CombiLatent: Neural Combinatorial Optimization via Latent Space Search under Sinkhorn Divergence Regularization

Younes BOUKACEM [1] **Hodhaifa BENOUAKLIL** [2] **Yassine AIT SAID** [3] **Houssem TABLI** [3]
Mehdi Zakaria ADJAL [4] **Faissal IZERMINE** [1]

Abstract

We introduce **CombiLatent**, a general two-phase neural framework for solving NP-hard combinatorial optimization problems via continuous latent-space optimization. A Transformer surrogate is first trained to predict the objective value of any candidate solution, embedding the combinatorial structure into a differentiable latent space. A learnable solution tensor is then optimized by gradient descent against the frozen surrogate, with Sinkhorn divergence (entropic optimal transport) enforcing permutation validity. As a first case study, we instantiate CombiLatent on the **Permutation Flow Shop Scheduling Problem (PFSP)**—a problem of significant industrial and environmental relevance that remains surprisingly underexplored in the neural combinatorial optimization literature. Our experiments show that CombiLatent is competitive with established PFSP heuristics (NEH, CDS, Palmer), and yield insights into how model capacity and regularization shape the latent optimization landscape.

1. Introduction

Combinatorial optimization problems are discrete by nature, making them fundamentally incompatible with gradient-based methods. Classical approaches—exact solvers, heuristics, meta-heuristics—operate directly in discrete spaces and do not scale gracefully. Neural combinatorial optimization (NCO) has emerged as a promising alternative, but most existing methods still sidestep gradients, relying on reinforcement learning with non-differentiable rewards or auto-regressive decoding over discrete tokens.

We propose **CombiLatent**, a framework that takes a different route: *embed the problem in a continuous latent space, then optimize directly with gradient descent*. It operates in two phases. First, a Transformer is trained to predict the objective value of any candidate solution, producing a differentiable surrogate of the combinatorial objective. Second, with the Transformer frozen, a learnable embedding tensor is optimized to minimize this surrogate; Sinkhorn divergence regularization (entropic optimal transport) keeps the continuous solution anchored to a valid discrete permutation, which is finally recovered via the Hungarian algorithm.

We instantiate CombiLatent on the **Permutation Flow Shop Scheduling Problem (PFSP)**, chosen for two reasons. First, it has clear real-world impact: efficient scheduling reduces manufacturing costs and industrial energy consumption. Second, it is notably underrepresented in the NCO literature, which has focused overwhelmingly on routing problems (TSP, VRP). Our experiments benchmark CombiLatent against NEH, CDS, and Palmer, and analyse the effects of model capacity and weight decay on the quality of the learned latent space.

2. Methodology: The Two-Phase Architecture

Our approach is divided into two distinct phases: (1) training a surrogate model to map continuous job embeddings to makespan estimations, and (2) optimizing a learnable set of embeddings to recover the optimal schedule. The global architecture is shown in Figure 1.

2.1. Phase 1: Surrogate Model Training

In the first phase, we train a sequence model (a Transformer with relative positional embeddings) to predict the makespan of a given schedule. Let the jobs be represented by a dictionary of embeddings $X_1 \in \mathbb{R}^{n \times d}$, where d is the embedding dimension. A sequence of jobs (a schedule) is passed through the Transformer layers f_θ.

The model is trained via supervised learning to minimize the Mean Squared Error (MSE) between the predicted

[1]Master 2 student at Institut Polytechnique de Paris / Ecole Nationale Des Ponts et Chaussées [2]Master 2 student at Universitée Grenoble Alpes [3]Master 2 student at Universitée Paris Saclay [4]PhD student at Universitée Paris Saclay. Correspondence to: Younes BOUKACEM <yo.boukacem@gmail.com>.

Accepted at ICML 2026 Workshop on Decision-Making from Offline Datasets to Online Adaptation: Black-Box Optimization to Reinforcement Learning.

