# Real-Time Personalization with Simple Transformers

**Lin An, Andrew A. Li, Vaisnavi Nemala, Gabriel Visotsky**
Carnegie Mellon University
`{linan,aali1,vnemala,gvisotsky}@andrew.cmu.edu`

## Abstract

Real-time personalization by and large relies on embedding-based models, which enable fast optimization via nearest-neighbors, but fail to capture complex user behavior. Transformer-based models successfully capture such behavior, but are provably hard to optimize. We study *simple transformers*, i.e. those with a single self-attention layer, and show they still capture rich user behavior despite their simplicity. We then develop a sub-linear time algorithm with near-optimal performance. On large-scale Spotify and Trivago datasets, simple transformers match the accuracy of deeper models while enabling real-time recommendations, improving objective values by over 20% against natural benchmarks.

## 1 Introduction

Personalization today is powered by ML models that leverage rich user data. Among recent advances, *transformers* in particular have revolutionized the modeling of sequential data and are well-suited for personalization, where users' interactions with items are naturally sequential. Indeed, larger firms (e.g., *Alibaba* [11], *Amazon* [25], *Spotify* [31], *Wayfair* [29]) have already seen strong empirical success with transformers in offline experiments. Yet there is little principled guidance on how they should be applied. In general, real-time optimization under transformers is provably hard (in a manner we make formal momentarily), creating a trade-off between modeling power and tractability. This raises two questions:

**1. Modeling Power:** If we restrict to a simpler subclass of transformers, do we lose significant ability to model user preferences? Proposition 1 (Section 3.1) shows that transformers with a single attention layer can represent a wide range of preference models from economics, marketing, and operations management relevant to personalization. Our experiments (Section 4.1) show that simple transformers captured user preferences almost as accurately as deeper transformer models. They achieved 43% higher accuracy than the best non-attention baselines (logistic regression, random forest, SVM) and were only 6.1% worse than general transformers. Thus, simple transformers deliver much higher accuracy than classical models while nearly matching deeper transformers.

**2. Fast Optimization:** Can such a subclass still support sublinear-time algorithms for millisecond-level recommendations? Theorem 1 (Section 3.2) shows that simple transformers can be optimized efficiently, and Proposition 2 proves this result is tight. Our experiments (Section 4.2) show that our algorithm completes simple-transformer-based recommendation tasks efficiently and accurately. It operates in two phases—retrieval and ranking—which we benchmark against approximate nearest neighbors (ANN) and beam search, respectively. Combining these methods yields four variants. With the same candidate budget, our algorithm achieved objective values 20.56% higher than using our retrieval with beam search, and 20.86% higher than using ANN with our ranking. Thus, it outperforms natural benchmarks in both phases.

Thus, simple transformers strike a practical balance—capturing rich user behavior while enabling efficient real-time personalization.

**Literature Review:** Transformer architectures, introduced by [46], are now central to recommender systems, with adoption by major platforms [11, 25, 31, 29] and research on diverse attention models

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: MLxOR: Mathematical Foundations and Operational Integration of Machine Learning for Uncertainty-Aware Decision-Making.

[21, 29, 51, 49, 11, 4, 8, 53, 55, 10, 16, 25, 43, 27]. Their representational power is well established [54, 50, 20, 34, 40, 39, 6, 15, 24, 48], and we show that even a single attention layer can capture sequential variety and complementarity/substitution effects. With simple transformers, the recommendation task reduces to binary quadratic optimization—NP-hard in general [36] but with tractable cases [36, 44, 35, 7]—and our contribution is to exploit the low non-negative rank of the softmax matrix, building on related low-rank optimization results [17, 30, 33, 12, 15, 2, 3].

## 2 Model

We begin with *pure embedding models*, widely used in personalization. These represent items and users as vectors in a shared low-dimensional space. Let $[n] = \{1, \dots, n\}$ denote the items and $V \in \mathbb{R}^{n\times d}$ the item matrix, with row $v_i^\top$ as the value vector of item $i$. Each user is represented by $u \in \mathbb{R}^d$, typically learned from historical interactions. The utility of recommending item $i$ to user $u$ is $f(v_i^\top u)$, where $f : \mathbb{R} \to \mathbb{R}$ is a non-decreasing Lipschitz function, often interpreted as a click/purchase probability. The real-time task is to select up to $k$ items maximizing total reward:

$$\max \sum_{i\in S} f(v_i^\top u) \qquad \text{s.t.} \quad S \subset [n],\ |S| \le k.$$

Since items contribute independently, the problem reduces to finding the top-$k$ items with largest $v_i^\top u$, i.e., a nearest-neighbor search query. Approximate nearest neighbor (ANN) algorithms achieve sublinear runtime in $n$, while guaranteeing near-optimality. For example, the ANN algorithm in [1] gives a $(1-\epsilon)$-approximation of the optimization problem above with expected amortized runtime $O\big(n^{1-c(\epsilon)}\big)$ for some function $c(\cdot)$ where $c(\epsilon) > 0$.

We now introduce *simple transformers*, our primary model. Unlike pure embedding models, which treat items independently, transformers capture dependencies among items through *self-attention*. Given $Q, K \in \mathbb{R}^{n\times d_{kq}}$, $V \in \mathbb{R}^{n\times d_v}$, and $X \in \mathbb{R}^{n\times n}$, a self-attention layer is

$$\text{SA}_{Q,K,V}(X) = \text{softmax}((XQ)(XK)^\top)XV.^{1}$$

For a subset $S \subset [n]$, let $X_S$ be the diagonal membership matrix with $(X_S)_{ii} = 1$ if $i \in S$. Then each $i \in S$ produces a representation by averaging $\{v_j : j \in S\}$, where the weight on $v_j$ reflects how strongly $q_i$ aligns with $k_j$. Thus the model naturally captures complementarities, substitutions, and other set effects absent in embeddings.

**Definition 1 (Simple Transformer)** *For $Q, K \in \mathbb{R}^{n\times d_{kq}}$, $V \in \mathbb{R}^{n\times d_v}$, user vector $u \in \mathbb{R}^{d_v}$, and non-decreasing Lipchitz activation functions $f$, the simple transformer is*

$$\mathcal{T}_{Q,K,V,f,u}(X) = \left[f(\text{SA}_{Q,K,V}(X)_1^\top u), \cdots, f(\text{SA}_{Q,K,V}(X)_n^\top u)\right]^\top.^{2}$$

As before, the recommendation task is to select up to $k$ items maximizing total reward:

$$\text{OPT} := \max \sum_{i\in S} f(\text{SA}_{Q,K,V}(X_S)_i^\top u) \qquad \text{s.t.} \quad S \subset [n],\ |S| \le k. \tag{P}$$

## 3 Theoretical Results

### 3.1 Modeling Preferences with Simple Transformers

Simple transformers are widely used in personalization [49, 11, 4, 8], though deeper architectures are also common. This raises the question: how well can simple transformers model user preferences? We address this by showing that two standard parametric models can be exactly represented within this framework. First, we consider a model of *sequential variety effects*. Marketing studies (e.g., [28, 19, 37]) show that utility depends not only on intrinsic quality but also on novelty relative to past items. Repeated exposure to similar items lowers utility, while diversity restores engagement. We now give a mathematical formulation.

[1]For $A \in \mathbb{R}^{n\times d}$, the row-wise softmax, interpreted as attention weights, is defined by $\text{softmax}(A)_{i,j} = \exp(A_{i,j}) / \sum_{j'=1}^{d} \exp(A_{i,j'})$.

[2]A simple transformer may also use multiple *attention heads*, where parallel self-attention layers with different $Q, K, V$ are concatenated before point-wise transformations. Our results extend to this setting, but for simplicity we focus on the single-head case.

**Model 1 (Sequential Variety Effects)** *Let* $[n] := \{1, \dots, n\}$ *denote the set of items. Each item* $i$ *has a* base utility $\hat{u}_i > 0$ *and a* similarity embedding $x_i \in \mathbb{S}^{d-1}$. *Define* pairwise similarity score *between item i and item j by* $s_{ij} := x_i^\top x_j \in [-1, 1]$. *Fix a sequence length* $k \geq 1$ *and nonnegative* lag weights $\lambda_1, \dots, \lambda_{k-1} \geq 0$.

*For a sequence* $S = (i_1, \dots, i_k)$ *of length* $k$*, the* sequential–variety–adjusted utility *of the item at position* $t$ *is*

$$g(S, i_t) = \hat{u}_{i_t} \exp\left(\beta \sum_{\ell=1}^{t-1} \lambda_\ell \, s_{i_t, i_{t-\ell}}\right), \qquad t = 1, \dots, k,$$

*where* $\beta \in \mathbb{R}$ *controls the strength and sign of the variety effect.*

Second, we consider a model of *complementarity* and *substitution*. In economics, these effects describe how the presence of one item influences the desirability of another. A common modeling approach represents items as vectors in a feature space, with pairwise interactions captured via inner products (see, e.g., [26, 5, 38]).

**Model 2 (Complementarity and Substitution Effects)** *Let* $[n]$ *denote the set of items. Each item* $i$ *has a value vector* $\hat{v}_i \in \mathbb{R}^d$. *The pairwise complementarity and substitution effects is parametrized by a matrix* $H \in \mathbb{R}^{n \times n}$ *where* $H_{ii} = 0$ *for every* $i \in [n]$. *For a user* $\hat{u} \in \mathbb{R}^d$ *and a subset of items* $S \subset [n]$*, the interaction-adjusted utility of item* $i \in S$ *is defined as*

$$g(S, i) = \hat{v}_i^\top \hat{u} + \sum_{j \in S} H_{ij}.$$

Since in both models an item's reward depends on the presence of others, they cannot be captured by pure embeddings. In contrast, we show that simple transformers can represent both.

**Proposition 1** *Model 1 and Model 2 can be represented by a simple transformer.*

### 3.2 Fast Optimization with Simple Transformers

Our main result is an algorithm that approximately solves Problem (P) in sub-linear time in $n$ and polynomial time in $k$. Let $W = \text{softmax}(QK^\top)$. We first show that such guarantees are impossible without rank assumptions on $d_{kq}$ and the non-negative rank of $W$, denoted $\text{rank}_+(W)$.[3]

**Proposition 2** *(a) If* $d_{kq} = \Omega(\log n)$*, Problem* (P) *subsumes finding the maximum clique in graphs with* $\Omega(n)$ *disjoint cliques. (b) Problem* (P) *has no* $(1-\epsilon)$*-approximation runs in time* $k^{o(\sqrt{\text{rank}_+(W)})}$*, unless Gap-ETH holds [13].*[4]

Thus, sub-linear runtime requires $d_{kq} = o(\log n)$, and polynomial dependence on $k$ requires $\text{rank}_+(W) = O(1)$. Under these assumptions we obtain:

**Theorem 1** *Suppose* $d_{kq} = o(\log n)$ *and* $\text{rank}_+(W) = O(1)$. *There exists an algorithm for Problem (*P*) which, for any* $k \in \mathbb{N}$ *and* $\epsilon > 0$*, achieves* $\text{ALG} \geq (1-\epsilon)\,\text{OPT}$*, in expected amortized runtime*

$$O\left(k^{\mu(\epsilon)} n^{1-\gamma(\epsilon,k)}\right)$$

*for functions* $\mu, \gamma$ *with* $\gamma(\epsilon, k) > 0$.

Finally, the result holds if $W$ can be well approximated by a low non-negative rank matrix, not necessarily if $W$ itself has low rank.

Real-time recommendation pipelines typically consist of *retrieval* (selecting a small candidate set) and *ranking* (choosing the final recommendations). Our algorithm follows this structure but introduces new techniques in both phases.

[3]The non-negative rank of $W \in \mathbb{R}_{\geq 0}^{n \times n}$ is the smallest $r$ such that $W = AB^\top$ for $A, B \in \mathbb{R}_{\geq 0}^{n \times r}$.

[4]The Gap Exponential Time Hypothesis (Gap-ETH) asserts that, for some constant $\epsilon > 0$, distinguishing between satisfiable 3SAT formulas and those that are not even $(1-\epsilon)$-satisfiable requires exponential time.

**Phase One: Retrieval.** We extend ANN-based retrieval to incorporate transformer structure. Items are partitioned offline by query and key vectors so that those in the same partition behave similarly under self-attention. Given a user $u$, we perform ANN search within each partition and take the union of retrieved items as the candidate set.

**Phase Two: Ranking.** We build on beam search, which greedily explores a decision tree of item sequences, but add a provable refinement. Instead of exploring all $k$ levels, we explore only a limited prefix and then apply LP rounding to optimize over the remaining items. This balances runtime (shallower exploration) and accuracy (limited rounding error), yielding near-optimal rewards in sub-linear time.

## 4 Experiments

### 4.1 Representation

We evaluate simple transformers on two large-scale datasets. Spotify Million Playlist Dataset [9]: one million playlists created between 2010–2017, each containing track metadata. Trivago Session-Based Hotel Recommendation Dataset [23]: roughly 900K hotel-search sessions from 730K users across 340K hotels.

We compare simple transformers against models that ignore sequence effects (e.g., logistic regression, random forests, SVM, nearest neighbor) and deeper transformers with multiple attention layers. The prediction task is to estimate user responses (click/no click) given past interactions.

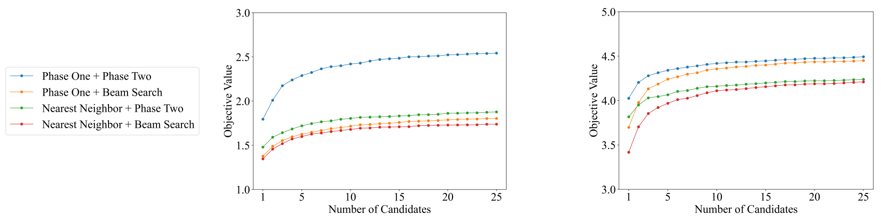
| | Random Forest | Logistic Regression | Simple Transformer | General Transformers |
|---|---|---|---|---|
| Spotify | 0.518 | 0.520 | 0.702 | 0.726 |
| Trivago | 0.268 | 0.333 | 0.503 | 0.552 |

Table 1: Average accuracy of different machine-learning models on Spotify and Trivago.

Our results show that, simple transformers consistently outperform models without sequence structure and approach the accuracy of deeper transformers. Thus, they are expressive enough to capture rich user behaviors while remaining efficient for real-time personalization.

