How Do Transformers Align Tokens?

Hadi Daneshmand
Department of Computer Science
University of Virginia
dhadi@virginia.edu

Abstract

How can language models align words in translated sentences with different syntactic structures? Can they compute edit distances—or even sort arbitrary sequences? These tasks are examples of *assignment problems*. We prove a carefully engineered prompt enables a transformer to approximate the solution of assignment. This prompt induces attention layers to simulate gradient descent on the dual objective of assignment. We establish an **explicit approximation bound** that improves with transformer depth. A striking implication is that a single transformer can sort inputs of **arbitrary** length—proving a form of *out-of-distribution generalization*.

1 Introduction

Token matching is a core operation in natural language processing. Consider the example of machine translation: “I really liked this book.” becomes “Ce livre m’a beaucoup plu.” Although “Ce livre” (meaning “this book”) appears first in French, it is the last phrase in English. How does a language model align words to their translations? Assignment (or matching) is a fundamental computational task with broad applications in data mining [Peyré et al., 2019], bioinformatics [Schiebinger et al., 2019], and natural language processing. Assignment includes computing the *edit distance* [Munkres, 1957] and *sorting* [Brockett, 1991].

Mathematically, the *assignment problem* is formulated as finding a permutation matrix P_* that matches two sets of d-dimensional points, $\{x_1, \dots, x_n\}$ and $\{y_1, \dots, y_n\}$, with minimal cost as

$$P_* := \arg \min_{P \in \mathbb{R}^{n \times n}} \sum_{ij} P_{ij} \|y_i - x_j\|_2^2 \quad \text{subject to } P \text{ being a permutation matrix.} \tag{1}$$

Motivated by the remarkable performance of language models, we ask: can transformers solve assignment problems via *in-context learning*?

Recent work highlights the ability of transformers to perform *in-context learning* [Brown, 2020]: inference based solely on data presented in the prompt, without parameter updates. For the assignment problem, in-context learning can be formulated as

$$\text{Prompt: } x_1, y_1, \dots, x_n, y_n \quad \Rightarrow \quad \text{Output: } P_* \begin{bmatrix} x_1 & \dots & x_n \end{bmatrix}^\top . \tag{2}$$

The matrix product on right aligns the x_j’s to their corresponding y_i’s using P_*. A model that learns in-context must generalize to arbitrary n and all input sets $\{x_1, \dots, x_n\}$. As an example, one can prompt GPT-4 with: Prompt: sort$(2, 1, 4, 3) \Rightarrow$ Output: $(1, 2, 3, 4)$. But can GPT-4 sort arbitrary lists of any length?

We prove that transformers are capable of in-context assignment. Specifically, we show that *softmax attention* can implement gradient descent (with adaptive step sizes) on the dual objective of assignment with entropy regularization. Each transformer layer with two attention heads simulates one gradient descent step, and stacking layers simulates multiple iterations. This yields an explicit approximation

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

bound: for any two sets of n points in $\mathbb{R}^d$, a transformer can estimate the solution up to

$$O\left(\frac{n^{3/2}}{\text{depth}^{1/2}}\right)\text{-accuracy for all integers } n. \tag{3}$$

This implies a form of *out-of-distribution generalization* in the number of tokens. Crucially, the result is enabled by careful *prompt engineering*, which allows the transformer to read and write intermediate computations across layers—thereby simulating an algorithm.

2 Motivation

Before delving into the theory, we motivate our study with a striking observation. The following experiment aims to demonstrate the in-context learning capability of transformers on the assignment problem. We generate random data and train a transformer to solve the assignment task by minimizing

$$\min_{\theta} \mathbb{E} \left\| f_\theta\left(\begin{bmatrix} x_1 & y_1 & \blacksquare \\ x_2 & y_2 & \blacksquare \\ \vdots & \vdots & \blacksquare \\ x_n & y_n & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \end{bmatrix}\right) - P_* \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \right\|^2, \tag{4}$$

where expectation is taken over randomly generated inputs $x_1, \ldots, x_n$, and the ■ marks engineered part of the input. For clarity, we defer the training details to Section A. After training with $n = 7$, we observe that the attention weights in the transformer approximate the assignment solution for both $n = 8$ and $n = 9$, as illustrated in Figure 1. This observation suggests a form of *out-of-distribution generalization*, since the model successfully adapts to a significantly different distribution from training data distribution.

Importantly, transformers naturally accept inputs of varying lengths, making them particularly well-suited for tasks involving variable-length sequences such as natural language. This contrasts with other neural architectures like convolutional neural networks, which typically require fixed-size inputs. Our goal is to theoretically investigate how this flexible input structure enables transformers to align tokens of different sizes.

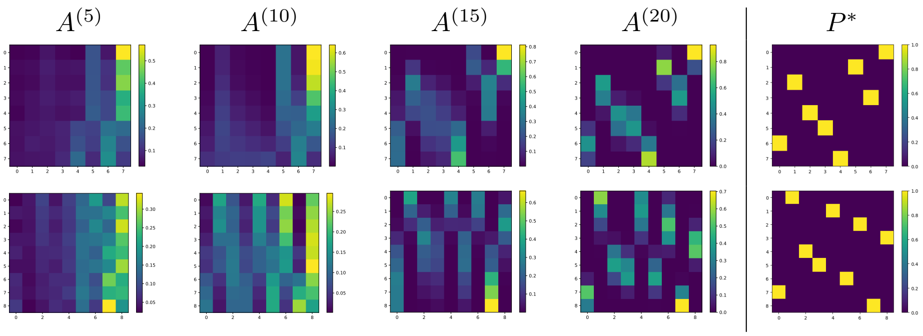


Figure 1: **In-context Assignment after Training.** *Rows:* $n = 8$ and 9; *Columns:* attention matrices $A^{(\ell)}$ defined in (8) for $\ell = 5, 10, 15, 20$. The last columns shows the assignment solution P^* in (1). We observe a good approximation with attenion in the last layer. See Appendix A for details.

3 Preliminaries

Regularized Assignment. Cuturi [2013] proposes a computationally efficient method to approximate P^* based on regularization with entropy:

$$P^*_\lambda := \arg\min_{P \in \mathbb{R}^{n\times n}} \sum_{ij} P_{ij} C_{ij} + \lambda P_{ij} \log(P_{ij}), \text{ subject to } P \text{ is a } \textit{doubly stochastic} \text{ matrix} \tag{5}$$

We will prove self-attention layers can approximate P^*_λ defined above.

Self-attention layers. Attention layers are fundamental building blocks of neural networks, developed over decades of research. We study the mechanism of self-attention layers rely on a convex combination [Vaswani, 2017]. Let $Z \in \mathbb{R}^{n\times d}$. An attention layer is a function denoted by $\text{atten}_\theta : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$ with parameters $\theta := \left[w_k, w_v, w_q \in \mathbb{R}^{d\times d}\right]$ defined as

$$\text{atten}_\theta(Z) = AZw_v, \quad A_{ij} = e^{\langle w_k z_i, w_q z_j\rangle} \left(\sum_{j=1}^n e^{\langle w_k z_i, w_q z_j\rangle}\right)^{-1},$$

where z_i and z_j are rows of Z. The convex combination of data points imposes a local dependency between representations of tokens, i.e. rows of Z.

The Engineered Prompt. We propose a particular input denoted by $Z_0 \in \mathbb{R}^{(n+1)\times(2d+9)}$ to encode the assignment problem:

$$Z_0 = \begin{bmatrix} x_1 & y_1 & \|x_1\|^2 & \|y_1\|^2 & 1_4 & 0_3 \\ x_2 & y_2 & \|x_2\|^2 & \|y_2\|^2 & 1_4 & 0_3 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \\ x_n & y_n & \|x_n\|^2 & \|y_n\|^2 & 1_4 & 0_3 \\ 0 & 0 & 0 & 0 & -v_4/n & 0_3 \end{bmatrix}. \tag{6}$$

The elements in blue are the original prompts, which are sufficient for the assignment. The elements in red are carefully engineered. 1_4 denotes the all-ones 4-dimensional vector, 0_3 denotes the all-zeros 3-dimensional vector, and $v_4 = [0, 0, 0, 1]$.