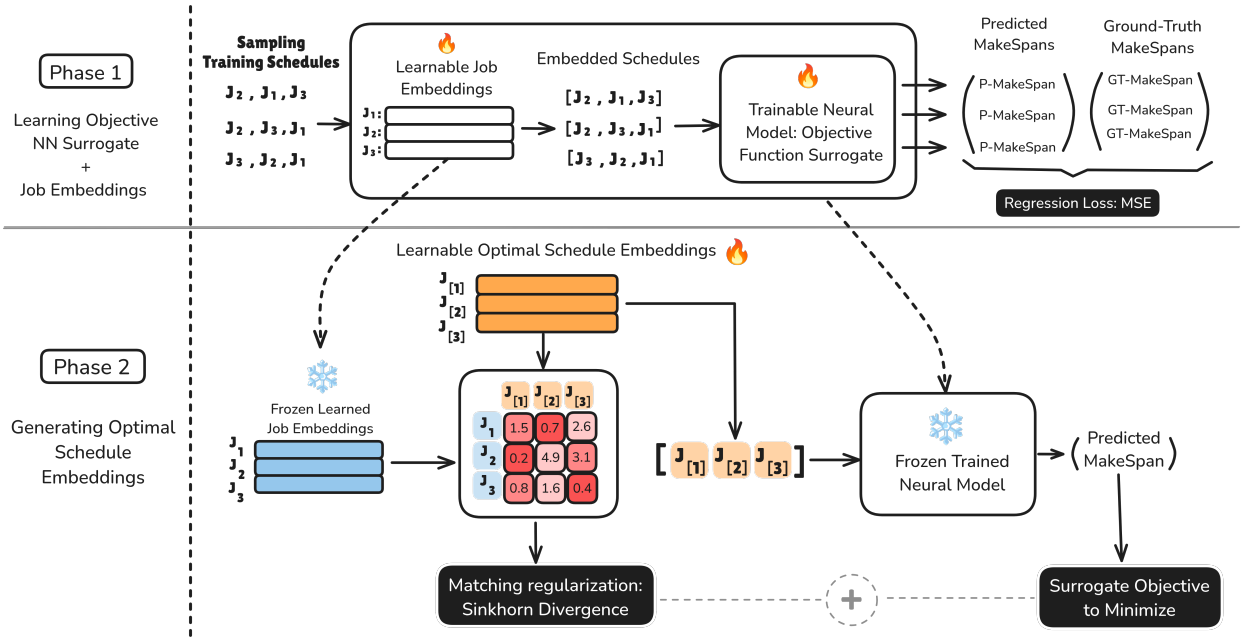

Figure 1. Global Architecture of the Flowshop Transformer Pipeline

makespan and the true makespan y:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{B}\sum_{b=1}^{B}\left(f_\theta(X_1^{(b)}) - y^{(b)}\right)^2 \tag{1}$$

Once trained, the Transformer accurately models the physical dynamics of the flow shop, and its token embedding table X_1 contains rich, latent representations of the jobs. We then **freeze** the Transformer's weights θ.

2.2. Phase 2: Latent Schedule Optimization

To find the optimal schedule, we introduce a new, learnable tensor $X_2 \in \mathbb{R}^{n\times d}$, which represents the embeddings of the jobs at each position in the optimal schedule. We initialize X_2 as a random permutation of X_1.

We optimize X_2 via gradient descent to minimize the makespan predicted by our frozen surrogate model. However, to ensure that the learned continuous vectors in X_2 actually correspond to the exact set of real jobs in X_1, we regularize the optimization using **Sinkhorn Divergence** (Entropic Optimal Transport).

The Sinkhorn distance between the fixed job embeddings X_1 and the learnable slots X_2 is defined as:

$$\mathcal{L}_{\text{Sinkhorn}}(X_1, X_2) = \min_{P\in U(1_n,1_n)} \sum_{i,j} P_{ij}C_{ij} - \epsilon H(P) \tag{2}$$

where:

- $C_{ij} = \|(X_1)_i - (X_2)_j\|^2$ is the pairwise cost matrix.
- P is the transport plan (a soft permutation matrix).
- $H(P) = -\sum_{i,j} P_{ij}\log P_{ij}$ is the entropic regularization term, making the assignment differentiable.

The total objective function optimized during Phase 2 is thus:

$$\mathcal{L}_{\text{total}} = f_\theta(X_2) + \lambda \cdot \mathcal{L}_{\text{Sinkhorn}}(X_1, X_2) \tag{3}$$

Once X_2 converges, we project it back to the discrete space using the Hungarian Algorithm to recover the exact job permutation.

3. Experimental Setup

To validate this approach and simulate real-world conditions where the absolute optimal schedule is unknown, we designed a robust, synthetic experimental framework illustrated in Figure 2.