### 4.2 Optimization

On Spotify and Trivago, we benchmark against ANN search (ignoring sequence effects) and beam search. For recommending five items, our method consistently improves user engagement. With ranking fixed, our retrieval phase outperforms ANN search by 20.86% on average; with retrieval fixed, our ranking phase outperforms Beam Search by 20.56%. Overall, the algorithm combines high efficiency with strong empirical accuracy in both phases.



(a) Spotify (b) Trivago

Figure 1: The $x$-axis is the number of candidate solutions generated by each algorithm, and the $y$-axis is the objective value of the current best solution. Each figure is averaged across 100 instances.

## References

[1] Alexandr Andoni, Piotr Indyk, Thijs Laarhoven, Ilya Razenshteyn, and Ludwig Schmidt. Practical and optimal lsh for angular distance. *Advances in neural information processing systems*, 28, 2015.

[2] Cristina Bazgan, Hadrien Hugot, and Daniel Vanderpooten. Implementing an efficient fptas for the 0–1 multi-objective knapsack problem. *European Journal of Operational Research*, 198(1):47–56, 2009.

[3] Cristina Bazgan, Hadrien Hugot, and Daniel Vanderpooten. Solving efficiently the 0–1 multi-objective knapsack problem. *Computers & Operations Research*, 36(1):260–279, 2009.

[4] Walid Bendada, Théo Bontempelli, Mathieu Morlon, Benjamin Chapus, Thibault Cador, Thomas Bouabça, and Guillaume Salha-Galvan. Track mix generation on music streaming services using transformers. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 112–115, 2023.

[5] Steve Berry, Ahmed Khwaja, Vineet Kumar, Andres Musalem, Kenneth C Wilbur, Greg Allenby, Bharat Anand, Pradeep Chintagunta, W Michael Hanemann, Przemek Jeziorski, et al. Structural models of complementary choices. *Marketing Letters*, 25:245–256, 2014.

[6] Alberto Bietti, Vivien Cabannes, Diane Bouchacourt, Herve Jegou, and Leon Bottou. Birth of a transformer: A memory viewpoint. *Advances in Neural Information Processing Systems*, 36, 2024.

[7] Valentina Cacchiani, Manuel Iori, Alberto Locatelli, and Silvano Martello. Knapsack problems—an overview of recent advances. part ii: Multiple, multidimensional, and quadratic knapsack problems. *Computers & Operations Research*, 143:105693, 2022.

[8] Marjan Celikik, Ana Peleteiro Ramallo, and Jacek Wasilewski. Reusable self-attention recommender systems in fashion industry applications. In *Proceedings of the 16th ACM Conference on Recommender Systems*, pages 448–451, 2022.

[9] Ching-Wei Chen, Paul Lamere, Markus Schedl, and Hamed Zamani. Recsys challenge 2018: Automatic music playlist continuation. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 527–528, 2018.

[10] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 335–344, 2017.

[11] Qiwei Chen, Huan Zhao, Wei Li, Pipei Huang, and Wenwu Ou. Behavior sequence transformer for e-commerce recommendation in alibaba. In *Proceedings of the 1st international workshop on deep learning practice for high-dimensional sparse data*, pages 1–4, 2019.

[12] Joel E Cohen and Uriel G Rothblum. Nonnegative ranks, decompositions, and factorizations of nonnegative matrices. *Linear Algebra and its Applications*, 190:149–168, 1993.

[13] Ilan Doron-Arad, Ariel Kulik, and Pasin Manurangsi. Fine grained lower bounds for multidimensional knapsack. *arXiv preprint arXiv:2407.10146*, 2024.

[14] Ilya Dumer. Covering spheres with spheres. *Discrete & Computational Geometry*, 38:665–679, 2007.

[15] Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1(1):12, 2021.

[16] Mingsheng Fu, Hong Qu, Dagmawi Moges, and Li Lu. Attention based collaborative filtering. *Neurocomputing*, 311:88–98, 2018.

[17] Vineet Goyal and Ramamoorthi Ravi. An fptas for minimizing a class of low-rank quasi-concave functions over a convex set. *Operations Research Letters*, 41(2):191–196, 2013.

[18] Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169–197, 1981.

[19] Stephen J Hoch, Eric T Bradlow, and Brian Wansink. The variety of an assortment. *Marketing Science*, 18(4):527–546, 1999.

[20] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[21] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*, pages 197–206. IEEE, 2018.

[22] Leonid G Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980.

[23] Peter Knees, Yashar Deldjoo, Farshad Bakhshandegan Moghaddam, Jens Adamczak, Gerard-Paul Leyson, and Philipp Monreal. Recsys challenge 2019: Session-based hotel recommendations. In *Proceedings of the Thirteenth ACM Conference on Recommender Systems*, RecSys '19, New York, NY, USA, 2019. ACM.

[24] Joohwan Ko and Andrew A Li. Modeling choice via self-attention. *arXiv preprint arXiv:2311.07607*, 2023.

[25] Thom Lake, Sinead A Williamson, Alexander T Hawk, Christopher C Johnson, and Benjamin P Wing. Large-scale collaborative filtering with product embeddings. *arXiv preprint arXiv:1901.04321*, 2019.

[26] Sanghak Lee, Jaehwan Kim, and Greg M Allenby. A direct utility model for asymmetric complements. *Marketing Science*, 32(3):454–470, 2013.

[27] Chengxi Li, Yejing Wang, Qidong Liu, Xiangyu Zhao, Wanyu Wang, Yiqi Wang, Lixin Zou, Wenqi Fan, and Qing Li. Strec: Sparse transformer for sequential recommendations. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 101–111, 2023.

[28] Leigh McAlister. A dynamic attribute satiation model of variety-seeking behavior. *Journal of consumer research*, 9(2):141–150, 1982.

[29] M Jeffrey Mei, Cole Zuber, and Yasaman Khazaeni. A lightweight transformer for next-item product recommendation. In *Proceedings of the 16th ACM Conference on Recommender Systems*, pages 546–549, 2022.

[30] Shashi Mittal and Andreas S Schulz. An fptas for optimizing a class of low-rank functions over a polytope. *Mathematical Programming*, 141:103–120, 2013.

[31] Dmitrii Moor, Yi Yuan, Rishabh Mehrotra, Zhenwen Dai, and Mounia Lalmas. Exploiting sequential music preferences via optimisation-based sequencing. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 4759–4765, 2023.

[32] Yurii Nesterov and Arkadii Nemirovskii. *Interior-point polynomial algorithms in convex programming*. SIAM, 1994.

[33] Trung Thanh Nguyen and Khaled Elbassioni. A ptas for a class of binary non-linear programs with low-rank functions. *Operations Research Letters*, 49(5):633–638, 2021.

[34] Jorge Pérez, Javier Marinković, and Pablo Barceló. On the turing completeness of modern neural network architectures. *arXiv preprint arXiv:1901.03429*, 2019.

[35] David Pisinger and Paolo Toth. Knapsack problems. *Handbook of Combinatorial Optimization: Volume1–3*, pages 299–428, 1998.

[36] David J Rader Jr and Gerhard J Woeginger. The quadratic 0–1 knapsack problem with series–parallel support. *Operations Research Letters*, 30(3):159–166, 2002.

[37] Omid Rafieian. Optimizing user engagement through adaptive ad sequencing. *Marketing Science*, 42(5):910–933, 2023.

[38] Francisco JR Ruiz, Susan Athey, and David M Blei. Shopper. *The Annals of Applied Statistics*, 14(1):1–27, 2020.

[39] Clayton Sanford, Daniel Hsu, and Matus Telgarsky. One-layer transformers fail to solve the induction heads task. *arXiv preprint arXiv:2408.14332*, 2024.

[40] Clayton Sanford, Daniel J Hsu, and Matus Telgarsky. Representational strengths and limitations of transformers. *Advances in Neural Information Processing Systems*, 36, 2024.

[41] A Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[42] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[43] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. *Advances in neural information processing systems*, 28, 2015.

[44] Richard Taylor. Approximation of the quadratic knapsack problem. *Operations Research Letters*, 44(4):495–497, 2016.

[45] Pravin M Vaidya. A new algorithm for minimizing convex functions over convex sets. *Mathematical programming*, 73(3):291–341, 1996.

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

[47] Jean-Louis Verger-Gaugry. Covering a ball with smaller equal balls in $\mathbb{R}^n$. *Discrete & Computational Geometry*, 33:143–155, 2005.

[48] Hanzhao Wang, Xiaocheng Li, and Kalyan Talluri. Transformer choice net: A transformer neural network for choice prediction. *arXiv preprint arXiv:2310.08716*, 2023.

[49] Shoujin Wang, Liang Hu, Longbing Cao, Xiaoshui Huang, Defu Lian, and Wei Liu. Attention-based transactional context embedding for next-item recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[50] Colin Wei, Yining Chen, and Tengyu Ma. Statistically meaningful approximation: a case study on approximating turing machines with transformers. *Advances in Neural Information Processing Systems*, 35:12071–12083, 2022.

[51] Timo Wilm, Philipp Normann, Sophie Baumeister, and Paul-Vincent Kobow. Scaling session-based transformer recommendations using optimized negative sampling and loss functions. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 1023–1026, 2023.

[52] Yihong Wu. Lecture 14: Covering and packing numbers. `http://www.stat.yale.edu/~yw562/teaching/598/lec14.pdf`, 2017. STAT 598: High Dimensional Statistics, Yale University.

[53] Boming Yang, Dairui Liu, Toyotaro Suzumura, Ruihai Dong, and Irene Li. Going beyond local: Global graph-enhanced personalized news recommendations. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 24–34, 2023.

[54] Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank J Reddi, and Sanjiv Kumar. Are transformers universal approximators of sequence-to-sequence functions? *arXiv preprint arXiv:1912.10077*, 2019.

[55] Zhi Zheng, Ying Sun, Xin Song, Hengshu Zhu, and Hui Xiong. Generative learning plan recommendation for employees: A performance-aware reinforcement learning approach. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 443–454, 2023.

## A Proofs in Section 3.1

**Proof of Proposition 1.**

**Model 1:** We construct a simple transformer with the following parameters:

- The input dimension is set to
$$N = nk + nkd + 1 + nk,$$
and is indexed as follows. Define index sets
$$\mathcal{I} = \{(i,t) : i \in [n],\ t \in [k]\}, \quad \mathcal{M} = \{(j,m,a) : j \in [n],\ m \in [k],\ a \in [d]\},$$
$$\mathcal{D} = \{\odot\} \quad \text{(one dummy row)}, \qquad \mathcal{B} = \{(i,t)^{\text{base}} : i \in [n],\ t \in [k]\},$$
and order the $N$ rows of $Q$ and $K$ as $[\mathcal{I};\ \mathcal{M};\ \mathcal{D};\ \mathcal{B}]$.
Before proceeding, we give intuitions on how these index sets are used:
  - $\mathcal{I} = \{(i,t)\}$: These rows are the only rows with nonzero queries $Q$. Each of them represents the effects of the past items to the current item.
  - $\mathcal{M} = \{(j,m,a)\}$: These rows have non-zero keys $K$ and scalar values $V$. They have zero queries $Q$. They encode the item $i_m$ at position $m$ and its similarity embedding $x_{i_m,a}$.
  - $\mathcal{D} = \{\odot\}$: This is a dummy row that dominates the softmax denominator, so that the softmax vector behaves like division by a constant.
  - $\mathcal{B} = \{(i,t)^{\text{base}}\}$: These rows encode the base utility $\hat{u}_i$ of each item $i$.
- The embedding dimension is set to $d_{kq} = k + d + 2$. This is split into a *position* block of length $k$, a *component* block of length $d$, a single *dummy* column (denoted $\Delta$), and a single *base* column (denoted $\Theta$).
- Let $M > 0$ be a sufficiently large constant and $b_0 \in \mathbb{R}$ a fixed constant. Later we will take $M$ large enough so that the simple transformer approximates $g(S, i_t)$ to arbitrary precision.
- For each position $t \in [k]$, define the row vector $r_t \in \mathbb{R}^{1\times k}$ by
$$(r_t)_m \;=\; \begin{cases} \log \lambda_{t-m}, & m < t, \\ -M, & m \geq t, \end{cases} \qquad m \in [k],$$
i.e.
$$r_t = \big[\log \lambda_{t-1},\ \log \lambda_{t-2},\ \cdots\ \log \lambda_1,\ \underbrace{-M,\ \ -M,\ \ \cdots,\ \ -M}_{k-t+1}\big].$$
For each $t \in [k]$, let $R_t \in \mathbb{R}^{n\times k}$ be the matrix with all rows equal to $r_t$:
$$R_t \;=\; \begin{bmatrix} r_t \\ r_t \\ \vdots \\ r_t \end{bmatrix} \qquad (n \text{ rows}).$$
Then the *position block* of the query matrix is the vertical stacking of these $k$ blocks:
$$Q_{\text{pos}} \;=\; \begin{bmatrix} R_1 \\ R_2 \\ \vdots \\ R_k \end{bmatrix} \in \mathbb{R}^{(nk)\times k}.$$
Next, define the $n \times d$ matrix $G$ collecting the (component-wise) logarithms of the similarity embeddings $x_i = (x_{i,1}, \ldots, x_{i,d}) \in S^{d-1}$:
$$G \;=\; \begin{bmatrix} (\log x_1)^\top \\ (\log x_2)^\top \\ \vdots \\ (\log x_n)^\top \end{bmatrix} \;=\; \begin{bmatrix} \log x_{1,1} & \log x_{1,2} & \cdots & \log x_{1,d} \\ \log x_{2,1} & \log x_{2,2} & \cdots & \log x_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ \log x_{n,1} & \log x_{n,2} & \cdots & \log x_{n,d} \end{bmatrix}.$$

The *component block* of the query matrix is $k$ identical copies of $G$ stacked vertically:

$$Q_{\text{cmp}} = \begin{bmatrix} G \\ G \\ \vdots \\ G \end{bmatrix} \in \mathbb{R}^{(nk)\times d}.$$

Let $\mathbf{1}_p \in \mathbb{R}^{p\times 1}$ be the all-ones column, and $0_p \in \mathbb{R}^{p\times 1}$ be the all-zeross column. Define the *dummy* and *base* query columns as

$$Q_\Delta = \begin{bmatrix} M^3\,\mathbf{1}_{nk} \\ 0_{nkd} \\ 0 \\ 0_{nk} \end{bmatrix}, \qquad Q_\Theta = \begin{bmatrix} b_0\,\mathbf{1}_{nk} \\ 0_{nkd} \\ 0 \\ 0_{nk} \end{bmatrix} \in \mathbb{R}^{N\times 1}.$$