Transformer. We consider a specific transformer architecture composed of multiple attention and feedforward layers, all connected via residual connections. Let Z_ℓ denote the intermediate representation of the input Z_0 at layer ℓ which obeys the following recurrence

$$Z_{\ell+1} = Z_\ell + \sum_{j=1}^2 \text{atten}_{\theta_j^{(\ell)}}(Z_\ell) B_j^{(\ell)}, \tag{7}$$

where $\theta_j^{(\ell)}$ are parameters of the attention layers, $B_j^{(\ell)} \in \mathbb{R}^{d'\times d'}$ are the weights to combine attention heads. Remarkably, the model has two attention heads.

Notations. Let $Z_0 \in \mathbb{R}^{n\times d}$ denote the input to the transformer. Its representation at layer ℓ is denoted by Z_ℓ. The input matrix Z_0 contains the points $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$ from the assignment problem. Let P^* be the optimal assignment solution, and let P^*_λ be its approximation obtained via entropy regularization. The transformer is parameterized by θ, which includes all attention parameters $\theta_j^{(\ell)}$ at layer ℓ and attention head j. The vectors u_ℓ and v_ℓ denote the gradient descent iterates on the dual objective function L, as defined in Equation (14). For a matrix M, we use the notation $[M]_{i,j}$ to refer to the entry in row i and column j.

4 Provable Assignment

Attention matrices can approximate the entropy-regularized assignment solution, denoted by P^*_λ, as defined in Equation (13). Consider the block of *attention matrices* $A^{(\ell)} \in \mathbb{R}^{n\times n}$, defined by

$$A^{(\ell)}_{ij} = e^{\langle w_k^{(\ell,1)} z_i^{(\ell)}, w_q^{(\ell,1)} z_j^{(\ell)}\rangle} \left(\sum_{j=1}^n e^{\langle w_k^{(\ell,1)} z_i^{(\ell)}, w_q^{(\ell,1)} z_j^{(\ell)}\rangle}\right)^{-1}, \tag{8}$$

where $z_i^{(\ell)}$ is the i-th row of Z_ℓ, representing the token embeddings at layer ℓ, and $w_q^{(\ell,1)}$, $w_k^{(\ell,1)}$ are the query and key weight matrices of attention head 1 at layer ℓ.

We prove the attention matrix $A^{(\ell)}$ converges to the regularized optimal assignment matrix P^*_λ in an appropriate metric for certain choice of parameters of attention layers,.

As discussed, solution P_λ^* in (13) can be written as

$$P_\lambda^* = \text{diag}(p^*)Q\text{diag}(q^*), \quad p^*, q^* \in \mathbb{R}_+^n, \quad Q \in \mathbb{R}_+^{n\times n}, \tag{9}$$

where $\text{diag}(v)$ is a diagonal matrix whose diagonal element in row i is v_i, $\quad Q_{ij} = e^{-\frac{C_{ij}}{\lambda}-1}$, $p_i^* = e^{v_i^*/\lambda}$ and $q_j^* = e^{u_j^*/\lambda}$. It is easy to check that replacing p^* and q^* with cp^* and q^*/c leads to the same matrix P_λ^* for all $c \in \mathbb{R}_+$. Franklin and Lorenz [1989] introduce a metric that accounts for this particular scaling invariance:

$$\mu(q, q') = \log\left(\max_{ij} \frac{q_i q_j'}{q_j q_i'}\right). \tag{10}$$

Remarkably, μ is a metric that satisfies the triangle inequality [Franklin and Lorenz, 1989]. However, μ is not a norm, as $\mu(q, q') = 0$ only implies that there exists a constant c such that $q = cq'$. The next theorem establishes an explicit convergence rate for the attention matrices $A^{(\ell)}$ to P_λ^* in μ.

Theorem 4.1. *There exists a choice of parameters* ***independent from*** n *that ensures* $A^{(\ell)}$ *can be written as* $A^{(\ell)} = diag(p_\ell)Qdiag(q_\ell)$*, where* Q *is defined in* (9) *and* $p_\ell, q_\ell \in \mathbb{R}_+^d$ *obey*

$$\min_{k\le\ell} \max\left\{\mu(p_k, p^*), \mu(q_k, q^*)\right\} \le \frac{36 n^{3/2} e^{r/\lambda}\sqrt{r}}{\sqrt{\ell}(1-\eta)}, \quad \begin{cases} (1/4)r^2 = \|p_1 - p^*\|_2^2 + \|q_1 - q^*\|_2^2, \\ \eta = (\phi(Q)^{1/2} - 1)/(\phi(Q)^{1/2} + 1), \\ \phi(Q) = \max_{ijkl} Q_{ik}Q_{jl}/(Q_{jk}Q_{il}), \end{cases}$$

as long as $\ell \ge 64 n^3 \exp(3r/\lambda) r$.

According to the theorem, the attention matrices converge to P_λ^* at a rate of $O\left(\frac{1}{\text{depth}^{1/2}}\right)$, implying that performance improves with increasing depth. This convergence holds for any integer n. Thus, a transformer can solve the assignment problem for any number of points without changes in parameters. This result mathematically proves transformers are capable of seen generalization in Figure 1.

An application of the last theorem is that transformers can be used to sort lists. As discussed, sorting is a specific instance of assignment problem for $d = 1$ with $y_1 < \cdots < y_n$. Thus, transformer can sort with an error that vanishes with λ. Graves [2014] experimentally demonstrate that the neural Turing machine can sort. Here, we theoretically prove this capability for transformers by establishing an approximation bound.

5 Limitations and Future Works

Our findings open several avenues for future research.

According to Theorem 4.1, a transformer with depth $O(\epsilon^{-2})$ can achieve an $O(\epsilon)$-accurate solution, following the established convergence rate for gradient descent with adaptive step sizes. However, only $O(\log(1/\epsilon))$ Sinkhorn iterations are needed for the same accuracy. We believe this gap arises from a loose convergence analysis, which can be refined in future work.

We prove that a transformer with fixed parameters can solve the assignment problem for any arbitrary n. This striking generalization has practical benefits: it can drastically reduce both training time and memory usage for assignment tasks, since the transformer can be trained on prompts with a constant number of tokens. A natural question arises: what is the minimal value of n sufficient for the model to learn the assignment task?

We theoretically and experimentally demonstrate that prompt engineering is essential for in-context assignment. However, the underlying mechanism of prompt engineering remains understudied in a broader context. Our findings motivate further study of prompt engineering from a computational perspective, highlighting its role in enhancing the computational expressivity of transformers.

Akhnowldgements

We thank Jacob Andreas for his helpful discussions and for inspiring the experiment in Appendix A. We also thank Amir Reiszade for his insightful discussions on the related literature on in-context learning. We acknowledge support from NSF TRIPODS program (award DMS-2022448).

References

Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

Geoffrey Schiebinger, Jian Shu, Marcin Tabaka, Brian Cleary, Vidya Subramanian, Aryeh Solomon, Joshua Gould, Siyan Liu, Stacie Lin, Peter Berube, et al. Optimal-transport analysis of single-cell gene expression identifies developmental trajectories in reprogramming. *Cell*, 176(4):928–943, 2019.

James Munkres. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38, 1957.

Roger W Brockett. Dynamical systems that sort lists, diagonalize matrices, and solve linear programming problems. *Linear Algebra and its applications*, 146:79–91, 1991.

Tom B Brown. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013.

A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Joel Franklin and Jens Lorenz. On the scaling of multidimensional matrices. *Linear Algebra and its applications*, 114:717–735, 1989.