Data Generation We generated 15 random PFSP instances, with the number of jobs $n \in [7, 9]$ and machines $m \in [2, 6]$. Execution times were sampled uniformly $\sim U(0, 1)$. Because n was kept relatively small, we were able to compute the exact makespan for *every possible permutation* of jobs, creating 15 "Exhaustive" datasets.

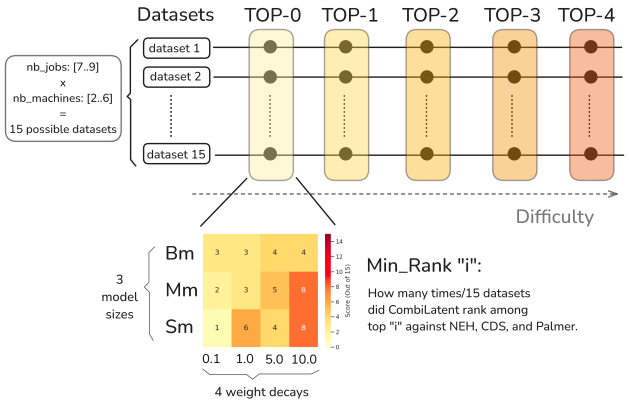

Figure 2. Datasets and experimental setup

Dataset Degradation Strategy In realistic, large-scale scenarios, the true global optimum is rarely present in the training data. To simulate this, we generated 5 subsets from each exhaustive dataset:

- **top_0**: The original exhaustive dataset.
- **top_1**: The exhaustive dataset with the absolute best schedule removed.
- **top_2 to top_4**: Datasets with the top 2, 3, and 4 best schedules removed, respectively.

Model Configurations We evaluated the architecture across three model sizes to study the effect of representational capacity, and four levels of weight decay to manage the convexity of the latent space:

- **Model Sizes**: Big (`Bm` $\approx$ 6M params), Medium (`Mm` $\approx$ 800K params), Small (`Sm` $\approx$ 100K params).
- **Weight Decays**: 0.1, 1.0, 5.0, 10.0.

To mitigate the risk of Phase 2 optimization getting trapped in local optima, we introduced Langevin-like training dynamics: Gaussian noise $\mathcal{N}(0, \sigma^2)$ is added to X_2 at uniform intervals during gradient descent, periodically perturbing the schedule embeddings out of potential local minima.

Benchmarking We benchmarked CombiLatent against standard heuristic algorithms specialized for PFSP: **CDS**, **Palmer**, and **NEH**. The metric used is "Min Rank k", representing the number of PFSP instances (out of 15) where the makespan of CombiLatent's recovered schedule ranked among the top k makespans against the heuristics ($k \in [1..3]$).

4. Results and Analysis

The heatmaps in Figure 3 display the "Min Rank" scores (out of 15) for all combinations of datasets, model sizes, and weight decay configurations, with darker red indicating higher performance. We present our observations and their analysis in the following paragraphs.

1. The Role of Weight Decay in Landscape Smoothing As we expected, the performance of CombiLatent is very sensitive to weight decay. We can observe that as the training dataset is degraded from `top_0` to `top_4` (meaning the model has never seen the optimal solutions during Phase 1), the models more heavily rely on high weight decay (e.g., 5.0 or 10.0).

Analysis: In Phase 2, a highly non-convex latent space will more easily trap X_2 in local minima. High weight decay acts as a strong regularizer that "smooths" the loss landscape of the surrogate model and prevent it from overfitting to combinatorial artifacts, making it easier for Phase 2 to navigate towards good optimum.

2. Effect of Model Capacity on Landscape Smoothing Here also, we confirm our intuition that bigger models do not necessarily produce better results. The Small model (`Sm`) frequently matches or outperforms the Big model (`Bm`), achieving the highest scores (e.g., `Sm` achieves 8/15 in Min Rank 2 for **top_4** while `Bm` only achieves 4).

Analysis: The Big model likely overfits the combinatorial artifacts of the training data. While it may predict makespans with low MSE during validation, its latent manifold is too complex for Phase 2 optimization. The Small model, restricted by its capacity, is forced to learn the broader "physics" of the scheduling problem, leading to a better inductive bias for the gradient descent in Phase 2.

3. Competitiveness Against Heuristics The results indicate that the latent optimization strategy shows signs of competitiveness. For the `Sm` model with high weight decay, the framework consistently ranks in the top 3—and frequently 1st or 2nd—against the established, highly specialized heuristics that we compared with (NEH, CDS, and Palmer) even as we degraded the datasets exhaustivity.