Putting the query blocks together, we define $Q$ to be

$$Q = \begin{bmatrix} Q_{\text{pos}} & Q_{\text{cmp}} & Q_\Delta & Q_\Theta \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \in \mathbb{R}^{N\times(k+d+2)}.$$

- The *position block* of the key matrix is defined as

$$K_{\text{pos}} = \begin{bmatrix} \mathbf{1}_{nd} & 0 & \cdots & 0 \\ 0 & \mathbf{1}_{nd} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{1}_{nd} \end{bmatrix} \in \mathbb{R}^{(nkd)\times k},$$

i.e., a block-diagonal matrix with $k$ diagonal blocks, each block the column $\mathbf{1}_{nd}$.
Let $I_d \in \mathbb{R}^{d\times d}$ be the identity matrix. The *component block* of the key matrix is defined as

$$K_{\text{cmp}} = \begin{bmatrix} I_d \\ I_d \\ \vdots \\ I_d \end{bmatrix} \in \mathbb{R}^{(nkd)\times d},$$

Thus, in the block associated with position $m$ and item $j$, the $d$ consecutive rows equal the identity $I_d$.
The *dummy* and *base* key columns are

$$K_\Delta = \begin{bmatrix} 0_{nk} \\ 0_{nkd} \\ 1 \\ 0_{nk} \end{bmatrix}, \qquad K_\Theta = \begin{bmatrix} 0_{nk} \\ 0_{nkd} \\ 0 \\ \mathbf{1}_{nk} \end{bmatrix} \in \mathbb{R}^{N\times 1}.$$

Putting the key blocks together, we define $K$ to be

$$K = \begin{bmatrix} 0 & 0 & 0 & 0 \\ K_{\text{pos}} & K_{\text{cmp}} & 0 & 0 \\ 0 & 0 & K_\Delta & 0 \\ 0 & 0 & 0 & K_\Theta \end{bmatrix} \in \mathbb{R}^{N\times(k+d+2)}.$$

- Let $v \in \mathbb{R}^{nd\times 1}$:

$$v = \begin{bmatrix} x_1^\top \\ x_2^\top \\ \vdots \\ x_n^\top \end{bmatrix} = \begin{bmatrix} x_{1,1} \\ \vdots \\ x_{1,d} \\ x_{2,1} \\ \vdots \\ x_{n,d} \end{bmatrix}.$$

We define $V$ (one scalar per row) by placing zeros on $\mathcal{I}$ and $\mathcal{D}$, the item–component entries on $\mathcal{M}$, and the base utilities on $\mathcal{B}$:

$$V \;=\; \begin{bmatrix} 0_{nk} \\ \beta\, e^{M^3}\, v \\ 0 \\ e^{M^3}\,\left(\frac{1}{\beta}\log \hat{u}_1, \ldots, \frac{1}{\beta}\log \hat{u}_n\right)^\top \\ \vdots \\ e^{M^3}\,\left(\frac{1}{\beta}\log \hat{u}_1, \ldots, \frac{1}{\beta}\log \hat{u}_n\right)^\top \end{bmatrix} \;\in\; \mathbb{R}^{N\times 1},$$

where $e^{M^3}\,\left(\frac{1}{\beta}\log \hat{u}_1, \ldots, \frac{1}{\beta}\log \hat{u}_n\right)^\top$ is repeated $k$ times. Set $u = 1$ so that $V_i^\top u = V_i$ for each row $i$.

- We set $f_i(x) = \exp(\beta x)$ for all $i \in [nk + nkd + 1 + nk]$.
- For a length-$k$ sequence $S = (i_1, \ldots, i_k)$, let $S' \subset [nk + nkd + 1 + nk] = [\mathcal{I};\ \mathcal{M};\ \mathcal{D};\ \mathcal{B}]$ such that $S'$ contains $\{(i_t, t) : t \in [k]\} \subset [\mathcal{I}]$ from the first set of rows, $\{(i_m, m, a) : m \in [k],\ a \in [d]\} \subset [\mathcal{M}]$ from the second set of rows, the dummy row $\mathcal{D}$, and $\{(i_t, t)^{\text{base}} : t \in [k]\} \subset \mathcal{B}$ from the set of base rows.

We show that the simple transformer above approximates $g(S, i_t)$ to arbitrary precision. The intuition of our construction is given below:

- $Q$ encodes lags ($\lambda$) and similarity embeddings ($\log x_i$).
- $K$ encodes positions ($t$) in the sequence.
- $V$ encodes similarity embeddings ($\log x_i$) and base utilities ($\hat{u}_i$).
- The dummy row makes softmax act like a constant divider.

Fix a length-$k$ sequence $S = (i_1, \ldots, i_k)$. We first compute $X_{S'}Q(X_{S'}K)^\top$ for each $(i_t, t) \in S'$:

$$\text{(context)}\quad (X_{S'}Q(X_{S'}K)^\top)_{(i_t,t),(i_m,m,a)} \;=\; r_t(m) + \log x_{i_t,a} = \begin{cases} \log \lambda_{t-m} + \log x_{i_t,a}, & m < t, \\ -M + \log x_{i_t,a}, & m \geq t, \end{cases}$$

$$\text{(dummy)}\quad (X_{S'}Q(X_{S'}K)^\top)_{(i_t,t),\odot} \;=\; M^3,$$

$$\text{(base)}\quad (X_{S'}Q(X_{S'}K)^\top)_{(i_t,t),(i_t,t)^{\text{base}}} \;=\; 0.$$

Hence the denominator of the softmax operation on row $X_{S'}Q(X_{S'}K)^\top$ is

$$Z_t \;=\; e^{M^3} + 1 + \sum_{m<t}\sum_{a=1}^{d} \lambda_{t-m} x_{i_t,a} + \sum_{m\geq t}\sum_{a=1}^{d} e^{-M} x_{i_t,a}.$$

Define the following (finite) constants

$$\Lambda_t = \sum_{m<t} \lambda_{t-m}, \qquad S_t^{(\leq)} = \sum_{m<t}\sum_{a} \lambda_{t-m} x_{i_t,a}\ , \qquad S_t^{(\geq)} = \sum_{m\geq t}\sum_{a} e^{-M} x_{i_t,a}\ .$$

Set

$$\delta_t = e^{-M^3} + e^{-M^3} S_t^{(\leq)} + e^{-M^3} S_t^{(\geq)}.$$

Then

$$Z_t = e^{M^3}(1 + \delta_t).$$

Therefore the attention weights on row $(i_t, t)$ are

$$\text{softmax}(X_{S'}Q(X_{S'}K)^\top)_{(i_t,t),(i_m,m,a)} = \frac{\exp(r_t(m) + \log x_{i_t,a})}{Z_t} = \frac{\lambda_{t-m}\, x_{i_t,a}}{e^{M^3}(1+\delta_t)} \quad (m < t),$$

and

$$\text{softmax}(X_{S'}Q(X_{S'}K)^\top)_{(i_t,t),(i_t,t)^{\text{base}}} = \frac{\exp(0)}{Z_t} = \frac{e^{-M^3}}{1+\delta_t}.$$

Recall that

$$V_{(i_m,m,a)} = e^{M^3}\, x_{i_m,a}, \qquad V_{\odot} = 0, \qquad V_{(i_t,t)^{\text{base}}} = e^{M^3} \cdot \frac{1}{\beta} \log \hat{u}_{i_t}.$$

So we have

$$\begin{aligned}
\mathrm{SA}_{Q,K,V}(X_{S'})_{(i_t,t)} &= \sum_{m<t} \sum_{a=1}^{d} \text{softmax}(X_{S'}Q(X_{S'}K)^\top)_{(i_t,t),(i_m,m,a)}\, V_{(i_m,m,a)} \\
&\quad + \text{softmax}(X_{S'}Q(X_{S'}K)^\top)_{(i_t,t),(i_t,t)^{\text{base}}}\, V_{(i_t,t)^{\text{base}}} \\
&= \frac{1}{1+\delta_t} \left( \sum_{m<t} \sum_{a=1}^{d} \frac{\lambda_{t-m} x_{i_t,a}}{e^{M^3}}\, e^{M^3} x_{i_m,a} + e^{-M^3}\, e^{M^3} \cdot \frac{1}{\beta} \log \hat{u}_{i_t} \right) \\
&= \frac{1}{1+\delta_t} \left( \sum_{m<t} \lambda_{t-m} \sum_{a=1}^{d} x_{i_t,a} x_{i_m,a} \; + \; \frac{1}{\beta} \log \hat{u}_{i_t} \right) \\
&= \frac{1}{1+\delta_t} \left( \sum_{\ell=1}^{t-1} \lambda_\ell\, x_{i_t}^\top x_{i_{t-\ell}} \; + \; \frac{1}{\beta} \log \hat{u}_{i_t} \right).
\end{aligned}$$

Fix any $\epsilon > 0$. Since $S_t^{(\leq)} \leq \Lambda_t \leq \Lambda_{k-1}$ and $S_t^{(\geq)} \leq d\, e^{-M}$, we can set $M$ large enough so that

$$e^{-M^3}\big(1 + \Lambda_{k-1}\big) + d\, e^{-(M^3+M)} \leq \epsilon,$$

which gives $\delta_t \leq \epsilon$ for every $t$. Finally, we have

$$\begin{aligned}
\mathcal{T}_{Q,K,V,f_1,\ldots,f_n,u}(X_{S'})_{(i_t,t)} &= f_{(i_t,t)}\big(\mathrm{SA}_{Q,K,V}(X_{S'})_{(i_t,t)}\big) \\
&= \exp\left( \frac{\beta}{1+\delta_t} \sum_{\ell=1}^{t-1} \lambda_\ell\, x_{i_t}^\top x_{i_{t-\ell}} + \frac{1}{1+\delta_t} \log \hat{u}_{i_t} \right) \\
&= \hat{u}_{i_t}^{\frac{1}{1+\delta_t}} \; \exp\left( \frac{\beta}{1+\delta_t} \sum_{\ell=1}^{t-1} \lambda_\ell\, x_{i_t}^\top x_{i_{t-\ell}} \right).
\end{aligned}$$

Hence, as $M \to \infty$ (so $\delta_t \to 0$ uniformly in $t$), we have

$$(1-\epsilon) g(S, i_t) \leq \mathcal{T}_{Q,K,V,f_1,\ldots,f_n,u}(X_{S'})_{(i_t,t)} \leq \mathcal{T}_{Q,K,V,f_1,\ldots,f_n,u}(X_{S'})_{(i_t,t)}.$$

Therefore the simple transformer approximate $g(S, i_t)$ to arbitrary precision.

**Model 2:** We construct a simple transformer with three self-attention heads. The first head and the second head represent the complementarity effects and the substitution effects, respectively. They have input dimensions $n+1$, output dimension 1, and embedding dimension $n+1$. The third head represents the base utility of each item. It has input dimensions $n$, output dimension $d$, and embedding dimension $n$. Note that this multi-head construction can be equivalently represented as a single-head simple transformer by arranging the $Q$, $K$, $V$, and $u$ matrices for all three heads into block form. However, for clarity, we present the proof by describing each head separately.

Let $A, B \in \mathbb{R}_+^{n\times n}$ be two matrices with positive entries such that $H_{ij} = \exp(A_{ij}) - \exp(B_{ij})$ for every $i, j \in [n]$. Let $M$ be a sufficiently large constant. Later we will set $M$ to be large enough so that the simple transformer approximates $g(S, i)$ to arbitrary precision.

**Head 1.** The first self-attention head has the following parameters:

- $Q^{(1)}, K^{(1)} \in \mathbb{R}^{(n+1)\times(n+1)}$ are set such that

$$Q^{(1)}(K^{(1)})^\top = \left[\begin{array}{c|c} & M \\ A & \vdots \\ & M \\ \hline 0 \cdots\cdots\cdots 0 & M \end{array}\right].$$

- $V^{(1)} \in \mathbb{R}^{n+1}$ is set to $V^{(1)}_i = 1$ for each $i \in [n]$ and $V^{(1)}_{n+1} = 0$. Set $u^{(1)} = 1$ so that $(V^{(1)}_i)^\top u^{(1)} = V^{(1)}_i$.
- Set $f^{(1)}_i(x) = \exp(M)x$ for every $i \in [n^2+1]$.
- For a subset $S \subset [n]$, we set $S^{(1)} \subset [n+1]$ where $S^{(1)} = S \cup \{n+1\}$.

We have

$$\operatorname{softmax}\big(X_{S^{(1)}} Q^{(1)} (X_{S^{(1)}} K^{(1)})^\top\big)_{ij} = \begin{cases} \dfrac{\exp(A_{ij})}{\sum_{j\in S}\exp(A_{ij}) + \exp(M) + (n-|S|)}, & i \in S,\ j \in S, \\ \dfrac{\exp(M)}{\sum_{j\in S}\exp(A_{ij}) + \exp(M) + (n-|S|)}, & i \in S,\ j = n+1, \\ \dfrac{1}{\sum_{j\in S}\exp(A_{ij}) + \exp(M) + (n-|S|)}, & i \in S,\ j \in [n]\setminus S, \\ \dfrac{1}{n+\exp(M)}, & i = n+1,\ j \in [n], \\ \dfrac{\exp(M)}{n+\exp(M)}, & i = n+1,\ j = n+1, \\ \dfrac{1}{n+1}, & i \in [n]\setminus S,\ j \in [n{+}1]. \end{cases}$$

Therefore, for every $i \in S$, we have

$$\mathrm{SA}_{Q^{(1)},K^{(1)},V^{(1)}}(X_{S^{(1)}})_i^\top u^{(1)} = \frac{\sum_{j\in S}\exp(A_{ij})}{\sum_{j\in S}\exp(A_{ij}) + \exp(M) + (n-|S|)}.$$

For any given $\epsilon > 0$, we can take $M$ to be sufficiently large such that

$$\frac{\sum_{j\in S}\exp(A_{ij}) - \frac{\epsilon}{2}}{\exp(M)} \le \mathrm{SA}_{Q^{(1)},K^{(1)},V^{(1)}}(X_{S^{(1)}})_i^\top u^{(1)} \le \frac{\sum_{j\in S}\exp(A_{ij})}{\exp(M)}.$$

Finally, we get

$$\sum_{j\in S}\exp(A_{ij}) - \frac{\epsilon}{3} \le \mathcal{T}_{Q^{(1)},K^{(1)},V^{(1)},f^{(1)}_1,\ldots,f^{(1)}_n,u^{(1)}}(X_{S^{(1)}})_i \le \sum_{j\in S}\exp(A_{ij}).$$

**Head 2.** The second self-attention head has exactly the same parameters, expect we replace $A$ with $B$ and we flip the sign of $f_i$:

- $Q^{(2)}, K^{(2)} \in \mathbb{R}^{(n+1)\times(n+1)}$ are set such that

$$Q^{(2)}(K^{(2)})^\top = \left[\begin{array}{c|c} & M \\ B & \vdots \\ & M \\ \hline 0 \cdots\cdots\cdots 0 & M \end{array}\right].$$

- $V^{(2)} \in \mathbb{R}^{n+1}$ is set to $V^{(2)}_i = 1$ for each $i \in [n]$ and $V^{(2)}_{n+1} = 0$. Set $u^{(2)} = 1$ so that $(V^{(1)}_i)^\top u^{(2)} = V^{(2)}_i$.
- Set $f^{(2)}_i(x) = -\exp(M)x$ for every $i \in [n^2+1]$.
- For a subset $S \subset [n]$, we set $S^{(2)} \subset [n+1]$ where $S^{(2)} = S \cup \{n+1\}$.

