Complexity Scaling Laws for Neural Models using Combinatorial Optimization

Lowell Weissman[†]
Virginia Tech

Michael Krumdick
Kensho Technologies

A. Lynn Abbott[†]
Virginia Tech

Abstract

Recent work on neural scaling laws demonstrates that model performance scales predictably with compute budget, model size, and dataset size. In this work, we develop scaling laws based on *problem complexity*. We analyze two fundamental complexity measures: solution space size and representation space size. Using the Traveling Salesman Problem (TSP) as a case study, we show that combinatorial optimization promotes smooth cost trends, and therefore meaningful scaling laws can be obtained even in the absence of an interpretable loss. We then show that suboptimality grows predictably for fixed-size models when scaling the number of TSP nodes or spatial dimensions, independent of whether the model was trained with reinforcement learning or supervised fine-tuning on a static dataset. We conclude with an analogy to problem complexity scaling in local search, showing that a much simpler gradient descent of the cost landscape produces similar trends.[1]

1 Introduction

Neural network performance usually improves with more parameters, more training, and more data. While small increases in budget improve performance unpredictably, model performance becomes highly predictable at scale, often with surprisingly simple trends [1, 2]. Such trends have been called neural scaling laws [3, 4], where performance smoothly improves when scaling model size, dataset size, or compute budget. This principle is summarized elegantly [3] by analogy to the ideal gas law, which describes the macroscopic behavior of a gas independent of its microscopic dynamics.

Neural scaling laws emerge over a wide variety of problems and problem scales [4, 5, 6, 7, 8, 9]. This observation implies the existence of a general underlying order, one that persists across the diverse natures of various tasks. Neural scaling laws capture slices of this behavior, fixing variables of the problem so that their influence is absorbed within the constants of a specific trend.

In this paper, we hypothesize that model performance is similarly predictable when scaling fundamental measures of problem complexity. We directly isolate this relationship and scale problem complexity for parameter-constrained, fixed-size models while approximating the regime in which compute and data are unconstrained. We take an initial step toward predicting the limit of performance as a function of the deep learning algorithm, the capacity of the model, and the properties of the task.

If we study measures of complexity that are task-specific, it will be difficult to draw meaningful conclusions that are general in nature, even if smooth scaling laws emerge. For example, Jones [10] extends compute scaling laws to also be a function of Hex board size. Measures like game-tree complexity and state-space complexity are well-studied for Hex [11] but have complicated relationships with board size [12] and are primarily applicable to sequential games. We approach this issue by distilling two fundamental measures of problem complexity inherent in any deep learning

[†]Bradley Department of Electrical and Computer Engineering. Correspondence to: <lowell6@vt.edu>

[1]Project code, along with our new dataset of 128 million optimal TSP solutions, is provided at: `https://github.com/lowellw6/complexity-scaling-laws`

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

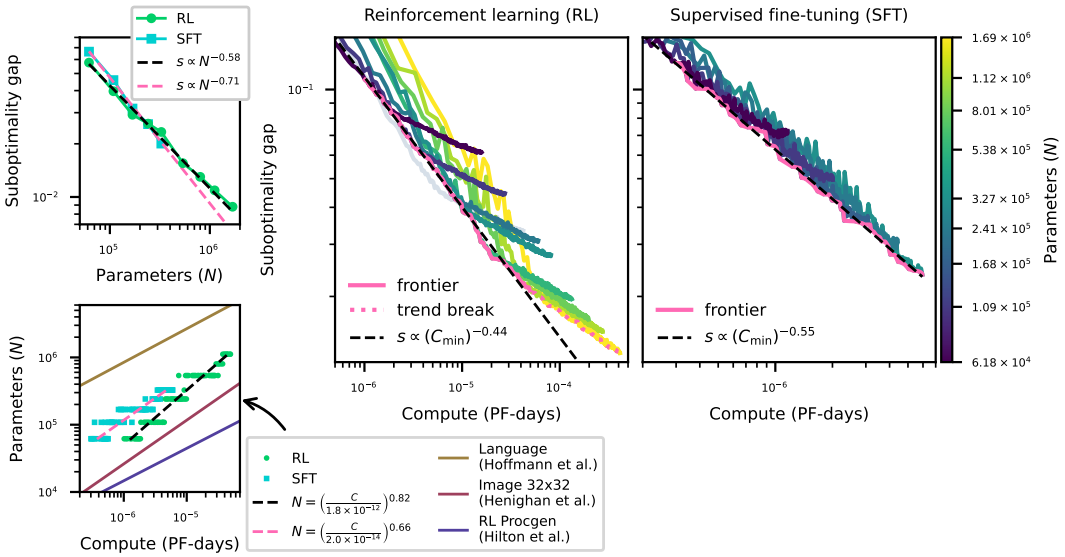

Figure 1: Suboptimality gap, defined as the difference between mean model performance and mean optimal performance, smoothly power decays with respect to model size and compute for both reinforcement learning (RL) and supervised fine-tuning (SFT) in TSP. Fits suggest that SFT is more compute-efficient than RL, and possibly more parameter-efficient as we scale to larger models, with faster decay toward optimal performance (larger α). **Top left:** Suboptimality w.r.t. parameters (N) for models evaluated near convergence. **Right:** Suboptimality w.r.t. compute (C) evaluated throughout training, where the compute-efficient frontier power decays. Note that the compute axis for SFT has been stretched for easier viewing. **Bottom left:** Optimal model size follows power growth w.r.t. compute budget. This relationship is strikingly consistent between domains [4, 13, 14].

task: size of the solution space and size of the representation space. Isolating each measure to a single variable is often not feasible in prevalent domains such as generative modeling and vision. We leverage combinatorial optimization toward this goal, using the Traveling Salesman Problem (TSP) to decouple the solution space and representation space, scaling TSP nodes or spatial dimensions independently. We then examine the limit of model performance when confronting the combinatorial growth of the solution space along with the curse of dimensionality.

Contributions: We obtain model size and compute budget scaling laws for TSP that predict model suboptimality, allowing us to make direct comparisons between reinforcement learning (RL) and supervised fine-tuning (SFT) on the same task (Section 3; Figure 1). We then obtain smooth problem complexity scaling laws characterized by simple trends for parameter-constrained models (Section 4). However, we show that these trends must break down at very large TSP node scales, reaffirming [3] that simple scaling behavior can become more nuanced beyond the scales studied. Informed by comparisons to local search, we provide potential explanations for our parameter-constrained complexity scaling laws (Section 5). Finally, we provide our newly created dataset of 128 million TSP solutions.

2 Experimental setup

2.1 Data design

All experiments were performed using the Symmetric Euclidean Traveling Salesman Problem. One problem instance consists of n $\mathbb{R}^d$ coordinates uniformly sampled between $[0, 1]$ for each of the d dimensions (illustrated in Figure 2). TSP's $O(n!)$ combinatorial solution space is defined as all closed loops that visit each node exactly once. Such loops are called *tours*, and the objective is to find the minimum-length tour. In combinatorial optimization language, tour length is referred to as *cost* (or *fitness* when maximizing), and the cost surface w.r.t. the solution space is referred to as the cost *landscape*.

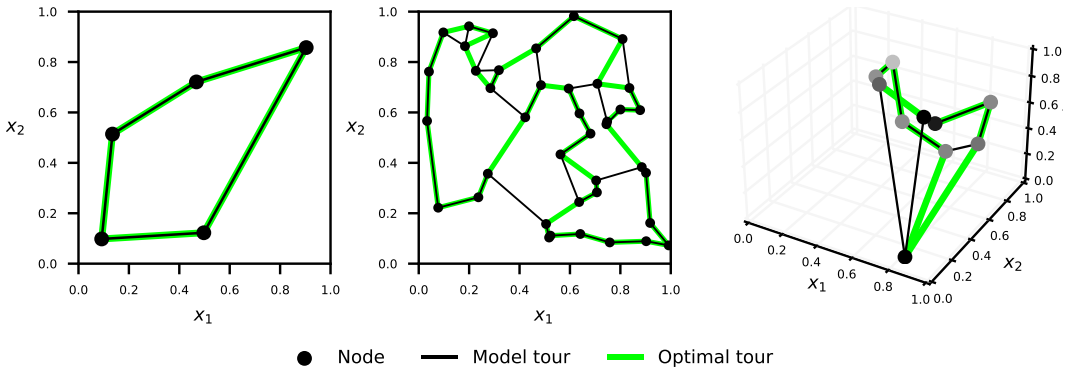

Figure 2: TSP has two convenient ways to adjust problem complexity: node count and spatial dimensionality. **Left:** 2D TSP instance with 5 nodes and a trivial 12 solutions. The solution tour sampled from a trained RL model is the optimal, minimum-length tour. **Center:** 2D TSP instance with 40 nodes and roughly 10^{46} solutions. The RL model tour is slightly suboptimal, 0.08 units longer than the optimal tour. **Right:** 3D TSP instance with 10 nodes, where brightness illustrates increased depth. Adding spatial dimensions does not modify the number of solutions but makes the problem representation more complex. This RL model solution is 0.05 units suboptimal.

2.2 Optimal tour generation

We leverage optimal solutions for model evaluation and supervised model training. For two-dimensional TSP, we use PyConcorde [15], a Python wrapper for the Concorde TSP solver [16, 17]. We generate optimal solutions to 128 million TSP problems, 12.8 million for each node scale we study, and share this dataset for academic use. We were unable to identify a pre-existing dataset of adequate size for supervised learning.

For our evaluations on higher-dimensional TSP, we closely approximate optimality with local search. We discuss our method and how we validated the resulting datasets in Appendix E. Although not exact, performance estimates derived from these datasets are statistically indistinguishable from their Concorde counterparts at two dimensions. We produce 128,000 of these solutions for each 10-node dimension scale we study, and 64,000 solutions for each 20-node dimension scale.

2.3 Model training

We train models separately using both reinforcement learning (RL) and supervised fine-tuning (SFT). Our models output tour sequences autoregressively, producing a policy distribution over each unvisited node via a Pointer Network head [18], adding an encoding of the sampled visitation to its Transformer decoder memory [19], and repeating until all nodes are visited. Details on our model architecture are provided in Appendix G, along with a brief comparison to alternative approaches. Hyperparameter choices are detailed separately in Appendix H.

Reinforcement learning: We formalize the RL environment as a bandit problem where return is the negative length of a solution tour. We use Proximal Policy Optimization (PPO) [20], training for one million gradient updates with hyperparameter optimization (HPO) informed learning rate decay.

Supervised fine-tuning: For SFT models, we minimize negative log-likelihood (NLL) loss between model edge selections and optimal edge selections with teacher forcing. We train for one epoch over the optimal solution dataset generated for the corresponding problem scale.[2] One epoch translates to roughly 73,000 gradient updates with a faster learning rate decay than that used for RL experiments.

2.4 Suboptimality estimation

We measure model performance in mean suboptimal tour length, $s = \mu_{model} - \mu_{opt}$, where μ_{model} is mean model tour length and μ_{opt} is mean optimal tour length. We refer to this regret-based metric as the *suboptimality gap*, or simply *suboptimality*. For neural scaling laws, raw tour length trends and

[2]Multiple epochs of training is a distinct scaling regime where data is also bottlenecked (Eq. 1.5 in [3]).

suboptimality trends both describe the same relationship since μ_{opt} remains constant. For problem complexity scaling μ_{opt} varies, so measuring suboptimality extracts the underlying model behavior.

Parameter and node scaling evaluations use the first 1.28 million problems of their corresponding Concorde-solved datasets. Compute scaling evaluations, performed throughout training every 4000 updates, use the first 12,800 problems. Spatial dimension scaling evaluations use their full approximate dataset for each scale. We sample one model tour for each optimal tour.

2.5 Scaling experiments

Neural scaling: For parameter and compute scaling laws, we study two-dimensional 20-node TSP. We train at 12 model sizes, scaling model width to achieve a geometric progression between roughly 60,000 and 6 million parameters. Previous work suggests that neural scaling is relatively insensitive to Transformer width/depth aspect ratio over several orders of magnitude [3]. We would expect similar results if, for instance, the number of layers were scaled instead, provided models are still trained sufficiently near convergence. However, for some tasks such as neural machine translation, proportionality of encoder and decoder blocks must also be considered [21].

Problem complexity scaling: For TSP node and spatial dimension scaling laws, we fix model width thereby fixing model capacity as we scale problem complexity. For node scaling, we train one RL and one SFT model per node scale. We study ten scales, $n \in \{5, 10, 15, ..., 50\}$. Symmetric TSP has $1/2\,(n-1)!$ possible tours[3] for $n \geq 3$, so these scales range from 12 solutions up to approximately 3×10^{62}. For spatial dimension scaling, we train one RL model per scale and forgo SFT training given the scarcity of optimal data as detailed in Section 2.2. We study $d \in \{2, 3, ..., 12, 15, 20, 30, 40, 50, 100\}$ over two node scales $n \in \{10, 20\}$, which allows us to better interpret otherwise ambiguous trends.

Fitting scaling laws: Figures 1 and 3 show experiments that converge to trend. These results are the inputs for scaling law regression fits. We discuss experiments that failed to converge in Appendix B. Critically, all trailing trend-breaking experiments are more suboptimal than predicted. If any were less suboptimal, we could not hypothesize that trend alignment is achievable with further training and further model improvement. Note that RL converges to trend at larger scales than SFT since the former is not constrained by the amount of available optimal data.

Compute requirements: Our 50-node RL experiment is the most computationally expensive. It trained for 24 V100-days and consumed roughly 3×10^{-3} PF-days of compute. Including preliminary experiments and HPO, training required several months. Further details are provided in Appendix F.

3 Neural scaling laws for combinatorial optimization

Figure 1 shows that model suboptimality smoothly power decays w.r.t. parameters or compute, for both RL and SFT with the chosen settings. At the scales tested, SFT is more compute efficient than RL (at the expense of node-level supervision), but both exhibit similar parameter efficiency. Note that SFT's parameter-scaling fit may slightly underestimate the true infinite-compute decay exponent. For the larger models shown in SFT's compute-efficient frontier, learning rate decay forces convergence before suboptimality visibly deviates from the frontier. Hence, further training may yield small improvements for these models, increasing SFT's parameter-scaling exponent and further differentiating it from RL's. This suggests SFT will surpass RL in parameter efficiency with larger models. However, with only one seed per model size, we cannot determine statistical significance.

Figure 1 also shows that optimal model size can be expressed as a function of compute budget by extracting each model's contribution to the compute-efficient frontier. This scaling law is very consistent between domains such as generative modeling [3, 4, 13] and reinforcement learning [8, 14]. Growth rate exponents are usually around 0.5 to 0.75, and our TSP results closely align with these values. However, the degree of overlap between adjacent model sizes produces non-trivial imprecision, especially for SFT, so extrapolation would be imprecise. We can infer that RL and SFT are compute efficient at similar model sizes for the tested range and TSP problem scale.

[3] From $n!$ permutations, starting node insensitivity yields n equivalent tours per permutation and tour reversal insensitivity yields 2 equivalent tours per permutation (dividing by n and 2, respectively).

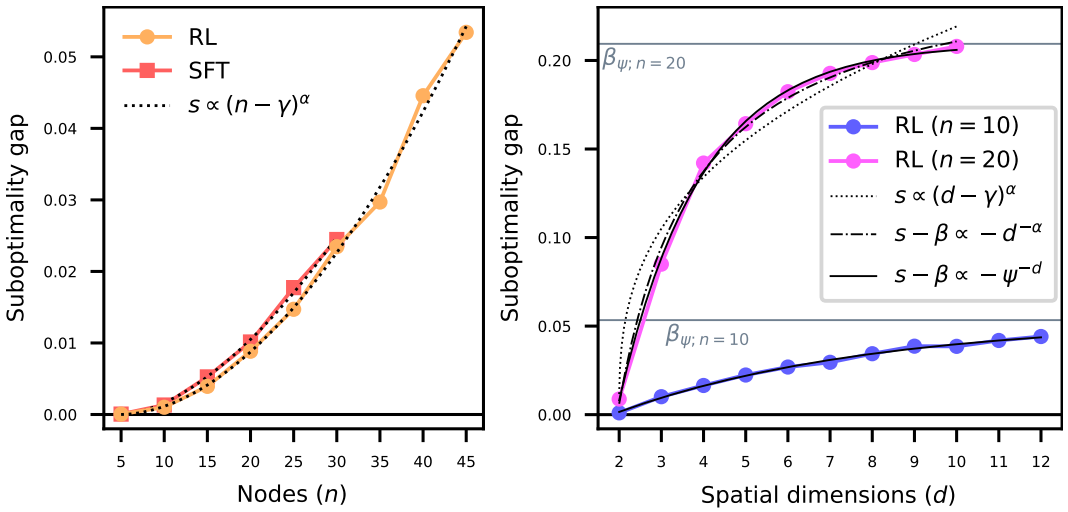

Figure 3: Suboptimality over problem scaling for models near convergence with a fixed number of non-embedding parameters. **Left:** Suboptimality follows superlinear power growth w.r.t. nodes, though we expect this trend eventually to break down before intersecting the near-linear random performance ceiling (Figure 4). **Right:** Suboptimality smoothly increases w.r.t. spatial dimensions, closely following negative exponential decay. Power growth (dashed), power decay (dash-dot), and exponential decay (solid) all predict the 10-node RL experiment (we show the latter). But power growth fails to find a convincing fit for its 20-node counterpart. Even random tour suboptimality is bounded as $d \to \infty$ (Theorem 6), so *any* better-than-random monotonic trend must converge. But the power decay asymptote obtained for 10 nodes is larger than that for 20 nodes, which is nonsensical. Exponential decay is most predictive while maintaining sound β_{ψ} asymptote ordering, as shown.

A broader implication of Figure 1 is that TSP tour length is a natural performance metric [14]. For TSP, RL obtains smooth scaling laws directly for the return signal with high precision. For more complex environments with sequential dynamics, return-based RL scaling laws have considerable variance and lose predictive power [9, 22, 23, 24].

Note that the RL compute-efficient frontier ignores the 168,000 parameter model, shown in lower opacity, which improves slightly faster than the trend otherwise predicts. We discuss this outlier in Appendix B. We include scaling results for loss in Appendix C and only summarize those findings here. First, loss variance can reveal a pattern of smooth power decay even when the corresponding mean has no trend. Second, PPO's mean critic loss predictably improves despite the absence of a mean actor loss trend and the interdependence between the two learning objectives.

4 Complexity scaling laws at the infinite-compute limit

We now study problem complexity scaling for deep models with unbottlenecked compute and data, observing patterns in the limit of performance under fixed model capacity. We find that predictable suboptimality trends emerge when scaling either the number of nodes or the number of spatial dimensions, despite the stark contrast in how these measures influence the problem. Further, our node scaling result fosters a useful critique of scaling law breakdown, and our dimension scaling result characterizes the distinct nature of embedding parameters in complexity scaling laws. We again discuss corresponding loss trends in Appendix C. Theorems introduced are proven in Appendix I.

4.1 Solution space scaling

The left plot in Figure 3 shows suboptimality for RL and SFT models near convergence as we scale the number of TSP nodes. Both experiments exhibit smooth power growth after including a third fitted constant γ as a node-scale offset.[4] We find that growth rates for RL ($\alpha \approx 1.86$) and SFT

[4] A node-scale offset is required because zero-node TSP is ill-defined.

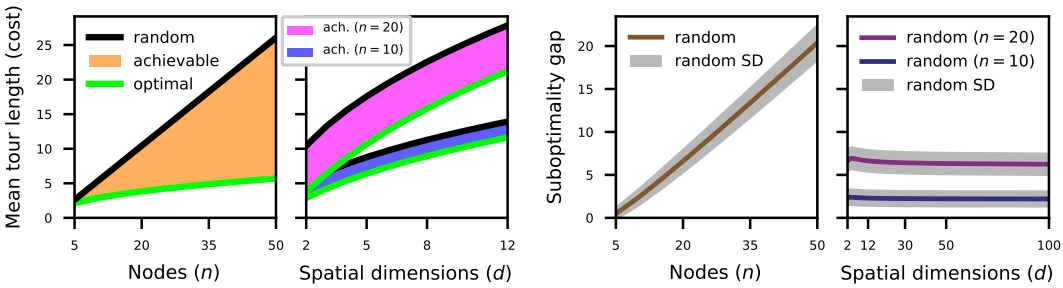

Figure 4: Node and spatial dimension scaling have distinct effects on the achievable performance span, the suboptimality gap of random performance. **Left:** Mean optimal tour length closely follows sublinear power growth w.r.t. either problem scale. Mean random tour length grows linearly w.r.t. nodes, and sublinearly w.r.t. dimensions at rate similar to optimal tour length growth. Each sublinear trend approaches square root growth in the limit ($\alpha = 0.5$; proof in Appendix I). **Right:** Suboptimality of random performance w.r.t. nodes is polynomial but approximately linear, being dominated by random tour length growth. Suboptimality of random performance w.r.t. dimensions produces a small, transient increase then decrease, but is provably constant in the limit (Theorem 6).