Analysis: These promising results show that CombiLatent approach manages to capture the dynamics of the underlying combinatorial problem to transfer it to a continuous manifold where optimization can be performed more predictively.

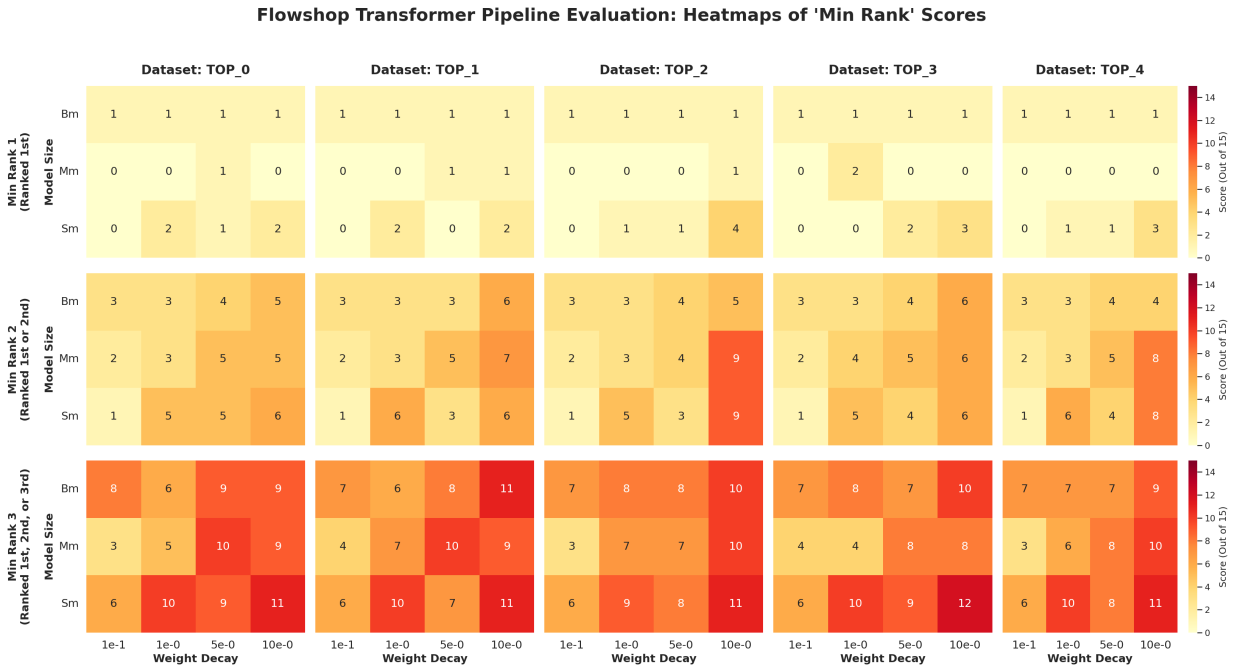

Figure 3. Obtained results: each row represents the results for a Min Rank "i"

5. Limitations and Future Directions

CombiLatent has three main limitations:

1. **Instance-specific retraining:** the surrogate must currently be retrained per instance. This could be mitigated by conditioning it on an MLP-encoded representation of the execution time matrix, which would enable the generalization to instances different from those seen in training.

2. **Latent landscape non-convexity:** As we have seen, Phase 2 gradient descent is prone to local minima. We partiallay mitigated this via high weight-decay, reasonable model sizes and Langevin-like dynamics, but more principled non-convex optimizers should be explored (e.g., a latent "Tabu-search" method that repulses the Phase 2 optimization from detected local minima by adding the proximity to them as a penalization in the Phase 2 loss.).

3. **Experimental scale:** experiments were restricted to small instances ($n \in [7, 9]$, $m \in [2, 6]$) due to compute constraints; scaling to practically relevant sizes (such as $n \in [20, 500]$ as in Taillard Benchmarks) is the most important direction for future work.

6. Conclusion

CombiLatent tested on PFSP successfully demonstrates that combinatorial scheduling problems can be solved in a purely continuous latent space. By leveraging sequence models for evaluation and Sinkhorn Divergence for permutation adherence, we can optimize schedules directly via backpropagation. Crucially, the experimental results prove that aggressive regularization (high weight decay) and smaller model capacities are the key to unlocking smooth latent manifolds that generalize well, even when the optimal schedules are hidden from the training data.