Alex Graves. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Rémi Flamary, Nicolas Courty, Alexandre Gramfort, Mehdi Alaya, Aurélien Boisbunon, Stanislas Chambon, Laetitia Chapel, Adrien Corenflos, Kevin Fatras, Nicolas Fournier, Lucas Gautheron, Nathalie Gayraud, Hicham Janati, Alain Rakotomamonjy, Ievgen Redko, Antoine Rolet, Antoine Schutz, Vivien Seguy, Leo Sellem, and Titouan Vayer. Pot: Python optimal transport. *Journal of Machine Learning Research*, 22(78):1–8, 2021.

Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. *Advances in Neural Information Processing Systems*, 35:30583–30598, 2022.

Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pages 35151–35174. PMLR, 2023.

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? investigations with linear models. *arXiv preprint arXiv:2211.15661*, 2022.

Kwangjun Ahn, Xiang Cheng, Hadi Daneshmand, and Suvrit Sra. Transformers learn to implement preconditioned gradient descent for in-context learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Jiuqi Wang, Ethan Blaser, Hadi Daneshmand, and Shangtong Zhang. Transformers learn temporal difference methods for in-context reinforcement learning. *arXiv preprint arXiv:2405.13861*, 2024.

Artur Back de Luca and Kimon Fountoulakis. Simulation of graph algorithms with looped transformers. *arXiv preprint arXiv:2402.01107*, 2024.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35: 22199–22213, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*, 2022.

Hongjin Su, Jungo Kasai, Chen Henry Wu, Weijia Shi, Tianlu Wang, Jiayi Xin, Rui Zhang, Mari Ostendorf, Luke Zettlemoyer, Noah A Smith, et al. Selective annotation makes language models better few-shot learners. *arXiv preprint arXiv:2209.01975*, 2022.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022.

Jiale Cheng, Xiao Liu, Kehan Zheng, Pei Ke, Hongning Wang, Yuxiao Dong, Jie Tang, and Minlie Huang. Black-box prompt optimization: Aligning large language models without model training. *arXiv preprint arXiv:2311.04155*, 2023.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with" gradient descent" and beam search. *arXiv preprint arXiv:2305.03495*, 2023.

Cho-Jui Hsieh, Si Si, Felix X Yu, and Inderjit S Dhillon. Automatic engineering of long prompts. *arXiv preprint arXiv:2311.10117*, 2023.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. Learning to retrieve prompts for in-context learning. *arXiv preprint arXiv:2112.08633*, 2021.

G Conte, AM Perdon, B Wyman, Roger W Brockett, and Wing Shing Wong. A gradient flow for the assignment problem. In *New Trends in Systems Theory: Proceedings of the Università di Genova-The Ohio State University Joint Conference, July 9–11, 1990*, pages 170–177. Springer, 1991.

Richard Sinkhorn. Diagonal equivalence to matrices with prescribed row and column sums. *The American Mathematical Monthly*, 74(4):402–405, 1967.

Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

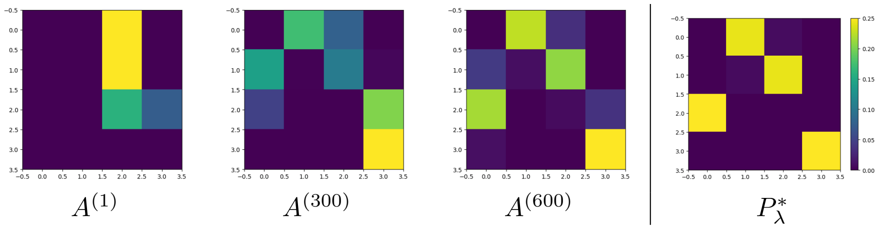
Appendix

A Experiments

Theorems 4.1 and D.1 convey four key insights: (i) Transformers approximate sorting, (ii) their attention matrices converge to P_λ^* from (13), (iii) they solve assignment for any n without parameter adaptation, and (iv) they handle high-dimensional inputs. We validate (i)-(ii) for a particular choice of parameters defined in the proof of Theorem D.1(see the Appendix). We study training in the Appendix A.

Data. $x_1, \ldots, x_n$ are a random permutation of $[1/n, 2/n, \ldots, n/n]$, and we set $y_i = i/n$ in our experiments. The default regularization constant is $\lambda = 0.005$.

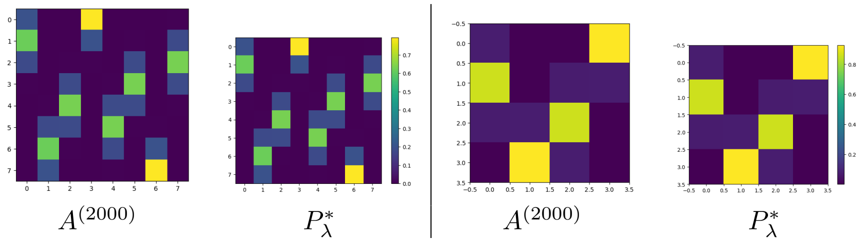
(i) Convergence. Figure 2 illustrates the convergence of the attention matrices $A^{(\ell)}$ to P_λ^*, as established in Theorem 4.1. Notably, the attention matrices retain high rank, contrasting with the rank collapse phenomenon observed in other settings; see the remarks in Section 4.



$A^{(1)}$ $A^{(300)}$ $A^{(600)}$ P_λ^*

Figure 2: **Convergence of attention matrices.** The plotted matrices are $A^{(1)}$, $A^{(300)}$ and $A^{(600)}$ defined in (8). We observe these matrices converge to P_λ^* (the rightmost plot), which is proven by Theorem 4.1.

(ii) Samplesize. Figure 3 illustrates that a single transformer can approximate the solution for different n without changing the parameters.



$A^{(2000)}$ P_λ^* $A^{(2000)}$ P_λ^*

Figure 3: **Sample Size.** left: $n = 8$, right: $n = 4$. The transformer weights remain exactly the same.

We have theoretically and experimentally demonstrated that transformers can solve the assignment problem. But can they learn the assignment directly from data? Figure 1, previously discussed, provides evidence that they can. In this section, we present the details of that experiment and elaborate on the critical role of prompt engineering in assigment.

Training Protocol. The training objective shown in (4) where $f_\theta : \mathbb{R}^{m \times d} \to \mathbb{R}^n$ is the output function of a transformer with parameters θ. To generate outputs, we use an attention layers as

$$f_\theta(Z_0) = [\text{atten}_\theta([Z_\ell]_{1:n,:})]_{:,0:d} \tag{11}$$

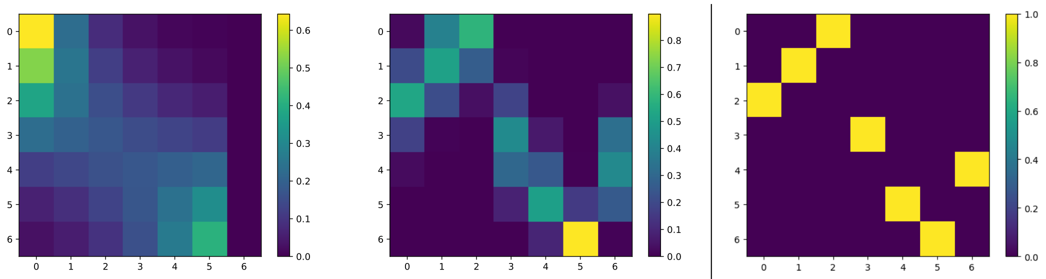
where $[Z_\ell]_{1:n,:}$ denotes the first n rows of Z_ℓ in (7). The above indexing allows us to generate outputs of size $n \times d$. The expectation in (4) is taken over 1000 random samples generated with $n = 7$ (see **Data**). For training, we run 10^4 iterations of Adam [Kingma, 2014] with stepsize 0.001. Parameters are initialized randomly from a Gaussian distribution with variance $1/(2d + 9)$. We set $B_j^{(\ell)} = (1/20) I_{d'}$, and optimize attention parameters $\theta := \{[w_k^{(m,j)}, w_q^{(m,j)}, w_v^{(m,j)}]\}_{m=1}^{\ell}$. Furthermore, we use the reparameterization $Q_\ell := w_k^{(\ell)} (w_q^{(\ell)})^\top$ and optimize Q_ℓ. All experiments are implement in PyTorch [Paszke et al., 2019] and executed on the Google Colab platform using a T4 GPU. We also used POT library [Flamary et al., 2021] to compute P_*.