($\alpha \approx 1.69$) are close for the tested model size, although their precise values are not necessarily meaningful. Power growth fits are sensitive to the suboptimalities obtained for the largest node scales, which require the most training to converge. If further training can produce small improvements at these scales, a fit may overestimate the true infinite-compute α. Similar to parameter scaling fits in Figure 1, SFT's node-scaling fit is more likely to be pessimistic than RL's given the former's shortened training. However, extrapolating SFT's small α advantage is less trustworthy here since SFT does not surpass RL by 30 nodes.

Even if we assume that α estimates are imprecise, the growth of suboptimality is clearly superlinear. This observation presents a subtle contradiction. Figure 4 shows that the suboptimality of random tour length grows almost linearly with node scale. Indeed, this growth is provably $O(n)$ as $n \to \infty$.[5] Thus, the obtained scaling laws imply model performance will eventually be worse than random performance, and with enough scale, exceed even the maximum possible tour length (also $O(n)$). Instead, we expect this superlinearity to break down before exceeding random performance, and we expect the obtained scaling laws to become pessimistic estimates at large node scales.

Model suboptimality may be better expressed as a broken scaling law [26] given the strong fit quality over initial scaling and the subsequent need for an inflection point. However, this functional form would increase the number of fitted constants, and piecewise scaling is less useful unless we can also predict where trend breaks occur. Conveniently, extrapolating RL's more pessimistic fit, model suboptimality does not intersect random tour suboptimality until roughly 40,000 nodes, a scale with an unfathomably large solution space. Analogous contradictions with huge model sizes have been observed for neural scaling laws (e.g., Figure 15 in [3]), reinforcing the caveat that more intricate scaling behavior can appear simple over the scales studied.

4.2 Representation space scaling

The right plot in Figure 3 shows suboptimality for RL models near convergence as we scale the number of TSP spatial dimensions. Both 10-node and 20-node experiments produce smooth bounded growth described by negative exponential decay toward a fitted asymptote β. We found that almost any regression model results in a good fit of the 10-node experiment since its growth is closer to being linear, but the 20-node experiment is more discriminative. Unbounded power growth produces a markedly poor fit given the observed trend is visibly convergent. Power decay regression obtains smaller residuals but appears to converge slightly slower than the observed growth. Further, power decay fits produce a likely contradiction: the bound obtained for the 10-node experiment ($\beta_\alpha \approx 0.31$) is higher than that obtained for the more complex 20-node counterpart ($\beta_\alpha \approx 0.25$). This implies that a lower suboptimality is achieved in the limit as $d \to \infty$ by raising the baseline solution complexity,

[5] Lemma 1 shows that expected random tour length grows linearly w.r.t. n, while the Beardwood-Halton-Hammersley Theorem [25] proves that optimal tour length is asymptotically proportional to $\sqrt{n}$ as $n \to \infty$.

unless increased node scale fundamentally alters the form of scaling w.r.t. dimensions. In contrast, we find that exponential decay is most predictive[6] while maintaining sound asymptote ordering between 10- and 20-node experiments (the β_ψ values shown). Like α in node scaling fits, β_ψ is sensitive to the suboptimality values obtained at larger scales. Because higher-dimensional runs require more training to converge, β_ψ estimates are likely imprecise, so the exact values shown may not be meaningful.

To explain the asymptotic nature of this scaling law, we again refer to analysis. L^p-norms such as Euclidean distance exhibit quite unintuitive behavior when scaling to higher dimensions [27, 28, 29, 30]. In expectation, both random tour length and optimal tour length diverge but at similar rates (Figure 4), so random tour suboptimality is roughly constant over the tested domain and is provably constant in the limit as $d \to \infty$ (Theorem 6). Several other properties of the cost landscape similarly converge as $d \to \infty$ (Theorem 5 and Appendix D). For local search algorithms, these observations imply that TSP approaches a mostly (if not entirely) stationary problem complexity with one critical exception: the increasing computational complexity of evaluating scalar distances in higher dimensions. (However, the average number of distances that search evaluates converges.)

This increasing computational complexity is confined to the embedding layer for deep models. If we assume that arbitrarily large feature vectors can be embedded without producing a learning bottleneck, then model performance converging to a better-than-random suboptimality becomes anticipated. This is a complementary view of a pattern previously established in neural scaling: the separability of embedding parameters from model capacity [3, 4, 14]. When embeddings are unbottlenecked for a fixed problem, scaling non-embedding parameters studies the bottleneck of model capacity (parameter scaling laws). When scaling the problem, embeddings must also be scaled such that model capacity remains the only bottleneck (parameter-constrained complexity scaling laws).

5 Interpretable complexity scaling with local search

We now reverse engineer the complexity scaling laws introduced in Section 4 using local search, studying comparable trends produced with a white-box algorithm. These similarities do *not* directly imply that model inference and local search share underlying mechanisms. Instead, they provide potential topics to explore toward the development of a formal theory for parameter-constrained deep learning in future work. We evaluate local search using the same datasets discussed in Section 2.4. For each problem, we initialize search with a random tour and then generate a locally optimum tour with the simple, well-studied 2-opt search move [31], performing gradient descent through the cost landscape. The landscape properties discussed at a high level below are detailed in Appendix D.

Figure 5 shows the suboptimality of 2-opt solutions w.r.t. TSP node and spatial dimension scaling. For the latter, extending to 100 dimensions reveals a clearly convergent trend that aligns with the reasoning introduced in Section 4.2. If we only observe the lower-dimensional scales studied in Figure 3, convergence is less obvious, as it is for RL. We find that subexponential decay produces a better fit than pure exponential decay, though at the expense of another fitted constant ϕ. This form may also generalize better for deep models, although we observe no improvement when fitting subexponential decay to the RL experiments in this paper.

Before trends converge, scaling spatial dimensions increases the density of local optima in the cost landscape. In expectation, search arrives at a local optimum faster, but one with increased suboptimality. This trend mirrors the result of parameter-constrained models. 10-node experiments are especially similar, where 2-opt's β asymptote is nearly aligned with RL's at the tested model size. One potential explanation is that fixed model capacity limits a model's ability to avoid poor local optima, matching the bottleneck observed with local search. For example, if models learn a latent representation of a smoothed cost landscape [32], more model capacity may permit more advanced smoothing, making low cost optima easier to identify.

When scaling nodes, 2-opt's baseline behavior is less informative. We observe near-linear growth with an inflection point, an unclear relationship that deviates from the power growth observed for RL and SFT. One key distinction between dimension scaling and node scaling is the depth of search required to reach a local optimum. When scaling dimensions before trend convergence, the required search depth slightly decreases due to increased local optima density. In contrast, as $n \to \infty$, the

[6] We also tested decays with the quasi-polynomial form $d^{-\log_\psi d}$ and the subexponential form ψ^{-d^ϕ}, $\phi \in (0, 1]$ but found extreme log base fits for the former and no visual improvement from either form.

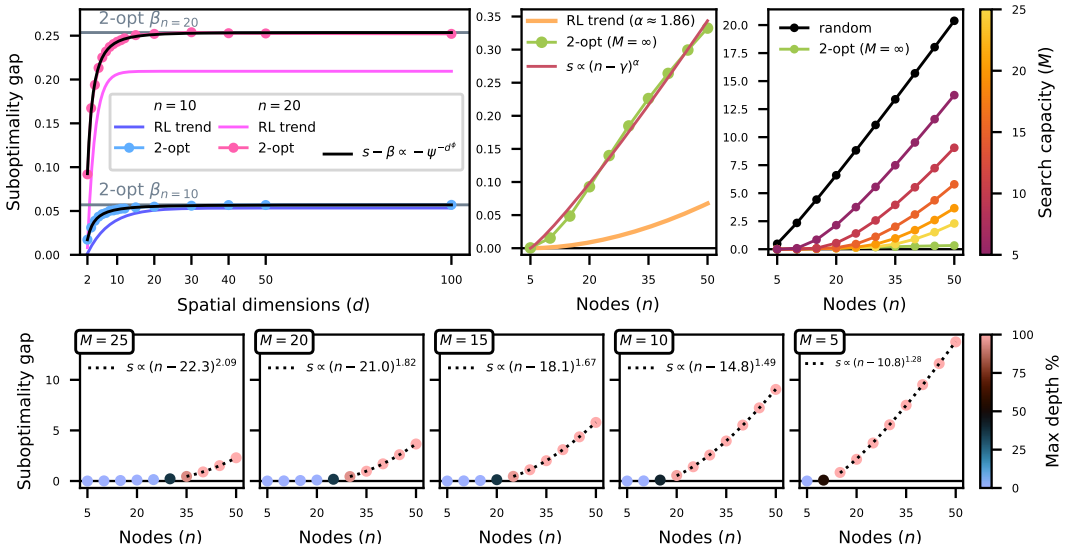

Figure 5: 2-opt local search suboptimality over problem complexity scaling. A simpler gradient descent of the cost landscape can produce trends similar to those of parameter-constrained deep models. **Top left:** 2-opt suboptimality w.r.t. spatial dimensions closely aligns with RL trends. Pure exponential fit attempts decay slightly too fast, but we obtain close fits with the subexponential generalization shown, where $\phi \in (0, 1]$. **Top center:** 2-opt suboptimality w.r.t. number of nodes. With unconstrained search depth, 2-opt produces an unclear trend with an inflection point. **Top right:** Contraining search depth (M) produces smooth superlinear growth. **Bottom:** Power growth emerges after saturating at 100% early stopping, aligning with the scaling form of parameter-constrained deep models (but these trends are *not* roughly equivalent, because proportionalities are quite different).

required search depth diverges to infinity. Attempting to align node scaling behavior, we constrain the maximum search depth (M). Doing so causes suboptimality to follow smooth superlinear power growth after search saturates at 100% early stopping (Figure 5 top-right and bottom). Although M-constrained suboptimality is much higher than model suboptimality, and we expect breakdown of superlinearity (before exceeding random tour suboptimality) to occur more rapidly.

Despite these distinctions, the power growth form of M-constrained suboptimality may still help explain this form for parameter-constrained models. As the cost landscape grows, fixed-depth local search stops at solutions that are further from the attractor local optimum. Is inference similarly forced to approximate local optima under constrained model capacity? If so, what measure of solution proximity is learned? We were able to reproduce superlinear power growth with M-constrained 2-exchange search, which uses a different definition of solution adjacency (shown in Appendix D). Hence, this finding is not overly sensitive to the definition of solution proximity.

6 Related work

Neural scaling laws for loss: Neural scaling laws where cross-entropy loss smoothly improves with model size, dataset size, or compute budget were popularized by Kaplan et al.'s study of large language models [3], which also first demonstrated the separable nature of embedding parameters. Henighan et al. [4] generalize neural scaling laws to other autoregressive generative modeling domains, showing that optimal model size w.r.t. compute budget is remarkably consistent. Sorscher et al. [33] demonstrate that an exponential decay of test error w.r.t. data can be achieved using data pruning (instead of the typical power law scaling [3, 5, 6]). Analogously, Frantar et al. [34] show that network sparsity influences scaling behavior. Supervised learning constitutes the majority of recent research on neural scaling laws [3, 4, 5, 6, 21, 33, 34, 35, 36, 37].

Neural scaling laws for RL: Reward-based metrics for reinforcement learning models often do not improve smoothly when scaling parameters, compute, or environment interactions [14]. Notable exceptions include ground-truth reward in RL from human feedback (RLHF) [7] and Elo ratings in

two-player competitive zero-sum games such as Hex [10], Connect Four and Pentago [8, 38], and to a lesser extent in two-team competitive games like football [39]. Hilton et al. [14] address this challenge by introducing intrinsic performance, which maps the compute-efficient frontier to a linear relationship. Caballero et al. [26] use broken neural scaling laws to predict more sophisticated return trends, like those observed in Procgen tasks [40]. We circumvent these challenges entirely because TSP natively exhibits smooth, pure power scaling of suboptimality w.r.t. parameters or compute.

Neural scaling law theory: Several theories have been proposed to explain the relationships between loss and parameters [41], between loss and data [42], and for both in conjunction [43, 44]. Most recently, Bordelon et al.'s dynamical mean field theory [45] recovers phenomena from parameter, data, and compute scaling. The Quantization Hypothesis [38, 44] also explains emergent abilities [46] by positing that there is an underlying discreteness to all abilities that deep models learn. However, Schaeffer et al. [47] show that emergent abilities often reflect the chosen performance metric rather than deep learning itself.

Problem complexity scaling laws: Jones [10] shows that the Elo score of a fixed-size AlphaZero [48] model playing Hex can be predicted as a function of both compute budget and board size. The latter primarily grows the solution space similar to TSP node scaling, arguably making that work the closest to our own. However, our key contribution is studying the bottleneck of fixed model capacity, which Jones' method cannot address because it achieves perfect play at each board size. More generally, when considering non-deep-learning algorithms applied to combinatorial optimization, the study of performance w.r.t. problem complexity goes back decades [49]. Merz et al. [50] predict the performance of memetic algorithms on the Quadratic Assignment Problem (QAP) using cost landscape properties. (TSP is a special case of QAP.) Ochoa et al. [51] predict local search performance on NK problems by introducing Local Optima Networks (LON) as a general model of cost landscapes. Tayarani et al. [52] broadly investigate local search trends for TSP, including several node scaling results for cost, some of which informed our methodology.

7 Discussion

Limitations: Infinite-compute performance can only be estimated. With modest resources, we could not experiment on larger models and problem scales. This is especially important to consider when extrapolating node scaling trends, since they must eventually break down (possibly well before the contradiction detailed in Section 4.1). We are also limited in precision when comparing SFT and RL scaling law fits given that fewer SFT models converge to trend with one epoch of training, and we train only a single seed per scale.[7] Nevertheless, the main limitation of this work is its focus on a single problem. Euclidean TSP is easier than most NP-hard problems, in part due to its cost metric satisfying the triangle inequality [49]. Euclidean TSP also admits polynomial-time approximation schemes (PTAS), allowing near-optimal tours to be computed in polynomial time with respect to the number of nodes [53]. These properties should be considered when hypothesizing how our findings generalize to other combinatorial optimization problems and beyond. Future work would need to demonstrate compelling generalization before complexity scaling laws can be substantiated as a general principle.

Beyond TSP scaling: Despite TSP's relative simplicity, there is some basis to be optimistic for parameter-constrained complexity scaling laws in other tasks. The TSP cost landscape was once widely accepted to be approximately globally convex [54, 55], but more recent work has found this convexity to be increasingly coarse [56, 57, 58]. Even so, in expectation, smooth complexity scaling laws emerge. This macroscopic order despite microscopic disorder mirrors the nature of neural scaling laws and supports the hypothesis that complexity scaling laws can be obtained for less structured tasks. A natural next step would be studying parameter-constrained complexity scaling with more elaborate Euclidean combinatorial optimization problems where the solution space and the representation space remain independently scalable. Potential testbeds include other routing problems like the Vehicle Routing Problem (VRP) [59], the Orienteering Problem (OP) [60], and the Ring Star Problem (RSP) [61]. Given the similarity between the RL node scaling curve and its SFT counterpart (Figure 3 left), we especially encourage future study of algorithm insensitivity for parameter-constrained complexity scaling in other problems.

[7] We provide leave-one-out fit statistics in Appendix A.

Real-world domains: Identifying scalable yet precise measures of problem complexity is very challenging in domains like language modeling, vision, robotics, and so on. Solution complexity and representation complexity are often tightly coupled. For instance, in language modeling, vocabulary size affects both the number of token representations and the space of possible texts to generate. Discriminative NLP tasks like multiple choice question answering maintain separability of vocabulary size and number of classes, but at the expense of diminished relevance to generative modeling. Another challenge arises for real-world tasks where relevant performance metrics do not smoothly improve, as is often the case for sequential decision processes and tasks with discontinuous performance metrics. However, the latter has recently been addressed for language modeling with Token Edit Distance [47], an edit-space analogue to suboptimality.

Complexity scaling law theory: Without an underlying theory that explains parameter-constrained complexity scaling laws, we can only obtain a limited understanding of the conditions in which they apply and the types of problems they generalize to. Relating findings between domains will require significant speculation and extrapolations may be unreliable. Explaining each form of scaling is the most obvious gap. We are especially interested in whether model size predicts suboptimality growth rates like α and ψ, or β limits for spatial dimension scaling, and explaining the underlying nature if so. Extending Section 5 to reverse engineer scaling laws with a white-box machine learning model is one possible approach. Identifying and explaining counter-examples is another potentially useful approach. For example, if network sparsity were to influence parameter-constrained complexity scaling laws (analogous to how sparsity influences neural scaling laws [34]), a theory explaining this distinction may also address related open questions. Separately, another gap in our current understanding is what determines an embedding bottleneck. Widening the coordinate projection layer forever is probably insufficient to avoid an embedding bottleneck while dimension scaling. Extending the positional encoding sequence forever may be insufficient to avoid one while node scaling.

Practical implications: Deep methods that excel at solving TSP usually excel at solving similar routing problems like VRP and OP [62, 63, 64], both of which have well-studied practical variants [59, 60]. Practical use cases for parameter-constrained complexity scaling laws are more challenging to identify given that training until convergence is usually compute-inefficient. One use case would be predicting performance for applications mainly constrained by model size, such as edge applications requiring fast inference with limited hardware acceleration. Another potential use case is algorithm benchmarking, where we can estimate the limit of performance for several algorithms across a range of problem complexities without training models on larger problem scales. Such benchmarks can also provide useful insights like sample efficiency tradeoffs with respect to problem complexity. Our single-seed node scaling results for RL and SFT demonstrate that SFT reaches intermediate suboptimality values faster at the scales evaluated (Figure 8 in Appendix B). Future work could use multiple-seed training to quantify this advantage with high statistical confidence.

Finite-compute complexity scaling: While this paper focuses on bottleneck from model capacity, we could have instead chosen to limit compute budget or dataset size (with early stopping) and observe the patterns that emerge with increased problem complexity, if any. Compute-constrained and data-constrained complexity scaling laws would have many practical use cases. For example, Jones [10] shows that we can predict model performance as a bivariate function of train-time compute and test-time compute for a single Hex board size. If this neural scaling law can be generalized over problem scale, one could estimate for complex tasks the tradeoff between train-time and test-time compute using only cheap experiments.

Acknowledgments and Disclosure of Funding

We thank Charles Lovering for his feedback on delivery and clarity. We thank Joseph Weissman for reviewing portions of our proofs. This work was performed without third-party support.

References

[1] Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

[2] Jonathan S Rosenfeld, Amir Rosenfeld, Yonatan Belinkov, and Nir Shavit. A constructive prediction of the generalization error across scales. In *International Conference on Learning Representations*, 2020.

[3] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[4] Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B Brown, Prafulla Dhariwal, Scott Gray, Chris Hallacy, Benjamin Mann, Alec Radford, Aditya Ramesh, Nick Ryder, Daniel M Ziegler, John Schulman, Dario Amodei, and Sam McCandlish. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020.

[5] Ibrahim M Alabdulmohsin, Behnam Neyshabur, and Xiaohua Zhai. Revisiting neural scaling laws in language and vision. *Advances in Neural Information Processing Systems*, 35:22300–22312, 2022.

[6] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12104–12113, 2022.

[7] Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pages 10835–10866. Proceedings of Machine Learning Research, 2023.

[8] Oren Neumann and Claudius Gros. Scaling laws for a multi-agent reinforcement learning model. *arXiv preprint arXiv:2210.00849*, 2022.

[9] Kuang-Huei Lee, Ofir Nachum, Mengjiao Sherry Yang, Lisa Lee, Daniel Freeman, Sergio Guadarrama, Ian Fischer, Winnie Xu, Eric Jang, Henryk Michalewski, and Igor Mordatch. Multi-game decision transformers. *Advances in Neural Information Processing Systems*, 35: 27921–27936, 2022.

[10] Andy L Jones. Scaling scaling laws with board games. *arXiv preprint arXiv:2104.03113*, 2021.

[11] H Jaap van den Herik, Jos WHM Uiterwijk, and Jack van Rijswijck. Games solved: Now and in the future. *Artificial Intelligence*, 134(1-2):277–311, 2002.

[12] Jack van Rijswijck. *Computer Hex: Are bees better than fruitflies?* MSc. thesis, U. Alberta, Edmonton, 2002.

[13] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. *Advances in Neural Information Processing Systems*, 2022.

[14] Jacob Hilton, Jie Tang, and John Schulman. Scaling laws for single-agent reinforcement learning. *arXiv preprint arXiv:2301.13442*, 2023.

[15] Joris Vankerschaver. PyConcorde. URL `https://github.com/jvkersch/pyconcorde`.

[16] David Applegate, Robert Bixby, William Cook, and Vasek Chvátal. On the solution of Traveling Salesman Problems. *Documenta Mathematica*, pages 645–656, 1998.

[17] William Cook. Concorde TSP solver. URL `https://www.math.uwaterloo.ca/tsp/concorde/index.html`.

[18] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. *Advances in Neural Information Processing Systems*, 28, 2015.

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

[20] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[21] Behrooz Ghorbani, Orhan Firat, Markus Freitag, Ankur Bapna, Maxim Krikun, Xavier Garcia, Ciprian Chelba, and Colin Cherry. Scaling laws for neural machine translation. *International Conference on Learning Representations*, 2022.