Then, similar to head 1, we get

$$-\sum_{j\in S}\exp(B_{ij}) \le \mathcal{T}_{Q^{(2)},K^{(2)},V^{(2)},f^{(2)}_1,\ldots,f^{(2)}_n,u^{(2)}}(X_{S^{(2)}})_i \le -\sum_{j\in S}\exp(B_{ij}) + \frac{\epsilon}{3}.$$

**Head 3.** The third self-attention head has the following parameters:

- $Q^{(3)}, K^{(3)} \in \mathbb{R}^{n\times n}$ are set such that

$$Q^{(3)}(K^{(3)})^\top = \begin{bmatrix} 0 & -M & \cdots & -M \\ -M & 0 & \cdots & -M \\ \vdots & \vdots & \ddots & \vdots \\ -M & -M & \cdots & 0 \end{bmatrix}.$$

- $V^{(3)} \in \mathbb{R}^{n\times d}$ is set to $V_i^{(3)} = \hat{v}_i$ for each $i \in [n]$.
- $u^{(3)} \in \mathbb{R}^d$ is set to $u^{(3)} = \hat{u}$.
- Set $f_i^{(3)}(x) = x$ for every $i \in [n]$.
- For a subset $S \subset [n]$, we set $S^{(3)} = S$.

For every $i \in S$, we have

$$\mathrm{SA}_{Q^{(3)},K^{(3)},V^{(3)}}(X_{S^{(3)}})_i^\top u^{(3)} = \frac{(\hat{v}_i)^\top \hat{u} + (n-|S|)\exp(-M)}{1+(n-|S|)\exp(-M)}.$$

For any given $\epsilon > 0$, we can take $M$ to be sufficiently large such that

$$(\hat{v}_i)^\top \hat{u} - \frac{\epsilon}{3} \le \mathcal{T}_{Q^{(3)},K^{(3)},V^{(3)},f_1^{(3)},\ldots,f_n^{(3)},u^{(3)}}(X_{S^{(3)}})_i \le (\hat{v}_i)^\top \hat{u}.$$

**Complete the Proof.** Because $\exp(A_{ij}) - \exp(B_{ij}) = H_{ij}$ Combining the above three self-attention heads, we have

$$\hat{v}_i^\top \hat{u} + \sum_{j\in S} H_{ij} - \epsilon \le \sum_{\ell=1}^{3} \mathcal{T}_{Q^{(\ell)},K^{(\ell)},V^{(\ell)},f_1^{(\ell)},\ldots,f_n^{(\ell)},u^{(\ell)}}(X_{S^{(\ell)}})_i \le \hat{v}_i^\top \hat{u} + \sum_{j\in S} H_{ij}.$$

Therefore the three attention heads approximate $g(S,i)$ to arbitrary precision.

## B Proof of Proposition 2

**Proof of Proposition 2 (a).** We prove that for any constant $M \ge 3$, there exists a number $c(M) > 0$ for which the following holds. For any $d_{kq}$ such that $\exp(c(M)\cdot d_{kq}) \le n-1$ and any $k \ge M+1$, Problem (P) subsumes the problem of finding a largest clique in a graph with

- $n-1$ vertices,
- $\exp(c(M)\cdot d_{kq})$ disjoint cliques,
- and all cliques have size at least $k-M$ and at most $k-1$.

Fix any constant $M \ge 1$. We construct an instance of Problem (P) by applying the Johnson-Lindenstrauss Lemma:

**Lemma 1 (Johnson-Lindenstrauss Lemma)** *For any $0 < \epsilon < 1$ and any set $S$ of $m$ points in $\mathbb{R}^n$, there exists a universal constant $c > 0$ and a linear function $f : \mathbb{R}^n \to \mathbb{R}^d$ with $d = c\epsilon^{-2}\log(m)$ such that*

$$(1-\epsilon)\|x_i\|_2^2 \le \|f(x_i)\|_2^2 \le (1+\epsilon)\|x_i\|_2^2$$

*for all $x_i \in S$ and*

$$(1-\epsilon)\|x_i - x_j\|_2^2 \le \|f(x_i) - f(x_j)\|_2^2 \le (1+\epsilon)\|x_i - x_j\|_2^2$$

*for all $x_i, x_j \in S$.*

Take $N$ to be large enough such that $\exp(-N) \le 1/2M$. Take $0 < \delta < 1$ to be small enough such that $\exp(-\delta N) \ge 1 - \exp(-N)$. Then $N$ and $\delta$ can be chosen to be both only depend on $M$. We obtain the following corollary:

**Corollary 1 (Corollary of Lemma 1.)** *There exists a number $c(M) > 0$ and a set $S$ of $\exp(c(M) \cdot d_{kq})$ unit vectors in $\mathbb{R}^{d_{kq}}$ such that $|u_i^\top u_j| \le \delta$ for every $u_i, u_j \in S$ and $u_i \neq u_j$.*

**Proof of Corollary 1.** Let $c > 0$ be the universal constant in Lemma 1 and let $n = \exp(c^{-1}(\delta/4)^2 \cdot d_{kq})$. Let $c(M) = c^{-1}(\delta/4)^2$, then $n = \exp(c(M) \cdot d_{kq})$. Because $\delta > 0$ only depends on $M$, we have $c(M) > 0$ also only depends on $M$. Consider $\{e_i\}_{i=1}^n \subset \mathbb{R}^n$ where $e_i \in \mathbb{R}^n$ is the unit vector where the $i$-th entry equals to 1 and all other entries equal to 0. Then we have $\|e_i - e_j\|_2^2 = 2$ for every $e_i \neq e_j$. By Lemma 1, there exists a linear function $f : \mathbb{R}^n \to \mathbb{R}^{d_{kq}}$ such that

$$\left(1 - \frac{\delta}{4}\right) \le \|f(e_i)\|_2^2 \le \left(1 + \frac{\delta}{4}\right)$$

for all $e_i$ and

$$2\left(1 - \frac{\delta}{4}\right) \le \|f(e_i) - f(e_j)\|_2^2 \le 2\left(1 + \frac{\delta}{4}\right)$$

for all $e_i \neq e_j$.

For every $e_i \neq e_j$, because

$$\|f(e_i) - f(e_j)\|_2^2 = \|f(e_i)\|_2^2 + \|f(e_j)\|_2^2 - 2f(e_i)^\top f(e_j),$$

we have

$$\begin{aligned} f(e_i)^\top f(e_j) &= (\|f(e_i)\|_2^2 + \|f(e_j)\|_2^2 - \|f(e_i) - f(e_j)\|_2^2)/2 \\ &\le \left(\left(1 + \frac{\delta}{4}\right) + \left(1 + \frac{\delta}{4}\right) - 2\left(1 - \frac{\delta}{4}\right)\right)/2 \\ &= \frac{\delta}{2}, \end{aligned}$$

and

$$\begin{aligned} f(e_i)^\top f(e_j) &= (\|f(e_i)\|_2^2 + \|f(e_j)\|_2^2 - \|f(e_i) - f(e_j)\|_2^2)/2 \\ &\ge \left(\left(1 - \frac{\delta}{4}\right) + \left(1 - \frac{\delta}{4}\right) - 2\left(1 + \frac{\delta}{4}\right)\right)/2 \\ &= -\frac{\delta}{2}. \end{aligned}$$

Let $u_i = f(e_i)/\|f(e_i)\|_2$ for all $i \in [n]$, then each $u_i$ is a unit vector. Moreover, for every $u_i \neq u_j$,

$$u_i^\top u_j = \frac{f(e_i)^\top f(e_j)}{\|f(e_i)\|_2 \|f(e_j)\|_2} \le \frac{\frac{\delta}{2}}{1 - \frac{\delta}{4}} = \delta\left(\frac{2}{4 - \delta}\right) < \delta,$$

where the last inequality follows since $0 < \delta < 1$. Similarly,

$$u_i^\top u_j = \frac{f(e_i)^\top f(e_j)}{\|f(e_i)\|_2 \|f(e_j)\|_2} \ge \frac{-\frac{\delta}{2}}{1 - \frac{\delta}{4}} = -\delta\left(\frac{2}{4 - \delta}\right) > -\delta.$$

Therefore $|u_i^\top u_j| \le \delta$ for every $u_i \neq u_j$. Take $S = \{u_i\}_{i=1}^n$ gives the desired set of unit vectors.

Corollary 1 states that there exists a set of unit vectors in $\mathbb{R}^{d_{kq}}$, with size exponential in $d_{kq}$, where all unit vectors in the set are approximately orthonormal.

Fix any $d_{kq}$ such that $\exp(c(M) \cdot d_{kq}) \le n - 1$ and any $k \ge M + 1$. Let $G$ be a graph with $n - 1$ vertices $v_1, \ldots, v_{n-1}$ and $\ell = \exp(c(M) \cdot d_{kq})$ disjoint cliques, each of size at least $k - M$ and at most $k - 1$. Let $I_1, \ldots, I_\ell \subset [n-1]$ be the index sets of vertices corresponding to these $\ell$ cliques. That is, $\{v_i\}_{i \in I_{\ell'}}$ forms a clique for each $\ell' \in [\ell]$. Without loss of generality we assume $\ell' \in I_{\ell'}$ for every $\ell' \in [\ell]$.

By Corollary 1, there exists $\ell$ unit vectors $u_1, \ldots, u_\ell \in \mathbb{R}^{d_{kq}}$ such that $|u_i^\top u_j| \le \delta$ for $i \neq j$. Let $u_{\ell+1}, \ldots, u_{n-1}$ be unit vectors such that $u_i = u_{\ell'}$ for $i \in I_{\ell'}$. That is, for indices $i, j$ such that $v_i$ and $v_j$ are in the same clique, we have $u_i = u_j$. Let $U \in \mathbb{R}^{(n-1)\times d_{kq}}$ where the $i$-th row of $U$ is $u_i^\top$. Let $A = UU^\top \in \mathbb{R}^{(n-1)\times(n-1)}$, then $A$ has rank at most $d_{kq}$.

Because $A$ has rank at most $d_{kq}$, there exists $Q, K \in \mathbb{R}^{n \times d_{kq}}$ such that

$$QK^\top = \left[\begin{array}{c|c} -N \cdot A & \begin{matrix} 0 \\ \vdots \\ 0 \end{matrix} \\ \hline 0 \cdots\cdots\cdots 0 & 0 \end{array}\right].$$

We create an explicit instance of $\mathsf{P}$ similar to the instance in the proof of Proposition 2 (a):

- $W = \text{softmax}(QK^\top)$. For simplicity of exposition, we replace $W$ with $W'$, defined to be

$$W'_{ij} = \begin{cases} 1 & \text{for } i = n \text{ or } j = n \\ \exp(-N|u_i^\top u_j|) & \text{for } i, j \neq n \text{ and } A_{ij} \neq 1 \\ \exp(-N) & \text{for } i, j \neq n \text{ and } A_{ij} = 1, \end{cases}$$

  As a sanity check, $W$ is simply $W'$ with each row rescaled to sum to one.
- $d_v = 1$, and we set $u = 1$, so that $Vu = V$.
- $V \in \mathbb{R}^{n \times 1}$ is set according to $V_i = 1$ for $i = 1, \dots, n-1$, and $V_n = 2$.
- For $i = 1, \dots, n-1$, we set $f_i(\cdot)$ to be:

$$f_i(x) = \begin{cases} 0 & \text{if } 0 \leq x \leq \frac{2+\exp(-N)k+\frac{1}{4}}{1+\exp(-N)k+\frac{1}{4}} \\ \dfrac{x - \frac{2+\exp(-N)k+\frac{1}{4}}{1+\exp(-N)k+\frac{1}{4}}}{\frac{2+\exp(-N)k}{1+\exp(-N)k} - \frac{2+\exp(-N)k+\frac{1}{4}}{1+\exp(-N)k+\frac{1}{4}}} & \text{if } \frac{2+\exp(-N)k+\frac{1}{4}}{1+\exp(-N)k+\frac{1}{4}} < x < \frac{2+\exp(-N)k}{1+\exp(-N)k} \\ 1 & \text{if } x \geq \frac{2+\exp(-N)k}{1+\exp(-N)k}. \end{cases}$$

  Because $\frac{2+\exp(-N)k}{1+\exp(-N)k} > \frac{2+\exp(-N)k+\frac{1}{4}}{1+\exp(-N)k+\frac{1}{4}}$, we have $f_i(\cdot)$ is continuous piece-wise linear for every $i = 1, \dots, n-1$. We set $f_n(x) = 0$ for every $x$.

Now notice that because $f_i(x) \leq 1$ for every $i = 1, \dots, n-1$, and $f_n(x) = 0$, the optimal value of this instance of Problem $\mathsf{P}$ is at most $k-1$. It will suffice to show that the largest clique $G$ has size $k'$ if and only if the optimal value is $k'$. We prove both directions separately.

**If $G$ has a clique of size $k'$, then the optimal value is at least $k'$:** Suppose $G$ has a clique of size $k'$. Without loss of generality, let $I_1 \subset [n-1]$ correspond the vertex set of a clique of size $k'$ in $G$. Consider the solution $x^*$, where $x_i^* = 1$ for $i \in I_1 \cup \{n\}$. Because $|I_1 \cup \{n\}| \leq k$, the solution $x^*$ is feasible. We will show that the objective value of $\mathsf{P}$ at $x^*$ is $k'$ (and thus the optimal value is at least $k'$).