Results. Figure 1 shows the attention matrices, $A^{(\ell)}$ in (8), across the layers where we observe that these patterns are converging to the optimal solution. This observation validates that transformers iteratively solve the assignment problem across their layers (similar to gradient descent on L). While the the transformer is trained for $n = 7$, we observe a good approximation for $n = 8$ and $n = 9$ after training for $n = 7$. Theorem 4.1 confirms this generalization capability.

The role of prompt engineering. To study the impact of the engineered prompt (6), we remove additional columns in the engineered prompt as

$$Z' = \begin{bmatrix} x_1 & y_1 & 0 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 0 \end{bmatrix} \in \mathbb{R}^{n\times 3}. \tag{12}$$

We train the transformer following the **Training Protocol**. Figure 4 shows that the trained transformer cannot approximate P_* without prompt engineering, but prompt engineering significantly improves the approximation.



without prompt engineering (12) with the engineered prompt (6) optimal solution P_*

Figure 4: **The Significance of Prompt Engineering.** *Left:* Attention pattern in the last layer (see (8)) without prompt engineering. *Middle:* Attention pattern in the last layer with the engineered prompt from (6). *Right:* Optimal solution. The attention pattern approximates P_* when proper prompt engineering is applied. See Section A for details.

B Related works

Our study is motivated by a recent line of research exploring how transformers can perform regression using in-context samples Garg et al. [2022], Von Oswald et al. [2023], Akyürek et al. [2022], Ahn et al. [2024]. Garg et al. [2022] propose analyzing regression tasks encoded directly within the prompts of language models. Von Oswald et al. [2023] show that linear attention mechanisms can implement gradient descent on the mean squared error to solve such tasks. Building on this, Ahn et al. [2024] investigate how data distribution influences the in-context learning behavior of transformers. Their results suggest that data can guide linear attention to learn a wider class of algorithms, including preconditioning strategies for ill-conditioned problems. In this work, we extend these ideas to the assignment problem—a fundamental computation with applications in natural language processing.

The computational power of transformers extends beyond statistical regression. Wang et al. [2024] prove that linear attention mechanisms can implement temporal difference algorithms for the evaluation problem in reinforcement learning. de Luca and Fountoulakis [2024] prove transformers are capable of simulating several graph algorithms. In this paper, we investigate how computational capabilities interact with prompt engineering and generalization.

Prompting language models is an art as a proper prompting can substantially enhance their response. For instance, using phrases like "let's think step by step" into prompts has been shown to helps language models to produce more accurate solution with logical reasoning [Kojima et al., 2022]. This insight has inspired a growing body of research aimed at automating prompting to enhance the performance of language models. Among the key directions are two prominent areas of focus: the automation of prompt engineering through reinforcement learning-based chain-of-thought prompting [Wei et al., 2022, Zhang et al., 2022, Su et al., 2022, Zhou et al., 2022], and prompt optimization techniques [Cheng et al., 2023, Pryzant et al., 2023, Hsieh et al., 2023, Rubin et al., 2021]. We study the essence of prompt engineering from a computational perspective, demonstrating that it can significantly enhance the computational capabilities of transformers.

While assignment involves optimization over the combinatorial set of permutation matrices, it can be relaxed to optimization over a continuous set. The state-of-the-art method is based on linear programming, namely

$$\widehat{P} = \arg\min_{P \in \mathbb{R}^{n\times n}} \sum_{ij} P_{ij} C_{ij}, \text{ subject to } P \text{ being a \textit{doubly stochastic} matrix,}$$

where $C \in \mathbb{R}^{n\times n}, C_{ij} = \|y_i - x_j\|^2$. Comparing the above problem with the original problem in (1), we notice that the constraint requiring P to be a permutation matrix has been relaxed to allowing P to be a doubly stochastic matrix. Recall the solutions of linear programs lie among the extreme points of the constraint set. Since the extreme points of doubly stochastic matrices are permutation matrices [Conte et al., 1991], the above linear program has the same solution as (1), i.e., $P_* = \widehat{P}$.

The above linear program has $O(n^2)$ variables. Due to the quadratic growth with n, solving the linear program becomes computationally challenging for large n. Cuturi [2013] proposes a computationally efficient alternative based on regularization with entropy:

$$P^*_\lambda := \arg\min_{P \in \mathbb{R}^{n\times n}} \sum_{ij} P_{ij} C_{ij} + \lambda P_{ij} \log(P_{ij}), \text{ subject to } P \text{ is a \textit{doubly stochastic} matrix} \tag{13}$$

The Lagrangian dual of the above program has only $O(n)$ variables which is considerably fewer than $O(n^2)$ variables for the original linear program. Introducing the dual parameters $v \in \mathbb{R}^n$ and $u \in \mathbb{R}^n$, the Lagrangian function is defined as follows:

$$\sum_{ij} P_{ij} C_{ij} + \lambda \sum_{ij} P_{ij} \log(P_{ij}) - u^\top \left(P 1_n - \frac{1}{n} 1_n \right) - v^\top \left(P^\top 1_n - \frac{1}{n} 1_n \right).$$

It is easy to check P is $P_{ij} = e^{\frac{-C_{ij}+v_j+u_i}{\lambda} - 1}$ minimizes the Lagrangian function. This structure inspired the use of Sinkhorn's fixed point iteration to find the solution of the dual problem. In particular, [Sinkhorn, 1967] proves that there exists a unique doubly stochastic matrix of the form $[P^*_\lambda]_{ij} = e^{\frac{-C_{ij}+v^*_i+u^*_j}{\lambda} - 1}$ that is the solution of a simple fixed-point iteration where u^*, v^* are unique up to scaling factors. $[M]_{ij}$ denotes the element of the matrix M located in row i and column j. Leveraging this theorem, Cuturi [2013] proves Sinkhorn recurrence can efficiently find P^*_λ.

Apart from Sinkhorn's fixed point iteration, there are many different methods to solve Lagrangian dual problem such as first-order optimization methods. Recall the minimizer $P_{ij} = e^{\frac{-C_{ij}+u_i+v_j}{\lambda} - 1}$. Plugging this into the Lagrangian function yields

$$\arg\min_{v,u \in \mathbb{R}^n} \left\{ L(u,v) := \lambda \left(\sum_{ij} e^{(-C_{ij}+u_i+v_j)/\lambda - 1} \right) - \frac{1}{n} \sum v_i - \frac{1}{n} \sum_i u_i \right\} \tag{14}$$

It is easy to check that L is convex in u and v as its Hessian is diagonally dominant, hence positive semi-definite. Thus, standard first-order optimization can optimize L such as gradient descent with adaptive coordinate-wise stepsizes:

$$\begin{cases} u_{\ell+1} = u_\ell - D_\ell \nabla_u L(u_\ell, v_\ell) \\ v_{\ell+1} = v_\ell - D'_\ell \nabla_v L(u_\ell, v_\ell) \end{cases}, \tag{15}$$

where $\nabla_u L$ denotes the gradient of L with respect to u and $D_\ell, D'_\ell \in \mathbb{R}^{n\times n}$ are diagonal matrices with positive diagonal elements (stepsizes).

