$, $\{1, 5, 10\}$, and $\{1, 5, 10, 25\}$. The score network was a Transformer with hidden dimension 256, time embedding dimension 128, feed-forward dimension 1024, 10 layers, 8 heads, and dropout 0.05. Training used batch size 128, learning rate 3×10^{-4}, weight decay 10^{-2}, and 500 epochs. Sampling used 100 reverse steps. This architecture has approximately 8.58M trainable parameters. Figure 5 reports the full experiment on this dataset.

B Training and Sampling Algorithms

Training. Training is standard denoising score matching after lifting permutations to permutahedral vertices. At each step, we sample a training pair (c, σ), noise the clean vertex v_σ with the VP transition, and regress the network output onto the conditional Gaussian score.

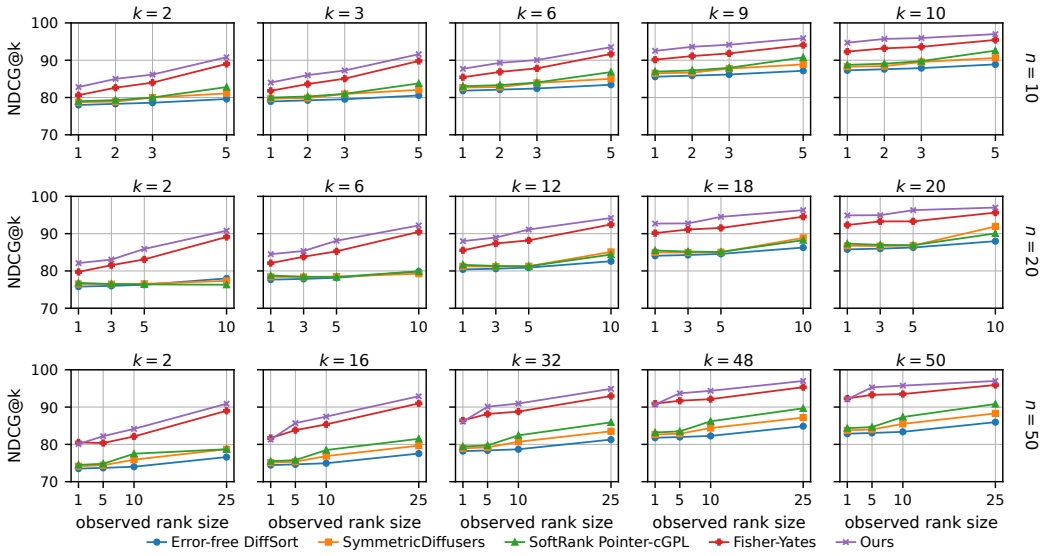


Figure 5: MovieLens 32M re-ranking results for $n \in \{10, 20, 50\}$.

Training procedure

1. Sample a minibatch of contexts and target permutations
$$\{(c_b, \sigma_b)\}_{b=1}^{B}.$$
2. Lift each permutation to the centered permutahedron:
$$x_b = v_{\sigma_b} \in H_n.$$
3. Sample diffusion times
$$t_b \sim \mathrm{Unif}(t_\varepsilon, T)$$
and centered Gaussian noise
$$\varepsilon_b \sim \mathcal{N}(0, I_{H_n}).$$
4. Form the noised inputs
$$z_b = \alpha_{t_b} x_b + \eta_{t_b} \varepsilon_b.$$
5. Compute the denoising score target
$$s_b^\star = -\frac{z_b - \alpha_{t_b} x_b}{\eta_{t_b}^2} = -\frac{\varepsilon_b}{\eta_{t_b}}.$$
6. Minimize the weighted score-matching loss
$$\mathcal{L}(\theta) = \frac{1}{B} \sum_{b=1}^{B} \lambda(t_b) \left\| s_\theta(z_b, t_b, c_b) - s_b^\star \right\|_{H_n}^2 .$$
7. Update θ with AdamW, apply global gradient clipping, and update the exponential moving average of the parameters.