Because $W'_{ij} = \exp(-N)$ whenever $i, j \in I_1$, we have

$$\begin{aligned} \sum_{i=1}^n x_i^* f_i\left(\frac{(w'_i \odot Vu)^\top x^*}{{w'_i}^\top x^*}\right) &= \sum_{i \in I_1} f_i\left(\frac{(w'_i \odot Vu)^\top x^*}{{w'_i}^\top x^*}\right) \\ &= \sum_{i \in I_1} f_i\left(\frac{2 + k' \exp(-N)}{1 + k' \exp(-N)}\right). \end{aligned}$$

Since $k' \leq k-1$. It follows that

$$\frac{2 + k' \exp(-N)}{1 + k' \exp(-N)} > \frac{2 + k \exp(-N)}{1 + k \exp(-N)}.$$

Therefore

$$\sum_{i=1}^n x_i^* f_i\left(\frac{(w'_i \odot Vu)^\top x^*}{{w'_i}^\top x^*}\right) = \sum_{i \in I_1} f_i\left(\frac{2 + k' \exp(-N)}{1 + k' \exp(-N)}\right) = |I_1| = k'.$$

**If the largest clique in $G$ has size $k'$, then the optimal value is at most $k'$:** By our assumption we have $k' \geq k - M$. Suppose for the sake of contradiction that the optimal value is $k^* > k'$. Let $x^*$ be an optimal solution to P. Notice if $x_n^* = 0$, then since $(w_i' \odot Vu)_j = W_{ij}'$ for all $j \in [n-1]$, we have

$$\sum_{i=1}^{n} x_i^* f_i \left( \frac{(w_i' \odot Vu)^\top x^*}{{w_i'}^\top x^*} \right) = \sum_{i=1}^{n} x_i^* f_i(1) = 0.$$

Therefore we must have $x_n^* = 1$.

Let $I \subset [n-1]$ be the index set where $x_i^* = 1$ for $i \in I$. Because $f_i(x) \leq 1$ for every $i = 1, \ldots, n-1$, we have $|I| \geq k^*$. Let $d(i)$ be the degree of vertex $i$ in the induced subgraph of $G$ with vertex set $\{v_i\}_{i \in I}$. Because $G$ consists of disjoint cliques with size at most $k'$, we have $d(i) \leq k' - 1$ for every $i \in C$.

Fix any $i \in I$ such that

$$f_i \left( \frac{(w_i' \odot Vu)^\top x^*}{{w_i'}^\top x^*} \right) > 0,$$

or, equivalently,

$$\frac{(w_i' \odot Vu)^\top x^*}{{w_i'}^\top x^*} > \frac{2 + \exp(-N) + \frac{1}{4}}{1 + \exp(-N) + \frac{1}{4}}.$$

Because $\exp(-N|u_i^\top u_j|) \geq \exp(-\delta N)$ for every $i, j \neq n$ and $A_{ij} \neq 1$, we have

$$\begin{aligned}
\frac{(w_i' \odot Vu)^\top x^*}{{w_i'}^\top x^*} &= \frac{2 + \sum_{j \in I, A_{ij} \neq 1} \exp(-N|u_i^\top u_j|) + (d(i)+1)\exp(-N)}{1 + \sum_{j \in I, A_{ij} \neq 1} \exp(-N|u_i^\top u_j|) + (d(i)+1)\exp(-N)} \\
&\leq \frac{2 + (|I| - d(i) - 1)\exp(-\delta N) + (d(i)+1)\exp(-N)}{1 + (|I| - d(i) - 1)\exp(-\delta N) + (d(i)+1)\exp(-N)} \\
&\leq \frac{2 + (k^* - d(i) - 1)\exp(-\delta N) + (d(i)+1)\exp(-N)}{1 + (k^* - d(i) - 1)\exp(-\delta N) + (d(i)+1)\exp(-N)} \\
&= \frac{2 + \exp(-\delta N)k^* - (\exp(-\delta N) - \exp(-N))(d(i)+1)}{1 + \exp(-\delta N)k^* - (\exp(-\delta N) - \exp(-N))(d(i)+1)} \\
&\leq \frac{2 + \exp(-\delta N)k^* - (\exp(-\delta N) - \exp(-N))k'}{1 + \exp(-\delta N)k^* - (\exp(-\delta N) - \exp(-N))k'},
\end{aligned}$$

where the second inequality follows since $|I| \geq k^*$, and the last inequality follows since $\exp(-\delta N) - \exp(-N) > 0$ and $d(i) \leq k' - 1$. Therefore

$$\frac{2 + \exp(-\delta N)k^* - (\exp(-\delta N) - \exp(-N))k'}{1 + \exp(-\delta N)k^* - (\exp(-\delta N) - \exp(-N))k'} > \frac{2 + \exp(-N)k + \frac{1}{4}}{1 + \exp(-N)k + \frac{1}{4}}.$$

So we have

$$\exp(-\delta N)k^* - (\exp(-\delta N) - \exp(-N))k' < \exp(-N)k + \frac{1}{4}.$$

Rearrange the above inequality gives

$$\exp(-\delta N)(k^* - k') < \exp(-N)(k - k') + \frac{1}{4}.$$

Because $k^* \geq k' + 1$ and $k' \geq k - M$, we get

$$\exp(-\delta N) < \exp(-N)M + \frac{1}{4}.$$

However, since $\exp(-N) \leq 1/2M$ and $\exp(-\delta N) \geq 1 - 1/2M$, we have

$$\exp(-\delta N) \geq 1 - \frac{1}{2M} \geq \frac{5}{6} > \exp(-N)M + \frac{1}{4}.$$

A contradiction. Therefore $k^* \leq k'$.

**Proof of Proposition 2 (b).**

Our proof is based on a reduction from Problem (P) to the well-known *Multi-dimensional Knapsack Problem* (MDKP), defined as follows:

**Definition 2 (Multi-dimensional Knapsack Problem)** *The Multi-dimensional Knapsack Problem (*MDKP*) with $c$ items and $d$ dimensions is defined as:*

$$\begin{aligned} \max \quad & f_{\mathsf{MDKP}}(x) = p^\top x \\ \text{s.t.} \quad & y_i^\top x \le t_i \quad \forall i \in [d], \\ & x \in \{0,1\}^c. \end{aligned} \tag{MDKP}$$

*Here, $p \in \mathbb{N}^c$, and for all $i \in [d]$, we have $y_i \in \mathbb{Z}^c_{\ge 0}$ and $t_i \in \mathbb{Z}_{\ge 0}$.*[5]

We present the reduction in the following proposition:

**Proposition 3** *Consider Problem* (P)*, written in the equivalent form* P*, reproduced below, which has parameters $n$, $k$, and $r_+$, where $r_+$ is the non-negative rank of $W$:*

$$\begin{aligned} \max \quad & f_{\mathsf{P}(I)}(x) = \sum_{i=1}^n x_i f_i\left(\frac{(w_i \odot Vu)^\top x}{w_i^\top x}\right) \\ \text{s.t.} \quad & x \in \{0,1\}^n, \quad 1 \le e^\top x \le k. \end{aligned} \tag{P}$$

*Now consider an instance of* MDKP *with parameters $c$ and $d$. Suppose there exists an algorithm* ALG *for solving* P *with parameters $n = k = c + d + 1$ and $\min\{c, d\} \le r_+ \le c + d + 1$, such that for any sufficiently small $\epsilon > 0$, the algorithm satisfies*

$$\mathrm{ALG}_{\mathsf{P}} \ge (1 - \epsilon)\, \mathrm{OPT}_{\mathsf{P}}$$

*with runtime $T$. Then we can construct an algorithm* ALG$'$ *for solving* MDKP *with parameters $c$ and $d$, such that for the same $\epsilon$, it satisfies*

$$\mathrm{ALG}'_{\mathsf{MDKP}} \ge (1 - \epsilon)\, \mathrm{OPT}_{\mathsf{MDKP}}$$

*with runtime $O(T)$.*

**Proof of Proposition 3.**

Fix an MDKP instance. We create an instance of P as follows:

- $n = k = d + c + 1$. Here, out of the $n$ total variables, the first $d$ variables will correspond to the $d$ constraints of MDKP, the $c$ variables after that will correspond to the $c$ variables of MDKP, and the last variable will be a dummy variable that must be selected in any optimal solution of P.
- For each $i = 1, \ldots, d$, set
$$w_{ij} = \begin{cases} 0 & \text{for } j = 1, \ldots, d \\ y_{i,j-d} & \text{for } j = d+1, \ldots, d+c \\ 1 & \text{for } j = d+c+1. \end{cases}$$
  That is, in block notation,
$$w_i^\top = \left[\ 0 \cdots 0 \ \middle|\ y_i^\top \ \middle|\ 1 \ \right].$$
  Then we have
$$w_i^\top x = \sum_{j=1}^c y_{ij} x_{d+j} + x_{d+c+1}.$$
  Moreover, since $y_i \in \mathbb{Z}^c_{\ge 0}$ for each $i \in [d]$, the non-negative rank of the sub-matrix of $W$ consisting of its first $d$ rows is at most $\min\{c, d\}$.
- For each $i = d+1, \ldots, d+c+1$, set $w_i \in \mathbb{R}^{d+c+1}_{\ge 0}$'s to be any non-zero vectors such that $W$ has the desired non-negative rank $r_+$. This is possible since $\min\{c, d\} \le r_+ \le c+d+1$.
- $d_v = 1$, and we set $u = 1$, so that $Vu = V$.

[5]We assume $p_{\min} > 0$ without loss of generality. If $p_j = 0$ for some $j \in [c]$, there always exists an optimal solution with $x_j = 0$. Hence, we can safely ignore the $j$-th entry of $p$, $x$, and each $y_i$.

- $V \in \mathbb{R}^{(d+c+1)\times 1}$ is set to be $V_i = 0$ for $i = 1, \dots, d$, and $V_i = 1$ for $i = d+1, \dots, d+c$, and $V_{d+c+1} = 2$.
- For each $i = 1, \dots, d$, set

$$f_i(x) = \begin{cases} -\left(\sum_{j=1}^{c} p_j + 1\right) & \text{for } x \le \frac{t_i+3}{t_i+2} \\ \dfrac{x - \frac{t_i+2}{t_i+1}}{\frac{t_i+2}{t_i+1} - \frac{t_i+3}{t_i+2}} \cdot \left(\sum_{j=1}^{c} p_j + 1\right) & \text{for } \frac{t_i+3}{t_i+2} < x < \frac{t_i+2}{t_i+1} \\ 0 & \text{for } x \ge \frac{t_i+2}{t_i+1}. \end{cases}$$

  Then $f_i(x)$ is continuous piecewise-linear.
- For each $i = d+1, \dots, d+c$, set $f_i(x) = p_{i-d}$. Set $f_{d+c+1}(x) = 0$.

Because $p_{\min} > 0$ and $\vec{0} \in \{0,1\}^c$ is a feasible solution to MDKP, we have $\mathrm{OPT}_{\mathsf{MDKP}} = 0$ if and only if $\vec{0} \in \{0,1\}^c$ is the only feasible solution to MDKP. Moreover, if $\vec{0} \in \{0,1\}^c$ is not the only feasible solution to MDKP, then $\mathrm{OPT}_{\mathsf{MDKP}} \ge p_{\min}$.

Our proof relies on the following lemma.

**Lemma 2** *Let $x \in \{0,1\}^n$ be a feasible solution to* P *such that $f_{\mathsf{P}}(x) > 0$. Then we can construct a feasible solution $z \in \{0,1\}^c$ to* MDKP *such that $f_{\mathsf{P}}(x) = f_{\mathsf{MDKP}}(z)$.*

*Conversely, let $z \in \{0,1\}^c$ be a feasible solution to* MDKP *such that $f_{\mathsf{MDKP}}(z) > 0$. Then we can construct a feasible solution $x \in \{0,1\}^n$ to* P *such that $f_{\mathsf{P}}(x) = f_{\mathsf{MDKP}}(z)$.*

Lemma 2 gives a correspondence between solutions to P and solutions to MDKP. In particular, as a corollary, Lemma 2 implies the relationship between the optimal objective values of P and MDKP.

**Corollary 2 (Corollary of Lemma 2.)** $\mathrm{OPT}_{\mathsf{P}} \le 0$ *if and only if* $\mathrm{OPT}_{\mathsf{MDKP}} = 0$. *Moreover, suppose* $\mathrm{OPT}_{\mathsf{MDKP}} > 0$. *Then* $\mathrm{OPT}_{\mathsf{P}} = \mathrm{OPT}_{\mathsf{MDKP}}$.

**Proof of Corollary 2.** Suppose $\mathrm{OPT}_{\mathsf{P}} > 0$. Let $x^*$ be an optimal solution to P. By Lemma 2 there exists a feasible solution $z$ to MDKP such that $f_{\mathsf{MDKP}}(z) > 0$ and

$$\mathrm{OPT}_{\mathsf{P}} = f_{\mathsf{P}}(x^*) = f_{\mathsf{MDKP}}(z) \le \mathrm{OPT}_{\mathsf{MDKP}}.$$

Conversely, suppose $\mathrm{OPT}_{\mathsf{MDKP}} > 0$. Let $z^*$ be an optimal solution to MDKP. By Lemma 2 there exists a feasible solution $x$ to MDKP such that

$$\mathrm{OPT}_{\mathsf{MDKP}} = f_{\mathsf{MDKP}}(z^*) = f_{\mathsf{P}}(x) \le \mathrm{OPT}_{\mathsf{P}}.$$

We also get

$$f_{\mathsf{P}}(x) \ge f_{\mathsf{MDKP}}(z^*) > 0.$$

Before proving Lemma 2, we first show that it is sufficient to prove Lemma 2. Assume we have an algorithm ALG for solving P such that, for any sufficiently small $\epsilon > 0$, we have ALG satisfies

$$\mathrm{ALG}_{\mathsf{P}} \ge (1-\epsilon)\,\mathrm{OPT}_{\mathsf{P}}.$$

Then our proposed algorithm ALG$'$ for solving MDKP works as follows:

- If $\mathrm{ALG}_{\mathsf{P}} \le 0$, ALG$'$ outputs $\vec{0} \in \{0,1\}^c$.
- If $\mathrm{ALG}_{\mathsf{P}} > 0$, let $x$ be the solution to P given by $\mathrm{ALG}_{\mathsf{P}}$. Then ALG$'$ outputs the solution $z \in \{0,1\}^c$ to MDKP given by Lemma 2. That is, $z \in \{0,1\}^c$ that satisfies

$$f_{\mathsf{P}}(x) = f_{\mathsf{MDKP}}(z).$$

**Performance Guarantee of ALG$'$:** Suppose $\mathrm{OPT}_{\mathsf{MDKP}} = 0$. Then by Corollary 2,

$$\mathrm{ALG}_{\mathsf{P}} \le \mathrm{OPT}_{\mathsf{P}} \le 0.$$

Therefore ALG$'$ correctly outputs $\vec{0} \in \{0,1\}^c$.