C The Aligning Transformer

The theoretical and experimental results presented are based on a specific choice of parameters for the attention layers. These parameters are used both to generate the plots in Figures 2–**??** and to establish the main theorems. Recall the attention layers $\mathrm{atten}_{\theta_j^{(\ell)}}$, where $\theta_j^{(\ell)}$ denotes the parameters of attention head j at layer ℓ, given by $\theta_j^{(\ell)} = [w_k^{(\ell,j)}, w_q^{(\ell,j)}, w_v^{(\ell,j)}]$.

We define $Q^{(\ell,j)} = w_k^{(\ell,j)}(w_q^{(\ell,j)})^\top$. Let $d' = 2d+9$, and let $e_i \in \mathbb{R}^{d'}$ denote the i-th standard basis vector, defined by $[e_i]_j = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise} \end{cases}$. We choose the parameters such that:

$$\begin{aligned}
\lambda Q^{(\ell,1)} &= [\mathbf{0}_{d'\times d} \quad 2e_1, 2e_2 \ldots, 2e_d \quad \mathbf{0}_{d'} \quad -e_{2d+3} \quad -e_{2d+1} \quad e_{2d+7} \quad -\lambda e_{2d+5} \quad \mathbf{0}_{d'\times 2} \quad e_{2d+5} \quad \mathbf{0}_{d'}] \\
\lambda Q^{(\ell,2)} &= [2e_{d+1}, \ldots, 2e_{2d} \quad \mathbf{0}_{d'\times d} \quad -e_{2d+3} \quad \mathbf{0}_{d'} \quad -e_{2d+2} \quad e_{2d+8} \quad -\lambda e_{2d+5} \quad \mathbf{0}_{d'} \quad e_{2d+5} \quad \mathbf{0}_{d'\times 2}] \\
[w_v^{(\ell,1)}]_{ij} &= \begin{cases} 1 & i = 2d+6 \text{ and } j = 2d+7 \\ 0 & \text{otherwise} \end{cases}, \\
[w_v^{(\ell,2)}]_{ij} &= \begin{cases} 1 & i = 2d+6 \text{ and } j = 2d+8 \\ 0 & \text{otherwise} \end{cases}, \\
B_j^{(\ell)} &= \gamma_\ell I_{d'\times d'}.
\end{aligned} \tag{16}$$

Notably, there are many choices for $w_k^{(\ell,j)}$ and $w_q^{(\ell,j)}$ that ensure the above equations hold.

D The Mechanism of Softmax Attention

As shown in Figure 1, the trained transformer successfully solves the assignment problem for varying values of n. We explain this capability by analyzing the computational power of the `softmax` attention mechanism. Specifically, we prove that softmax attention can simulate gradient descent on the objective function L defined in (14).

Theorem D.1. *There exists a configuration of parameters* ***indepdent from*** n *such that*

$$\begin{cases} [Z_\ell]_{(1:n),(2d+7)} = u_\ell - D_\ell \nabla_u L(u_\ell, v_\ell) \\ [Z_\ell]_{(1:n),(2d+8)} = v_\ell - D'_\ell \nabla_v L(u_\ell, v_\ell) \end{cases},$$

holds for all integer values of n*, where* u_ℓ *and* v_ℓ *are gradient descent in* (15) *iterations starting from* $u_0 = v_0 = 0$ *with the following adaptive stepsizes*

$$(\gamma[D_\ell]_{ii})^{-1} = \sum_j e^{(-C_{ij}+[u_\ell]_i+[v_\ell]_j)/\lambda - 1} + 1, \quad (\gamma[D'_\ell]_{jj})^{-1} = \sum_i e^{(-C_{ij}+[u_\ell]_i+[v_\ell]_j)/\lambda - 1} + 1.$$

Note: γ_ℓ *are arbitrary scalars.*

Linear attention has been shown to simulate gradient descent on mean squares to solve linear regression [Ahn et al., 2024, Von Oswald et al., 2023]. The result above holds for standard **softmax attention**. In particular, we show that softmax attention is especially well-suited to implementing gradient descent on the objective function L, as defined in (14). Remarkably, attention mechanisms cannot simulate the gradient of an arbitrary function and exhibit inherent limitations in their computational capabilities. In particular, *linear attention* is well-suited for problems such as linear regression, where the underlying objective is quadratic [Ahn et al., 2024, Von Oswald et al., 2023]. In contrast, *softmax attention* is better aligned with tasks involving token alignment, which has profound applications in natural language processing, including translation and sequence matching.

Prompt engineering is essential for the proof of Theorem D.1. Expanding the input size by adding columns and rows creates an extended data representation matrix across the layers. Attention layers can utilize a part of this matrix **as memory to store the iterates of gradient descent** . Furthermore, the input dependent part of the prompt supplies the necessary statistics for the attention layers to implement gradient descent.

The result above holds with a choice of parameters that is independent of the input size n. This implies that a single attention layer with fixed parameters can implement gradient descent on the dual objective function L, regardless of sequence length. We leverage this key property to show that a transformer can align an arbitrary number of tokens which theoretically explains the observation in Figure 1.

E Proof of Theorem D.1

We demonstrate that two attention heads can jointly implement a single step of gradient descent (with adaptive step sizes) on $L(u, v)$. By induction, multiple attention heads can implement several iterations of gradient descent with adaptive step sizes. The proof is constructive, explicitly determining the choice of parameters specified in Section C.

Gradient descent iterates in closed form. Recall $u_\ell, v_\ell \in \mathbb{R}^n$ are iterations of gradient descent (with adaptive coordinate-wise stepsize) on Lagrangian function L defined in (15):

$$\begin{cases} u_{\ell+1} = u_\ell - D_\ell \nabla_u L(u_\ell, v_\ell) \\ v_{\ell+1} = v_\ell - D'_\ell \nabla_v L(u_\ell, v_\ell) \end{cases} \quad , L(u, v) := \lambda \left(\sum_{ij} e^{(-C_{ij}+u_i+v_j)/\lambda - 1} \right) - \frac{1}{n} \sum v_i - \frac{1}{n} \sum_i u_i$$

where the coordinate-wise stepsizes are

$$(\gamma_\ell [D_\ell]_{ii})^{-1} = \sum_j e^{(-C_{ij}+[u_\ell]_i+[v_\ell]_j)/\lambda - 1} + 1, \quad (\gamma_\ell [D'_\ell]_{jj})^{-1} = \sum_i e^{(-C_{ij}+[u_\ell]_i+[v_\ell]_j)/\lambda - 1} + 1.$$

Remarkably, the subscript ℓ on γ was omitted in the main submission.

Define $x = [x_1, \ldots, x_n], y = [y_1, \ldots, y_n] \in \mathbb{R}^{n \times d}$ and $x^2 = [\|x_1\|^2, \ldots, \|x_n\|^2], y^2 = [\|y_1\|^2, \ldots, \|y_n\|^2] \in \mathbb{R}^n$. Let $H^{(\ell)} \in \mathbb{R}^{n \times n}$ obeys

$$H^{(\ell)} = \tfrac{1}{\lambda} \left(-x^2 \mathbf{1}_n^\top + 2xy^\top - \mathbf{1}_n (y^2)^\top + u_\ell \mathbf{1}_n^\top + \mathbf{1}_n (v_\ell)^\top \right) - \mathbf{1}_n \mathbf{1}_n^\top \tag{17}$$

Define $M^{(\ell)} \in \mathbb{R}^{n \times n}$ defines as $M_{ij}^{(\ell)} = e^{H_{ij}^{(\ell)}}$. Gradient descent thus follows:

$$\mu_{\ell+1} = \mu_\ell - D_\ell (M^{(\ell)} \mathbf{1}_n - \frac{1}{n} \mathbf{1}_n) \tag{18}$$

$$v_{\ell+1} = v_\ell - D'_\ell ((M^{(\ell)})^\top \mathbf{1}_n - \frac{1}{n} \mathbf{1}_n) \tag{19}$$