Sampling. At inference time, the sampler starts from a centered Gaussian at terminal time T, integrates the learned reverse VP SDE, and decodes only once at the end. With a decreasing time grid and $M \in \mathbb{N}$ fixed:

$$T = t_M > t_{M-1} > \cdots > t_0 = t_\varepsilon, \qquad \Delta_k = t_k - t_{k-1},$$

Euler–Maruyama gives the update

$$z_{k-1} = z_k + \left[\frac{1}{2}\beta(t_k)z_k + \beta(t_k)s_\theta(z_k, t_k, c)\right]\Delta_k + \sqrt{\beta(t_k)\Delta_k}\,\xi_k, \qquad \xi_k \sim \mathcal{N}(0, I_{H_n}).$$

After every step we project back to H_n by subtracting the coordinate mean.

Euler–Maruyama sampling

1. Given a context c, sample the initial state
$$z_M \sim \mathcal{N}(0, I_{H_n}).$$
2. For $k = M, M-1, \ldots, 1$, sample $\xi_k \sim \mathcal{N}(0, I_{H_n})$ and compute
$$\widetilde{z}_{k-1} = z_k + \left[\frac{1}{2}\beta(t_k)z_k + \beta(t_k)s_\theta(z_k, t_k, c)\right]\Delta_k + \sqrt{\beta(t_k)\Delta_k}\,\xi_k.$$
3. Center the state:
$$z_{k-1} = \widetilde{z}_{k-1} - \frac{1}{n}\mathbf{1}\mathbf{1}^\top \widetilde{z}_{k-1}.$$
4. Decode the final utility vector by hard sorting:
$$\widehat{\sigma} = \mathrm{argsort}(z_0).$$

This procedure keeps the full reverse trajectory in the Euclidean space H_n. The only discrete operation is the final argsort, which guarantees that every generated sample is an exact permutation.

C Proofs for Section 5

Throughout this appendix we write $H = H_n$ to lighten notation. All Lebesgue measures, surface measures, gradients, and divergences are intrinsic to H. For a density u_t on H, the notation $\mathrm{argsort}_{\#}\, u_t$ means the pushforward of the measure $u_t(z)\,\mathrm{d}z$.

We use the following regularity convention. A density-current pair (u_t, J_t) is called regular if u_t, J_t, and their derivatives are sufficiently integrable for differentiation under the integral sign and for the divergence theorem on polyhedral chambers. Since the chambers are unbounded, we also assume the flux through $C_\sigma \cap \partial B_R$ vanishes as $R \to \infty$. These conditions hold for the Gaussian-mixture VP marginals used in the paper.

C.1 Proof of Theorem 5.1

We first record the elementary geometric fact used in the lifting part.

Lemma C.1 (Adjacent permutahedral edges cross the corresponding braid facet)**.** *Let $\tau = \sigma s_k$. Then*

$$v_\tau - v_\sigma = e_{\sigma(k+1)} - e_{\sigma(k)} = \sqrt{2}\, n_{\sigma,k}.$$

Moreover, the segment $[v_\sigma, v_\tau]$ crosses the braid arrangement only through the facet $F_{\sigma,k}$. It crosses this facet exactly once.

Proof. By definition, the coordinate of v_σ attached to item $\sigma(\ell)$ is $\bar{r}_\ell$, with

$$\bar{r}_\ell = \frac{n+1-2\ell}{2}.$$

Hence

$$\bar{r}_k - \bar{r}_{k+1} = 1.$$

The adjacent transposition $\tau = \sigma s_k$ only swaps the values attached to $\sigma(k)$ and $\sigma(k+1)$. Therefore

$$v_\tau - v_\sigma = (\bar{r}_{k+1} - \bar{r}_k)e_{\sigma(k)} + (\bar{r}_k - \bar{r}_{k+1})e_{\sigma(k+1)} = e_{\sigma(k+1)} - e_{\sigma(k)} = \sqrt{2}\, n_{\sigma,k}.$$

Let

$$z(r) = (1-r)v_\sigma + rv_\tau, \qquad r \in [0,1].$$

Along this segment, all coordinates are constant except the coordinates of $\sigma(k)$ and $\sigma(k+1)$. These two coordinates are

$$z_{\sigma(k)}(r) = \bar{r}_k - r, \qquad z_{\sigma(k+1)}(r) = \bar{r}_{k+1} + r.$$

They are equal if and only if $r = 1/2$. Since the transposed entries are adjacent in the sorted order, no other coordinate equality can occur along the segment. Thus the segment crosses exactly the facet where $z_{\sigma(k)} = z_{\sigma(k+1)}$, namely $F_{\sigma,k}$, and no other braid facet. □

We shall also use the following standard local divergence solver.