[22] Jakob Bauer, Kate Baumli, Feryal Behbahani, Avishkar Bhoopchand, Nathalie Bradley-Schmieg, Michael Chang, Natalie Clay, Adrian Collister, Vibhavari Dasagi, Lucy Gonzalez, Karol Gregor, Edward Hughes, Sheleem Kashem, Maria Loks-Thompson, Hannah Openshaw, Jack Parker-Holder, Shreya Pathak, Nicolas Perez-Nieves, Nemanja Rakicevic, Tim Rocktäschel, Yannick Schroecker, Satinder Singh, Jakub Sygnowski, Karl Tuyls, Sarah York, Alexander Zacherl, and Lei M Zhang. Human-timescale adaptation in an open-ended task space. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 1887–1935, 23–29 Jul 2023.

[23] Sebastian Sartor and Neil Thompson. Neural scaling laws for embodied AI. *arXiv preprint arXiv:2405.14005*, 2024.

[24] Jens Tuyls, Dhruv Madeka, Kari Torkkola, Dean Foster, Karthik Narasimhan, and Sham Kakade. Scaling laws for imitation learning in single-agent games. *arXiv preprint arXiv:2307.09423*, 2023.

[25] Jillian Beardwood, John H Halton, and John Michael Hammersley. The shortest path through many points. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 55, pages 299–327. Cambridge University Press, 1959.

[26] Ethan Caballero, Kshitij Gupta, Irina Rish, and David Krueger. Broken neural scaling laws. In *The Eleventh International Conference on Learning Representations*, 2023.

[27] Charu C Aggarwal, Alexander Hinneburg, and Daniel A Keim. On the surprising behavior of distance metrics in high dimensional space. In *International Conference on Database Theory*, pages 420–434. Springer, 2001.

[28] Damien François, Vincent Wertz, and Michel Verleysen. The concentration of fractional distances. *IEEE Transactions on Knowledge and Data Engineering*, 19(7):873–886, 2007.

[29] Gérard Biau and David M Mason. High-dimensional p-norms. *Mathematical Statistics and Limit Theorems: Festschrift in Honour of Paul Deheuvels*, pages 21–40, 2015.

[30] Evgeny M Mirkes, Jeza Allohibi, and Alexander Gorban. Fractional norms and quasinorms do not help to overcome the curse of dimensionality. *Entropy*, 22(10):1105, 2020.

[31] Georges A Croes. A method for solving Traveling-Salesman Problems. *Operations Research*, 6 (6):791–812, 1958.

[32] Jun Gu and Xiaofei Huang. Efficient local search with search space smoothing: A case study of the Traveling Salesman Problem (TSP). *IEEE Transactions on Systems, Man, and Cybernetics*, 24(5):728–735, 1994.

[33] Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari Morcos. Beyond neural scaling laws: beating power law scaling via data pruning. *Advances in Neural Information Processing Systems*, 35:19523–19536, 2022.

[34] Elias Frantar, Carlos Riquelme Ruiz, Neil Houlsby, Dan Alistarh, and Utku Evci. Scaling laws for sparsely-connected foundation models. In *International Conference on Learning Representations*, 2024.

[35] Danny Hernandez, Jared Kaplan, Tom Henighan, and Sam McCandlish. Scaling laws for transfer. *arXiv preprint arXiv:2102.01293*, 2021.

[36] Danny Hernandez, Tom Brown, Tom Conerly, Nova DasSarma, Dawn Drain, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Tom Henighan, Tristan Hume, Scott Johnston, Ben Mann, Chris Olah, Catherine Olsson, Dario Amodei, Nicholas Joseph, Jared Kaplan, and Sam McCandlish. Scaling laws and interpretability of learning from repeated data. *arXiv preprint arXiv:2205.10487*, 2022.

[37] Aidan Clark, Diego de Las Casas, Aurelia Guy, Arthur Mensch, Michela Paganini, Jordan Hoffmann, Bogdan Damoc, Blake Hechtman, Trevor Cai, Sebastian Borgeaud, George van den Driessche, Eliza Rutherford, Tom Hennigan, Matthew Johnson, Katie Millican, Albin Cassirer, Chris Jones, Elena Buchatskaya, David Budden, Laurent Sifre, Simon Osindero, Oriol Vinyals, Jack Rae, Erich Elsen, Koray Kavukcuoglu, and Karen Simonyan. Unified scaling laws for routed language models. In *International Conference on Machine Learning*, pages 4057–4086. Proceedings of Machine Learning Research, 2022.

[38] Oren Neumann and Claudius Gros. AlphaZero neural scaling and Zipf's law: a tale of board games and power laws. *arXiv preprint arXiv:2412.11979*, 2024.

[39] Siqi Liu, Guy Lever, Zhe Wang, Josh Merel, SM Ali Eslami, Daniel Hennes, Wojciech M Czarnecki, Yuval Tassa, Shayegan Omidshafiei, Abbas Abdolmaleki, Noah Y Siegel, Leonard Hasenclever, Luke Marris, Saran Tunyasuvunakool, H Francis Song, Markus Wulfmeier, Paul Muller, Tuomas Haarnoja, Brendan D Tracey, Karl Tuyls, Thore Graepel, and Nicolas Heess. From motor control to team play in simulated humanoid football. *Science Robotics*, 7(69): eabo0235, 2022.

[40] Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *International Conference on Machine Learning*, pages 2048–2056. Proceedings of Machine Learning Research, 2020.

[41] Utkarsh Sharma and Jared Kaplan. Scaling laws from the data manifold dimension. *Journal of Machine Learning Research*, 23(9):1–34, 2022.

[42] Marcus Hutter. Learning curve theory. *arXiv preprint arXiv:2102.04074*, 2021.

[43] Yasaman Bahri, Ethan Dyer, Jared Kaplan, Jaehoon Lee, and Utkarsh Sharma. Explaining neural scaling laws. *Proceedings of the National Academy of Sciences*, 121(27):e2311878121, 2024.

[44] Eric Michaud, Ziming Liu, Uzay Girit, and Max Tegmark. The quantization model of neural scaling. *Advances in Neural Information Processing Systems*, 36:28699–28722, 2023.

[45] Blake Bordelon, Alexander Atanasov, and Cengiz Pehlevan. A dynamical model of neural scaling laws. In *Proceedings of the 41st International Conference on Machine Learning*, pages 4345–4382, 2024.

[46] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856.

[47] Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. Are emergent abilities of large language models a mirage? *Advances in Neural Information Processing Systems*, 36:55565–55581, 2023.

[48] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[49] Kate Smith-Miles and Leo Lopes. Measuring instance difficulty for combinatorial optimization problems. *Computers & Operations Research*, 39(5):875–889, 2012.

[50] Peter Merz and Bernd Freisleben. Fitness landscape analysis and memetic algorithms for the quadratic assignment problem. *IEEE Transactions on Evolutionary Computation*, 4(4):337–352, 2000.

[51] Gabriela Ochoa, Sébastien Verel, Fabio Daolio, and Marco Tomassini. Local optima networks: A new model of combinatorial fitness landscapes. *Recent Advances in the Theory and Application of Fitness Landscapes*, pages 233–262, 2014.

[52] Mohammad-H Tayarani-N and Adam Prügel-Bennett. An analysis of the fitness landscape of Travelling Salesman Problem. *Evolutionary Computation*, 24(2):347–384, 2016.

[53] Sanjeev Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *Journal of the ACM (JACM)*, 45(5):753–782, 1998.

[54] Kenneth D Boese, Andrew B Kahng, and Sudhakar Muddu. A new adaptive multi-start technique for combinatorial global optimizations. *Operations Research Letters*, 16(2):101–113, 1994.

[55] Eric Angel and Vassilis Zissimopoulos. On the landscape ruggedness of the quadratic assignment problem. *Theoretical Computer Science*, 263(1-2):159–172, 2001.

[56] Doug R Hains, L Darrell Whitley, and Adele E Howe. Revisiting the big valley search space structure in the TSP. *Journal of the Operational Research Society*, 62(2):305–312, 2011.

[57] Gabriela Ochoa, Nadarajen Veerapen, Darrell Whitley, and Edmund K Burke. The multi-funnel structure of TSP fitness landscapes: a visual exploration. In *Artificial Evolution: 12th International Conference, Evolution Artificielle, EA*, pages 1–13. Springer, 2016.

[58] Gabriela Ochoa and Nadarajen Veerapen. Deconstructing the big valley search space hypothesis. In *Evolutionary Computation in Combinatorial Optimization: 16th European Conference, EvoCOP*, pages 58–73. Springer, 2016.

[59] Kris Braekers, Katrien Ramaekers, and Inneke Van Nieuwenhuyse. The Vehicle Routing Problem: State of the art classification and review. *Computers & Industrial Engineering*, 99: 300–313, 2016.

[60] Aldy Gunawan, Hoong Chuin Lau, and Pieter Vansteenwegen. Orienteering Problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research*, 255(2):315–332, 2016.

[61] Martine Labbé, Gilbert Laporte, Inmaculada Rodríguez Martín, and Juan José Salazar González. The Ring Star Problem: Polyhedral analysis and exact algorithm. *Networks: An International Journal*, 43(3):177–189, 2004.

[62] Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019.

[63] Wouter Kool, Herke van Hoof, Joaquim Gromicho, and Max Welling. Deep Policy Dynamic Programming for Vehicle Routing Problems. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 190–213. Springer, 2022.

[64] Liang Xin, Wen Song, Zhiguang Cao, and Jie Zhang. NeuroLKH: Combining deep learning model with Lin-Kernighan-Helsgaun heuristic for solving the Traveling Salesman Problem. *Advances in Neural Information Processing Systems*, 34:7472–7483, 2021.

[65] Eric Angel and Vassilis Zissimopoulos. On the classification of NP-complete problems in terms of their correlation coefficient. *Discrete Applied Mathematics*, 99(1-3):261–277, 2000.

[66] Leticia Hernando, Jose A Pascual, Alexander Mendiburu, and Jose A Lozano. A study on the complexity of TSP instances under the 2-exchange neighbor system. In *IEEE Symposium on Foundations of Computational Intelligence (FOCI)*, pages 15–21, 2011.

[67] Shen Lin and Brian W Kernighan. An effective heuristic algorithm for the Traveling-Salesman Problem. *Operations Research*, 21(2):498–516, 1973.

[68] Alfonsas Misevicius and Andrius Blazinskas. Combining 2-opt, 3-opt and 4-opt with k-swap-kick perturbations for the Traveling Salesman Problem. In *17th International Conference on Information and Software Technologies*, 2011.

[69] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *Neural Information Processing Systems Workshop*, 2017.

[70] Xavier Bresson and Thomas Laurent. The transformer network for the Traveling Salesman Problem. *arXiv preprint arXiv:2103.03012*, 2021.

[71] Luke Metz, Julian Ibarz, Navdeep Jaitly, and James Davidson. Discrete sequential prediction of continuous actions for deep RL. *arXiv preprint arXiv:1705.05035*, 2017.

[72] Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1437–1446, 10–15 Jul 2018.

[73] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 2623–2631, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362016. doi: 10.1145/3292500.3330701.

[74] Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhkopf, René Sass, and Frank Hutter. SMAC3: A versatile Bayesian optimization package for hyperparameter optimization. *Journal of Machine Learning Research*, 23(54):1–9, 2022.

[75] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In J Shawe-Taylor, R Zemel, P Bartlett, F Pereira, and KQ Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

[76] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018.

[77] Shuhei Watanabe, Archit Bansal, and Frank Hutter. PED-ANOVA: Efficiently quantifying hyperparameter importance in arbitrary subspaces. In Edith Elkind, editor, *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pages 4389–4396. International Joint Conferences on Artificial Intelligence Organization, 2023. doi: 10.24963/ijcai.2023/488.

[78] Frank Hutter, Holger Hoos, and Kevin Leyton-Brown. An efficient approach for assessing hyperparameter importance. In Eric P Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 754–762, Bejing, China, 22–24 Jun 2014.

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: The introduction mentions our key hypothesis and summarizes our contributions in the last paragraph. We scope the problem studied (TSP) and the scaling regime we focus on (model capacity bottleneck from a limited number of parameters).

   Guidelines:

   - The answer NA 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 NA 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: First paragraph of the discussion, Section 7.

   Guidelines:

   - The answer NA 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: Proof of relations and limits referenced in Section 4 are provided in the supplementary materials.

Guidelines:

- The answer NA 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: Our method is outlined in Section 2. Details on our model architecture and hyperparameters for experiments (along with their motivation) are then provided in the supplementary materials.

Guidelines:

- The answer NA 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: Code for all experiments and figure generation will be provided with the paper when released, but are not provided with this submission. Most experiments take on the order of weeks (if not months) to reproduce assuming access to a mid-range GPU cluster. The optimal solution dataset we used and plan to contribute is much larger than the 100MB limit for supplementary materials.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.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://nips.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, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Key training and test settings are discussed in Section 2. Hyperparameters and our HPO method are provided in the supplementary materials.

   Guidelines:

   - The answer NA 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: [No]

   Justification: TSP data distribution means are estimated with large enough datasets to produce very tight confidence intervals. Intervals are similarly tight for seed-specific algorithm performance. However, due to resource constraints, we train only one seed per scale for deep learning experiments. We cannot determine statistical significance for specific fit constants, which we mention where relevant in the paper. Otherwise, we do include statistical significance information where feasible and appropriate, such as with approximations of higher-dimensional TSP optimality in the supplementary materials.

Guidelines:

- The answer NA 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: Provided in Section 2.5 and the supplementary materials.

   Guidelines:

   - The answer NA 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: We adhere to every requirement.

   Guidelines:

   - The answer NA 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: [No]

Justification: This paper is foundational research. We discuss potential applicability to domains beyond TSP and combinatorial optimization in Section 7. However, until this is better characterized in future work, any discussion of broader societal impact would be purely speculative.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA 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., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: We do not release any pretrained models and our dataset only contains solved TSP problems, which pose no such risks.

Guidelines:

- The answer NA 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: For optimal tour generation we use PyConcorde (BSD-3-Clause) [15] which wraps Concorde. Concorde is available for all academic research use [17]. Our repository follows PyConcorde's BSD-3-Clause requirements, and we cite all creators in Section 2.

Guidelines:

- The answer NA 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: [No]

    Justification: As discussed in Question 5, we plan to release our project code and TSP solution dataset, but do not include them in this submission. However, dataset details are documented in Section 2.

    Guidelines:

    - The answer NA 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: [NA]

    Justification: Our research did not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA 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: [NA]

Justification: Our research did not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer NA 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 rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: We occasionally used LLMs for word or phrase choice.

Guidelines:

- The answer NA 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 (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

Contents

Appendix

A Scaling law fit constants

We provide the fit constants for scaling laws discussed in the main paper. Remaining fits can be found in the project repository at `https://github.com/lowellw6/complexity-scaling-laws`. For example, our repository provides fits of the scaling laws for loss discussed in Appendix C.

RL and SFT scaling laws may have non-trivial imprecision due to training with a single seed per scale. This should be considered when extrapolating, or when comparing values between RL and SFT such as α in node scaling and optimal model size fits. To mitigate this concern, we performed jackknife (leave-one-out) resampling of RL and SFT fits and provide growth rate statistics in Table 4. We provide individual leave-one-out (LOO) fits in the project repository. While LOO distributions are distinct from those obtained using multiple seeds, they quantify sensitivity to individual training runs.

Optimal model size fits are relatively sensitive due to the frontier overlap between adjacent model sizes. We observe higher variance and the largest SFT LOO α exceeds the smallest RL LOO α (despite the full-data RL α being larger than the full-data SFT α). Node-scaling fits are relatively sensitive to the suboptimalities obtained at the largest node scales. For RL, we obtain α values as large as 1.95 when fixing the γ offset at its full-data value and 2.22 when fitting γ. Demonstrated by the large positive skewness, these larger α fits occur at larger LOO node scales. However, fitting γ can shift the offset more than 3 nodes and we know near optimality is achieved at 5 nodes, so fixed-γ LOO variance is likely closer to multi-seed variance. Remaining LOO fits are quite stable, and we found that β asymptote fits for dimension scaling experiments are nearly constant (roughly ± 0.001).

Table 1: Neural scaling law fits.

Scaling law	Form	Algorithm	α	k
Parameters (N)	$s = \left(\frac{k}{N}\right)^{\alpha}$	RL	0.582	4.42×10^{2}
		SFT	0.712	1.38×10^{3}
Compute ($C_{\min}$)	$s = \left(\frac{k}{C_{\min}}\right)^{\alpha}$	RL	0.439	6.62×10^{-9}
		SFT	0.555	6.78×10^{-9}
Optimal model size	$N = \left(\frac{C}{k}\right)^{\alpha}$	RL	0.816	1.75×10^{-12}
		SFT	0.658	1.98×10^{-14}

Table 2: TSP node (n) scaling law fits.

Form	Algorithm	M	α	γ	k
$s = \left(\frac{n-\gamma}{k}\right)^{\alpha}$	RL	–	1.86	4.99	1.91×10^{2}
	SFT	–	1.69	4.99	2.24×10^{2}
	2-opt	25	2.09	22.3	18.6
	2-opt	20	1.82	21.0	14.2
	2-opt	15	1.67	18.1	11.1
	2-opt	10	1.49	14.8	8.05
	2-opt	5	1.27	10.8	4.94

Table 3: TSP spatial dimension (d) scaling law fits. RL fits use pure exponential decay ($\phi = 1$).

Form	Algorithm	n	β	ψ	ϕ	k
$s = \beta - \left(\frac{k}{\psi^{d^{\phi}}}\right)$	RL	10	5.33×10^{-2}	1.18	1.00	7.24×10^{-2}
	RL	20	0.209	1.66	1.00	0.557
	2-opt	10	5.70×10^{-2}	66.8	0.215	5.22
	2-opt	20	0.254	98.6	0.251	38.0

Table 4: Leave-one-out (LOO) statistics for RL and SFT fits of exponent α and base ψ. LOO indices (idx) are in ascending scale order.

Scaling law	Algorithm	Fit	μ	σ	Skew	Min (idx)	Max (idx)
Parameters (N)	RL	α	0.579	0.013	-1.9	0.545 (0)	0.593 (4)
	SFT	α	0.718	0.015	0.87	0.701 (2)	0.745 (0)
Compute ($C_{\min}$)	RL	α	0.440	0.011	0.83	0.423 (5)	0.463 (0)
	SFT	α	0.554	0.011	-0.28	0.536 (2)	0.570 (0)
Optimal model size	RL	α	0.811	0.064	-0.47	0.690 (7)	0.903 (5)
	SFT	α	0.655	0.071	-0.22	0.556 (4)	0.734 (2)
Nodes (n) - fix γ	RL	α	1.87	0.032	1.4	1.83 (6)	1.95 (8)
	SFT	α	1.71	0.059	1.4	1.65 (3)	1.83 (5)
Nodes (n) - fit γ	RL	α	1.90	0.116	2.2	1.81 (0)	2.22 (8)
	SFT	α	1.69	0.134	0.46	1.48 (0)	1.94 (5)
Dimensions (d) $10\,n$	RL	ψ	1.18	0.007	-1.0	1.16 (0)	1.19 (5)
Dimensions (d) $20\,n$	RL	ψ	1.68	0.023	-0.46	1.62 (2)	1.71 (0)

Table 5: Performance bounds obtained for TSP node and spatial dimension scaling. Random tour length w.r.t. nodes is derived (Lemma 1). Otherwise, these fits should *not* be extrapolated.

Scaling variable	Form	Algorithm	n	α	γ	k
Nodes (n)	$\mu_{\text{cost}} = \left(\frac{n-\gamma}{k}\right)^{\alpha}$	Optimal	–	0.433	-0.215	0.906
		Random	–	1.000	0.000	1.918
Dimensions (d)	$\mu_{\text{cost}} = \left(\frac{d-\gamma}{k}\right)^{\alpha}$	Optimal	10	0.593	0.981	0.175
		Random	10	0.501	0.368	6.04×10^{-2}
		Optimal	20	0.656	1.24	0.102
		Random	20	0.500	0.369	1.50×10^{-2}

In Table 5, for random tour length and optimal tour length, we provide the scaling trends that we obtain for the tested TSP node and spatial dimension domains. Random tour length w.r.t. number of nodes is derived (Lemma 1). Other constants listed are empirical fits that may be useful for interpolation. For example, mean optimal tour length w.r.t. nodes produces errors less than 5×10^{-3} within the tested domain of 5 to 50 nodes. However, these fits should *not* be extrapolated to much larger scales because several of these fits provably break down. For mean optimal tour length, $\alpha \to 0.5$ as $n \to \infty$ [25], where our empirical fit is $\alpha = 0.433$. For mean random tour length w.r.t. spatial dimensions, α fits are close to the true limiting behavior $\alpha = 0.5$ (Theorem 4), but error eventually accumulates due to inexact proportionality constant fits.

B Trend breakdown and assessment of learning convergence

This section details larger-scale experiments that did not converge to trend. We also elaborate on relevant details of our training method. We show models which visibly failed to converge using this method and we identify boundary cases where convergence is more difficult to assess. These models underperform our scaling law predictions, evaluating to larger suboptimality values. If any model were to outperform our scaling law predictions, trend breakdown could not be attributed to insufficient training.