Induction statement. Assuming that the statement holds for ℓ, we prove that it holds for $\ell + 1$. Thus, the induction hypothesis is

$$Z_\ell = \begin{bmatrix} x_1 & y_1 & \|x_1\|^2 & \|y_1\|^2 & 1_4 & [u_\ell]_1 & [v_\ell]_1 & 0 \\ x_2 & y_2 & \|x_2\|^2 & \|y_2\|^2 & 1_4 & [u_\ell]_2 & [v_\ell]_2 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & \|x_n\|^2 & \|y_n\|^2 & 1_4 & [u_\ell]_d & [v_\ell]_d & 0 \\ 0 & 0 & 0 & 0 & -v_4/n & ? & ? & 0 \end{bmatrix} \in \mathbb{R}^{(n+1) \times (2d+9)},$$

where 1_4 denotes the all-ones 4-dimensional vector, $v_4 = [0, 0, 0, 1]$, and $[u]_i$ denotes element i of vector u. It is easy to check that the above equation holds for $\ell = 0$ as $u_0 = v_0 = 0_n$. The choice of $w_v^{(\ell,j)}$ ensure that only the $2d + 7$-th and $2d + 8$-th columns of Z_ℓ change with ℓ, which are highlighted in . We prove that

$$Z_{\ell+1} = \begin{bmatrix} x_1 & y_1 & \|x_1\|^2 & \|y_1\|^2 & 1_4 & [u_{\ell+1}]_1 & [v_{\ell+1}]_1 & 0 \\ x_2 & y_2 & \|x_2\|^2 & \|y_2\|^2 & 1_4 & [u_{\ell+1}]_2 & [v_{\ell+1}]_2 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & \|x_n\|^2 & \|y_n\|^2 & 1_4 & [u_{\ell+1}]_d & [v_{\ell+1}]_d & 0 \\ 0 & 0 & 0 & 0 & -v_4/n & ? & ? & 0 \end{bmatrix} \in \mathbb{R}^{(n+1) \times (2d+9)},$$

Indeed, the extended prompt offers sufficient memory to store the iterates of gradient descent.

Induction proof. We begin by computing the output of the first attention head in layer $\ell + 1$, step by step. Through straightforward algebra, we obtain the following:

$$Z_\ell Q^{(\ell,1)} = \begin{bmatrix} 0 & 2x/\lambda & \mathbf{0}_n & -\mathbf{1}_n/\lambda & -\|x\|^2/\lambda & u_\ell/\lambda & -\mathbf{1}_n & \mathbf{0}_{2\times n} & \mathbf{1}_n/\lambda & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{0}_{2\times 1} & 0 & 0 \end{bmatrix} \in \mathbb{R}^{(n+1)\times d'}$$

where matrices x and y are defined earlier in the proof. With these notations and the preceding equation, we proceed as follows:

$$Z_\ell Q^{(\ell,1)} Z_\ell^\top = \begin{bmatrix} H^{(\ell)} & \mathbf{0}_n \\ \mathbf{0}_n^\top & 0 \end{bmatrix}$$

which obtains

$$\exp(Z_\ell Q^{(\ell,1)} Z_\ell^\top) = \begin{bmatrix} M^{(\ell)} & \mathbf{1}_n \\ \mathbf{1}_n^\top & 1 \end{bmatrix} \tag{20}$$

Furthermore, the choice of parameters $w_v^{(\ell,1)}$ obtains

$$Z_\ell w_v^{(\ell,1)} = -\gamma \begin{bmatrix} \mathbf{0}_n & \dots & \mathbf{0}_n & \mathbf{1}_n & \mathbf{0}_n & \mathbf{0}_n \\ 0 & \dots & 0 & -1/n & 0 & 0 \end{bmatrix}$$

Stitching all equations together yields

$$\mathrm{atten}_{\theta_1^{(\ell)}}(Z_\ell) B_1^{(\ell)} = \begin{bmatrix} \mathbf{0}_n & \dots & \mathbf{0}_n & -D_\ell(M^{(\ell)}\mathbf{1}_n - \frac{1}{n}\mathbf{1}_n) & \mathbf{0}_n & \mathbf{0}_n \\ 0 & \vdots & 0 & n - 1/n & 0 & 0 \end{bmatrix}$$

Observe that the matrix above contains the gradient $\nabla_u L(u_\ell, v_\ell)$ from Eq. (19), which is required to compute the next gradient descent iterate $u_{\ell+1}$. Similarly, we can show that

$$\mathrm{atten}_{\theta_2^{(\ell)}}(Z_\ell) B_2^{(\ell)} = \begin{bmatrix} \mathbf{0}_n & \dots & \mathbf{0}_n & \mathbf{0}_n & -D'_\ell(M^{(\ell)})^\top\mathbf{1}_n - \frac{1}{n}\mathbf{1}_n) & \mathbf{0}_n \\ 0 & \vdots & 0 & 0 & n - 1/n & 0 \end{bmatrix},$$

which computes $v_{\ell+1}$ in parallel using a second attention head. Thus, replacing the last two equations into (7) concludes the induction proof.

F Proof of Theorem 4.1

According to Thm. D.1, a transformer can implement gradient descent. Therefore, the proof casts to analyzing gradient descent (with specific coordinate-wise stepsizes) on the convex L. However, we cannot directly apply existing convergence results from convex optimization. The existing convergence results for smooth convex optimization are in terms of function value L when L is not strongly convex[1]. But, the theorem statement aims at the convergence to a minimizer.

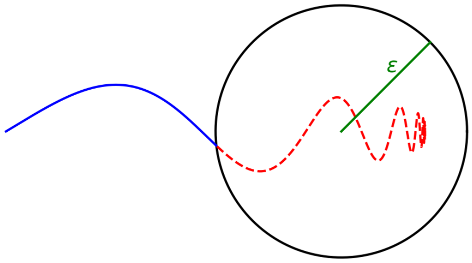


Figure 5: **Proof sketch for Theorem 4.1.** We first prove that the attention matrix converges to a local neighborhood of the set of doubly stochastic matrices. This convergence is illustrated by the blue curves converging to a small circle. Next, we show that this convergence implies convergence to the minimizer. To establish this, we hypothetically run Sinkhorn's iterations and leverage their known convergence rate. The red curve illustrates these hypothetical Sinkhorn iterations. This figure was generated with the assistance of GPT-4o.

The proof consists of two key steps: (i) the convergence of attention matrix $A^{(\ell)}$ to a matrix that is approximately doubly stochastic, and (ii) a hypothetical simulation of Sinkhorn's recurrence. See Figure 5 for the illustration.

[1] It is easy to check that L is not strongly convex since it Hessian has a zero eigenvalue.

(i) As shown in Thm. D.1, the transformer can perform gradient descent with an adaptive step size on the convex function L. Since L is convex, gradient descent is guaranteed to converge to a stationary point where the gradient norm becomes zero. Specifically, it is straightforward to verify that $\nabla_u L = M\mathbf{1} - \frac{1}{n}\mathbf{1}$ and $\nabla_v L = M^\top \mathbf{1} - \frac{1}{n}\mathbf{1}$, where $M_{ij} = \exp\left(\frac{-C_{ij}+u_i+v_j}{\lambda} - 1\right)$. Therefore, small gradients for u and v imply that M is close to being doubly stochastic.

(ii) We demonstrate that when the matrix M is approximately doubly stochastic, it is near the desired solution P_λ^*. To establish this, we (hypothetically) run Sinkhorn's recurrence starting from M and use its contraction property proven by [Franklin and Lorenz, 1989].

Before elaborating on the details of (i) and (ii), we present two propositions.