Lemma C.2 (Local divergence solver)**.** *Let $U \subset H$ be a bounded connected open set with smooth boundary, and let $f \in C_c^\infty(U)$ satisfy*

$$\int_U f(z)\,\mathrm{d}z = 0.$$

Then there exists $W \in C_c^\infty(U; H)$ such that

$$\operatorname{div} W = f.$$

Proof. For $\dim H = 1$, the result follows by taking, in a coordinate on H,

$$W(x) = \int_{-\infty}^{x} f(r)\,\mathrm{d}r,$$

which is compactly supported because $\int f = 0$. For $\dim H \geq 2$, this is the standard Bogovskii divergence construction on a smooth bounded domain. Since $\operatorname{supp}(f) \Subset U$, one may choose a smooth bounded domain D with $\operatorname{supp}(f) \Subset D \Subset U$, apply the Bogovskii operator on D, and extend the resulting field by zero outside D. This gives a compactly supported smooth vector field in U. □

We now prove the theorem.

Proof of Theorem 5.1. We prove the quotient statement and the converse lift separately.

From Euclidean flow to adjacent-swap dynamics. For each $\sigma \in S_n$, define

$$p_t(\sigma) = \int_{C_\sigma} u_t(z)\,\mathrm{d}z.$$

Since $\partial_t u_t + \operatorname{div} J_t = 0$, differentiation under the integral gives

$$\dot{p}_t(\sigma) = \int_{C_\sigma} \partial_t u_t(z)\,\mathrm{d}z = -\int_{C_\sigma} \operatorname{div} J_t(z)\,\mathrm{d}z.$$

Apply the divergence theorem on $C_\sigma \cap B_R$ and let $R \to \infty$. The flux through $C_\sigma \cap \partial B_R$ vanishes by the decay assumption. The boundary of C_σ is the union of the codimension-one facets $F_{\sigma,k}$, $k = 1, \ldots, n-1$, plus strata of codimension at least two. The latter do not contribute to surface measure. Hence

$$\dot{p}_t(\sigma) = -\sum_{k=1}^{n-1} \int_{F_{\sigma,k}} J_t(z) \cdot n_{\sigma,k}\,\mathrm{d}S.$$

By definition,

$$j_t(\sigma, \sigma s_k) = \int_{F_{\sigma,k}} J_t(z) \cdot n_{\sigma,k}\,\mathrm{d}S,$$

so

$$\dot{p}_t(\sigma) = -\sum_{k=1}^{n-1} j_t(\sigma, \sigma s_k) = -\sum_{\tau \sim \sigma} j_t(\sigma, \tau).$$

The antisymmetry is immediate. The same geometric facet viewed from the neighboring chamber has the opposite normal:

$$F_{\sigma,k} = F_{\sigma s_k,k}, \qquad n_{\sigma s_k,k} = -n_{\sigma,k}.$$

Therefore

$$j_t(\sigma, \sigma s_k) = -j_t(\sigma s_k, \sigma).$$

From adjacent-swap dynamics to a smooth Euclidean lift. Let $p_t \in \Delta(S_n)$ and let j_t be a smooth antisymmetric adjacent-swap current satisfying

$$\dot{p}_t(\sigma) = -\sum_{\tau \sim \sigma} j_t(\sigma, \tau).$$

Choose, for every $\sigma \in S_n$, a smooth nonnegative bump

$$\chi_\sigma \in C_c^\infty(C_\sigma), \qquad \int_{C_\sigma} \chi_\sigma(z)\,\mathrm{d}z = 1.$$

This is possible because $v_\sigma \in C_\sigma$.