A significant portion of neural scaling laws research is performed by organizations with access to abundant computational resources. Experiments that initially fail to converge (if any) can be rapidly iterated. Researchers with more computational constraints often do include results for trend-breaking experiments. However, these results are sometimes mentioned in passing without evaluating model convergence. Besides promoting transparency, we provide this appendix to document the behavior of scaling law breakdown under compute budget constraints.

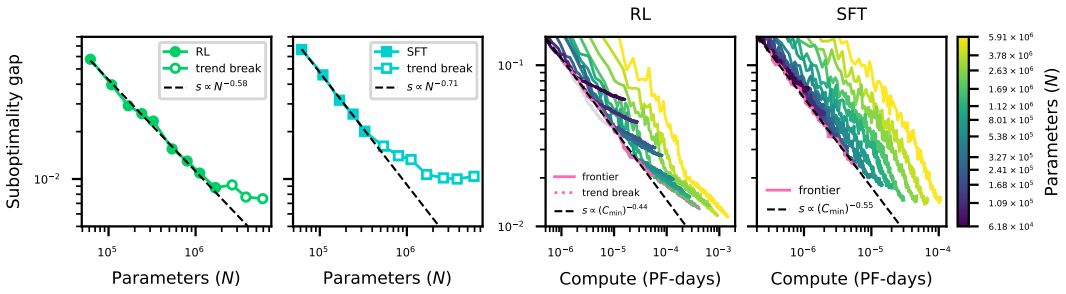

Figure 6: Suboptimality w.r.t. parameter and compute scaling, including experiments which did not converge to trend. **Left pair:** Parameter scaling experiments evaluated at the end of training. Unfilled markers indicate models not shown in Figure 1 that are excluded from power law fits. **Right pair:** Suboptimality evaluated throughout training w.r.t. compute. Models excluded from parameter scaling fits fail to reach the compute-efficient frontier with our training settings.

B.1 Training details

Without data constraints, RL models trained for one million gradient updates with a three-stage learning rate schedule: a brief linear warm-up, then cosine decay over early convergence, then a slow linear decay to zero. Cosine decay terminates at a learning rate of 10^{-5} at update number 170,000. This termination update was informed by hyperparameter optimization results that we detail in Appendix H. For the remainder of our training budget, 830,000 updates, it was a design choice to use a slow linear learning rate decay. For models which nearly converge within the cosine decay window, this approach reliably extracts remaining marginal improvements. However, late-stage training consumes most of the training budget. Models which insufficiently converge within the cosine decay window make less progress than they would if the learning rate was instead reduced more gradually. We opted for this trade-off to prioritize ensuring convergence for experiments at smaller scales, providing more precise results for the scaling laws we present in the main paper.

For SFT models, data constraints restricted training duration given we stop after one epoch to avoid potential performance bottlenecks from dataset size. For these experiments, after warm-up, the learning rate cosine decays to zero over the remainder of the epoch, which yields 73,143 updates with our dataset and batch size settings. Despite this abbreviation in training, SFT's strong supervised learning signal results in sufficient convergence for several smaller model scales and problem scales.

B.2 Neural scaling assessment

Figure 6 shows the complete set of parameter and compute scaling experiments we evaluated. This includes the fitted results shown in Figure 1 along with results from larger models that failed to converge. Parameter scaling evaluations drift right of trend and level off. Temporal compute scaling trends demonstrate that these models failed to converge with our training settings. Learning curves for larger SFT models maintain steep descents throughout training. Curves for larger RL models level off only slightly, which is attributable to the very small learning rates used in late-stage training.

168,000 parameter outlier: We also exclude the roughly 168,000 parameter model (shown in lower opacity) when fitting the RL compute-efficient frontier. This outlier is unrelated to convergence. However, fitting with this model separates the frontier from several other models that also converge. Scaling laws predict the expectation of model performance over random seed variables like parameter initializations and training batch sampling, and this outlier's random seed appears to perform unusually well. Averaging evaluations over several seeds would likely address this discrepancy.

B.3 Complexity scaling assessment

Figure 7 shows the full set of TSP node and spatial dimension scaling experiments we evaluated. For node scaling results, excluded SFT experiments only slightly break trend, whereas the excluded 50-node RL experiment sharply breaks trend. Figure 8 explains this behavior with suboptimality evaluations throughout training. Learning curves for excluded SFT experiments are more divergent

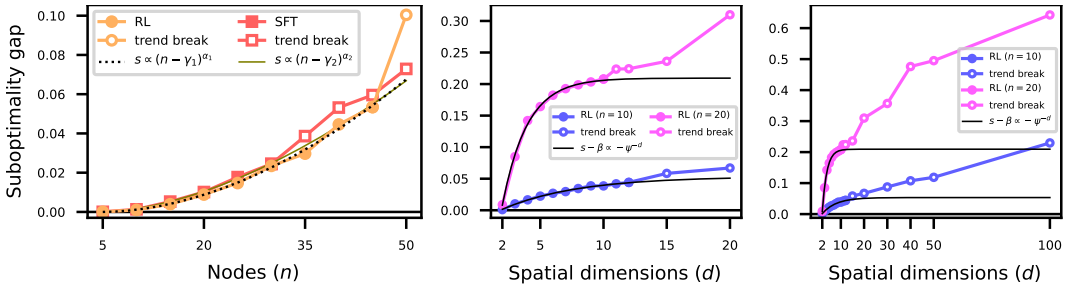

Figure 7: Suboptimality w.r.t. TSP node and spatial dimension scaling, including experiments which did not converge to trend. Unfilled markers indicate models not shown in Figure 3 that are excluded from scaling law fits. **Left:** Node scaling experiments evaluated at the end of training. SFT's power growth fit is shown in a different color only for clarity. **Right pair:** Spatial dimension scaling experiments evaluated at the end of training. We show early trend breakdown (center) along with the full domain up to 100 dimensions (right).

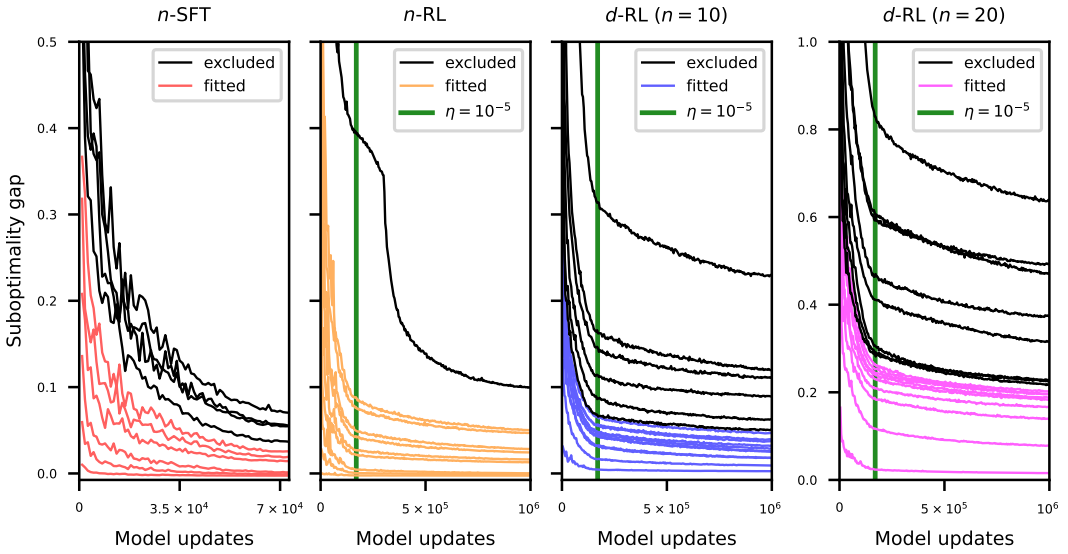

Figure 8: Suboptimality w.r.t. TSP node (n) and spatial dimension (d) scaling evaluated throughout training. For RL experiments, the end of cosine decay is marked with a green vertical line. Subsequent updates use learning rates below 10^{-5}, which impedes the convergence of models training on larger problem scales. Note that we use smaller 12,800 sample datasets to evaluate each learning curve. Each dataset is distinct since we scale the problem, so small differences between adjacent scales may not be statistically significant. The reduced sample size also (falsely) produces negative suboptimality values for 2D TSP with 5-node and 10-node scales where model performance becomes near-optimal.

and maintain steeper descents until learning rate decay impedes progress. And the excluded 50-node RL experiment performs particularly poorly in early training during the cosine decay period. Subsequent training makes significant progress, so this model may have initially approached a poor local optimum in loss landscape.

For spatial dimension scaling results, excluded experiments at scales beyond 20 dimensions fail to sufficiently converge in early training and maintain steeper descents in later training (right pair of Figure 8). However, excluded experiments between 11 and 15 dimensions are not as distinguishable from their fitted counterparts, especially with non-trivial imprecision from using smaller evaluation datasets that vary for each scale. These experiments correspond to the lowest black curve for the 10-node plot (15 dimensions) and the lowest three black curves for the 20-node plot (11, 12, and 15 dimensions). For these boundary cases, we solely rely on evidence from adjacent scales to surmise that further training can produce trend alignment.

C Loss behavior

For generative modeling, the majority of research on neural scaling laws studies cross-entropy loss [3, 4, 13]. However, in general, we do not expect loss to trend for RL algorithms [14]. Further, the scaling behavior of RL loss is often irrelevant to task performance, whereas cross-entropy loss in language modeling, for example, is directly relevant to task performance. This detachment from performance makes scaling laws for RL loss less interpretable.

Even so, we observe several loss behaviors worth documenting. We detail these behaviors below and summarize them here. First, separate loss components can concurrently trend and not trend even when these components have interdependent learning objectives. Second, variance of loss can reveal an underlying trend when mean loss does not trend. Third, when scaling problem complexity, token-level cross-entropy loss may not correlate with task performance.

C.1 Relevant experiment details

SFT models were trained to predict optimal tour length with a critic head alongside the primary objective of minimizing negative log-likelihood (NLL) of optimal next-node prediction. This setup maintains identical architectures (and equal parameter counts) between SFT and RL experiments, isolating differences to the policy learning objective.

Loss evaluations use the same datasets as corresponding suboptimality evaluations. We evaluate the same scales used for the main paper as we found that loss trend breakdown closely aligns with suboptimality trend breakdown when using our training method (with limited compute). Suboptimality trend breakdown is discussed in Appendix B.

C.2 Neural scaling loss behaviors

Figure 9 shows model test loss w.r.t. parameter and compute scaling. Supervised NLL loss power decays as expected from previous neural scaling laws work. PPO's clipped surrogate objective [20] (actor loss) has no obvious trend other than being zero-centered, also as expected. However, PPO's critic loss does smoothly power decay. The existence of this scaling law is more surprising given the value target distribution is non-stationary (following on-policy tour length) and the mean actor loss does not trend.

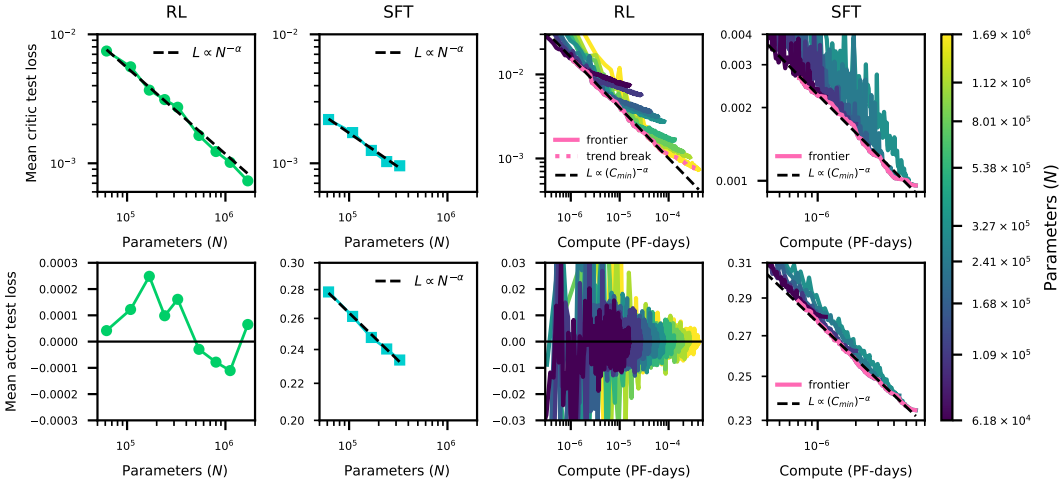

Figure 9: Loss w.r.t. parameter and compute scaling. Note that SFT's compute axes (right column) have been stretched for easier viewing, and the loss axes for those subplots do not share scale with the RL compute scaling subplots to their left (unlike Figure 1). **Top:** Self-supervised critic loss smoothly power decays when predicting on-policy tour length (RL) or predicting optimal tour length (SFT). **Bottom:** Supervised NLL actor loss smoothly power decays but PPO actor loss has no mean trend other than being zero-centered w.r.t. the non-stationary critic baseline.

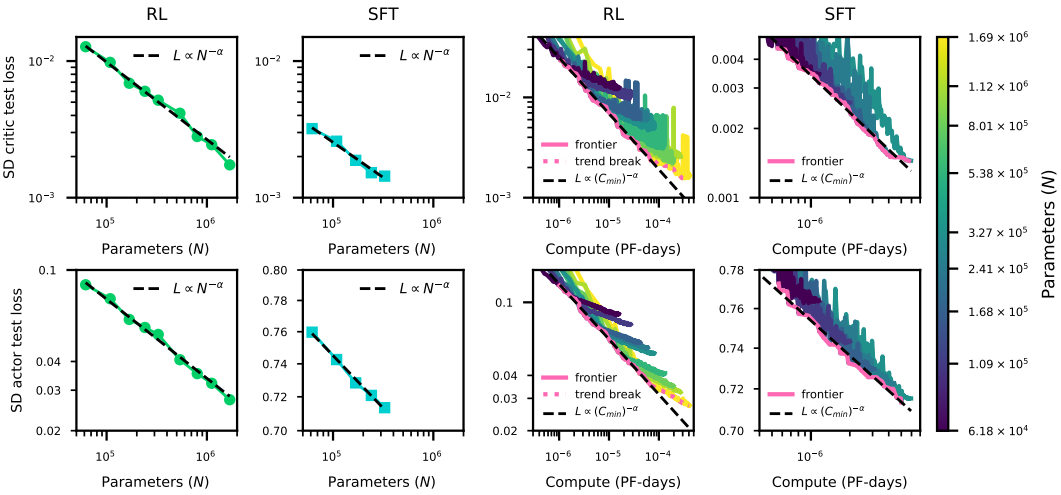

Figure 10: Standard deviation (SD) of loss w.r.t. parameter and compute scaling. Subplot layout matches that used for Figure 9. The variance of each loss component exhibits smooth power decay, even for the PPO actor loss which does not produce scaling laws for mean loss.

The behavior PPO actor loss *variance* may partially explain why scaling laws emerge for mean PPO critic loss. Figure 10 shows that variance of loss is also predictable when scaling parameters or compute. This observation is perhaps intuitive for loss components where the mean trends. However, we obtain similarly precise scaling laws for the variance of PPO actor loss, revealing that an underlying predictable nature persists despite the absence of a predictable mean. The existence of scaling laws for mean critic loss may depend on this predictable actor loss behavior, and vice versa, given the strong interdependence between the learning objectives.

Lastly, a subtle additional finding from these results is that scaling laws for loss appear more robust against outliers when compared to their suboptimality counterparts. For suboptimality trends in Figure 1, including the (roughly) 168,000 parameter model in the RL compute-efficient frontier fit noticeably skews the result. In contrast, while this outlier is still visible in the RL compute-efficient frontier for critic loss (Figure 9 top, third column), excluding it hardly alters the obtained fit.

C.3 Complexity scaling loss behaviors

Figure 11 shows model test loss w.r.t. TSP node and spatial dimension scaling. Aligning with observations for parameter and compute scaling, self-supervised critic loss components mirror the scaling law forms of their suboptimality counterparts, and PPO actor loss is zero-centered. However, supervised NLL loss w.r.t. node scale deviates from the power growth trend that SFT suboptimality and optimal tour length prediction adhere to. Instead, NLL loss exponentially decays with increased node scale over the tested domain.[8] Hence, as suboptimality accumulates with increasing solution space complexity, next-node prediction improves.

One potential explanation is that next-node prediction benefits from the increasing negative supervision: each label for next-node selection provides one positive example and $(n-1)$ negative examples. If we were to assume this trend extrapolates, then next-node prediction NLL converges as suboptimality diverges. If so, the joint NLL of full-tour prediction may still diverge given the number node predictions per tour increases, although we did not evaluate this metric. Alternatively, the trend observed for node-level NLL loss may break down shortly after the tested scales. In any case, node-level cross-entropy loss does not correlate with task performance. This complication is avoided when scaling laws are obtained for metrics that correlate with task performance by definition (for example, suboptimality).

We find that standard deviation (SD) of critic loss follows the corresponding scaling law form of suboptimality and mean critic loss, as shown in Figure 12. However, scaling behaviors are not as

[8] We also tested power decay but this form converges too slowly and fits a negative β asymptote (NLL is strictly non-negative).

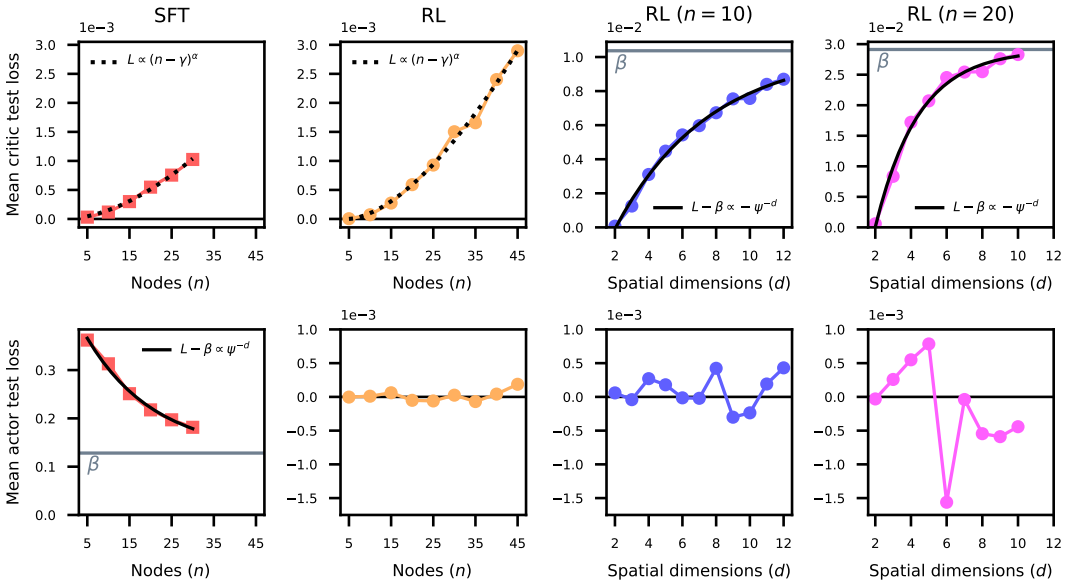

Figure 11: Loss w.r.t. TSP node and spatial dimension scaling. **Top:** Self-supervised critic loss adheres to the corresponding scaling law form observed for suboptimality (Figure 3). **Bottom:** PPO actor loss only follows a zero-centered trend, as it does for neural scaling experiments (Figure 9). Supervised NLL loss adheres to exponential decay when scaling nodes, deviating from the power growth pattern observed for suboptimality and critic loss. This trend implies that next-node prediction improves with node scale for SFT models, at least over smaller scales. However, this trend does *not* imply that full-tour prediction improves because the number of nodes predicted per tour increases.