Preliminaries. Define the functions $row : \mathbb{R}_+^{n\times n} \to \mathbb{R}_+^n$ and $col : \mathbb{R}_+^{n\times n} \to \mathbb{R}_+^n$ as

$$\text{row}(A)_i = \frac{1}{n\sum_j A_{ij}}, \quad \text{col}(A)_j = \frac{1}{n\sum_i A_{ij}}.$$

We also introduce functions $f, g : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ defined as

$$f(A) = A\text{diag}(\text{col}(A)), \quad g(A) = \text{diag}(\text{row}(A))A.$$

Indeed, $f(A)$ (resp. $g(A)$) normalizes the columns (resp. rows) of A by a scaling factor of their average. We will later use f and g to formulate Sinkhorn's recurrence, which iteratively normalizes the rows and columns of a matrix with positive elements. The next proposition proves that an almost doubly stochastic matrix remains almost doubly stochastic under f and g. To formulate the statement, we introduce a set containing matrices that almost doubly stochastic matrices:

$$\mathcal{S}_\epsilon := \left\{ A \in \mathbb{R}_+^{n\times n} \mid \|A\mathbf{1}_n - \tfrac{1}{n}\mathbf{1}_n\|_\infty \leq \epsilon \text{ and } \|A^\top \mathbf{1}_n - \tfrac{1}{n}\mathbf{1}_n\|_\infty \leq \epsilon \right\}.$$

Proposition F.1. *Suppose that $A \in \mathcal{S}_\epsilon$; then $f(A) \in \mathcal{S}_{3\epsilon}$ and $g(A) \in \mathcal{S}_{3\epsilon}$, as long as $\epsilon < 1/(3n)$.*

Recall the metric d defined in (10). The next proposition establishes a particular property of f and g with respect to d.

Proposition F.2. *Let $A \in \mathcal{S}_\epsilon$ be decomposed as $A = diag(w)Qdiag(q)$, where $w, q \in \mathbb{R}_+^n$.*

(i) For $f(A) = diag(w)Qdiag(q')$, $d(q, q') \leq 4n\epsilon$ holds for $\epsilon < \frac{1}{4n}$.

(ii) For $g(A) = diag(w')Qdiag(q)$, $d(w, w') \leq 4n\epsilon$ holds for $\epsilon < \frac{1}{4n}$.

(i) Convergence Analysis. According to Theorem D.1, there is a choice of parameters such that

$$A_{ij}^{(\ell)} = e^{\frac{-C_{ij}+[u_\ell]_i+[v_\ell]_j}{\lambda}-1},$$

where u_ℓ and v_ℓ are the iterates defined in (15). The following lemma establishes that, as the number of iterations ℓ increases, $A^{(\ell)}$ meets a neighborhood of doubly stochastic matrices.

Lemma F.3. *For $\gamma_k^{-1} = (n+2)e^{2r/\lambda}$, there exists a $k \leq \ell$ such that*

$$A^{(k)} \in \mathcal{S}_\epsilon, \quad \textit{where } \epsilon^2 := \left(\frac{1}{\ell}\right) 3ne^{3r/\lambda} r.$$

Notably, the matrix $A^{(k)}$ has a specific structure that ensures $A^{(k)} \in \mathcal{S}_\epsilon$ is sufficient to approximate P_λ^*. To prove this statement, we leverage the contraction property of Sinkhorn's recurrence.

Contractive Sinkhorn's Process. According to the last lemma, there exists an iteration $k \leq \ell$ such that $A^{(k)} \in \mathcal{S}_\epsilon$. We then apply Sinkhorn's recurrence starting from $A_1 = g(A^{(k)})$ as:

$$A_{m+1/2} = f(A_m), \quad A_{m+1} = g(A_{m+1/2}).$$

Notably, we utilize the above recurrence solely for the proof; hence, there is no need for a transformer to implement this recurrence. According to the definition, A_m can be decomposed as diag$(w_m)Q$diag(q_m), where $Q_{ij} = e^{-C_{ij}/\lambda - 1}$ and $q_m, w_m \in \mathbb{R}^n_+$. Sinkhorn [1967] proves that there exist unique vectors $w^*, q^* \in \mathbb{R}^n_+$ such that $P^*_\lambda =$ diag$(w^*)Q$diag(q^*), where $w^*_i = e^{u^*_i/\lambda}$ and $q^*_j = e^{v^*_j/\lambda}$. [Franklin and Lorenz, 1989] establish the linear convergence of (w_m, q_m) to (w^*, q^*):

$$\begin{cases} \mu(w_{m+1}, w^*) \leq \eta \mu(w_m, w^*) \\ \mu(q_{m+1}, q^*) \leq \eta \mu(q_m, q^*) \end{cases}, \quad \eta = \frac{\phi(A_1)^{1/2} - 1}{\phi(A_1)^{1/2} + 1} < 1, \tag{21}$$

where $\phi(A) = \max_{ijkl} \frac{A_{ik} A_{jl}}{A_{jk} A_{il}}$. Since $A_1 =$ diag$(w_1)Q$diag(q_1), we have $\phi(A_1) = \phi(Q)$.

(ii) Approximating the Optimal Solution. Propositions F.1 and F.2 enable us to demonstrate that there exists a constant c such that cA_1 lies within a neighborhood of P^*_λ. Proposition F.1 combined with Lemma F.3 ensure $A_1 \in \mathcal{S}_{3\epsilon}$. Thus, we can apply Proposition F.2 to obtain: $\mu(q_2, q_1) \leq 12n\epsilon$. Using Proposition F.1 again, we find that $A_{1+1/2} \in \mathcal{S}_{9\epsilon}$. Consequently, we can invoke Proposition F.2 once more to yield: $\mu(w_2, w_1) \leq 36n\epsilon$. Applying the triangle inequality together with 21 completes the proof:

$$\begin{aligned} 36n\epsilon &\geq \mu(w_2, w_1) \geq \mu(w_1, w^*) - \mu(w_2, w^*) \geq (1 - \eta) d(w_1, w^*) \\ 12n\epsilon &\geq \mu(q_2, q_1) \geq \mu(q_1, q^*) - \mu(q_2, q^*) \geq (1 - \eta)\mu(q_1, q^*). \end{aligned}$$

G Proof of Lemma F.3

Notations. Define the concatenated vector of iterates as

$$\theta_k = \begin{bmatrix} u_k \\ v_k \end{bmatrix},$$

and consider the following block diagonal matrix:

$$\Lambda_k = \begin{bmatrix} D_k & 0 \\ 0 & D'_k \end{bmatrix},$$

where D_k and D'_k are diagonal matrices at iteration k defined in Theorem D.1. Define the ball $B(r) = \{\theta \in \mathbb{R}^n \mid \|\theta\| \leq r\}$. If $\theta_k \in B(r)$, then

$$\frac{\gamma_k}{n \exp(r/\lambda) + 1} I_n \preceq \Lambda_k \preceq \gamma_k I_n. \tag{22}$$

Smoothness of L. The Hessian of L has the following form

$$\nabla^2 L := \begin{bmatrix} \nabla^2_{uu} L & \nabla^2_{uv} L \\ \nabla^2_{vu} L & \nabla^2_{vv} L \end{bmatrix} = \begin{bmatrix} \text{diag}(\sum_i M_{ij}) & M \\ M^\top & \text{diag}(\sum_j M_{ij}) \end{bmatrix} \tag{23}$$

We will prove that the Hessian bounded within the domain $\theta \in B(r)$. For all $v := \begin{bmatrix} s \in \mathbb{R}^n \\ s' \in \mathbb{R}^n \end{bmatrix}$ such that $\|v\|^2 = 1$, we have

$$v^\top \nabla^2 L v = \|s\|^2_{\text{diag}(\sum_i M_{ij})} + 2s^\top M s' + \|s'\|^2_{\text{diag}(\sum_j M_{ij})} \tag{24}$$

where $\|v\|^2_A = v^\top A v$. Recall $M_{ij} = e^{\frac{-C_{ij} + u_i + v_j}{\lambda} - 1}$, hence

$$\begin{aligned} \sum_{ij} s_i s'_j M_{ij} &= \sum_{ij} s_i e^{u_i/\lambda} s_j e^{v_j/\lambda} e^{-C_{ij}/\lambda - 1} && (25) \\ &\leq \sum_{ij} s_i e^{u_i/\lambda} s_j e^{v_j/\lambda} && (26) \\ &\leq \sqrt{\sum_i s_i^2 e^{2u_i/\lambda}} \sqrt{\sum_i (s'_i)^2 e^{2v_i/\lambda}} && (27) \\ &\leq e^{2r/\lambda} && (28) \end{aligned}$$

It is easy to check that $\|\text{diag}(\sum_i M_{ij})\|$ and $\|\text{diag}(\sum_j M_{ij})\|$ are bounded by $ne^{r/\lambda}g$. Replacing these inequalities into (24) yields

$$v^\top \nabla^2 L v \leq ne^{r/\lambda} \left(\underbrace{\|s\|^2 + \|s'\|^2}_{=1} \right) + 2e^{2r/\lambda} \leq (n+2)e^{2r/\lambda}. \tag{29}$$

Thus, $L(\theta)$ is ζ-smooth for $\zeta := (n+2)e^{2r/\lambda}$ when $\theta \in B(r)$.