References

Bello, I., Pham, H., Le, Q. V., Norouzi, M., and Bengio, S. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

Bengio, Y., Lodi, A., and Prouvost, A. Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, 2021.

Campbell, H. G., Dudek, R. A., and Smith, M. L. A heuristic algorithm for the n job, m machine sequencing problem. *Management Science*, 16(10):B630–B637, 1970.

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., et al. Au-

tomatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4(2): 268–276, 2018.

Kool, W., van Hoof, H., and Welling, M. Attention, learn to solve routing problems! In *International Conference on Learning Representations (ICLR)*, 2019.

Li, Y., Gama, F., Yolcu, A., Smola, A., Russell, S., and Garg, S. T2t: From distribution learning in training to gradient search in testing for combinatorial optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023.

Li, Y. et al. Fast t2t: Optimization consistency speeds up diffusion-based combinatorial solvers. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, 2024.

Nawaz, M., Enscore Jr, E. E., and Ham, I. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, 11(1):91–95, 1983.

Palmer, D. S. Sequencing jobs through a multi-stage process in the minimum total time—a quick method of obtaining a near optimum. *Operational Research Quarterly*, 16(1): 101–107, 1965.

Shaw, P., Uszkoreit, J., and Vaswani, A. Self-attention with relative position representations. In *Proceedings of NAACL-HLT*, 2018.

Trabucco, B., Kumar, A., Geng, X., and Levine, S. Conservative objective models for effective offline model-based optimization. In *International Conference on Machine Learning (ICML)*, 2021.

Vinyals, O., Fortunato, M., and Jaitly, N. Pointer networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 28, 2015.

Yu, S., Ahn, S., Song, L., and Shin, J. Roma: Robust model adaptation for offline model-based optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021.

A. Related Work

Solving combinatorial optimization (CO) problems with deep learning is a recent and rapidly evolving field, often referred to as **Neural Combinatorial Optimization (NCO)** (Bengio et al., 2021). NCO models typically fall into one of three paradigms.

Supervised Learning. Early NCO methods trained models to directly predict or construct solutions from labeled instances (Vinyals et al., 2015). The fundamental drawback is that training requires optimal or near-optimal labels, which are themselves NP-hard to obtain at scale—creating a circular dependency at the core of the approach.

Reinforcement Learning. To bypass the need for labels, RL-based methods train a policy to construct solutions sequentially, using solution quality as a reward signal (Bello et al., 2016; Kool et al., 2019). While label-free, these approaches suffer from the well-known instability of policy gradient optimization: high variance, sensitivity to reward shaping, and a tendency to converge to local optima.

Test-Time Optimization (TTO). A more recent and emerging paradigm recognizes that minimizing average loss across training instances is fundamentally misaligned with the true CO goal of solving *each* test instance optimally. TTO methods instead train a model to encode problem structure, then perform an instance-specific optimization at inference time. The most closely related work is **T2T** (Li et al., 2023), which trains a discrete diffusion model and, at test time, steers the denoising process via the gradient of the objective. Our approach differs from T2T along every technical dimension: (i) we train a *discriminative* Transformer surrogate (predicting makespan) rather than a *generative* diffusion model; (ii) our test-time search operates entirely in a continuous embedding space rather than through iterative denoising over discrete variables; and (iii) permutation feasibility is enforced by Sinkhorn divergence between the learned slot embeddings X_2 and the frozen job embeddings X_1, rather than by the transition kernels of a diffusion chain. A NeurIPS 2024 follow-up, **Fast T2T** (Li et al., 2024), accelerates T2T's test-time search via optimization consistency mappings, confirming that this paradigm is actively being developed.

Offline Model-Based Optimization (MBO). Our Phase 1 surrogate training is also related to the offline MBO literature (Trabucco et al., 2021; Yu et al., 2021), where a proxy model is trained on a fixed dataset of designs and their scores, and is then optimized at test time. Standard offline MBO targets continuous or biological design spaces using MLP surrogates and VAE-based latent spaces (Gómez-Bombarelli et al., 2018). CombiLatent departs from this template in three ways: (i) it targets a *permutation-structured* NP-hard problem, to our knowledge the first offline MBO application in this combinatorial regime; (ii) it uses a Transformer surrogate, which naturally captures the relational structure between jobs; and (iii) it replaces the encoder-decoder of VAE-based methods with a direct Sinkhorn alignment between X_1 and X_2, eliminating the need for a separate generative model.