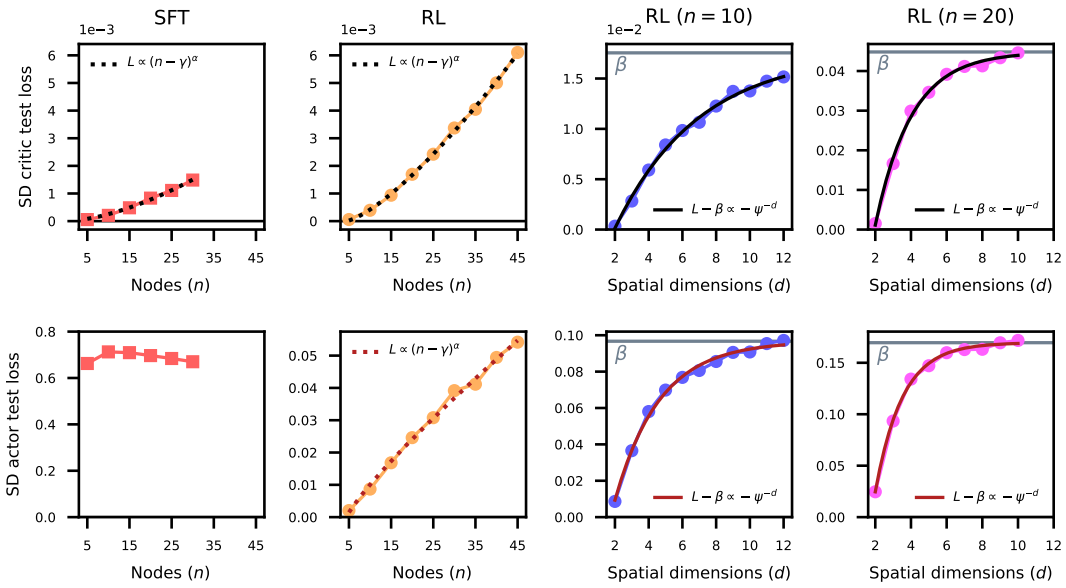

Figure 12: Standard deviation (SD) of loss w.r.t. TSP node and spatial dimension scaling. Subplot layout matches that used for Figure 11. **Top:** Standard deviation of critic loss adheres to the corresponding scaling law form of mean critic loss (and suboptimality). **Bottom:** Standard deviation of PPO actor loss roughly aligns with the corresponding suboptimality and critic loss trends, but fit quality is markedly worse (illustrated in red). Standard deviation of supervised NLL loss w.r.t. node scale does not follow a consistent trend (bottom left).

predictable for SD of PPO actor loss or supervised NLL. For the latter, node scale has no obvious correlation with SD. For the PPO actor loss, SD trends resemble their critic loss and suboptimality counterparts but fits are clearly inferior. We observe contiguous residuals of the same sign, and for spatial dimension scaling experiments exponential decay converges too fast, fitting β values below the SD values obtained at the largest scales. Compared to neural scaling results in Figure 10, predicting SD of loss appears to be more nuanced when scaling complexity. However, note that SD and variance would not share the dimension scaling law form due to the additive β term. Actor loss variance may produce a closer fit, but we did not evaluate these trends.

D Local search supplement

Section D.1 details the 2-opt cost landscape properties referenced in Sections 4.2 and 5. Section D.2 provides complexity scaling results for 2-exchange search, which uses a different definition of solution adjacency than 2-opt search.

D.1 Properties of the 2-opt cost landscape

For each complexity scaling scaling experiment, we measure three fundamental properties of the 2-opt cost landscape in expectation: the number of local optima, the size of a local optimum's basin of attraction, and the relative size of the global optimum's basin of attraction. Results in Figure 13 suggest that these properties converge when scaling TSP spatial dimensions. In contrast, these properties diverge when scaling the number of TSP nodes.

D.1.1 Evaluation method

To measure these properties, we perform numerous search descents (k) for each TSP problem evaluated. Each descent starts at a random tour and terminates when arriving at a locally optimal tour (w.r.t. one additional 2-opt move). We then count the number of unique local optima found, count the number of search moves required to reach them (basin of attraction size), and evaluate the fraction of descents that arrived at the best-found local optima (relative size of the global optimum's basin of attraction).

Estimating the number of local optima requires adequate coverage of the landscape, as does estimating the visitation rate of the global optimum given we approximate the global optimum using the best-found local optima (BLO). For these estimations, we perform 100,000 descents per problem (large-k) while sampling small batches for each scale: 2048 problems for 10-node dimension scaling, 512 problems for 20-node dimension scaling, and 32 problems for node scaling.

Evaluating mean search moves does not require adequate coverage of the landscape, so we use fewer descents per problem and sample larger problem batches (small-k, large-batch). For dimension scaling, we reuse the approximately optimal datasets detailed in Appendix E. For node scaling, we use 100 descents per problem and sample 64,000 problems. We also show evaluations for local optima count and BLO visitation rate using these datasets for comparison (blue triangles in Figure 13).

D.1.2 Property convergence as $d \to \infty$

Figure 13 shows that expected search depth and BLO visitation rate both power decay when increasing spatial dimensions. The former implies that basin of attraction size converges in expectation. Because each 2-opt search move evaluates $O(n^2)$ adjacent solutions, and TSP node scale remains constant, convergent search depth implies convergence of the number of solutions evaluated per descent. Alongside the convergence of the BLO visitation rate, these observations suggest that, with many-descent local search, finding the optimal solution approaches a stationary computational complexity apart from the $O(d)$ complexity of evaluating each visited solution's cost.

Unlike search depth and BLO visitation rate, the number of local optima has no predictable convergent trend and instead grows non-monotonically. This is not an artifact of sample size as we observe the same behavior for small-k, large-batch evaluations (with a significant underestimation bias). Growth visibly slows by 100 dimensions, and growth is upper bounded by the fixed solution space size, so convergence is plausible but not directly implied by these results.

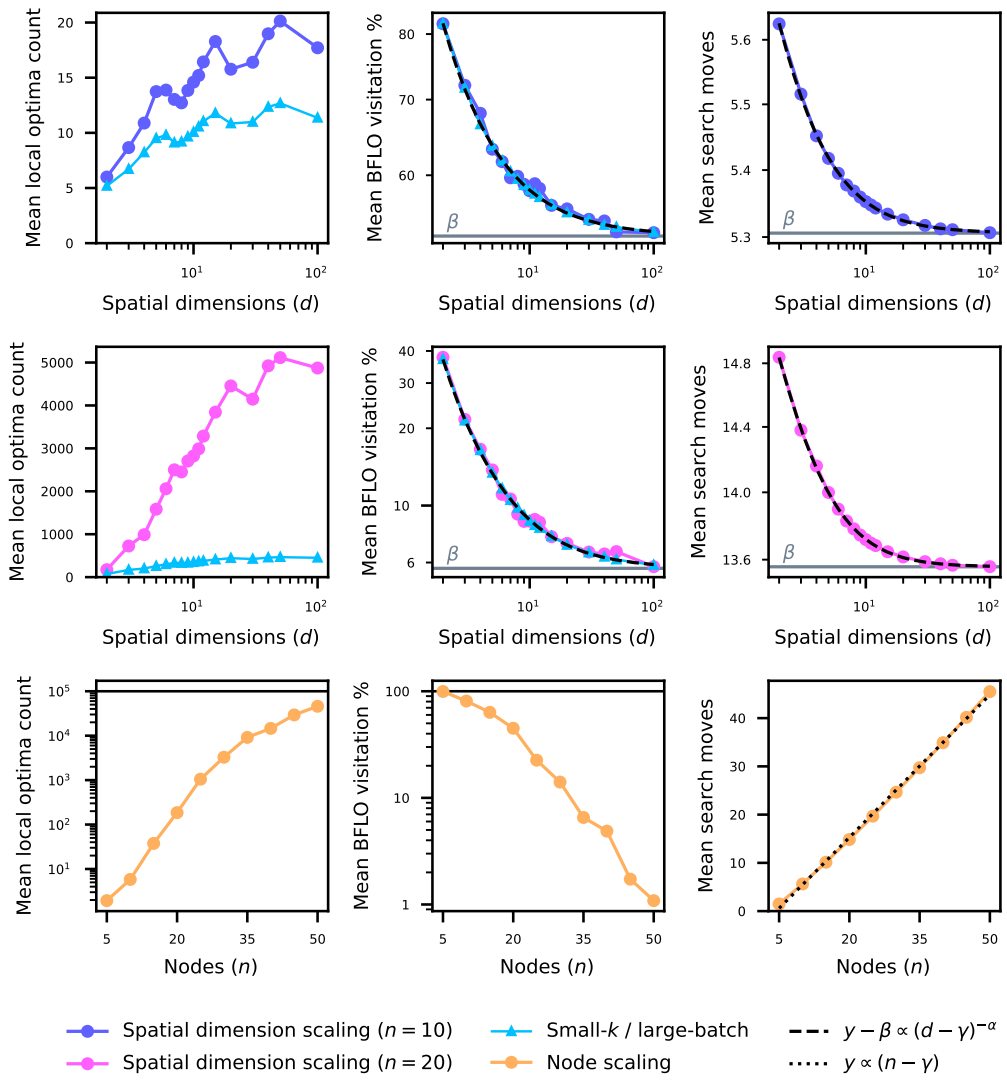

Figure 13: 2-opt cost landscape properties w.r.t. TSP node and spatial dimension scaling. From top to bottom: 10-node spatial dimension scaling, 20-node spatial dimension scaling, and 2-dimensional node scaling. From left to right: mean local optima count, mean frequency that search arrives at the best-found local optimum (BLO), and mean search depth before arriving at a local optimum. **Dimension scaling:** The number of local optima exhibits non-monotonic growth that is upper bounded by the fixed size of the solution space. BLO visitation rate and search depth adhere to power decay. We found that including a γ scale offset improves fit quality, but excluding γ still produces reasonable fits. From left to right axis scalings are: [semi-log-x, log-log, log-log]. **Node scaling:** Due to the expanding solution space, the number of local optima diverges as does the search depth required to reach one. BLO visitation rate decays roughly exponentially. Note that local optima count is underestimated by 100,000 descents per problem [52]. From left to right scalings are: [semi-log-y, semi-log-y, linear].

Lastly, these properties are closely related to landscape ruggedness measures like autocorrelation [55, 65]. Because the expected range of cost values also converges (Theorem 6), the previous findings may imply autocorrelation converges as $d \to \infty$. However, we did not directly evaluate this measure.

D.1.3 Property divergence as $n \to \infty$

Node scaling produces none of the convergent complexity behaviors discussed above. Cost landscape analysis over solution space scaling is a well-studied topic in TSP and combinatorial optimization [52] (Section 1.1 in the cited work provides a useful overview). Our result is provided for convenience.

When scaling nodes, the number of local optima rapidly exceeds what we can accurately estimate with 100,000 descents per problem. For 3-opt search moves, Tayarani et al. [52] show that this growth appears to be $O(e^{n \ln(n)})$. 2-opt's search depth grows almost linearly and 2-opt's BLO visitation rate decays roughly exponentially. Both observations align with 3-opt findings [52]. Note that BLO visitation rate converging toward zero does not reflect a steady state property of the cost landscape: we are measuring the relative size of the global optimum's basin of attraction as the absolute size of the landscape diverges.

D.2 2-exchange search results

In this paper, we define "2-exchange" as the naive search move that swaps two nodes in the tour sequence. We are not the first to use this definition [66]; however, the term is sometimes used to refer to 2-opt in informal sources. 2-opt is the distinct search move that inverts a subtour [31].

Figure 14 shows that 2-exchange reproduces 2-opt's alignment with the complexity scaling law forms of parameter-constrained models (Figure 5). However, both moves modify only a few edges and both adjacency sets scale $O(n^2)$, so this result demonstrates limited generalization. Analyzing the conditions in which this alignment generalizes for k-opt moves [67] is one potential direction for future work.

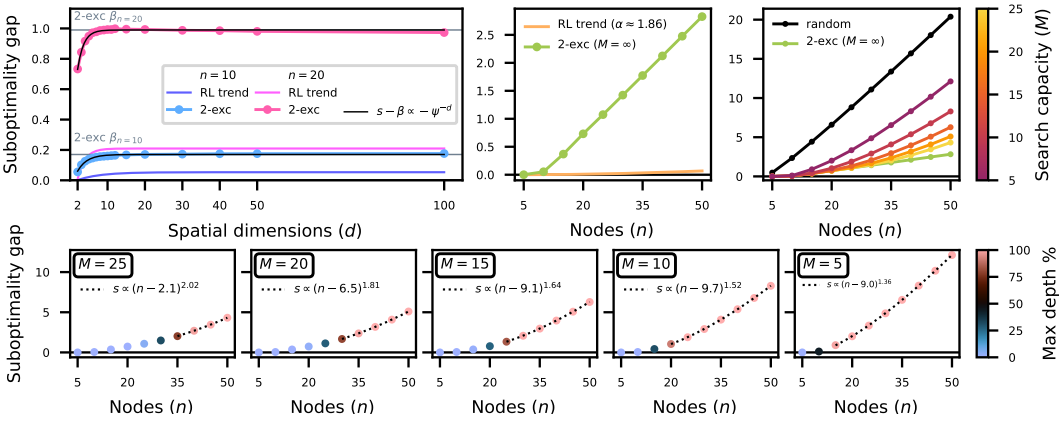

Figure 14: 2-exchange local search suboptimality over problem complexity scaling. When scaling nodes and constraining search depth, suboptimality follows superlinear power growth after saturating at 100% max depth (top right and bottom), aligning with 2-opt behavior and the scaling form of parameter-constrained models. Like RL fits, 2-exchange fits do not benefit from fitting the subexponential constant ϕ when scaling spatial dimensions. However, 2-exchange dimension scaling fits are slightly inferior to their 2-opt counterparts, which may be attributable to 2-exchange exhibiting several of the random-suboptimality behaviors shown in Figure 4. We observe a slight decrease near convergence when scaling dimensions for the 20-node experiment, and unconstrained 2-exchange search produces linear suboptimality growth when scaling nodes (beyond 10 nodes).

E Optimal solution approximation in higher dimensions

For node scaling evaluations we leverage the Concorde-generated optimal solutions introduced in Section 2.2. But evaluating suboptimality in higher dimensions requires a different approach since the PyConcorde software stack expects 2D TSP. To the best of our knowledge, applying Concorde to higher-dimensional TSP has not been explored. This appears to be only an implementation constraint, though, and not a limitation of the Concorde algorithm itself since it uses the cutting-plane method [16] and additional spatial dimensions only modify the edge cost calculation. Regardless, we opted to closely approximate optimality for convenience.

E.1 Approximation method

Near-optimal solution tours are generated via mass repetition of local search. For a single TSP instance, k random starting tours are sampled, and for each starting tour a distinct descent of local search is performed. Among the k generated locally optimal tours, the tour with lowest cost is selected as a *surrogate* for the globally optimal tour. This full process is repeated B times using a large batch of sampled TSP instances to acquire near-optimal datasets at a desired problem scale.

We use 2-opt moves when generating all surrogate optimal datasets. There are more sophisticated local search algorithms that usually find lower cost solutions for TSP [67, 68], though often with increased search time or compute. But we find 2-opt surrogate optimality to be sufficiently accurate at the needed problem scales despite its simplicity.

E.2 Dataset validation

Near-optimal suboptimality scaling laws are sensitive to the precision of optimality estimates, so here we investigate approximation quality. For clarity, we refer to surrogate optimal solutions as "surrogate" and reserve "optimal" for strictly optimal solutions.

First, we compare surrogate and optimal solution distributions in 2D TSP where we have datasets for the latter via Concorde [17]. Figure 15 shows that surrogate solution tour length closely approximates the distribution of optimal solution tour length at both 10 and 20 nodes using best-of-100 and best-of-1000 2-opt descents, respectively. Being normally distributed, we perform two sample t-tests on the means in Table 6. For each node scale, the mean surrogate tour length is slightly below the mean optimal tour length, which is possible since surrogate and optimal datasets sample distinct

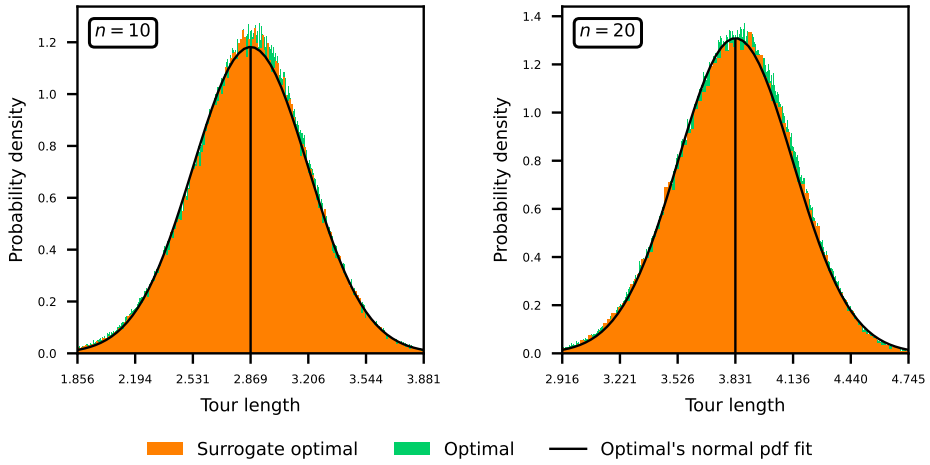

Figure 15: Histograms of surrogate tour lengths overlaying optimal tour lengths for 10-node and 20-node dataset pairs. Both are clearly normally distributed, justifying the t-test performed in Table 6, and the two distributions are visually similar with the optimal histogram's probability density function (pdf) fit aligning closely with the surrogate histogram. X-ticks are arranged in standard deviations of the optimality pdf fit.

Table 6: Two sample t-test comparing mean tour lengths of surrogate solutions and optimal solutions for 2D Euclidean TSP. P-values are obtained for both 10-node and 20-node dataset pairs. We fail to refute the null hypothesis that surrogate solutions come from the same distribution as optimal solutions.

	10 nodes		20 nodes	
	Surrogate	Optimal	Surrogate	Optimal
Samples	128,000	1,280,000	64,000	1,280,000
Mean	2.86839	2.86870	3.82970	3.83082
Standard deviation	0.33727	0.33753	0.30534	0.30478
t-value	-0.314		-0.906	
$\mathbb{P}(t <$ t-value$)$	0.377		0.183	
$\mathbb{P}(t >$ t-value$)$	0.623		0.817	

problems. Thus, along with the surrogate algorithm itself, we are testing whether the smaller size of the surrogate datasets can sufficiently approximate 1.28 million samples from the optimal distribution.

At both node scales, we are unable to reject the null hypothesis that mean surrogate tour length is equivalent to mean optimal tour length. Because these surrogate datasets underestimate optimality, from these results we clearly cannot suggest the surrogate mean is larger than the optimal mean. But we also do not find spurious evidence that the surrogate distribution underestimates the optimal distribution on average, with lower one-sided p-values of 0.377 and 0.183 at 10 and 20 nodes, respectively. With our chosen surrogate dataset settings, for 2D TSP with no more than 20 nodes, surrogate solutions appear statistically indistinguishable from optimal solutions (as desired).

One problem would arise if the best-found local optimum (BLO) visitation frequency were overestimated by fewer descents in surrogate solution generation. This overestimation would imply surrogate solutions require more descents to find the true global optima located in relatively small basins of attraction. Instead, we find the BLO visitation rate of surrogate solution generation is almost identical to that observed in the small batch 100,000 descents-per-problem experiment in Figure 13 (center and top center). The error magnitudes do not exceed 1.5% and 0.7% for 10-node and 20-node dimension scaling arrays, respectively. Alongside the high BLO visitation rates themselves, this suggests global optima are often found with only a few 2-opt descents, a finding consistent with existing analysis of 3-opt [52] (Figure 15 in cited work).

Another problem would arise if surrogate solution generation were to visit only a tiny fraction of local optima. Even with a high BLO visitation rate, eventually the number of missed global optima may accumulate. Figure 16 plots the mean local optima discovery fraction of surrogate solution generation, where we estimate the true mean (divisor) using the 100,000 descents-per-problem experiments. The 10-node surrogate solutions uphold a greater than 60% discovery rate by 100 dimensions, increasing confidence in global optima discovery considering the BLO visitation rate maintains values greater than 50%. In contrast, 20-node surrogate solutions' local optima discovery decreases to around 10% by 100 dimensions. This discovery rate may still be sufficient to find the global optima in the overwhelming majority of problems. But given the 20-node BLO visitation rate sinks to roughly 6% (Figure 13 center), we are of course less confident.

There are three smaller caveats for Figure 16 also worth noting. First, the confidence intervals assume that the number of local optima at each problem scale are normally distributed. We did not verify as we only saved summary statistics for these experiments. If the local optima count was (for instance) long-tailed toward larger values, this could add uncertainty to our estimates. The second caveat is that we ignore uncertainty in the surrogate solutions' local optima discovery rate, which is relatively small since the surrogate datasets are orders of magnitude larger than those used to estimate the true discovery rate (2048 and 512 samples for 10 and 20 nodes, respectively). Lastly, 100,000 descents-per-problem may still miss elusive local optima with small basins of attraction. Even so, these local optima are quite unlikely to be global optima if 3-opt trends generalize to higher-dimensional 2-opt solutions [52] (Figure 14 in cited work).