On the other hand, suppose $\mathrm{OPT}_{\mathsf{MDKP}} > 0$. Then by Corollary 2,

$$\mathrm{ALG}'_{\mathsf{MDKP}} = f_{\mathsf{MDKP}}(z) = f_{\mathsf{P}}(x) \geq (1-\epsilon)\mathrm{OPT}_{\mathsf{P}} = (1-\epsilon)\mathrm{OPT}_{\mathsf{MDKP}}.$$

This proves the desired performance guarantee of $\mathrm{ALG}'$. To finish the proof, we prove Lemma 2 and analyze the runtime of $\mathrm{ALG}'$.

**Proof of Lemma 2.** Recall for each $i = 1, \dots, d$, we set

$$w_{ij} = \begin{cases} 0 & \text{for } j = 1, \dots, d \\ y_{i,j-d} & \text{for } j = d+1, \dots, d+c \\ 1 & \text{for } j = d+c+1. \end{cases}$$

Also, $(Vu)_i = 0$ for $i = 1, \dots, d$, and $(Vu)_i = 1$ for $i = d+1, \dots, d+c$, and $(Vu)_{d+c+1} = 2$. Therefore for $i = 1, \dots, d$ we have

$$\frac{(w_i \odot Vu)^\top x}{w_i^\top x} = \frac{\sum_{j=1}^c y_{ij}x_{d+j} + 2x_{d+c+1}}{\sum_{j=1}^c y_{ij}x_{d+j} + x_{d+c+1}}.$$

Recall for each $i = d+1, \dots, d+c$ we set $f_i(x) = p_{i-d}$, and we set $f_{d+c+1}(x) = 0$. Therefore we have

$$\begin{aligned} f_{\mathsf{P}}(x) &= \sum_{i=1}^{c+d+1} x_i f_i\left(\frac{(w_i \odot Vu)^\top x}{w_i^\top x}\right) \\ &= \sum_{i=1}^{d} x_i f_i\left(\frac{\sum_{j=1}^c y_{ij}x_{d+j} + 2x_{d+c+1}}{\sum_{j=1}^c y_{ij}x_{d+j} + x_{d+c+1}}\right) + \sum_{i=1}^{c} x_{d+i}p_i. \end{aligned}$$

First, let $x \in \{0,1\}^n$ be a feasible solution to $\mathsf{P}(I)$ such that $f_{\mathsf{P}}(x) > 0$. Let $z \in \{0,1\}^c$ where $z_j = x_{d+j}$ for $j \in [c]$. We claim that $z$ is a feasible solution to $\mathsf{MDKP}$ and

$$f_{\mathsf{P}}(x) = f_{\mathsf{MDKP}}(z).$$

Because $\sum_{i=1}^c x_{d+i}p_i < \sum_{j=1}^c p_j + 1$, we must have $x_i = 1$ for every $i \in [d]$. Moreover, if $x_{d+c+1} = 0$, then

$$f_i\left(\frac{\sum_{j=1}^c y_{ij}x_{d+j} + 2x_{d+c+1}}{\sum_{j=1}^c y_{ij}x_{d+j} + x_{d+c+1}}\right) = f_i(1) < 0$$

for every $i \in [d]$. Hence we must have $x_{d+c+1} = 1$. Because $y_i \in \mathbb{Z}^c_{\geq 0}$ and $t_i \in \mathbb{Z}_{\geq 0}$ for all $i \in [d]$, if $\sum_{j=1}^c y_{ij}x_{d+j} > t_i$, we must have $\sum_{j=1}^c y_{ij}x_{d+j} \geq t_i + 1$. Then

$$f_i\left(\frac{\sum_{j=1}^c y_{ij}x_{d+j} + 2x_{d+c+1}}{\sum_{j=1}^c y_{ij}x_{d+j} + x_{d+c+1}}\right) \leq f_i\left(\frac{t_i+3}{t_i+2}\right) = -\left(\sum_{j=1}^c p_j + 1\right).$$

Therefore we must have $\sum_{j=1}^c y_{ij}x_{d+j} \leq t_i$ for all $i \in [d]$. Hence $\sum_{j=1}^c y_{ij}z_j \leq t_i$ for all $i \in [d]$, which shows $z$ is a feasible solution to $\mathsf{MDKP}$. Finally, because

$$\frac{\sum_{j=1}^c y_{ij}x_{d+j} + 2x_{d+c+1}}{\sum_{j=1}^c y_{ij}x_{d+j} + x_{d+c+1}} \geq \frac{t_i+2}{t_i+1}$$

for all $i \in [d]$, we have

$$f_i\left(\frac{\sum_{j=1}^c y_{ij}x_{d+j} + 2x_{d+c+1}}{\sum_{j=1}^c y_{ij}x_{d+j} + x_{d+c+1}}\right) = 0$$

for all $i \in [d]$. Therefore

$$\begin{aligned} f_{\mathsf{P}}(x) &= \sum_{i=1}^{d} x_i f_i\left(\frac{\sum_{j=1}^c y_{ij}x_{d+j} + 2x_{d+c+1}}{\sum_{j=1}^c y_{ij}x_{d+j} + x_{d+c+1}}\right) + \sum_{i=1}^{c} x_{d+i}p_i \\ &= \sum_{i=1}^{c} z_i p_i \\ &= f_{\mathsf{MDKP}}(z). \end{aligned}$$

Conversely, let $z \in \{0,1\}^c$ be a feasible solution to MDKP such that $f_{\mathsf{MDKP}}(z) > 0$. Let $x$ be a solution to P where $x_i = 1$ for $i = 1, \dots, d$ and $i = d + c + 1$, and $x_{d+i} = z_i$ for $i = 1, \dots, c$. We claim that $x$ is a feasible solution to MDKP and

$$f_{\mathsf{P}}(x) = f_{\mathsf{MDKP}}(z).$$

Because $z$ is a feasible solution to MDKP, we have $\sum_{j=1}^{c} y_{ij} z_j \leq t_i$ for all $i \in [d]$. Therefore $\sum_{j=1}^{c} y_{ij} x_{d+j} \leq t_i$ for all $i \in [d]$. Then

$$\frac{\sum_{j=1}^{c} y_{ij} x_{d+j} + 2x_{d+c+1}}{\sum_{j=1}^{c} y_{ij} x_{d+j} + x_{d+c+1}} \geq \frac{t_i + 2}{t_i + 1}$$

for all $i \in [d]$. Therefore we have

$$f_i\left(\frac{\sum_{j=1}^{c} y_{ij} x_{d+j} + 2x_{d+c+1}}{\sum_{j=1}^{c} y_{ij} x_{d+j} + x_{d+c+1}}\right) = 0$$

for all $i \in [d]$. Hence

$$\begin{aligned} f_{\mathsf{P}}(x) &= \sum_{i=1}^{d} x_i f_i\left(\frac{\sum_{j=1}^{c} y_{ij} x_{d+j} + 2x_{d+c+1}}{\sum_{j=1}^{c} y_{ij} x_{d+j} + x_{d+c+1}}\right) + \sum_{i=1}^{c} x_{d+i} p_i \\ &= \sum_{i=1}^{c} z_i p_i \\ &= f_{\mathsf{MDKP}}(z). \end{aligned}$$

Given an MDKP instance, our construction of the corresponding P instance takes $O(1)$ runtime. Also, the procedure of the construction of a feasible solution $z \in \{0,1\}^c$ to MDKP given a feasible solution to P described in Lemma 2 takes $O(1)$ runtime: we simply set $z \in \{0,1\}^c$ where $z_j = x_{d+j}$ for $j \in [c]$. Therefore, if ALG has runtime $T$, our ALG$'$ has runtime $O(T)$.

By Proposition 3, any $(1-\epsilon)$-approximation scheme for P can be directly translated into a $(1-\epsilon)$-approximation scheme for MDKP with comparable runtime. Therefore, many hardness results for MDKP carry over directly to P. We cite such an result below:

**Proposition 4 (Corollary of [13])** *For general $d$,* MDKP *admits no $(1-\epsilon)$-approximation scheme with runtime*

$$k^{o(\sqrt{d})},$$

*assuming Gap Exponential Time Hypothesis holds.*

This result, along with Proposition 3, gives our desired lower bound statement in Proposition 2 (b).

## C Proof of Theorem 1

We first formally define the $\epsilon$-Approximate $k$-Nearest Neighbor Search Algorithm.

**Definition 3 ($\epsilon$-Approximate $k$-Nearest Neighbor Search Algorithm)** *An $\epsilon$-Approximate $k$-Nearest Neighbor search algorithm builds a data structure on any given set of points $\{v_1, \dots, v_n\} \subset \mathbb{R}^d$, and takes any query $u \in \mathbb{R}^d$, any $1 \leq k \leq n$, and any $\epsilon > 0$ as inputs. Let $\pi : [n] \to [n]$ be a permutation of the indices such that $v_{\pi(1)}^\top u \geq \cdots \geq v_{\pi(n)}^\top u$. Let $(i_1^*, \dots, i_k^*) = (\pi(1), \dots, \pi(k))$. The oracle outputs $k$ indices $i_1, \dots, i_k \in [n]$ such that $v_{i_j}^\top u \geq v_{i_j^*}^\top u - \epsilon$ for each $j = 1, \dots, k$ with expected amortized runtime $k$-ANN$(n, d, k, \epsilon)$.*

Many $\epsilon$-ANN searches can be naturally modified to perform the $\epsilon$-approximate $k$-Nearest Neighbor search, with a similar expected amortized runtime, that is, sub-linear in $n$.

For all $i \in [n]$ and $\epsilon > 0$, the function $f_i$ satisfies:

$$f_i(x - \epsilon) \geq (1 - g(\epsilon)) f_i(x) - h(\epsilon) \quad \text{for all } x,$$

where $0 \leq g(\epsilon) \leq 1$ and $h(\epsilon) \geq 0$ are non-negative functions of $\epsilon$. This parametrization captures the idea that a small additive error in the input $x$ leads to a controlled multiplicative and additive error in the output $f_i(x)$.

Recall that while our algorithm's runtime is parametrized by the non-negative rank of $W$, achieving sub-linear dependence on $n$ and polynomial dependence on $k$ requires that the non-negative rank of $W$ be small. Importantly, our main result does not require $W$ itself to have low non-negative rank, but only that $W$ can be well approximated entry-wise by a low non-negative rank matrix.

Formally, suppose there exists a non-negative matrix $W' \in \mathbb{R}^{n\times n}_{\geq 0}$ such that

$$1-\gamma \leq \frac{W_{ij}}{W'_{ij}} \leq 1+\gamma \quad \text{for all } i, j,$$

with $\operatorname{rank}_+(W') = r_+$.[6] Then our main result is parametrized by $r_+$ and $\gamma$. Just as in the case of the parameter $d_{kq}$, our guarantee is non-trivial when $r_+ = O(1)$.

We are now prepared to state our main result in full, which is that our algorithm achieves the following:

**Theorem 2** *Let $k$-ANN$(n, d, k, \epsilon)$ be the expected amortized runtime of an $\epsilon$-Approximate $k$-Nearest Neighbor search algorithm, which we assume is concave in $n$ (e.g. sub-linear suffices).*

*Let $\tau$ be the number of distinct functions among $f_1, \dots, f_n$. Suppose there exists $W' \in \mathbb{R}^{n\times n}_+$ such that $1-\gamma \leq W_{ij}/W'_{ij} \leq 1+\gamma$ for all $i, j$, and $W'$ has non-negative rank $r_+$ with an explicit non-negative factorization. Assume $g(x), h(x) = O(x)$.*

*Given any $\epsilon > 0$, our algorithm* ALG *achieves*

$$\text{ALG} \geq (1-\epsilon)(1-\gamma)\cdot \text{OPT} - c\epsilon\gamma k$$

*where $c$ is a universal constant. Moreover, the expected amortized runtime of our algorithm is*

$$O\left(\epsilon^{-2d_{kq}} \cdot \tau \cdot k\text{-ANN}\left(\frac{n}{\tau}, d_v, k, \epsilon\right) + \epsilon^{-2d_{kq} r_+^2/\epsilon}(1+\gamma)^{r_+}(\tau k)^{r_+^2/\epsilon}\right),$$

*where the Big-Oh depends only on $\|Q\|_{2,\infty}, \|K\|_{2,\infty}, (Vu)_{\max}, W_{\min}$, and $\max_{i\in[n]}\{f_i((Vu)_{\max})\}$.*

## C.1 Phase One (Retrieval)

In phase one, our algorithm aims to identify a small subset $I \subset [n]$ of items such that the optimal objective value of $\mathsf{P}$ does not decrease significantly when restricted to $I$. To achieve this, our algorithm first partitions items offline based on their query vectors $q_i$, key vectors $k_i$, and reward functions $f_i$. Since both $q_i$ and $k_i$ lie in a space of dimension $d_{kq}$, the number of partitions is much smaller than $n$. Items in the same partition are designed to behave similarly under the self-attention layer. That is, they produce similar outputs when interacting with other items. When a user $u$ arrives, the algorithm applies an $\epsilon$-approximate $k$-nearest neighbor search within each partition to select at most $k$ items whose value vectors have the highest approximate inner product similarity with $u$. Because items within the same partition respond similarly under attention, user preferences within each partition are primarily determined by similarity to the user vector. Thus, our algorithm retrieves a small subset $I$ containing high-reward items tailored to the user.

**Proposition 5 (Phase One)** *Suppose we have an $\epsilon$-Approximate $k$-Nearest Neighbor search algorithm with expected amortized runtime $k$-ANN$(n, d, k, \epsilon)$. Let $\tau = |\bigcup_{i=1}^n \{f_i\}|$ be the number of distinct functions among $f_1, \dots, f_n$. Given any $\epsilon > 0$, our algorithm returns an index set $I \subset [n]$ such that*

$$|I| = \tau k \left\lceil \frac{140 \max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\})^2}{(Vu)_{\max}\cdot \epsilon} \right\rceil^{2d_{kq}},$$

*and the optimal value to the following Problem $\mathsf{P}(I)$*

$$\begin{aligned} \max \quad & f_{\mathsf{P}(I)}(x) = \sum_{i=1}^n x_i f_i\left(\frac{(w_i \odot Vu)^\top x}{w_i^\top x}\right) \qquad (\mathsf{P}(I)) \\ \text{s.t.} \quad & x \in \{0,1\}^n,\ x_i = 0 \, for\, i \notin I,\ 1 \leq e^\top x \leq k \end{aligned}$$

[6] Actually, we only require a particular sub-matrix to be of non-negative rank $r_+$. This sub-matrix is of significantly smaller size, corresponding to the entries which survives a certain pruning procedure.