PFSP in the NCO Literature. The Permutation Flow Shop Scheduling Problem is a well-studied combinatorial problem in the operations research community, where classical heuristics such as NEH (Nawaz et al., 1983), CDS (Campbell et al., 1970), and Palmer (Palmer, 1965) remain competitive baselines. Despite its industrial relevance, the PFSP has received little attention in the NCO literature, which has focused predominantly on routing problems such as TSP and VRP (Kool et al., 2019). CombiLatent is, to our knowledge, the first NCO approach to apply the TTO/offline-MBO paradigm to the PFSP.

B. Ablation Studies

To assess the contribution of the two key components of Phase 2 — the Sinkhorn divergence regularization and the Langevin-like noise injection — we ran two ablation experiments using as a baseline the best-performing configuration identified in section 4, namely the `Sm` model with weight decay 10^1), whose results from section 4 are reported again in Figure 4.

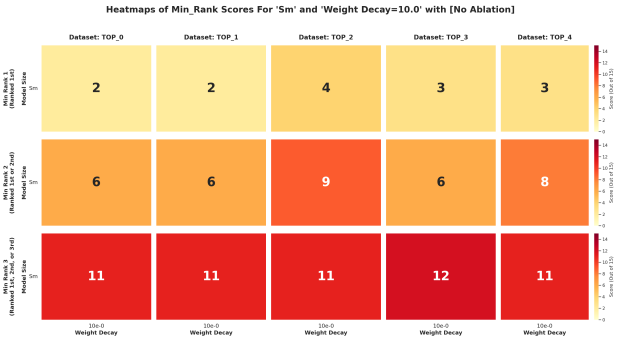

Figure 4. Baseline: `Sm` model, weight decay 10^1, no ablation.

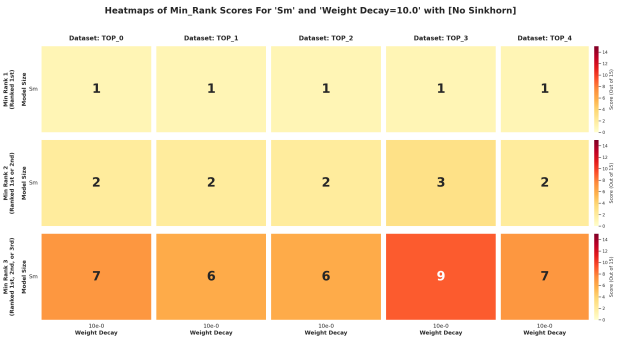

Figure 5. results of ablation 1 where Sinkhorn regularization is removed.

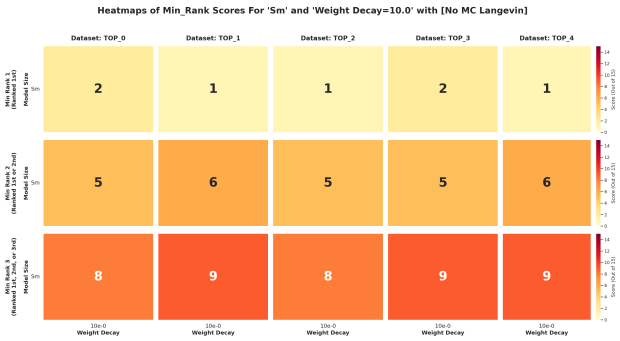

Figure 6. results of ablation 2 where Langevin-like noise injection is removed.

Removing Sinkhorn regularization (Figure 5). This is the most damaging ablation. Without Sinkhorn, Min Rank 1 collapses to 1/15 across all dataset degradation levels, and Min Rank 3 drops to 6–9/15 compared to 11–12/15 in the baseline. This is expected: without the optimal transport penalty anchoring X_2 to valid permutations of X_1, the gradient descent over the schedule embeddings is free to converge to continuous vectors that do not correspond to any real job, rendering the Hungarian projection unreliable. Sinkhorn regularization is therefore not merely a regularizer but a *structural necessity* of the framework.