Fix an arbitrary orientation E^+ of the adjacent-swap graph. For each oriented edge $e = (\sigma, \tau) \in E^+$, choose a thin smooth tube $U_e \subset H$ around the segment $[v_\sigma, v_\tau]$, enlarged near its endpoints so that

$$\operatorname{supp}(\chi_\sigma) \cup \operatorname{supp}(\chi_\tau) \subset U_e.$$

By Lemma C.1, the segment $[v_\sigma, v_\tau]$ crosses exactly the facet shared by C_σ and C_τ. Since the braid arrangement has finitely many facets, the tube can be chosen thin enough so that U_e also intersects no other braid facet.

For $e = (\sigma, \tau)$, define

$$f_e = \chi_\sigma - \chi_\tau.$$

Then $\int_{U_e} f_e = 0$. By Lemma C.2, there exists

$$W_e \in C_c^\infty(U_e; H)$$

such that

$$\operatorname{div} W_e = \chi_\sigma - \chi_\tau.$$

We claim that W_e has unit flux through the facet crossed by the oriented edge e. Suppose first that $\tau = \sigma s_k$ and that $e = (\sigma, \tau)$. Integrating $\operatorname{div} W_e$ over $U_e \cap C_\sigma$ gives

$$\int_{U_e \cap C_\sigma} \operatorname{div} W_e(z)\,\mathrm{d}z = \int_{U_e \cap C_\sigma} \chi_\sigma(z)\,\mathrm{d}z = 1.$$

The vector field W_e is compactly supported in U_e, so there is no flux through the outer boundary of the tube. Also, U_e intersects only the facet $F_{\sigma,k}$. Therefore the divergence theorem yields

$$\int_{F_{\sigma,k}} W_e(z) \cdot n_{\sigma,k}\,\mathrm{d}S = 1.$$

If the orientation is the reverse edge $e = (\tau, \sigma)$, the same argument gives flux $+1$ from C_τ to C_σ, hence flux -1 from C_σ to C_τ.

Now define

$$u_t(z) = \sum_{\sigma \in S_n} p_t(\sigma)\chi_\sigma(z),$$

and

$$J_t(z) = \sum_{(\sigma,\tau) \in E^+} j_t(\sigma, \tau) W_{(\sigma,\tau)}(z).$$

The density u_t is smooth, nonnegative, compactly supported, and has total mass one. Since each χ_σ is supported inside C_σ,

$$\int_{C_\eta} u_t(z)\,\mathrm{d}z = \sum_\sigma p_t(\sigma) \int_{C_\eta} \chi_\sigma(z)\,\mathrm{d}z = p_t(\eta).$$

Thus $\operatorname{argsort}_{\#} u_t = p_t$.

Next,

$$\partial_t u_t = \sum_\eta \dot{p}_t(\eta)\chi_\eta.$$

Moreover,

$$\operatorname{div} J_t = \sum_{(\sigma,\tau)\in E^+} j_t(\sigma,\tau)(\chi_\sigma - \chi_\tau).$$

Collecting the coefficient of χ_η, and using antisymmetry of j_t, we get

$$\operatorname{div} J_t = \sum_\eta \left(\sum_{\tau\sim\eta} j_t(\eta,\tau) \right) \chi_\eta.$$

Therefore

$$\partial_t u_t + \operatorname{div} J_t = \sum_\eta \left(\dot{p}_t(\eta) + \sum_{\tau\sim\eta} j_t(\eta,\tau) \right) \chi_\eta = 0.$$

So (u_t, J_t) satisfies the Euclidean continuity equation.

Finally, let $\tau = \sigma s_k$. The only tube crossing $F_{\sigma,k}$ is the tube associated with the edge $\{\sigma,\tau\}$. If this edge is oriented as (σ,τ), the flux from C_σ to C_τ is $j_t(\sigma,\tau)\cdot 1$. If it is oriented as (τ,σ), the unit flux is -1 in the direction from C_σ to C_τ, and the coefficient is $j_t(\tau,\sigma) = -j_t(\sigma,\tau)$. The product is again $j_t(\sigma,\tau)$. Hence

$$\int_{F_{\sigma,k}} J_t(z)\cdot n_{\sigma,k}\,\mathrm{d}S = j_t(\sigma,\sigma s_k).$$