*satisfies*

$$\mathrm{OPT}_{\mathsf{P}(I)} \geq (1-g(\epsilon))\mathrm{OPT}_{\mathsf{P}} - kh(\epsilon).$$

*Moreover, suppose $k$-ANN$(n,d,k,\epsilon)$ is concave in $n$. Then the expected amortized runtime of our algorithm is*

$$\left\lceil \frac{140(\max\{\|Q\|_{2,\infty},\|K\|_{2,\infty}\})^2 (Vu)_{\max}}{\epsilon}\right\rceil^{2d_{kq}} \tau$$
$$\cdot\, k\text{-ANN}\left(\frac{n}{\tau}, d_v, k, \frac{\epsilon}{35\max\{\|Q\|_{2,\infty},\|K\|_{2,\infty}\}(Vu)_{\max}}\right).$$

A few remarks are in order:

- Suppose $d_{kq} = o(\log n)$ and $\|Q\|_{2,\infty}, \|K\|_{2,\infty}, (Vu)_{\max}$ are all viewed as constants, then for any given constant $\epsilon > 0$, we have $|I| = \tau k n^{o(1)}$. Moreover, suppose $k$-ANN $(n, d_v, k, \epsilon)$ is sub-linear in $n$ when $d_v, k, \epsilon$ are fixed, then the expected amortized runtime of our algorithm is also sub-linear in $n$.
- Many $\epsilon$-Approximate $k$-Nearest Neighbor search algorithms have $k$-ANN $(n,d,k,\epsilon)$ sub-linear and concave in $n$.
- In practice, the number of distinct functions $\tau$ is typically very small – often just one.

**Proof of Proposition 5.**

Fix any $\epsilon > 0$. Let $\delta > 0$ be a parameter such that

$$\delta \leq \frac{1}{140\max\{\|Q\|_{2,\infty},\|K\|_{2,\infty}\}} \quad \text{and} \quad \epsilon \geq 34\delta\max\{\|Q\|_{2,\infty},\|K\|_{2,\infty}\}(Vu)_{\max} + \delta. \tag{1}$$

First we partition the rows of $Q$ and $K$. Because we can cover a ball with radius $\|Q\|_{2,\infty}$ in $\mathbb{R}^{d_{kq}}$ using $\lceil 4\|Q\|_{2,\infty}/\delta\rceil^{d_{kq}}$ number of balls with radius $\delta/2$ (see e.g. [47, 14, 42]), we can create a partition of the index set $[n]$ such that the size of the partition is $\lceil 4\|Q\|_{2,\infty}/\delta\rceil^{d_{kq}}$, and $\|q_i - q_{i'}\|_2 \leq \delta$ for every $i, i'$ in the same partition. [7] Similarly, we can create a partition of the index set $[n]$ such that the size of the partition is $\lceil 4\|K\|_{2,\infty}/\delta\rceil^{d_{kq}}$, and $\|k_j - k_{j'}\|_2 \leq \delta$ for every $j, j'$ in the same partition. Let $\mathcal{I} = \{I_1,\ldots,I_\ell\}$ be the product partition of the above two partitions. Then $\ell = \lceil 4\max\{\|Q\|_{2,\infty},\|K\|_{2,\infty}\}/\delta\rceil^{2d_{kq}}$, and for every $\ell' \in [\ell]$ and $i, i' \in I_{\ell'}$, we have $\|q_i - q_{i'}\|_2 \leq \delta$ and $\|k_i - k_{i'}\|_2 \leq \delta$. Therefore, for any $i, i' \in I_{\ell'}$ and $j, j' \in I_{\ell''}$, we have

$$\begin{aligned}
|q_i^\top k_j - q_{i'}^\top k_{j'}| &= |q_i^\top (k_j - k_{j'}) + k_{j'}^\top (q_i - q_{i'})| \\
&\leq |q_i^\top (k_j - k_{j'})| + |k_{j'}^\top (q_i - q_{i'})| \\
&\leq \|q_i\|_2 \|k_j - k_{j'}\|_2 + \|k_{j'}\|_2 \|q_i - q_{i'}\|_2 \\
&\leq 2\delta\max\{\|Q\|_{2,\infty},\|K\|_{2,\infty}\}.
\end{aligned}$$

Then, because $2\delta\max\{\|Q\|_{2,\infty},\|K\|_{2,\infty}\} \leq 1$, we have

$$\begin{aligned}
\left|\frac{\exp(q_i^\top k_j)}{\exp(q_{i'}^\top k_{j'})} - 1\right| &\leq \left|\exp(|q_i^\top k_j - q_{i'}^\top k_{j'}|) - 1\right| \\
&\leq |\exp(2\delta\max\{\|Q\|_{2,\infty},\|K\|_{2,\infty}\}) - 1| \\
&\leq 4\delta\max\{\|Q\|_{2,\infty},\|K\|_{2,\infty}\},
\end{aligned}$$

[7] We can create such a partition via the packing-covering duality in the following way (see e.g. Theorem 14.1, Theorem 14.2, and Example 14.1 of [52]). First we create a maximal packing of the ball $B(\vec{0}, \|Q\|_{2,\infty}) \subset \mathbb{R}^{d_{kq}}$ using balls with radius $\delta/4$ greedily, where we greedily choose points $x_1, x_2, \ldots$ such that the balls $B(x_i, \delta/4)$ are disjoint. We stop when no more such points can be added in $B(\vec{0}, \|Q\|_{2,\infty}) \subset \mathbb{R}^{d_{kq}}$. Then, by the packing-covering duality, the balls $B(x_i, \delta/2)$ cover $B(\vec{0}, \|Q\|_{2,\infty})$. This is because any uncovered point can be added to the packing we created before, which contradicts the maximality of the packing.

where the last inequality follows by $\exp(x) \leq 1 + 2x$ for $0 \leq x \leq 1$. Therefore, because $4\delta \max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\} \leq 1/35$, we have

$$\begin{aligned}
\left|\frac{w_{ij}}{w_{i'j'}} - 1\right| &= \left|\frac{\exp(q_i^\top k_j)/\sum_{m=1}^n \exp(q_i^\top k_m)}{\exp(q_{i'}^\top k_{j'})/\sum_{m'=1}^n \exp(q_{i'}^\top k_{m'})} - 1\right| \\
&\leq \left|(1 + 4\delta \max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\})^2/(1 - 4\delta \max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\})^2 - 1\right| \\
&\leq \frac{4 \cdot 35^2}{34^2} \cdot 4\delta \max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\} \\
&\leq 17\delta \max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\},
\end{aligned}$$

where the first inequality follows since $1 - 4\delta \max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\} \leq \exp(q_i^\top k_j)/\exp(q_{i'}^\top k_{j'}) \leq 1 + 4\delta \max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\}$, and the second inequality follows since $(1+x)^2/(1-x)^2 \leq 1 + (4a^2/(a-1)^2)x$ for every $a > 1$ and $0 \leq x \leq 1/a$.

Now we construct our desired index set $I$. Let $\{S_1, \ldots, S_\tau\}$ be a partition of the index set $[n]$ such that $f_i = f_j$ for every $\tau' \in [\tau]$ and $i, j \in S_{\tau'}$. We first construct an index set $J_{\tau'} \subset S_{\tau'}$ for each $\tau' \in [\tau]$, and then combine them to obtain $I$.

Fix any $\tau' \in \tau$. For each index set $I_{\ell'}$, we only choose $k$ indices out of it to include in $J_{\tau'}$, namely the $k$ indices that are approximately the $k$ highest indices in $\{(Vu)_i\}_{i \in S_{\tau'} \cap I_{\ell'}}$.[8] Specifically, for each $\ell' \in [\ell]$, we run the given $\delta$-Approximate $k$-Nearest Neighbor oracle with given set of points $\bigcup_{i \in S_{\tau'} \cap I_{\ell'}} \{V_i\} \subset \mathbb{R}^{d_v}$, and query $u$, numbers $k$ and $\delta$ as inputs. We let $J_{\tau'}$ be the collection of all output indices for each $\ell' \in [\ell]$. Then because $\ell = \lceil 4\max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\}/\delta \rceil^{2d_{kq}}$, we have

$$|J_{\tau'}| = k\lceil 4\max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\}/\delta \rceil^{2d_{kq}},$$

and the expect amortized runtime of constructing each $J_{\tau'}$ is

$$\lceil 4\max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\}/\delta \rceil^{2d_{kq}} \cdot k\text{-ANN}(|S_{\tau'}|, d_v, k, \delta).$$

Let $I = \cup_{\tau' \in [\tau]} J_{\tau'}$. Then we have

$$|I| = \tau k\lceil 4\max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\}/\delta \rceil^{2d_{kq}},$$

and the expect amortized runtime of constructing $I$ is

$$\begin{aligned}
&\sum_{\tau'=1}^{\tau} \lceil 4\max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\}/\delta \rceil^{2d_{kq}} \cdot k\text{-ANN}(|S_{\tau'}|, d_v, k, \delta) \\
&\leq \lceil 4\max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\}/\delta \rceil^{2d_{kq}} \tau \cdot k\text{-ANN}\left(\frac{n}{\tau}, d_v, k, \delta\right).
\end{aligned}$$

The inequality follows since $\sum_{\tau'=1}^{\tau} |S_{\tau'}| = n$ and $k$-ANN$(n, d, k, \epsilon)$ is concave in $n$.

Finally, we show that $\mathrm{OPT}_{\mathsf{P}(I)} \geq (1 - g(\epsilon))\mathrm{OPT}_{\mathsf{P}} - kh(\epsilon)$ with our choice of $\delta$. Let $i_1^*, \ldots, i_k^*$ be the non-zero coordinates of an optimal solution $x^*$ to the original problem (for simplicity we assume $x^*$ has $k$ non-zero entries, and other cases can be handled similarly). For each $m = 1, \ldots, k$, let $i_m$ be the index such that $i_m$ and $i_m^*$ are in the same $S_{\tau'} \cap I_{\ell'}$ and $(Vu)_{i_m} \geq (Vu)_{i_m^*} - \delta$. Then $f_{i_m} = f_{i_m^*}$. Let $x \in \{0,1\}^n$ such that $x_{i_m} = 1$, then $x$ is feasible to $\mathsf{P}(I)$. We prove that the objective value of $\mathsf{P}(I)$ at $x$ is at least $(1 - g(\epsilon))\mathrm{OPT}_{\mathsf{P}} - kh(\epsilon)$. Indeed, because $\epsilon \geq 34\delta \max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\}(Vu)_{\max} + \delta$, then for each $m = 1, \ldots, k$ we have

[8]For simplicity we assume $|S_{\tau'} \cap I_{\ell'}| \geq k$. Otherwise we simply choose all indices in $S_{\tau'} \cap I_{\ell'}$.

$$\begin{aligned}
f_{i_m}\left(\frac{(w_{i_m}\odot Vu)^\top x}{w_{i_m}^\top x}\right) &= f_{i_m}\left(\frac{\sum_{j=1}^k (w_{i_m})_{i_j}(Vu)_{i_j}}{\sum_{j=1}^k (w_{i_m})_{i_j}}\right)\\
&\geq f_{i_m}\left(\frac{\sum_{j=1}^k (w_{i_m})_{i_j}(Vu)_{i_j^*}}{\sum_{j=1}^k (w_{i_m})_{i_j}} - \delta\right)\\
&\geq f_{i_m^*}\left(\frac{(1-\epsilon)\sum_{j=1}^k (w_{i_m^*})_{i_j^*}(Vu)_{i_j^*}}{(1+\epsilon)\sum_{j=1}^k (w_{i_m^*})_{i_j^*}} - \delta\right)\\
&\geq f_{i_m^*}\left(\frac{\sum_{j=1}^k (w_{i_m^*})_{i_j^*}(Vu)_{i_j^*}}{\sum_{j=1}^k (w_{i_m^*})_{i_j^*}} - 34\delta\max\{\|Q\|_{2,\infty},\|K\|_{2,\infty}\}(Vu)_{\max} - \delta\right)\\
&\geq f_{i_m^*}\left(\frac{\sum_{j=1}^k (w_{i_m^*})_{i_j^*}(Vu)_{i_j^*}}{\sum_{j=1}^k (w_{i_m^*})_{i_j^*}} - \epsilon\right)\\
&\geq (1-g(\epsilon))f_{i_m^*}\left(\frac{\sum_{j=1}^k (w_{i_m^*})_{i_j^*}(Vu)_{i_j^*}}{\sum_{j=1}^k (w_{i_m^*})_{i_j^*}}\right) - h(\epsilon),
\end{aligned}$$

where the first inequality follows since $(Vu)_{i_j} \geq (Vu)_{i_j^*} + \delta$, the second inequality follows since $1-\epsilon \leq (w_{i_m^*})_{i_j^*}/(w_{i_m})_{i_j} \leq 1+\epsilon$ for every $i_m, i_m^*$ that are in the same partition in $\mathcal{I}$ and $i_j, i_j^*$ that are in the same partition in $\mathcal{I}$, and the third inequality follows since $(1-\epsilon)/(1+\epsilon) \geq 1-2\epsilon$ and $\sum_{j=1}^k (w_{i_m^*})_{i_j^*}(Vu)_{i_j^*}/\sum_{j=1}^k (w_{i_m^*})_{i_j^*} \leq (Vu)_{\max}$. Then

$$\begin{aligned}
\mathrm{OPT}_{\mathsf{P}(I)} &\geq \sum_{m=1}^k x_{i_m} f_{i_m}\left(\frac{(w_{i_m}\odot Vu)^\top x}{w_{i_m}^\top x}\right)\\
&\geq \sum_{m=1}^k x_{i_m^*}^*\left(g(\epsilon)f_{i_m^*}\left(\frac{\sum_{j=1}^k (w_{i_m^*})_{i_j^*}(Vu)_{i_j^*}}{\sum_{j=1}^k (w_{i_m^*})_{i_j^*}}\right) - h(\epsilon)\right)\\
&\geq (1-g(\epsilon))\mathrm{OPT}_{\mathsf{P}} - kh(\epsilon)
\end{aligned}$$

as desired. By our choice of $\delta$, we have

$$|I| = k\left\lceil \frac{140(\max\{\|Q\|_{2,\infty},\|K\|_{2,\infty}\})^2}{(Vu)_{\max}\cdot\epsilon}\right\rceil^{2d_{kq}},$$

and the expected amortized runtime of finding $I$ is

$$\begin{aligned}
&\left\lceil \frac{140(\max\{\|Q\|_{2,\infty},\|K\|_{2,\infty}\})^2(Vu)_{\max}}{\epsilon}\right\rceil^{2d_{kq}}\tau\\
&\cdot k\text{-ANN}\left(\frac{n}{\tau}, d_v, k, \frac{\epsilon}{35\max\{\|Q\|_{2,\infty},\|K\|_{2,\infty}\}(Vu)_{\max}}\right).
\end{aligned}$$