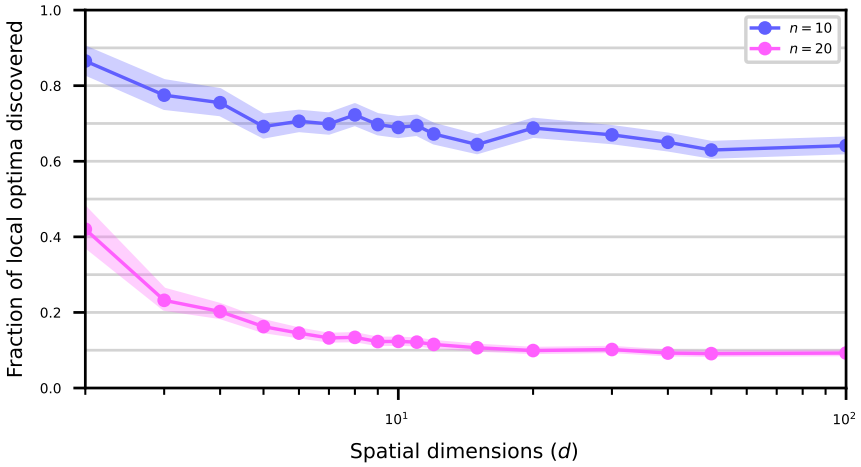

Figure 16: Surrogate solution generation's estimated mean local optima discovery fraction. Fills show 99% confidence intervals assuming local optima count is normally distributed. True mean local optima count is estimated with small batches of 2-opt local search performing 100,000 descents-per-problem. 10-node surrogate generation never falls below 60% discovery, increasing confidence that best-found local optima are usually the true global optima. 20-node generation falls below 10% discovery by 20 spatial dimensions, suggesting that best-found local optima at these scales may be suboptimal more often.

F Compute requirements

Each model was trained on an HPC node using one Nvidia Tesla V100 (16GB), 24 Intel Xeon Gold 6136 cores, and 384GB of memory. GPU usage was the bottleneck, so fewer CPU cores and less memory would be sufficient. As mentioned in Section 2.5, the 50-node RL run trained for 24 days and consumed roughly 3×10^{-3} PF-days of compute. The longest spatial dimension scaling runs and parameter scaling runs trained for about half that time and consumed around an order of magnitude less compute.

Optimal solution generation only used CPU compute and was performed on the same HPC nodes. Using Concorde [17] via PyConcorde [15], generating 1.28 million optimal solutions to 50-node 2D TSP required roughly 10 days. We provide 10 of these chunks for each TSP node scale we experimented on: 12.8 million solutions per scale and 128 million solutions in total. Generating approximately optimal datasets for higher-dimensional TSP (Section 2.2 and Appendix E) required around 2 hours per 10-node scale and 60 hours per 20-node scale. Note that we used a Cython implementation of 2-opt, which we found to be several times faster than a pure Python implementation.

G Model architecture

Our model design prioritizes simplicity where possible. Besides making experiments easier to implement, this choice was intended to improve the reproducibility of our results and promote their generalization to other deep architectures. All core model components are implemented using PyTorch [69] Transformer modules. Model code, algorithm code, and training scripts are provided in the project repository at `https://github.com/lowellw6/complexity-scaling-laws`. Specific hyperparameter choices are discussed separately in Appendix H.

G.1 Policy network

Our policy network is most similar to that introduced by Kool et al. [62] with two key simplifications. First, our encoder does not produce a full graph embedding, and so does not attend over a context node embedding during the subsequent forward passes of the autoregressive decoder. Second, unlike Kool et al. and Bresson et al. [70], our decoder does not include a final single-head attention layer. Instead, our decoder performs multi-head attention over all node encodings for each decoding step

while building a positionally-encoded partial tour in the decoder memory. No masking occurs until directly before logits are softmaxed and sampled, where probabilities are set to zero for previously selected nodes.

We moved forward with these simplifications after obtaining near-optimal performance without them at our experimented problem scales. But Kool et al.'s design has compelling motivation, especially concerning efficiency. Their graph embedding allows each decoder step to attend over the reduced subset of selectable nodes while still referencing a representation of the full problem. This linear ramp down of attention length still incurs $O(n^3)$ complexity to decode a full tour, as does our approach which maintains constant attention length.[9] But pruning previously selected nodes requires only a third the compute, all else being equal. Given that autoregressive decoding is most of the forward pass, we suggest reconsidering this design choice when experimenting with large node scales.

G.2 Critic network

Our architecture learns values for the policy gradient baseline. Kool et al. instead use deterministic greedy rollouts from a baseline policy network, referencing the difficulty of multi-objective actor-critic learning. But we found the contrary to be true in our experiments. Our greedy rollout implementation often diverges early in training (Figure 17 pink), which may stem from one of several distinctions. First, we use a PPO actor loss rather than REINFORCE. Second, we synchronize the baseline policy every 625 gradient updates, matching Kool et al.'s period, but they only synchronize if a paired t-test determines statistically significant improvement. Lastly, we do not regenerate the evaluation dataset after synchronizing (to avoid overfitting), though we do use a larger samples size of 100,000. Where possible we also use hyperparameters from Appendix H, which are optimized for compositional value learning (discussed below), creating a less fair comparison.

Regardless, learning a value function was useful for studying critic loss trends (Appendix C). To learn problem-level state values, we use a second Transformer decoder which receives node encodings from the (now shared) policy encoder. But encodings are input as decoder memory (without positional encoding) and a zero-fill start token is input for the target. The output is projected into a scalar value estimate which we train to predict on-policy tour cost for PPO, and optimal tour cost for SFT. Requiring just one forward pass, this is a relatively lightweight addition. We found this simple approach converges well in our experiments (Figure 17 blue).

However, problem-level value estimates provide an increasingly stale baseline the longer decoding iterates. For example, if just two nodes remain unvisited in a partial tour of 50, the range of possible cost outcomes dramatically reduces. From the perspective of the 49$^{\text{th}}$ decoding, all previous node selections can be considered as components of the current node-level state. Note that our framework of problem-level states is merely the outcome of our *choice* to formalize the TSP environment as a bandit problem. We could have instead chosen to formalize the environment as an MDP with node-level decisions, and if we set a discount factor $\gamma = 1$, this MDP generates identical returns to our bandit problem framing. (Because no rewards occur before the terminal step.)

Attempting to learn more granular credit assignment, we implemented node-level value estimation. We refer to this alternative as *compositional value learning*, and refer to problem-level value learning as state value learning. We compute compositional values through a straightforward modification to our state-value critic architecture: concatenating the positionally-encoded node selection sequence to the start token for target input. This produces $n + 1$ compositional values, each attending over incremental partial tours in one forward pass via a causal mask. The first output remains the problem-level state-value, attending only over the start token and node encodings. The final output is the Q-value, attending over all node selections. We explain how compositional values influence learning objectives in the following subsection.

G.3 PPO using a compositional value baseline

This section describes how we set up the PPO actor-critic loss to learn and leverage node-level compositional value estimates. We use this style of PPO for all RL experiments in the main paper.

[9] Self-attention has $O(n^2)$ complexity w.r.t. to the node scale and attention length n [19], which we iterate n times.

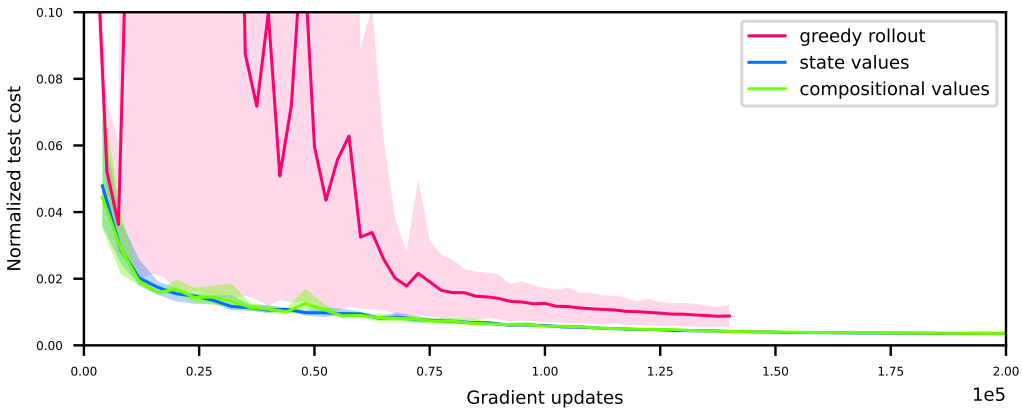

Figure 17: PPO baseline alternatives training on 2D 30-node TSP, evaluated on a held out set of 100,000 problems. We normalize suboptimality between optimal performance and random performance in the range $[0, 1]$. We summarize 5 random seeds for each baseline, where lines showing the means and fills showing the min/max values. Performance converges well when learning problem-level (state) values or node-level (compositional) values with a critic network, with no meaningful difference between approaches. But using greedy rollouts via a separate policy (updated sparsely like a target Q-network) usually produces temporary divergence. We truncate this curve because two runs timed out due to external load on our HPC cluster. Of the three runs that reached 200,000 updates, none surpassed learned value function performance.

We omit these details for simplicity and because we observe no evidence of performance differences between node- and problem-level baselines (Figure 17 green and blue).

To leverage compositional values in the actor loss, the PPO clipped surrogate objective [20], for each node selection we substitute the problem-level value for the node-specific value of the preceding partial tour. Formally, we minimize:

$$L^{CLIP}(\theta) = \mathbb{E}_{t,i}\left[\max\left(r_{t,i}(\theta)A_{t,i}, \text{clip}(r_{t,i}(\theta), 1-\epsilon, 1+\epsilon)A_{t,i}\right)\right]$$
$$A_{t,i} = c_t - V_{t,i}(\theta)$$

for on-policy tour t and node selection index i, where we instead take the maximum term since we are minimizing cost c_t. The ratio function $r_{t,i}$ is unchanged besides specifying the policy's node selection index. Note that we do not add an entropy bonus since optimal edge selection is deterministic, and doing so may incentivize suboptimal infinite-compute performance, entangling the entropy objective with the model limitations we intend to observe.

Modifying the critic loss to learn compositional values is less straightforward since cost is defined only for a complete solution. Transitioning from problem- to node-level credit assignment resembles the transformation from top MDP to bottom MDP illustrated by Metz et al. [71].[10] Because the return is shared between node selections within a particular tour, the corresponding compositional value targets are identical as well. Our critic loss is then:

$$L^{CRT}(\theta) = \mathbb{E}_{t,i}\left[\left(c_t - V_{t,i}(\theta)\right)^2\right]$$

In theory, this loss still allows compositional value targets to diversify in expectation. If rollouts from a given partial tour usually result in low-cost solutions, the expected compositional value target is small, and the inverse is also true. In preliminary experiments, compositional value learning appeared slightly more sample efficient compared to strictly learning problem-level state values, motivating our selection of the former.

[10]Their related work section provides a useful introduction to other deep learning algorithms which address large discrete action spaces.

However, this distinction disappears when carefully controlling for other variables and averaging over several seeds, at least for 30-node TSP. After initial transient effects, performance distributions are indistinguishable (Figure 17), and we noticed that compositional value estimates vary little within a given tour. These findings are not surprising given the critic attempts to learn granular values with a coarse reward signal, an analogue to value estimation in a sparsely rewarded MDP. Further, given tours are sampled on-policy for problems generated online, the critic sees each partial tour exactly once.

H Hyperparameter settings

Here we provide the hyperparameter (HP) settings used for model training (Table 7), along with how we determined them through hyperparameter optimization (HPO). We optimize HP selection for PPO experiments, but we also use the obtained settings for SFT experiments where applicable.

Critically, previous work suggests scaling law fits are relatively insensitive to model and algorithm HPs like Transformer model shape and learning rate decay [3]. Within reasonable bounds, we do not expect different HP settings to severely alter the trends found in this paper so long as training sufficiently converges.

Setting HPs effectively by hand would be tedious and rely heavily on intuition. Reinforcement learning often involves more HPs than supervised learning, and setting model HPs for our custom architecture lacks the grounding of existing results. Algorithmic HPO has the added benefit of reducing wall-clock time that is required to reach sufficient convergence. With finite resources, unbottlenecked compute can only be approximated in the scaling laws that require it. Some of our intuition-informed preliminary experiments required several million gradient updates (and multiple learning rate warm restarts) to sufficiently converge, dramatically slowing research flow. Our HPO-informed experiments converged well in less than half the gradient updates.

Table 7: Hyperparameters used for RL node-scaling experiments. Other experiments, including SFT training runs, default to these settings where applicable unless otherwise specified. After learning rate cosine decay finishes, RL runs switch to a slow linear decay for the remainder of training.

Category	Name	Value	HPO-informed
PPO	Minibatch size	175	No
	Minibatches	4	Yes
	Ratio clip	0.17	Yes
	Critic loss coefficient	0.52	Yes
Optimizer	Algorithm	Adam	No
	Gradient norm clip	0.24	Yes
Scheduler	Max learning rate	9.37×10^{-5}	Yes
	Linear warm-up updates	3,000	No
	Cosine decay finish update	170,000	Yes
	Cosine decay floor	10^{-5}	No
Model	Encoder layers	3	Yes
	Policy decoder layers	2	Yes
	Critic decoder layers	2	Yes
	Transformer width	184	Yes
	Transformer feedforward	736	No
	Transformer attention heads	8	No
	Transformer dropout	0	No

H.1 HPO method

We use BOHB [72] implemented with Optuna [73] to optimize update efficiency for 50-node TSP. We truly wish to optimize serial-compute efficiency, so we more strictly limit model depth, especially for the autoregressive policy decoder. We chose our maximum scale of 50 nodes to optimize the slowest run, and because we expect high-performing HPs for harder TSP scales to generalize well

to smaller, easier scales. But HPs optimized for small scales may fail to converge on larger scales within the same update budget.

We chose BOHB for its balance of anytime and final performance [72]. BOHB is no longer state-of-the-art in final performance but has comparable anytime performance [74] and is relatively easy to implement. We implemented BOHB using Optuna 3.5's multivariate Tree-structured Parzen Estimator (TPE) [75] for HP sampling (the Bayesian optimization "BO" in BOHB) paired with their Hyberband [76] pruning algorithm (the "HB" in BOHB). Table 8 details the HPs we used for the BOHB algorithm itself along with our evaluation setup. We found BOHB especially sensitive to the minimum gradient updates budgeted per trial. Setting this value too low results in pruning based on noise before initial convergence. Setting too high wastes a significant amount of time and compute.

Several HPs in Table 7 were fixed for various reasons. PPO minibatch size was maximized under GPU memory constraints for our search space. The learning rate linear warm-up schedule was fixed to avoid incentivizing rapid ascents that are more likely to diverge. Cosine learning rate decay was selected because we observed other schedules like exponential decaying too quickly to sufficiently converge; this issue was also noted in related work [3]. We fixed the learning rate decay floor to allow for a subsequent linear decay to zero over the remainder of training. This design choice couples fast, HPO-informed early convergence with meticulous late-stage convergence that appears to better approximate the infinite-compute performance limit.

Table 9 details our BOHB search space. We set relaxed bounds in preliminary attempts then trimmed unpromising regions. Runs often diverged when surpassing the reported upper bounds for max learning rate, gradient norm clip, and PPO ratio clip. GPU memory constraints limited model width's upper bound and our results suggest wider networks would perform better (Figure 18 top-left).

Lastly, we found that forcing a warm-start, human-intuition HP configuration for trial 0 allows BOHB to find high-performing regions of the search space considerably faster than when using only random startup trials.

H.2 BOHB results

Figure 18 summarizes the BOHB search results that were used to inform our HP selection. Using larger model widths, a tight range near 10^{-4} for max learning rate, and a small gradient norm clip less than 1 are most important for achieving high performance with fewer model updates. Convergence was particularly sensitive to gradient norm clip values greater than 1. In preliminary attempts without gradient clipping, pruning simply selected trials which most recently diverged.

PED-ANOVA importance ranking [77] was chosen over the more common f-ANOVA [78] algorithm as we found the latter to be overly sensitive to poor performing regions of HP space (attributing majority importance to gradient norm clip). PED-ANOVA importance quantifies a HP's degree of influence in attaining a top quantile of performance [77], which we found produces a more balanced ranking that also roughly aligns with concentration of high-performing clusters. We used a top-0.22 quantile which computed the local HP importance of the top 7 of 32 complete trials.

Optuna tracks objective scores by the final value achieved at the maximum update budget, not the best value achieved over the trial, which poses a question of statistical significance in near-optimal improvements. For example, trial 300 had the best outcome with a suboptimality gap of 0.169 at completion. But its minimum validation score over training was slightly below this at 0.166, though this happens to be the minimum score reported over all validations for all trials, which is encouraging. We evaluated the impact of our validation dataset size relative to these small performance deltas. This evaluation was performed retrospectively using the 50-node model from our main experiments since no checkpoints were saved during HPO. We rescored with 100 randomly sampled datasets of 10,000 problems each, obtaining a standard deviation of less than 0.003 suboptimality. The difference in suboptimality between the top two trials is over four standard deviations. Even comparing the final validation score of the top trial with the minimum score of the second-place trial returns a difference of over two standard deviations, demonstrating that our sample size is statistically meaningful assuming validation scores are normally distributed.

Table 8: BOHB algorithm and data settings. Remaining configurations use the defaults in Optuna 3.5.0. PPO minibatch size is fixed at 144 due to GPU memory constraints, and Transformer feedforward dimensions are set at $4\,d_{model}$. Otherwise, non-HPO-informed settings match Table 7.

Category	Name	Value
Evaluation setup	Trials	400
	TSP nodes	50
	Validation dataset size	10,000
	Validation period (gradient updates)	100
Hyperband pruner	Reduction factor η	3
	Min gradient updates	1,000
	Max gradient updates	100,000
TPE sampler	Startup trials	10
	Multivariate KDE	True

Table 9: Experiment variables and corresponding search spaces for BOHB HPO. Continuous intervals are log-transformed. Transformer width's interval is the minimum value for 8 attention heads.

Category	Name	Min	Max	Interval
PPO	Minibatches	1	12	1
	Ratio clip	0.05	0.2	0.01
	Critic loss coefficient	0.1	2.0	0.01
Optimizer	Gradient norm clip	0.1	10	Continuous
Scheduler	Max learning rate	5×10^{-5}	5×10^{-4}	Continuous
	Cosine decay finish update	100,000	500,000	1,000
Model	Encoder layers	1	4	1
	Policy decoder layers	1	2	1
	Critic decoder layers	1	12	1
	Transformer width	96	192	8

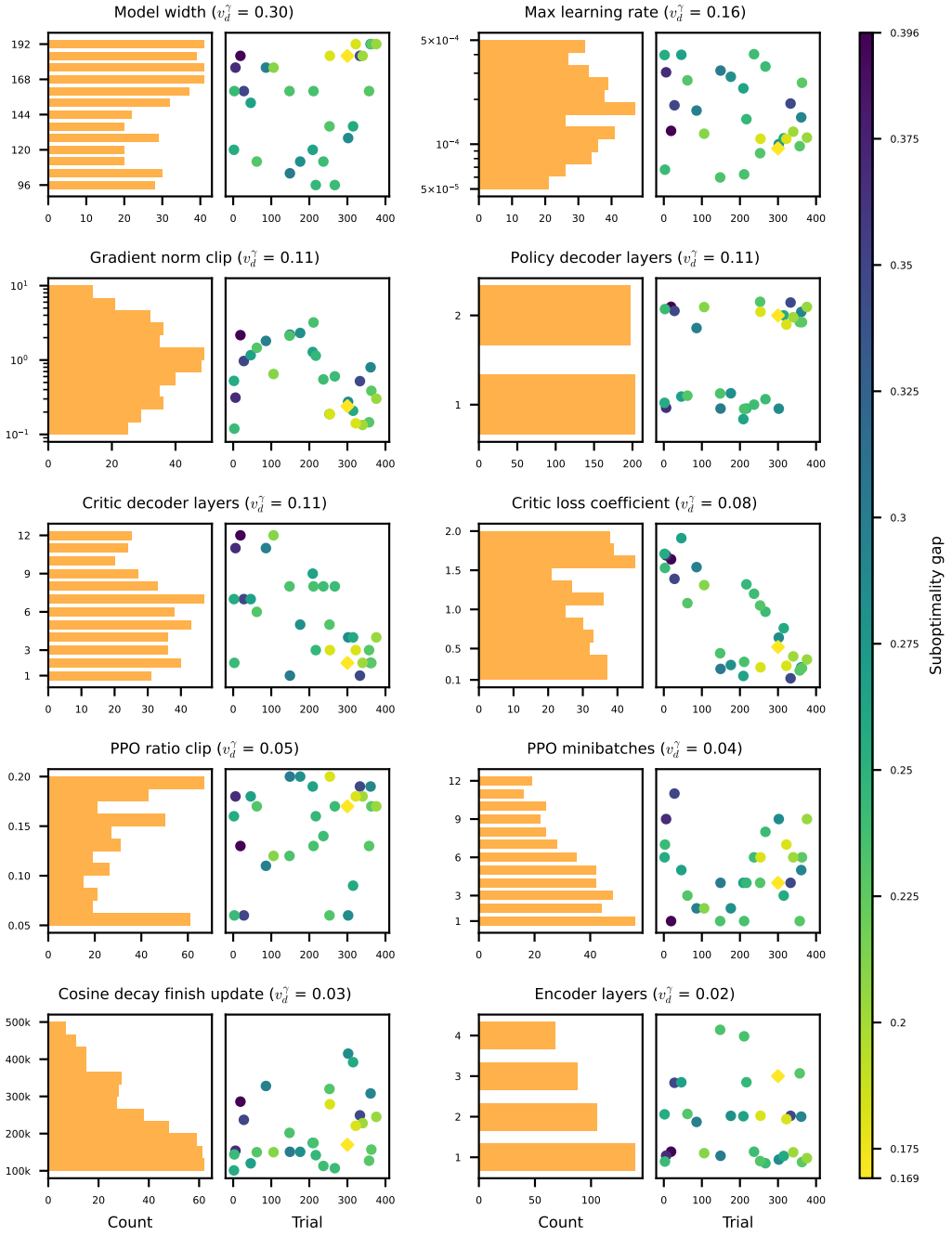

Figure 18: Search results of BOHB HPO on 50-node TSP for each HP in the search space. Y-axis labels are placed above subplots for clarity. The bar plots show HP search frequency over either discrete (bar gaps) or binned (no bar gaps) ranges, referencing all 400 trials. The scatter plots illustrate the performance of these HP ranges through the final scores of non-outlier completed trials, similarly to the plot generated by Optuna's visualization.plot_slice() function. Random Y-jitter is added to the policy decoder layers and encoder layers scatter plots to reduce point overlap. The best performing trial 300 is marked by a diamond (the wholeness of this number is just a coincidence and irrelevant to the HP sampling procedure). HPs are ranked by PED-ANOVA importance v_d^γ [77] in descending order left to right, top to bottom. Note the distinctive clusters of high-performing (warmer) regions found over time for higher-importance HPs.