This proves the smooth exact lift and completes the proof. □

C.2 Proof of Proposition 5.2

Proof of Proposition 5.2. Fix a context c, and write

$$p_\sigma = p_0^c(\sigma).$$

For $t > 0$, the noised density is

$$u_t^c(z) = \sum_{\sigma\in S_n} p_\sigma \mathcal{N}(z; \alpha_t v_\sigma, \eta_t^2 I_H).$$

Since all v_σ are permutations of the same centered rank vector $\bar{r}$, their norms are equal:

$$\|v_\sigma\|_H = \|\bar{r}\|_H \qquad \text{for all } \sigma.$$

Therefore, up to a factor independent of σ,

$$\mathcal{N}(z; \alpha_t v_\sigma, \eta_t^2 I_H) \propto \exp\left(-\frac{\|z - \alpha_t v_\sigma\|_H^2}{2\eta_t^2} \right) \propto \exp\left(\frac{\alpha_t}{\eta_t^2} \langle z, v_\sigma \rangle \right).$$

With

$$\kappa_t = \frac{\alpha_t}{\eta_t^2},$$

Bayes' rule gives

$$\mathbb{P}(V = v_\sigma \mid Z_t = z, c) = \frac{p_\sigma \exp\left(\kappa_t \langle z, v_\sigma \rangle\right)}{\sum_{\rho\in S_n} p_\rho \exp\left(\kappa_t \langle z, v_\rho \rangle\right)}.$$

This proves the posterior-weight formula.

Now define

$$S_{t,c}(z) = \sum_{\sigma\in S_n} p_\sigma \exp\left(\kappa_t \langle z, v_\sigma \rangle\right).$$

Then

$$A_{t,c}(z) = \frac{1}{\kappa_t} \log S_{t,c}(z).$$

Since κ_t is independent of z,

$$\nabla_z A_{t,c}(z) = \frac{\sum_\sigma p_\sigma \exp\left(\kappa_t \langle z, v_\sigma \rangle\right) v_\sigma}{\sum_\rho p_\rho \exp\left(\kappa_t \langle z, v_\rho \rangle\right)}.$$

The right-hand side is exactly the posterior barycenter

$$m_t(z, c) = \mathbb{E}[V \mid Z_t = z, c].$$

It is a convex combination of the vertices v_σ, hence

$$m_t(z, c) \in \Pi_n^\circ.$$

It remains to compute the score. Expanding the Gaussian mixture gives

$$u_t^c(z) = K_t \exp\left(-\frac{\|z\|_H^2}{2\eta_t^2}\right) S_{t,c}(z),$$

where K_t is independent of z. Hence

$$\mathfrak{s}_t(z, c) = \nabla_z \log u_t^c(z) = -\frac{z}{\eta_t^2} + \nabla_z \log S_{t,c}(z).$$

But

$$\nabla_z \log S_{t,c}(z) = \kappa_t m_t(z, c) = \frac{\alpha_t}{\eta_t^2} m_t(z, c).$$

Therefore

$$\mathfrak{s}_t(z, c) = \frac{\alpha_t m_t(z, c) - z}{\eta_t^2}.$$

Equivalently,

$$m_t(z, c) = \frac{z + \eta_t^2 \mathfrak{s}_t(z, c)}{\alpha_t}.$$

This proves the proposition. □

Low-noise consequence. The proposition also clarifies the precise meaning of the low-noise limit. Let

$$\mathcal{M}_c(z) = \arg\max_{\sigma:\, p_0^c(\sigma)>0} \langle z, v_\sigma \rangle .$$

As $t \downarrow 0$, one has $\kappa_t \to \infty$, and the posterior weights concentrate on $\mathcal{M}_c(z)$. If p_0^c has full support and z is away from the tie set, the rearrangement inequality implies that the unique maximizer of $\langle z, v_\sigma \rangle$ is

$$\sigma = \mathrm{argsort}(z).$$

In that case,

$$m_t(z, c) \longrightarrow v_{\mathrm{argsort}(z)}.$$

Without full support, the same statement holds only if $\mathrm{argsort}(z)$ belongs to the support of p_0^c and is the unique maximizer among supported permutations.