Removing Langevin-like dynamics (Figure 6). The degradation is clear but more moderate. Min Rank 3 drops to 8–9/15 (from 11–12/15), and Min Rank 2 drops to 5–6/15 (from 6–9/15). Without periodic Gaussian noise injections, Phase 2 optimization is more susceptible to getting trapped in local minima of the latent landscape, confirming that the noise plays a meaningful role in escaping poor basins. The fact that performance does not collapse entirely suggests that high weight decay already provides significant landscape smoothing on its own, with Langevin dynamics acting as a complementary mechanism.

C. Implementation Details

Surrogate Architecture. The surrogate model is a decoder-only Transformer with relative positional embeddings (Shaw et al., 2018), trained via supervised regression on (permutation, makespan) pairs. This architectural choice is motivated by a key inductive bias: computing the makespan is essentially a dynamic programming algorithm that processes jobs sequentially, which is well aligned with the auto-regressive nature of decoder-only Transformers.

Model Configurations. The three model sizes are as follows:

- **Small (`Sm`)**: embedding dim $d = 64$, 4 attention heads, 2 layers $\approx$ 100K parameters.
- **Medium (`Mm`)**: embedding dim $d = 128$, 8 attention heads, 4 layers $\approx$ 800K parameters.
- **Big (`Bm`)**: embedding dim $d = 256$, 16 attention heads, 8 layers $\approx$ 6M parameters.

All models share a FFN width multiplier of 4 (i.e., the inner FFN dimension is $4d$); remaining hyperparameters are provided in the released code.

Training Details. All experiments were conducted on a single NVIDIA RTX 3070 (8GB VRAM). Phase 1 training was run for 5 epochs across all model sizes. Phase 2 optimization was run for 1500 gradient descent iterations, with the Sinkhorn divergence coefficient $\lambda = 2$ in $\mathcal{L}_{\text{total}}$. Langevin-like Gaussian noise was injected into X_2 every 200 iterations to facilitate escape from local minima.

D. Additional Experiments

D.1. More Fine-grained Study of the Effect of Surrogate Parameterization

Building on the observation from our initial experiments in section 4 that the smallest surrogate architecture ($n_embd = 64$, $n_head = 4$, $n_layer = 2$, referred to as Sm) consistently delivered the best Phase 2 recovery performance, we decide to study the importance of adjusting the parametric expressive power of the surrogate model by varying the key parameters that control it i.e. n_embd, n_head, and n_layer.

We thus conduct an exploration of the low-parameterization region of the architecture space, focusing on the hardest available instance i.e. 9 jobs, 6 machines. We fix the weight decay at 10.0 and the Phase 2 protocol without Langevin-like dynamics, and sweep all combinations of $n_head \in \{1, 2, 3, 4\}$, $n_layer \in \{1, 2, 3, 4\}$, and n_embd taking all values from 6 to 66 (constrained to be divisible by n_head), yielding 200 distinct configurations in total. A configuration is deemed a winner if the best makespan recovered by Phase 2 matches the global optimum of the instance (we use the top-0 version of the dataset i.e. all the schedules are available in the dataset including the optimal ones). We summarize results through two complementary visualizations:

- Figure 7 presents the win rate by n_embd bar chart aggregates: for each embedding dimension, we compute the fraction of (n_head, n_layer) combinations that reach the global optimum, revealing how raw model width interacts with performance across the tested range.
- Figure 8 presents the win rate heatmap by (n_head, n_layer) pairs, marginalizing for each such pair over the n_embd dimension.

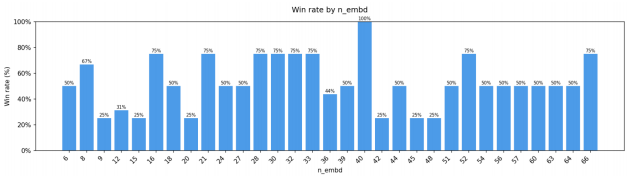

Figure 7. Win rate (fraction of configurations matching the global optimum makespan) as a function of the embedding dimension n_{embd}, aggregated over all tested combinations of $n_{\text{head}} \in \{1, 2, 3, 4\}$ and $n_{\text{layer}} \in \{1, 2, 3, 4\}$ on the 9-job 6-machine PFSP instance.