I Proof of scaling relations and limits

Here we prove the node and spatial dimension scaling relations asserted in Section 4 mostly through extensions of existing analysis. We exclusively use the term "tour length" rather than "cost" for these demonstrations to maintain precise language.

For node scaling in 2D TSP, we simply show linear growth for both the expectation and variance of random tour length. The Beardwood–Halton–Hammersley Theorem [25] already shows that expected optimal tour length grows proportionally to $\sqrt{n}$ as $n \to \infty$.

For spatial dimension scaling with a fixed number of nodes, we show that expected random tour length is strictly upper bounded by growth proportional to $\sqrt{d}$. We then show that expected random tour length itself grows proportionally to $\sqrt{d}$ as $d \to \infty$ while random tour length variance and random tour suboptimality both approach a constant value.

Lemma 1. *Expected random tour length grows linearly w.r.t.* n.

Proof. Assume n node coordinates are sampled within a unit square such that each coordinate dimension is i.i.d. randomly sampled $x_i \in [0,1]; \forall i \in \{1,2\}$. Let $\chi_i^{(p,q)} \in [-1,1]$ be the random variable describing the difference between nodes p and q along the i^{th} axis:

$$\chi_i^{(p,q)} = x_i^{(p)} - x_i^{(q)}$$

The Euclidean norm between nodes p and q is then

$$\|X\|_2^{(p,q)} = \sqrt{\left(\chi_1^{(p,q)}\right)^2 + \left(\chi_2^{(p,q)}\right)^2}$$

and we describe the full length of a random tour through each sampled point as

$$\|X\|_2^T = \sum_{k=1}^{n} \|X\|_2^{(p_k,q_k)} \quad \text{s.t. } p_k = q_{k-1},\ p_1 = q_n$$

Note that each $\|X\|_2^{(p_k,q_k)}$ variable shares sampled coordinates with their adjacent edges in the sum, meaning $\|X\|_2^T$ is *not* a sum of independent variables. But these random edge norms are still identically distributed given that the choice of starting node p_1 is arbitrary. Thus,

$$\mathbb{E}\left[\|X\|_2^T\right] = \sum_{k=1}^{n} \mathbb{E}\left[\|X\|_2^{(p_k,q_k)}\right] = n\,\mathbb{E}\left[\|X\|_2^{(p,q)}\right] = n\,\mu_{\|X\|} \tag{1}$$

where $\mu_{\|X\|}$ is the expected Euclidean distance between arbitrary nodes p and q. ■

For the uniform coordinate sampling used in this paper, evaluating $\mu_{\|X\|}$ analytically involves a cumbersome quadruple integral for which several informal solutions can be found online. The exact solution is

$$\mu_{\|X\|} = \frac{2 + \sqrt{2} + 5\ln(\sqrt{2}+1)}{15} \approx 0.521$$

which our empirical fit approximates with trivial error.

Lemma 2. *Variance of random tour length grows linearly w.r.t.* n.

Proof. Because $\|X\|_2^T$ is *not* a sum of independent random variables,

$$\mathrm{Var}\left[\|X\|_2^T\right] \neq \sum_{k=1}^{n} \mathrm{Var}\left[\|X\|_2^{(p_k,q_k)}\right]$$

This slightly complicates the proof. However, we can make use of the fact that dependencies are limited to adjacent edge variables. We start with the following equation:

$$\mathrm{Var}\left[\|X\|_2^T\right] = \mathrm{Var}\left[\sum_{k=1}^{n} \|X\|_2^{(p_k,q_k)}\right] = \sum_{k=1}^{n}\sum_{j=1}^{n} \mathrm{Cov}\left(\|X\|_2^{(p_k,q_k)}, \|X\|_2^{(p_j,q_j)}\right)$$

which is true for any sum of random variables.[11] Since $\|X\|_2^{(p_k,q_k)}$ only depends on $\|X\|_2^{(p_{k-1},q_{k-1})}$ and $\|X\|_2^{(p_{k+1},q_{k+1})}$ (with circular indexing for edges $k \in \{1, n\}$), all non-adjacent edge combinations have zero covariance. Thus, the previous expression simplifies to

$$\begin{aligned}\sum_{k=1}^{n} \Big[&\mathrm{Cov}\left(\|X\|_2^{(p_k,q_k)}, \|X\|_2^{(p_{k-1},q_{k-1})}\right) \\ &+ \mathrm{Cov}\left(\|X\|_2^{(p_k,q_k)}, \|X\|_2^{(p_k,q_k)}\right) \\ &+ \mathrm{Cov}\left(\|X\|_2^{(p_k,q_k)}, \|X\|_2^{(p_{k+1},q_{k+1})}\right)\Big]\end{aligned}$$

Given that covariances between adjacent edge lengths are identically distributed, we obtain

$$\begin{aligned}\mathrm{Var}\left[\|X\|_2^T\right] &= \sum_{k=1}^{n}\left[\mathrm{Var}\left(\|X\|_2^{(p_k,q_k)}\right) + 2\,\mathrm{Cov}\left(\|X\|_2^{(p_k,q_k)}, \|X\|_2^{(p_{k+1},q_{k+1})}\right)\right] \\ &= n\left[\mathrm{Var}\left(\|X\|_2^{(p,q)}\right) + 2\,\mathrm{Cov}\left(\|X\|_2^{(p,q)}, \|X\|_2^{(q,s)}\right)\right] \\ &= n\left(\xi^{(p,q,s)}\right)^2\end{aligned} \tag{2}$$

where (p, q, s) denotes an arbitrary 3-node sequence in the tour with edges (p, q) and (q, s), and $\left(\xi^{(p,q,s)}\right)^2$ is defined as the summation term obtained above. ■

Hence, the standard deviation of random tour length grows proportionally to $\sqrt{n}$, which our empirical results closely approximate (Figure 20 top left, $\alpha \approx 0.497$).

We could not find existing solutions for $\xi^{(p,q,s)}$ and make no attempt to evaluate it ourselves. But our empirical fit suggests $\xi^{(p,q,s)} \approx 0.276$ for uniform coordinate sampling.

Lemma 3. *Expected random tour length w.r.t. d is upper bounded by a function proportional to $\sqrt{d}$.*

Proof. In a unit hypercube of d spatial dimensions consider a fixed number of n nodes i.i.d. randomly sampled as previously described. First, we obtain the expected value of the square of the high-dimensional Euclidean norm between two random nodes:

$$\mathbb{E}\left[\left(\|X\|_2^{(p,q)}\right)^2\right] = \mathbb{E}\left[\sum_{k=1}^{d}\left(\chi_k^{(p,q)}\right)^2\right] = d\;\mathbb{E}\left[\left(\chi^{(p,q)}\right)^2\right] = d\,\mu_{\chi^2}$$

where $\mu_{\chi^2} \in [0, 1]$ is the expected squared difference between p and q along an arbitrary spatial dimension axis. From the concave form of Jensen's Inequality, it follows that

$$\mathbb{E}\left[\|X\|_2^{(p,q)}\right] = \mathbb{E}\left[\sqrt{\left(\|X\|_2^{(p,q)}\right)^2}\right] \le \sqrt{\mathbb{E}\left[\left(\|X\|_2^{(p,q)}\right)^2\right]} = \sqrt{d\,\mu_{\chi^2}}$$

Thus,

$$\mathbb{E}\left[\|X\|_2^{(p,q)}\right] \le \sqrt{d\,\mu_{\chi^2}}$$

Note that $\sqrt{d}$ is the hypercube's maximal Euclidean norm, measuring from one corner to its farthest opposing corner. Hence the expected Euclidean distance between two randomly sampled nodes

[11] Derived from the equality $\mathrm{Var}\left[\sum_{i=1}^{n} X_i\right] = \mathbb{E}\left[\left(\sum_{i=1}^{n} X_i\right)^2\right] - \mathbb{E}\left[\sum_{i=1}^{n} X_i\right]^2$

is bounded below that maximal length scaled by $\sqrt{\mu_{\chi^2}}$. Since Equation 1 generalizes to higher dimensions, extending this result to full tour length is simple:

$$\mathbb{E}\left[\|X\|_2^T\right] = n\ \mathbb{E}\left[\|X\|_2^{(p,q)}\right] \leq n\sqrt{d\,\mu_{\chi^2}}$$

So,

$$\mathbb{E}\left[\|X\|_2^T\right] \leq n\sqrt{\mu_{\chi^2}\,d}$$

■

If we wish to specify this bound for the uniform coordinate sampling used in this paper, it is straightforward to obtain μ_{χ^2} directly from the continuous definition of expectation. Let f be a random variable's probability density function and ρ and ω the integrand variables for $x_i^{(p)}$ and $x_i^{(q)}$, respectively. Then,

$$\mu_{\chi^2} = \mathbb{E}\left[\left(\chi^{(p,q)}\right)^2\right] = \int_0^1 \int_0^1 f(\omega) f(\rho)(\rho-\omega)^2\, d\rho\, d\omega$$

Since $f(\rho)$ and $f(\omega)$ are uniform between 0 and 1, this simplifies to

$$\mu_{\chi^2} = \int_0^1 \int_0^1 (\rho-\omega)^2\, d\rho\, d\omega = \int_0^1 \left(\omega^2 - \omega + \frac{1}{3}\right) d\omega = \frac{1}{6}$$

Thus, for any TSP problem distribution in this paper,

$$\mathbb{E}\left[\|X\|_2^T\right] \leq n\sqrt{\frac{d}{6}}$$

We plot the difference between this upper bound and our empirical dimension-scaling results in Figure 19. Interestingly, this gap also forms a smooth power law.

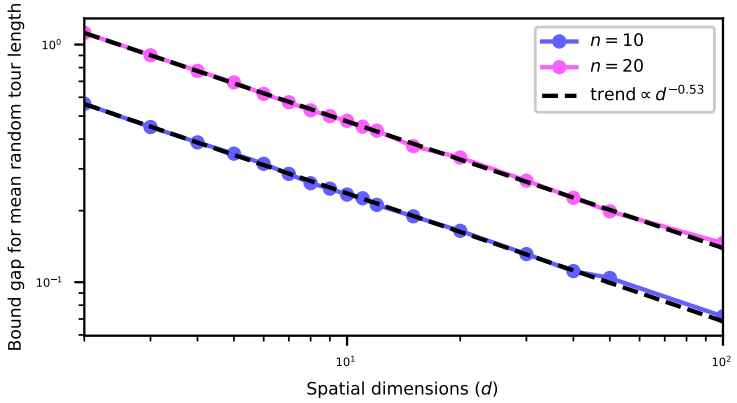

Figure 19: Upper bound for expected random tour length subtracted by observed means for both the 10-node and 20-node spatial dimension scaling experiments. Means evaluate 128,000 and 64,000 samples, respectively. This gap below the upper bound asymptotically decays toward zero (provably). Between node scales, decay rates are both close to square root power ($\alpha \approx 0.53$) so increasing the number of nodes simply increases the proportionality constant from roughly 0.82 to 1.62. This is notably close to the proportionality of node increase (doubling).

Theorem 4. *Expected random tour length is proportional to $\sqrt{d}$ as $d \to \infty$.*

Proof. Extending existing analysis, it is simple to show that the previous upper bound becomes arbitrarily tight in the limit as $d \to \infty$. Specifying the Euclidean norm case[12] for the proof of Lemma 1 in François et al. [28] (Section 5.1.1), it directly follows that

$$\lim_{d\to\infty} \frac{\mathbb{E}\left[\|X\|_2^{(p,q)}\right]}{\sqrt{d}} = \sqrt{\mu_{\chi^2}} \tag{3}$$

[12] $p = 2$ in the cited work. Our usage of p as the norm starting node is unrelated.

Borrowing Equation 1 once again, for the full tour length we obtain

$$\lim_{d\to\infty} \frac{\mathbb{E}\left[\|X\|_2^T\right]}{\sqrt{d}} = \lim_{d\to\infty} \frac{n\,\mathbb{E}\left[\|X\|_2^{(p,q)}\right]}{\sqrt{d}} = n \lim_{d\to\infty} \frac{\mathbb{E}\left[\|X\|_2^{(p,q)}\right]}{\sqrt{d}} = n\sqrt{\mu_{\chi^2}}$$

or

$$\lim_{d\to\infty} \frac{\mathbb{E}\left[\|X\|_2^T\right]}{\sqrt{d}} = n\sqrt{\mu_{\chi^2}}$$

■

For this paper's uniform coordinate sampling distribution, this becomes

$$\lim_{d\to\infty} \frac{\mathbb{E}\left[\|X\|_2^T\right]}{\sqrt{d}} = \frac{n}{\sqrt{6}}$$

implying that the upper bound gaps observed in Figure 19 indeed converge toward zero.

Theorem 5. *Variance of random tour length approaches a constant value as $d \to \infty$.*

Proof. For this limit we extend Lemma 2 in François et al. (proved in Section 5.1.2 of their paper). For the Euclidean norm case, their Lemma shows

$$\lim_{d\to\infty} \operatorname{Var}\left[\|X\|_2^{(p,q)}\right] = \frac{\left(\sigma_{\chi^2}\right)^2}{4\,\mu_{\chi^2}}$$

where σ_{χ^2} is the standard deviation[13] of $\left(\chi^{(p,q)}\right)^2$ for an arbitrary spatial dimension. Since Equation 2 holds for higher dimensions,

$$\begin{aligned}
\operatorname{Var}\left[\|X\|_2^T\right] &= n\left[\operatorname{Var}\left(\|X\|_2^{(p,q)}\right) + 2\operatorname{Cov}\left(\|X\|_2^{(p,q)}, \|X\|_2^{(q,s)}\right)\right] \\
\lim_{d\to\infty} \operatorname{Var}\left[\|X\|_2^T\right] &= n\left[\lim_{d\to\infty} \operatorname{Var}\left(\|X\|_2^{(p,q)}\right) + 2\lim_{d\to\infty} \operatorname{Cov}\left(\|X\|_2^{(p,q)}, \|X\|_2^{(q,s)}\right)\right] \\
&= n\left[\frac{\left(\sigma_{\chi^2}\right)^2}{4\,\mu_{\chi^2}} + 2\lim_{d\to\infty} \operatorname{Cov}\left(\|X\|_2^{(p,q)}, \|X\|_2^{(q,s)}\right)\right] \qquad (4)
\end{aligned}$$

We now follow reasoning analogous to François et al.'s initial steps proving Lemma 2 of their work, but for covariance instead of variance. By definition,

$$\operatorname{Cov}\left(\|X\|_2^{(p,q)}, \|X\|_2^{(q,s)}\right) = \mathbb{E}\left[\left(\|X\|_2^{(p,q)} - \mathbb{E}\left[\|X\|_2^{(p,q)}\right]\right)\left(\|X\|_2^{(q,s)} - \mathbb{E}\left[\|X\|_2^{(q,s)}\right]\right)\right]$$

Thus, using the Fatou–Lebesgue theorem to move the limit inside expectation,

$$\begin{aligned}
&\lim_{d\to\infty} \operatorname{Cov}\left(\|X\|_2^{(p,q)}, \|X\|_2^{(q,s)}\right) \\
&= \mathbb{E}\left[\lim_{d\to\infty}\left(\left(\|X\|_2^{(p,q)} - \mathbb{E}\left[\|X\|_2^{(p,q)}\right]\right)\left(\|X\|_2^{(q,s)} - \mathbb{E}\left[\|X\|_2^{(q,s)}\right]\right)\right)\right] \qquad (5)
\end{aligned}$$

We may rewrite the first product term as

$$\|X\|_2^{(p,q)} - \mathbb{E}\left[\|X\|_2^{(p,q)}\right] = \frac{\left(\|X\|_2^{(p,q)}\right)^2 - \mathbb{E}\left[\|X\|_2^{(p,q)}\right]^2}{\|X\|_2^{(p,q)} - \mathbb{E}\left[\|X\|_2^{(p,q)}\right]} = \frac{\frac{\left(\|X\|_2^{(p,q)}\right)^2 - \mathbb{E}\left[\|X\|_2^{(p,q)}\right]^2}{\sqrt{d}}}{\frac{\|X\|_2^{(p,q)}}{\sqrt{d}} + \frac{\mathbb{E}\left[\|X\|_2^{(p,q)}\right]}{\sqrt{d}}} \qquad (6)$$

Referencing Equation 3, along with François et al.'s intermediate step 1 result from their Lemma 1 proof, we know the limit of the denominator almost surely approaches a constant:

$$\lim_{d\to\infty}\left(\frac{\|X\|_2^{(p,q)}}{\sqrt{d}} + \frac{\mathbb{E}\left[\|X\|_2^{(p,q)}\right]}{\sqrt{d}}\right) = \lim_{d\to\infty}\frac{\|X\|_2^{(p,q)}}{\sqrt{d}} + \lim_{d\to\infty}\frac{\mathbb{E}\left[\|X\|_2^{(p,q)}\right]}{\sqrt{d}}$$

[13] François et al. at first describe σ_{χ^2} as variance, but later in their proof show it to be standard deviation.

and

$$\mathbb{P}\left[\lim_{d\to\infty}\frac{\|X\|_2^{(p,q)}}{\sqrt{d}}=\sqrt{\mu_{\chi^2}}\right]=1$$

so

$$\mathbb{P}\left[\lim_{d\to\infty}\left(\frac{\|X\|_2^{(p,q)}}{\sqrt{d}}+\frac{\mathbb{E}\left[\|X\|_2^{(p,q)}\right]}{\sqrt{d}}\right)=\sqrt{\mu_{\chi^2}}+\sqrt{\mu_{\chi^2}}=2\sqrt{\mu_{\chi^2}}\right]=1$$

Meanwhile, the limit of the numerator of Equation 6 is simplified using Equation 3 again. First, note that

$$\begin{aligned}\left(\|X\|_2^{(p,q)}\right)^2-\mathbb{E}\left[\|X\|_2^{(p,q)}\right]^2&=\sum_{k=1}^{d}\left(\chi_k^{(p,q)}\right)^2-\mathbb{E}\left[\|X\|_2^{(p,q)}\right]^2\\&=\sum_{k=1}^{d}\left(\left(\chi_k^{(p,q)}\right)^2-\frac{\mathbb{E}\left[\|X\|_2^{(p,q)}\right]^2}{d}\right)\end{aligned}$$

and since

$$\lim_{d\to\infty}\frac{\mathbb{E}\left[\|X\|_2^{(p,q)}\right]^2}{d}=\lim_{d\to\infty}\left(\frac{\mathbb{E}\left[\|X\|_2^{(p,q)}\right]}{\sqrt{d}}\right)^2=\left(\lim_{d\to\infty}\frac{\mathbb{E}\left[\|X\|_2^{(p,q)}\right]}{\sqrt{d}}\right)^2=\mu_{\chi^2}$$

taking the limit and substituting yields

$$\begin{aligned}\lim_{d\to\infty}\left(\left(\|X\|_2^{(p,q)}\right)^2-\mathbb{E}\left[\|X\|_2^{(p,q)}\right]^2\right)&=\lim_{d\to\infty}\sum_{k=1}^{d}\left(\left(\chi_k^{(p,q)}\right)^2-\frac{\mathbb{E}\left[\|X\|_2^{(p,q)}\right]^2}{d}\right)\\&=\lim_{d\to\infty}\sum_{k=1}^{d}\left(\left(\chi_k^{(p,q)}\right)^2-\mu_{\chi^2}\right)\end{aligned}$$