C.3 Proof of Theorem 5.3

Proof of Theorem 5.3. The forward VP SDE has drift

$$b_t^{\rightarrow}(z) = -\frac{\beta(t)}{2} z$$

and diffusion coefficient $\sqrt{\beta(t)}$. Its Fokker–Planck equation can be written

$$\partial_t u_t + \operatorname{div} J_t^{\rightarrow} = 0,$$

with current

$$J_t^{\rightarrow} = b_t^{\rightarrow} u_t - \frac{\beta(t)}{2} \nabla u_t = -\frac{\beta(t)}{2} \left(z\, u_t + \nabla u_t\right).$$

The same curve of marginals traversed in reverse time satisfies

$$-\partial_t u_t + \operatorname{div} J_t^{\leftarrow} = 0, \qquad J_t^{\leftarrow} = -J_t^{\rightarrow}.$$

Thus

$$J_t^{\leftarrow} = \frac{\beta(t)}{2} \left(z\, u_t + \nabla u_t\right).$$

Since

$$\mathfrak{s}_t = \nabla \log u_t \qquad \Longleftrightarrow \qquad \nabla u_t = u_t \mathfrak{s}_t,$$

we may also write

$$J_t^{\leftarrow}(z) = \frac{\beta(t)}{2} u_t(z) \left(z + \mathfrak{s}_t(z)\right).$$

We keep the original VP clock t, where $t = 0$ is data time and $t = T$ is noise time. Under reverse-time transport, the equation is

$$-\partial_t u_t + \operatorname{div} J_t^{\leftarrow} = 0.$$

Integrating over C_σ gives

$$-\dot{p}_t(\sigma) + \sum_{k=1}^{n-1} \int_{F_{\sigma,k}} J_t^{\leftarrow}(z) \cdot n_{\sigma,k} \, \mathrm{d}S = 0.$$

Therefore

$$\dot{p}_t(\sigma) = \sum_{k=1}^{n-1} j_t^{\leftarrow}(\sigma, \sigma s_k),$$

where

$$j_t^{\leftarrow}(\sigma, \sigma s_k) = \int_{F_{\sigma,k}} J_t^{\leftarrow}(z) \cdot n_{\sigma,k} \, \mathrm{d}S.$$

It remains to simplify the facet current. If $z \in F_{\sigma,k}$, then

$$z_{\sigma(k)} = z_{\sigma(k+1)}.$$

Since

$$n_{\sigma,k} = \frac{e_{\sigma(k+1)} - e_{\sigma(k)}}{\sqrt{2}},$$

we have

$$z \cdot n_{\sigma,k} = \frac{z_{\sigma(k+1)} - z_{\sigma(k)}}{\sqrt{2}} = 0.$$

Consequently, the Ornstein–Uhlenbeck part of the reverse current is tangent to the braid facet:

$$\left(\frac{\beta(t)}{2} z \, u_t(z)\right) \cdot n_{\sigma,k} = 0.$$

Thus

$$J_t^{\leftarrow}(z) \cdot n_{\sigma,k} = \frac{\beta(t)}{2} \nabla u_t(z) \cdot n_{\sigma,k} = \frac{\beta(t)}{2} u_t(z) \left\langle \mathfrak{s}_t(z), n_{\sigma,k} \right\rangle.$$

Integrating over $F_{\sigma,k}$ gives

$$j_t^{\leftarrow}(\sigma, \sigma s_k) = \frac{\beta(t)}{2} \int_{F_{\sigma,k}} u_t(z) \left\langle \mathfrak{s}_t(z), n_{\sigma,k} \right\rangle \mathrm{d}S.$$

This proves the theorem. □

Combining this result with Proposition 5.2 gives the posterior-barycenter form of the same current. On $F_{\sigma,k}$, $\langle z, n_{\sigma,k} \rangle = 0$, so

$$\left\langle \mathfrak{s}_t(z, c), n_{\sigma,k} \right\rangle = \frac{\alpha_t}{\eta_t^2} \left\langle m_t(z, c), n_{\sigma,k} \right\rangle.$$

Hence, conditionally on context c,

$$j_t^{\leftarrow}(\sigma, \sigma s_k \mid c) = \frac{\beta(t)\alpha_t}{2\eta_t^2} \int_{F_{\sigma,k}} u_t^c(z) \left\langle m_t(z, c), n_{\sigma,k} \right\rangle \mathrm{d}S.$$

Thus the adjacent-swap current is controlled by posterior permutahedral rank gaps on the shared sorting facet.