Boundedness of iterates. The recurrence relation of the iterates defined in (15) leads to the following inequality:

$$\|\theta_{k+1} - \theta^*\|^2_{\Lambda_k^{-1}} = \|\theta_k - \theta^*\|^2_{\Lambda_k^{-1}} - 2\langle \theta_k - \theta^*, \nabla L(\theta_k) \rangle + \|\nabla L(\theta_k)\|^2_{\Lambda_k}, \tag{30}$$

recall $\|v\|_A^2 = v^\top A v$. Since L is ζ-smooth within $B(r)$, by Theorem 2.1.5 of Nesterov [2013], we have

$$\langle \nabla L(\theta), \theta - \theta^* \rangle \geq \frac{1}{\zeta} \|\nabla L(\theta)\|^2. \tag{31}$$

Substituting the above inequality into (30) yields

$$\|\theta_{k+1} - \theta^*\|^2_{\Lambda_k^{-1}} \leq \|\theta_k - \theta^*\|^2_{\Lambda_k^{-1}} - \left(\frac{2}{\zeta} - \gamma_k \right) \|\nabla L(\theta_k)\|^2. \tag{32}$$

Let $\Delta_k := \|\theta_k - \theta^*\|^2_{\Lambda_k^{-1}}$. For $\gamma_k = \frac{1}{\zeta}$, the above inequality ensures that Δ_k is monotonically decreasing:

$$\Delta_{k+1} \leq \Delta_k - \left(\frac{2}{\zeta} - \gamma_k \right) \|\nabla L(\theta_k)\|^2 \leq \Delta_k.$$

To maintain $\theta_k \in B(r)$ for all k, choose r such that

$$\|\theta_k\| \leq \Delta_k + \|\theta^*\|_{\Lambda_k^{-1}} \leq \|\Delta_1\|_{\Lambda_1^{-1}} + \|\theta^*\|_{\Lambda_1^{-1}} \leq 2\left(\|\theta_1 - \theta^*\| + \|\theta^*\|\right) = r.$$

We now show that $\theta_k \in B(r)$ concludes the proof.

Convergence to a stationary point. Since $\theta_k \in B(r)$, we can take the average of (32) over $k = 1, \ldots, \ell$:

$$\sum_{k=1}^{\ell} \|\nabla L(\theta_k)\|^2 \leq \zeta \left(\sum_{k=1}^{\ell} \Delta_k - \Delta_{\ell+1} \right) \leq \zeta \Delta_1 \leq \zeta \left(ne^{r/\lambda} + 1 \right) r.$$

The above inequality leads to the following bound on the minimum gradient norm:

$$\min_{k \leq \ell} \|\nabla L(\theta_k)\|^2 \leq \tfrac{1}{\ell} \sum_{k=1}^{\ell} \|\nabla L(\theta_k)\|^2 \leq \left(\frac{1}{\ell} \right) \zeta (ne^{r/\lambda} + 1) r. \tag{33}$$

Closeness to Doubly Stochastic Matrices. By definition,

$$\nabla L(\theta_k) = \begin{bmatrix} M^{(k)}\mathbf{1} - \frac{1}{n}\mathbf{1} \\ (M^{(k)})^\top \mathbf{1} - \frac{1}{n}\mathbf{1} \end{bmatrix}, \tag{34}$$

where $\mathbf{1}$ denotes the vector of all ones. Substituting the expression for $\nabla L(\theta_k)$ into (33) gives

$$\min_{k \leq \ell} \left(\|M^{(k)}\mathbf{1} - \frac{1}{n}\mathbf{1}\|^2 + \|(M^{(k)})^\top \mathbf{1} - \frac{1}{n}\mathbf{1}\|^2 \right) \leq \frac{\zeta(n \exp(r/\lambda) + 1)r}{\ell}. \tag{35}$$

H Proof of Proposition F.1

We prove $f(A) \in \mathcal{S}_{3\epsilon}$ and the proof for $g(A) \in \mathcal{S}_{3\epsilon}$ follows exactly the same. Since $A \in \mathcal{S}_\epsilon$, the following two inequalities hold

$$\left| \sum_i A_{ij} - \tfrac{1}{n} \right| \leq \epsilon \implies \sum_i A_{ij} \geq \tfrac{1}{n} - \left| \frac{1}{n} - \sum_i A_{ij} \right| \geq \tfrac{1}{n} - \epsilon \tag{36}$$

Using the above two inequalities, we proceed as

$$\begin{aligned} |\tfrac{A_{ij}}{n\sum_i A_{ij}} - A_{ij}| &= A_{ij}|1 - \tfrac{1}{n\sum_i A_{ij}}| \qquad (37)\\ &\le \tfrac{A_{ij}\epsilon}{\sum_i A_{ij}} \qquad (38)\\ &\le \tfrac{A_{ij}\epsilon}{\frac{1}{n}-\epsilon}. \qquad (39) \end{aligned}$$

We use the above inequality to complete the proof:

$$\begin{aligned} \left|\sum_j \tfrac{A_{ij}}{n\sum_i A_{ij}} - \frac{1}{n}\right| &\le \left|\sum_j \tfrac{A_{ij}}{n\sum_i A_{ij}} - \sum_j A_{ij}\right| + |\sum_j A_{ij} - \tfrac{1}{n}| \qquad (40)\\ &\le \sum_j |\tfrac{A_{ij}}{n\sum_i A_{ij}} - A_{ij}| + \epsilon \qquad (41)\\ &\le \tfrac{\epsilon}{1/n-\epsilon}\sum_j A_{ij} + \epsilon \qquad (42)\\ &\le \epsilon\left(1 + \tfrac{1/n+\epsilon}{1/n-\epsilon}\right) \qquad (43) \end{aligned}$$

I Proof of Proposition F.2

We prove part (i), and the proof for part (ii) follows exactly the same. The following inequality holds for $A \in \mathcal{S}_\epsilon$:

$$\forall j : \left|\sum_i A_{ij} - \tfrac{1}{n}\right| \le \epsilon. \qquad (44)$$

Using the above inequality, we get:

$$\begin{aligned} |q'_j - q_j| &= \left|\tfrac{q_j}{n\sum_i A_{ij}} - q_j\right| \qquad (45)\\ &= q_j|\frac{1}{n\sum_i A_{ij}} - 1| \qquad (46)\\ &\le q_j\left(\tfrac{\epsilon}{\frac{1}{n}-\epsilon}\right) \qquad (47) \end{aligned}$$

Plugging the above inequality into μ concludes the proof:

$$\frac{q_i q'_j}{q_j q'_i} \le \frac{1}{1-2n\epsilon} \implies \mu(q, q'_i) \le \log(\tfrac{1}{1-2n\epsilon}) \le \tfrac{2n\epsilon}{1-2n\epsilon}. \qquad (48)$$