Returning to Equation 5, notice that all analysis from Equation 6 up until this point produces analogous results for the second product term $\left(\|X\|_2^{(q,s)}-\mathbb{E}\left[\|X\|_2^{(q,s)}\right]\right)$ since $\|X\|_2^{(p,q)}$ and $\|X\|_2^{(q,s)}$ are identically distributed. Thus,

$$\begin{aligned}&\lim_{d\to\infty}\left(\left(\|X\|_2^{(p,q)}-\mathbb{E}\left[\|X\|_2^{(p,q)}\right]\right)\left(\|X\|_2^{(q,s)}-\mathbb{E}\left[\|X\|_2^{(q,s)}\right]\right)\right)\\&=\lim_{d\to\infty}\left(\frac{\frac{\left(\|X\|_2^{(p,q)}\right)^2-\mathbb{E}\left[\|X\|_2^{(p,q)}\right]^2}{\sqrt{d}}}{\frac{\|X\|_2^{(p,q)}}{\sqrt{d}}+\frac{\mathbb{E}\left[\|X\|_2^{(p,q)}\right]}{\sqrt{d}}}\,\frac{\frac{\left(\|X\|_2^{(q,s)}\right)^2-\mathbb{E}\left[\|X\|_2^{(q,s)}\right]^2}{\sqrt{d}}}{\frac{\|X\|_2^{(q,s)}}{\sqrt{d}}+\frac{\mathbb{E}\left[\|X\|_2^{(q,s)}\right]}{\sqrt{d}}}\right)\\&=\lim_{d\to\infty}\left(\frac{\left(\left(\|X\|_2^{(p,q)}\right)^2-\mathbb{E}\left[\|X\|_2^{(p,q)}\right]^2\right)\left(\left(\|X\|_2^{(q,s)}\right)^2-\mathbb{E}\left[\|X\|_2^{(q,s)}\right]^2\right)}{d}}{\left(\frac{\|X\|_2^{(p,q)}}{\sqrt{d}}+\frac{\mathbb{E}\left[\|X\|_2^{(p,q)}\right]}{\sqrt{d}}\right)\left(\frac{\|X\|_2^{(q,s)}}{\sqrt{d}}+\frac{\mathbb{E}\left[\|X\|_2^{(q,s)}\right]}{\sqrt{d}}\right)}\right)\end{aligned}$$

and with probability 1,

$$\begin{aligned}&=\frac{\lim_{d\to\infty}\frac{\left(\left(\|X\|_2^{(p,q)}\right)^2-\mathbb{E}\left[\|X\|_2^{(p,q)}\right]^2\right)\left(\left(\|X\|_2^{(q,s)}\right)^2-\mathbb{E}\left[\|X\|_2^{(q,s)}\right]^2\right)}{d}}{\lim_{d\to\infty}\left(\frac{\|X\|_2^{(p,q)}}{\sqrt{d}}+\frac{\mathbb{E}\left[\|X\|_2^{(p,q)}\right]}{\sqrt{d}}\right)\lim_{d\to\infty}\left(\frac{\|X\|_2^{(q,s)}}{\sqrt{d}}+\frac{\mathbb{E}\left[\|X\|_2^{(q,s)}\right]}{\sqrt{d}}\right)}\\&=\frac{\lim_{d\to\infty}\frac{\left(\sum_{k=1}^{d}\left(\left(\chi_k^{(p,q)}\right)^2-\mu_{\chi^2}\right)\right)\left(\sum_{j=1}^{d}\left(\left(\chi_j^{(q,s)}\right)^2-\mu_{\chi^2}\right)\right)}{d}}{4\,\mu_{\chi^2}}\end{aligned}\tag{7}$$

Here, the numerator differs slightly from that in the proof from François et al. given we are proving for covariance rather than variance. But we can still apply reasoning analogous to their remaining steps. By the definition of covariance, and since $\mathbb{E}\left[\sum_{k=1}^{d}\left(\left(\chi_k^{(p,q)}\right)^2-\mu_{\chi^2}\right)\right]=0$,

$$\begin{aligned}
&\operatorname{Cov}\left(\sum_{k=1}^{d}\left(\left(\chi_k^{(p,q)}\right)^2-\mu_{\chi^2}\right),\sum_{j=1}^{d}\left(\left(\chi_j^{(q,s)}\right)^2-\mu_{\chi^2}\right)\right)\\
&=\mathbb{E}\left[\left(\sum_{k=1}^{d}\left(\left(\chi_k^{(p,q)}\right)^2-\mu_{\chi^2}\right)\right)\left(\sum_{j=1}^{d}\left(\left(\chi_j^{(q,s)}\right)^2-\mu_{\chi^2}\right)\right)\right]\\
&\quad-\mathbb{E}\left[\sum_{k=1}^{d}\left(\left(\chi_k^{(p,q)}\right)^2-\mu_{\chi^2}\right)\right]\mathbb{E}\left[\sum_{j=1}^{d}\left(\left(\chi_j^{(q,s)}\right)^2-\mu_{\chi^2}\right)\right]\\
&=\mathbb{E}\left[\left(\sum_{k=1}^{d}\left(\left(\chi_k^{(p,q)}\right)^2-\mu_{\chi^2}\right)\right)\left(\sum_{j=1}^{d}\left(\left(\chi_j^{(q,s)}\right)^2-\mu_{\chi^2}\right)\right)\right]-0
\end{aligned}$$

Thus,

$$\begin{aligned}
&\mathbb{E}\left[\left(\sum_{k=1}^{d}\left(\left(\chi_k^{(p,q)}\right)^2-\mu_{\chi^2}\right)\right)\left(\sum_{j=1}^{d}\left(\left(\chi_j^{(q,s)}\right)^2-\mu_{\chi^2}\right)\right)\right]\\
&=\operatorname{Cov}\left(\sum_{k=1}^{d}\left(\left(\chi_k^{(p,q)}\right)^2-\mu_{\chi^2}\right),\sum_{j=1}^{d}\left(\left(\chi_j^{(q,s)}\right)^2-\mu_{\chi^2}\right)\right)\\
&=\sum_{k=1}^{d}\sum_{j=1}^{d}\operatorname{Cov}\left(\left(\left(\chi_k^{(p,q)}\right)^2-\mu_{\chi^2}\right),\left(\left(\chi_j^{(q,s)}\right)^2-\mu_{\chi^2}\right)\right)
\end{aligned}$$

and since coordinates are independently sampled between dimensions,

$$=\sum_{k=1}^{d}\operatorname{Cov}\left(\left(\left(\chi_k^{(p,q)}\right)^2-\mu_{\chi^2}\right),\left(\left(\chi_k^{(q,s)}\right)^2-\mu_{\chi^2}\right)\right)$$

With the observation $\mathbb{E}\left[\left(\chi_k^{(p,q)}\right)^2-\mu_{\chi^2}\right]=0$ alongside another recursion through the definition of covariance, we obtain

$$\begin{aligned}
&\sum_{k=1}^{d}\operatorname{Cov}\left(\left(\left(\chi_k^{(p,q)}\right)^2-\mu_{\chi^2}\right),\left(\left(\chi_k^{(q,s)}\right)^2-\mu_{\chi^2}\right)\right)\\
&=\sum_{k=1}^{d}\left(\mathbb{E}\left[\left(\left(\chi_k^{(p,q)}\right)^2-\mu_{\chi^2}\right)\left(\left(\chi_k^{(q,s)}\right)^2-\mu_{\chi^2}\right)\right]\right.\\
&\qquad\left.-\mathbb{E}\left[\left(\chi_k^{(p,q)}\right)^2-\mu_{\chi^2}\right]\mathbb{E}\left[\left(\chi_k^{(q,s)}\right)^2-\mu_{\chi^2}\right]\right)\\
&=\sum_{k=1}^{d}\left(\mathbb{E}\left[\left(\left(\chi_k^{(p,q)}\right)^2-\mu_{\chi^2}\right)\left(\left(\chi_k^{(q,s)}\right)^2-\mu_{\chi^2}\right)\right]-0\right)\\
&=d\left(\varphi^{(p,q,s)}\right)^2
\end{aligned}$$

since the product inside the expected value is identically distributed for each summation term over d. We define this summation term as $\left(\varphi^{(p,q,s)}\right)^2$. Condensing, we arrive at

$$\mathbb{E}\left[\left(\sum_{k=1}^{d}\left(\left(\chi_k^{(p,q)}\right)^2-\mu_{\chi^2}\right)\right)\left(\sum_{j=1}^{d}\left(\left(\chi_j^{(q,s)}\right)^2-\mu_{\chi^2}\right)\right)\right]=d\left(\varphi^{(p,q,s)}\right)^2 \tag{8}$$

Returning to Equation 5, we can now see that with probability 1,

$$
\begin{aligned}
&\lim_{d\to\infty} \operatorname{Cov}\left(\|X\|_2^{(p,q)}, \|X\|_2^{(q,s)}\right) \\
&= \mathbb{E}\left[\lim_{d\to\infty}\left(\left(\|X\|_2^{(p,q)} - \mathbb{E}\left[\|X\|_2^{(p,q)}\right]\right)\left(\|X\|_2^{(q,s)} - \mathbb{E}\left[\|X\|_2^{(q,s)}\right]\right)\right)\right] \\
&= \mathbb{E}\left[\frac{\lim_{d\to\infty} \frac{\left(\sum_{k=1}^{d}\left(\left(\chi_k^{(p,q)}\right)^2 - \mu_{\chi^2}\right)\right)\left(\sum_{j=1}^{d}\left(\left(\chi_j^{(q,s)}\right)^2 - \mu_{\chi^2}\right)\right)}{d}}{4\,\mu_{\chi^2}}\right] \quad \text{via Eq. 7} \\
&= \frac{\lim_{d\to\infty} \frac{\mathbb{E}\left[\left(\sum_{k=1}^{d}\left(\left(\chi_k^{(p,q)}\right)^2 - \mu_{\chi^2}\right)\right)\left(\sum_{j=1}^{d}\left(\left(\chi_j^{(q,s)}\right)^2 - \mu_{\chi^2}\right)\right)\right]}{d}}{4\,\mu_{\chi^2}} \quad \text{via Fatou–Lebesgue theorem} \\
&= \frac{\lim_{d\to\infty} \frac{d\left(\varphi^{(p,q,s)}\right)^2}{d}}{4\,\mu_{\chi^2}} \quad \text{via Eq. 8} \\
&= \frac{\left(\varphi^{(p,q,s)}\right)^2}{4\,\mu_{\chi^2}}
\end{aligned}
$$

Thus,

$$
\mathbb{P}\left[\lim_{d\to\infty} \operatorname{Cov}\left(\|X\|_2^{(p,q)}, \|X\|_2^{(q,s)}\right) = \frac{\left(\varphi^{(p,q,s)}\right)^2}{4\,\mu_{\chi^2}}\right] = 1
$$

Referencing step 2 in François et al.'s proof of their Lemma 1, the same reasoning shows

$$
\lim_{d\to\infty} \operatorname{Cov}\left(\|X\|_2^{(p,q)}, \|X\|_2^{(q,s)}\right) = \frac{\left(\varphi^{(p,q,s)}\right)^2}{4\,\mu_{\chi^2}}
$$

Finally, returning to Equation 4 obtains the desired result:

$$
\begin{aligned}
\lim_{d\to\infty} \operatorname{Var}\left[\|X\|_2^T\right] &= n\left(\frac{\left(\sigma_{\chi^2}\right)^2}{4\,\mu_{\chi^2}} + 2\lim_{d\to\infty} \operatorname{Cov}\left(\|X\|_2^{(p,q)}, \|X\|_2^{(q,s)}\right)\right) \\
&= n\left(\frac{\left(\sigma_{\chi^2}\right)^2}{4\,\mu_{\chi^2}} + 2\frac{\left(\varphi^{(p,q,s)}\right)^2}{4\,\mu_{\chi^2}}\right) \\
&= n\left(\frac{\left(\sigma_{\chi^2}\right)^2 + 2\left(\varphi^{(p,q,s)}\right)^2}{4\,\mu_{\chi^2}}\right)
\end{aligned}
$$

■

We can then evaluate $\left(\sigma_{\chi^2}\right)^2$ and $\left(\varphi^{(p,q,s)}\right)^2$ for the uniform coordinate sampling used in this paper, like we did for μ_{χ^2} after proving Lemma 3:

$$
\begin{aligned}
\left(\sigma_{\chi^2}\right)^2 &= \int_0^1\int_0^1 f(\omega)f(\rho)((\rho-\omega)^2 - \mu_{\chi^2})^2 d\rho\, d\omega = \int_0^1\int_0^1 \left((\rho-\omega)^2 - \frac{1}{6}\right)^2 d\rho\, d\omega = \frac{7}{180} \\
\left(\varphi^{(p,q,s)}\right)^2 &= \int_0^1\int_0^1\int_0^1 f(\tau)f(\omega)f(\rho)((\rho-\omega)^2 - \mu_{\chi^2})((\omega-\tau)^2 - \mu_{\chi^2}) d\rho\, d\omega\, d\tau \\
&= \int_0^1\int_0^1\int_0^1 \left((\rho-\omega)^2 - \frac{1}{6}\right)\left((\omega-\tau)^2 - \frac{1}{6}\right) d\rho\, d\omega\, d\tau \\
&= \frac{1}{180}
\end{aligned}
$$

Plugging in values yields:

$$\lim_{d\to\infty} \operatorname{Var}\left[\|X\|_2^T\right] = \frac{3}{40}n$$

Evaluating for 10 and 20 nodes, respectively, we obtain variance limits of $\frac{3}{4}$ and $\frac{3}{2}$, translating to standard deviation limits of approximately 0.87 and 1.22. Our empirical results for standard deviation of random tour length in Figure 20 very closely align.

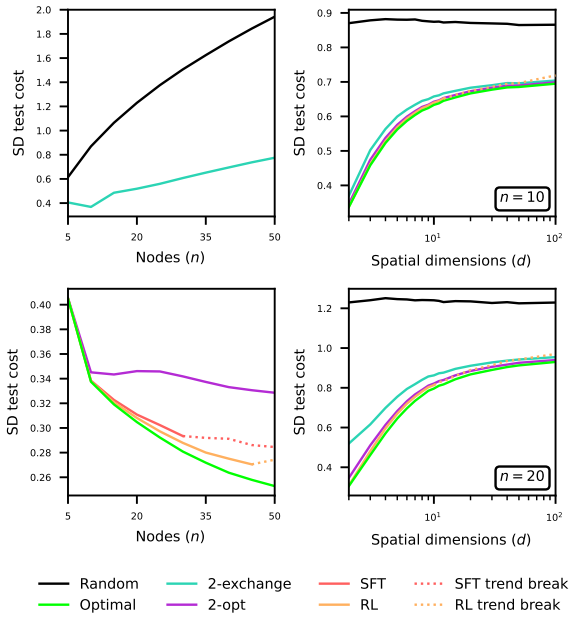

Figure 20: Standard deviation (SD) of cost over TSP node and spatial dimension scaling, shown for random, optimal, and algorithm distributions. Behavior of random tour length variance aligns with our analysis in Lemma 2 and Theorem 5. **Top left:** Tour length SD for random performance and mediocre algorithms like 2-exchange diverge w.r.t. number of nodes. **Bottom left:** Tour length SD for sufficiently near-optimal distributions decreases w.r.t. nodes, implying optimal tours become more similar in length with increasing node density in the representation space (at least for 2D TSP). **Right:** Like random tour suboptimality, random tour variance w.r.t. spatial dimensions is roughly constant over the tested domain and provably constant in the limit (Theorem 5). All other distributions produce an increasing convergent trend.

Theorem 6. *The difference between expected random tour length and expected optimal tour length (random tour suboptimality) approaches a constant value as* $d \to \infty$.

Proof. Here we leverage an existing derivation that bounds the limit of expected absolute contrast for a fixed number of nodes. Taking the norm from an arbitrary query point to each node coordinate, absolute contrast is defined as the maximum difference between norms. Specifying the Euclidean case for Corollary 2 in Aggarwal et al. [27], the bound can be formally described as follows:

$$\lambda \leq \lim_{d\to\infty} \mathbb{E}\left[\max_{u\in Q}\|X\|_2^{(p,u)} - \min_{v\in Q}\|X\|_2^{(p,v)}\right] \leq (n-2)\lambda \tag{9}$$

where node p is our chosen query point,[14] Q is the set of remaining nodes such that $p \notin Q$, and λ is a non-negative constant. This bound applies to any node coordinate distribution satisfying i.i.d. sampling for each dimension.

[14] Aggarwal et al. consistently use the origin as their query point but without loss of generality. Our upper bound term is scaled by $(n-2)$ to account for removing node p from the end point set Q.

Let $\|X\|_2^{T^*}$ be defined as the length of the optimal tour between the sampled nodes, that which incurs minimum Euclidean distance:

$$\|X\|_2^{T^*} = \sum_{k=1}^{n} \|X\|_2^{(p_k, q_k^*)} \quad \text{s.t. } p_k = q_{k-1}^*,\ p_1 = q_n^*$$

where node q_k^* forms the optimal subsequent edge from node p_k. Random tour suboptimality can then be expressed as $\mathbb{E}\left[\|X\|_2^T - \|X\|_2^{T^*}\right]$, the expected difference between random and optimal tour length.

First, we extend Aggarwal et al.'s Corollary 2 to upper bound the limit of random tour suboptimality. Let $q \in Q$ be a randomly selected node and let $q^* \in Q$ form (p, q^*), an optimal edge.[15] It follows that

$$\begin{aligned}
\|X\|_2^{(p,q)} - \|X\|_2^{(p,q^*)} &\leq \max_{u \in Q} \|X\|_2^{(p,u)} - \min_{v \in Q} \|X\|_2^{(p,v)} \\
\lim_{d \to \infty} \mathbb{E}\left[\|X\|_2^{(p,q)} - \|X\|_2^{(p,q^*)}\right] &\leq \lim_{d \to \infty} \mathbb{E}\left[\max_{u \in Q} \|X\|_2^{(p,u)} - \min_{v \in Q} \|X\|_2^{(p,v)}\right] \\
&\leq (n-2)\lambda \quad \text{via Rel. 9}
\end{aligned}$$

Repeating this reasoning with each node taking the perspective as the query point p then summing all inequalities yields:

$$\sum_{k=1}^{n} \lim_{d \to \infty} \mathbb{E}\left[\|X\|_2^{(p_k, q_k)} - \|X\|_2^{(p_k, q_k^*)}\right] \leq n(n-2)\lambda$$

Because we know each limit inside the sum does not diverge,[16] we can bring the limit outside the sum:

$$\begin{aligned}
\lim_{d \to \infty} \sum_{k=1}^{n} \mathbb{E}\left[\|X\|_2^{(p_k, q_k)} - \|X\|_2^{(p_k, q_k^*)}\right] &\leq n(n-2)\lambda \\
\lim_{d \to \infty} \left(\sum_{k=1}^{n} \mathbb{E}\left[\|X\|_2^{(p_k, q_k)}\right] - \sum_{j=1}^{n} \mathbb{E}\left[\|X\|_2^{(p_j, q_j^*)}\right]\right) &\leq n(n-2)\lambda
\end{aligned}$$

Like each random edge, each optimal edge variable $\|X\|_2^{(p_j, q_j^*)}$ is identically distributed given the choice of starting node p_1 is arbitrary. Therefore, the above relation is equivalent to

$$\begin{aligned}
\lim_{d \to \infty} \left(\mathbb{E}\left[\|X\|_2^T\right] - \mathbb{E}\left[\|X\|_2^{T^*}\right]\right) &\leq n(n-2)\lambda \\
\lim_{d \to \infty} \mathbb{E}\left[\|X\|_2^T - \|X\|_2^{T^*}\right] &\leq n(n-2)\lambda
\end{aligned}$$

showing the desired upper bound. From here it is straightforward to show a non-negative lower bound. By the definition of optimality,

$$\begin{aligned}
\|X\|_2^{T^*} &\leq \|X\|_2^T \\
0 &\leq \|X\|_2^T - \|X\|_2^{T^*} \\
0 &\leq \lim_{d \to \infty} \mathbb{E}\left[\|X\|_2^T - \|X\|_2^{T^*}\right]
\end{aligned}$$

Thus,

$$0 \leq \lim_{d \to \infty} \mathbb{E}\left[\|X\|_2^T - \|X\|_2^{T^*}\right] \leq n(n-2)\lambda$$

■

[15] Each node p has two alternatives for q^*, but this choice does not affect the analysis.

[16] We can easily show each limit is lower bounded using the mirroring relation:
$\|X\|_2^{(p,q)} - \|X\|_2^{(p,q^*)} \geq \min_{v \in Q} \|X\|_2^{(p,v)} - \max_{u \in Q} \|X\|_2^{(p,u)}$

In words, in the limit as $d \to \infty$, random tour suboptimality approaches a non-negative constant value. The upper bound is dependent on the number of nodes and, through λ, dependent on the coordinate sampling distribution.

If one can show $\mathbb{E}\left[\|X\|_2^T - \|X\|_2^{T^*}\right] > 0$ for arbitrary spatial dimensions, we could obtain the stricter bound

$$\epsilon \leq \lim_{d\to\infty} \mathbb{E}\left[\|X\|_2^T - \|X\|_2^{T^*}\right] \leq n(n-2)\lambda$$

for some $0 < \epsilon \leq n(n-2)\lambda$ where ϵ may be a function of n. For uniform coordinate sampling, this seems almost certainly true. Using Markov's inequality, it would be sufficient to show that for arbitrary spatial dimensions

$$\mathbb{P}\left[\left(\|X\|_2^T - \|X\|_2^{T^*}\right) \geq \epsilon\right] > 0$$

While intuitive, we forgo attempting a proof.