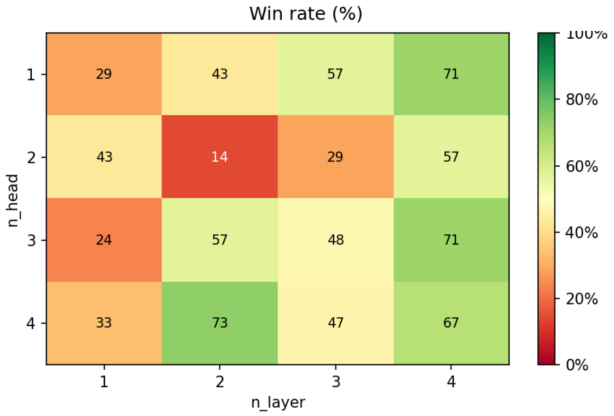

Figure 8. Win rate (%) for each combination of $n_{\text{head}} \in \{1, 2, 3, 4\}$ and $n_{\text{layer}} \in \{1, 2, 3, 4\}$, aggregated over all tested values of n_{embd} on the 9-job 6-machine PFSP instance. Each cell reports the percentage of n_{embd} configurations that recovered the global optimum makespan in Phase 2.

The analysis of Figure 7 reveals a substantial variance in the scores delivered by each embedding dimension (with scores ranging from 25% to 100%), suggesting the importance of properly calibrating this parameter. Furthermore, we find that one of the embedding dimensions, `n_embed=40` delivers a perfect score of 100% win rate across all (n_head, n_layer) pairs, suggesting that the performance of the surrogate model may be mostly sensitive to the embedding dimension of the jobs, rather than the "processing power" of the model which is controlled by its number of attention heads and layers.

This analysis if further confirmed by Figure 8, which shows that no single combination of (n_head, n_layer) achieves

a perfect score across all n_embd values. Additionally, Figure 8 reveals two close best scores, 73% and 71%, directly followed by a large drop to 67%. Each of these scores is achieved by either having a high n_head and low n_layer (for the 73% score) of inversely a high n_layer and low n_head (for the 71% score). This suggests that it may be beneficial to only augment one of these two parameters at the expense of the other to avoid spurious surrogate representations that will hinder Phase 2 optimization.

D.2. More Fine-grained Study of the Effect of Data Quality

To further assess how the quality of training data affects CombiLatent's ability to recover near-optimal solutions, we designed a controlled degradation experiment on the exhaustive 9-job, 6-machine PFSP instance. Starting from the complete dataset of all 9! = 362,880 permutations and their exact makespans, we systematically removed the top-p% schedules with the lowest (best) makespan values, for p in {5, 10, . . . , 95} (i.e. we remove increasing values of best percentiles). At each degradation level, we trained the surrogate model from scratch using the best configuration identified in the previous experiment from subsection D.1 (n_embd=40, n_head=4, n_layer=2) and ran Phase 2 latent optimisation without Langevin-like dynamics. This simulates a a setting where the training corpus is biased toward poor-quality solutions – as would be the case when labels are collected from cheap, suboptimal heuristics rather than exhaustive enumeration. The experiment isolates the effect of data quality independently of model capacity or optimisation dynamics, revealing whether the latent space learned from degraded data still carries enough signal for Phase 2 to navigate toward the global optimum. The results are shown in Figure 9

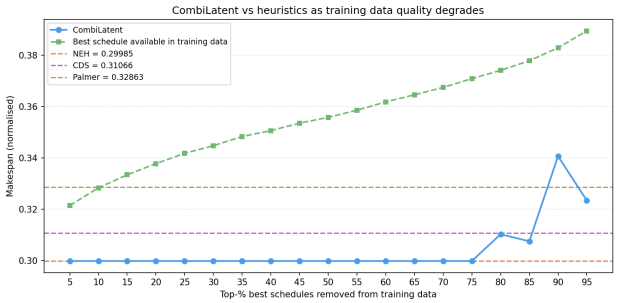

Figure 9. Best makespan recovered by CombiLatent as a function of the fraction of optimal schedules withheld from training, on the exhaustive 9-job 6-machine PFSP instance. The dashed orange line at the bottom indicates where CombiLatent matches the global optimum.

The analysis of Figure 9 reveals that despite increasing the value of the removed percentile, CombiLatent manages to reach the optimal schedule up until 75% of removed schedules. Passed that point, the model performance varies oscillates instead of necessarily systematically degrading. This suggests that under appropriate pre-hoc hyper-parameter tuning of the surrogate, CombiLatent may remain robust under low quality training schedules

This utlimately encourages us to scale CombiLatent to larger, more realistic PSFP instances, potentially combining it with some other mechanisms such as working in tandem with heuristic methods to make it operational and competitive.