C.4 Proof of Theorem 5.4

Proof. Fix c and write $D = \text{argsort}$. Let $\mathbb{P}^\star$ be the exact reverse VP path measure initialized from u_T^c, and let $\mathbb{P}^\theta$ be the learned reverse VP path measure initialized from $\mathcal{N}(0, I_H)$. On the reverse clock $\tau \in [0, T-\varepsilon]$, with $t = T - \tau$, the two drifts are

$$b_\tau^\star(y) = \frac{1}{2}\beta(t)y + \beta(t)\mathfrak{s}_t^c(y), \qquad b_\tau^\theta(y) = \frac{1}{2}\beta(t)y + \beta(t)s_\theta(y, t, c),$$

with the same diffusion coefficient $\sqrt{\beta(t)}I_H$.

By Girsanov's theorem,

$$\mathrm{KL}(\mathbb{P}^\star \| \mathbb{P}^\theta) = \mathrm{KL}(u_T^c \| \mathcal{N}(0, I_H)) + \frac{1}{2}\int_\varepsilon^T \beta(t)\mathbb{E}_{Z_t \sim u_t^c}\left[\|s_\theta(Z_t, t, c) - \mathfrak{s}_t^c(Z_t)\|_H^2\right] \mathrm{d}t.$$

Using the definition of $\Delta_\theta(c)$, this becomes

$$\mathrm{KL}(\mathbb{P}^\star \| \mathbb{P}^\theta) = \mathrm{KL}(u_T^c \| \mathcal{N}(0, I_H)) + \frac{1}{2}\Delta_\theta(c).$$

Let $e(\omega) = \omega_{T-\varepsilon}$ be the endpoint map. Since

$$e_\# \mathbb{P}^\star = u_\varepsilon^c, \qquad e_\# \mathbb{P}^\theta = \widehat{u}_\varepsilon^{\theta,c},$$

data processing gives

$$\mathrm{KL}\big(D_\# u_\varepsilon^c \big\| D_\# \widehat{u}_\varepsilon^{\theta,c}\big) \le \mathrm{KL}(\mathbb{P}^\star \| \mathbb{P}^\theta).$$

Pinsker's inequality then yields

$$\mathrm{TV}\big(D_\# u_\varepsilon^c, D_\# \widehat{u}_\varepsilon^{\theta,c}\big) \le \left[\frac{1}{2}\mathrm{KL}(u_T^c \| \mathcal{N}(0, I_H)) + \frac{1}{4}\Delta_\theta(c)\right]^{1/2}.$$

This proves the theorem. □

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We believe that the abstract provides a good summary of our main contribution, namely, the study of a diffusion score in the centered Euclidean space for solving permutation problems.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The two main limitations are performance at large n, which is better than the SOTA but still limited for practical use, and the complexity of the architecture, which reduces the benefits of our linear-dimensional solution. These points are discussed in the conclusion.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in

favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: We have made our best effort to state all assumptions for each theorem, with the corresponding proofs provided in the supplementary material.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We have included a dedicated section in the supplementary material that lists all hyperparameters and model parameter counts.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: While we plan to release the source code, we did not include it in the supplementary material.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We have included all necessary information in the supplementary material for all benchmarks used.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Since our method is stochastic, we report results as averages over 5 runs, which is common practice in our subfield.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We have included the details of our computational resources.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our work does not raise specific ethical concerns.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: We do not identify any direct societal impacts of our work, as we address a combinatorial task with no known direct societal implications.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification:

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We do have quoted dataset and the main framework used. We also quote the inspiration for our neural network architecture.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification:

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification:

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification:

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification:

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.