## C.2 Phase Two (Ranking)

In phase two, our algorithm approximately solves $\mathsf{P}(I)$, which is $\mathsf{P}$ over the retrieved subset of items $I \subset [n]$. Without loss of generality, assume $I = [m]$. By the first remark of Proposition 5, we may treat $m = kn^{o(1)}$ under mild assumptions. Then since $x_i = 0$ for $i > m$, we may consider only the first $m$ entries of $Vu$ and the top-left $m \times m$ principal sub-matrix of $W$. Therefore, with slight abuse of notation, we redefine $Vu \in \mathbb{R}^m$ to include its first $m$ entries, and $W, W' \in \mathbb{R}_+^{m\times m}$ to be the top-left $m \times m$ principal sub-matrices of the corresponding matrices, respectively. Moreover, note that the quantity

$$\frac{(w_i \odot Vu)^\top x}{w_i^\top x}$$

remains unchanged if the vector $w_i$ is multiplied by a non-zero constant. Thus, rescaling the rows of $W$ does not change $\mathsf{P}(I)$. Because $W \in \mathbb{R}_+^{m\times m}$ is now the $m \times m$ principal sub-matrix, the sum of its rows is not normalized to 1. So for simplicity of exposition, we assume each row of $W$ is rescaled so that $\sum_{j=1}^m W_{ij} = 1$, and each row of $W'$ is rescaled accordingly so that $1-\gamma \le W_{ij}/W'_{ij} \le 1+\gamma$ for all $i, j$. Then we may rewrite $\mathsf{P}(I)$ as

$$\begin{aligned} \max \quad & f_{\mathsf{P}(I)}(x) = \sum_{i=1}^m x_i f_i\left(\frac{(w_i \odot Vu)^\top x}{w_i^\top x}\right) \\ \text{s.t.} \quad & x \in \{0,1\}^m,\ 1 \le e^\top x \le k. \end{aligned} \tag{$\mathsf{P}(I)$}$$

From this point onward, we will work with this new form of $\mathsf{P}(I)$.

Our algorithm begins by replacing $W$ with a low non-negative rank surrogate $W'$ and showing that solving the problem under this approximation is sufficient. Rather than exhaustively enumerating all possible solutions, our algorithm then focuses on a restricted collection of partial solutions that retain the key structural information. The nonlinear terms in the objective are handled by introducing a family of auxiliary linearized problems, which can be further simplified through discretization. This reduction ensures that only a small number of auxiliary linearized problems need to be solved.

To address each auxiliary linearized problem, our algorithm employs a rounding procedure that converts fractional linear-programming solutions into valid discrete ones. At this stage, the central trade-off emerges: exploring too many partial solutions increases runtime beyond practical limits, while exploring too few places excessive burden on the rounding step, leading to higher approximation error. By carefully balancing this trade-off, the ranking phase achieves both computational efficiency – through controlled exploration – and strong accuracy – by minimizing the loss introduced during rounding.

**Proposition 6** *Suppose there exists $W' \in \mathbb{R}_+^{n\times n}$ such that $1-\gamma \le W_{ij}/W'_{ij} \le 1+\gamma$ for all $i, j$, and $W'$ has non-negative rank $r_+$ with an explicit non-negative factorization. Given any $\epsilon > 0$, our algorithm* ALG *achieves*

$$\begin{aligned} \mathrm{ALG}_{\mathsf{P}(I)} \ge\ & (1-g(2\gamma(Vu)_{\max}))^2(1-g(c_{\epsilon,\gamma,W'_{\min}}))^2\mathrm{OPT}_{\mathsf{P}(I)} \\ & - k(1-g(2\gamma(Vu)_{\max}))(2\epsilon(1-g(c_{\epsilon,\gamma,W'_{\min}})) + g(c_{\epsilon,\gamma,W'_{\min}})h(c_{\epsilon,\gamma,W'_{\min}}) \\ & + (1-g(c_{\epsilon,\gamma,W'_{\min}}))^2 h(2\gamma(Vu)_{\max}) + h(2\gamma(Vu)_{\max})), \end{aligned}$$

*where*

$$c_{\epsilon,\gamma,W'_{\min}} = \frac{(1+\gamma)\epsilon}{W'_{\min}},$$

*with runtime*

$$\begin{aligned} & r_+ m\log_2 m + \lambda m^\lambda + \lambda r_+^2 m^{\lambda r_+} \\ & + \left\lceil\left(\frac{4(1+\gamma)k}{\epsilon W'_{\min}}\right)^{r_+}\right\rceil \cdot \left\lceil\frac{(Vu)_{\max} - \min\{0,(Vu)_{\min}\}}{\epsilon}\right\rceil^{r_+} \\ & \cdot \left\lceil\frac{\max_{i\in[m]}\{f_i((Vu)_{\max})\}}{\epsilon}\right\rceil \cdot \lambda' m^{\lambda r_+ + \lambda'} T_{\mathrm{LP}}, \end{aligned}$$

*where $\lambda = \lceil(2r_+ + 2)(Vu)_{\max}/\epsilon\rceil$ and $\lambda' = \lceil(2r_+ + 2)\max_{i\in[m]}\{f_i((Vu)_{\max})\}/\epsilon\rceil$. Here, $T_{\mathrm{LP}} = \mathrm{LP}(m, 3m + r_+ + 2) + \mathrm{LP}(m, 2m + 2r_+ + 2)$ and $\mathrm{LP}(m, n)$ is the runtime of solving a linear program with $m$ variables and $n$ constraints.*

The proof is completed according to the following steps:

1. **Low Non-negative Rank Approximation:** In Lemma 3, we prove that in order to approximately solve $\mathsf{P}(I)$, it is sufficient to approximately solve $\mathsf{P}'(I)$, where $\mathsf{P}'(I)$ is obtained by replacing $W$ with $W'$.
2. **Enumeration of Partial Solutions:** We took guesses on some index sets $X_1, \ldots, X_{r_+}$, which corresponds to the non-negative factorization of $W'$. Let $X = (X_1, \ldots, X_{r_+})$ and let $\mathsf{P}(X)$ denote the problem where $\mathsf{P}'(I)$ has additional constraints that $x_i = 1$ for all $i \in \cup_j X_j$. We showed that the total number of guesses $X$ is bounded above, so it is sufficient to solve $\mathsf{P}(X)$ for each guess $X$.

3. **Linearization of Fractional Objective Terms:** In order to solve $\mathsf{P}(X)$, we linearize the fractional terms in the objective function by defining a set of auxiliary problems $\mathsf{P}(X,t)$ for each $t \in \mathbb{R}_+^m$. These problems are parameterized by the denominator terms in the objective function of $\mathsf{P}(X)$. In Lemma 5, we prove it suffices to find a $t^*$ for which $\mathsf{P}(X,t^*)$ has the highest optimal value among all $\mathsf{P}(X,t)$'s, as the corresponding optimal $x^*$ is an optimal solution to $\mathsf{P}(X)$.
4. **Dimensionality Reduction and Discretization of Auxiliary Problems:** In order to approximately solve $\mathsf{P}(X,t)$ for all $t \in \mathbb{R}_+^m$, we discretize $t$-space and show in Lemma 6 that it suffices to solve $\mathsf{P}(X,t)$ for a small number of $t$'s.
5. **Complete Linearization of Auxiliary Problems:** Fix a given $t$, the objective functions of $\mathsf{P}(X,t)$ inside $f$ has rank $r_+$. We discretized the value space of those objective functions. In Lemma 7, we showed that in order to solve $\mathsf{P}(X,t)$, it is sufficient to give an oracle that, for each discretization of the value space, identify whether there exists a feasible solution to $\mathsf{P}(X,t)$ with objective values that are approximately inside the discretization.
6. **Approximation of Linearized Auxiliary Problems via LP Rounding:** Finally, we gave such an oracle by a rounding procedure. Lemma 8 and Lemma 9 proved that the oracle is correct by using the properties of our guess $X$.

## C.3 Step 1: Low Non-negative Rank Approximation

First we bound the loss incurred by replacing $W$ with $W'$:

**Lemma 3** *Let Problem* $\mathsf{P}'(I)$ *be defined as*

$$\begin{aligned} \max \quad & f_{\mathsf{P}'(I)} = \sum_{i=1}^{m} x_i f_i\left(\frac{(w_i' \odot Vu)^\top x}{{w_i'}^\top x}\right) \\ \text{s.t.} \quad & x \in \{0,1\}^m,\ 1 \le e^\top x \le k. \end{aligned} \tag{$\mathsf{P}'(I)$}$$

*Let $x$ be a feasible solution to* $\mathsf{P}'(I)$ *(and hence also a feasible solution to* $\mathsf{P}(I)$*), and suppose $x$ satisfies*

$$f_{\mathsf{P}'(I)}(x) \ge (1-\alpha)\mathrm{OPT}_{\mathsf{P}'(I)} - \beta.$$

*Then we have*

$$\begin{aligned} f_{\mathsf{P}(I)}(x) \ge\ & (1-\alpha)(1-g(2\gamma(Vu)_{\max}))^2\mathrm{OPT}_{\mathsf{P}(I)} \\ & - kh(2\gamma(Vu)_{\max})(1+(1-\alpha)(1-g(2\gamma(Vu)_{\max})) - \beta(1-g(2\gamma(Vu)_{\max})). \end{aligned}$$

**Proof of Lemma 3.** Let $x$ be a feasible solution to $\mathsf{P}'(I)$. Because $1-\gamma \le W_{ij}/W'_{ij} \le 1+\gamma$, we have

$$\frac{(w_i' \odot Vu)^\top x}{{w_i'}^\top x} \ge \frac{(1-\gamma)}{(1+\gamma)}\frac{(w_i \odot Vu)^\top x}{{w_i}^\top x} \ge (1-2\gamma)\frac{(w_i \odot Vu)^\top x}{{w_i}^\top x}.$$

Therefore, for any feasible solution $x$, we have

$$\begin{aligned} f_{\mathsf{P}'(I)}(x) &= \sum_{i=1}^{m} x_i f_i\left(\frac{(w_i' \odot Vu)^\top x}{{w_i'}^\top x}\right) \\ &\ge \sum_{i=1}^{m} x_i f_i\left((1-2\gamma)\frac{(w_i \odot Vu)^\top x}{w_i^\top x}\right) \\ &\ge \sum_{i=1}^{m} x_i f_i\left(\frac{(w_i \odot Vu)^\top x}{w_i^\top x} - 2\gamma(Vu)_{\max}\right) \\ &\ge (1-g(2\gamma(Vu)_{\max}))f_{\mathsf{P}(I)}(x) - kh(2\gamma(Vu)_{\max}), \end{aligned} \tag{2}$$

where the third inequality follows since $(w_i \odot Vu)^\top x^*/w_i^\top x^* \le (Vu)_{\max}$.

Similarly, we also have

$$\frac{(w_i \odot Vu)^\top x}{{w_i}^\top x} \ge (1-2\gamma)\frac{(w_i' \odot Vu)^\top x}{{w_i'}^\top x},$$

which gives

$$f_{\mathsf{P}(I)}(x) \geq (1 - g(2\gamma(Vu)_{\max}))f_{\mathsf{P}'(I)}(x) - kh(2\gamma(Vu)_{\max}). \tag{3}$$

Now Let $x^*_{\mathsf{P}(I)}$ be an optimal solution to $\mathsf{P}(I)$. Then by Eq. (2), we have

$$\begin{aligned}\mathrm{OPT}_{\mathsf{P}'(I)} &\geq f_{\mathsf{P}'(I)}(x^*_{\mathsf{P}'(I)})\\ &\geq (1 - g(2\gamma(Vu)_{\max}))f_{\mathsf{P}(I)}(x^*_{\mathsf{P}(I)}) - kh(2\gamma(Vu)_{\max})\\ &= (1 - g(2\gamma(Vu)_{\max}))\mathrm{OPT}_{\mathsf{P}(I)} - kh(2\gamma(Vu)_{\max}).\end{aligned}$$

Finally, applying Eq. (3), we conclude that

$$\begin{aligned}f_{\mathsf{P}(I)}(x) &\geq (1 - g(2\gamma(Vu)_{\max}))f_{\mathsf{P}'(I)}(x) - kh(2\gamma(Vu)_{\max})\\ &\geq (1-\alpha)(1 - g(2\gamma(Vu)_{\max}))\mathrm{OPT}_{\mathsf{P}'(I)} - kh(2\gamma(Vu)_{\max}) - \beta(1 - g(2\gamma(Vu)_{\max}))\\ &\geq (1-\alpha)(1 - g(2\gamma(Vu)_{\max}))^2\mathrm{OPT}_{\mathsf{P}(I)}\\ &\quad - kh(2\gamma(Vu)_{\max})(1 + (1-\alpha)(1 - g(2\gamma(Vu)_{\max})) - \beta(1 - g(2\gamma(Vu)_{\max})).\end{aligned}$$

Lemma 3 shows that, in order to approximately solve $\mathsf{P}(I)$, it is enough to approximately solve $\mathsf{P}'(I)$.

Let $W' = AB^\top$ be the known non-negative factorization, where $A, B \in \mathbb{R}^{m\times r_+}_{\geq 0}$. Let $a_i^\top \in \mathbb{R}^{r_+}_{\geq 0}$ be the $i$-th row of $A$ and $b_j \in \mathbb{R}^m_{\geq 0}$ be the $j$-th column of $B$. Then $w'_i = \sum_{j=1}^{r_+} a_{ij}b_j$. Let

$$d_j = b_j \odot (Vu). \tag{4}$$

Then $\mathsf{P}'(I)$ can be rewritten as:

$$\begin{aligned}\max \quad & f_{\mathsf{P}'(I)} = \sum_{i=1}^m x_i f_i\left(\frac{\sum_{j=1}^{r_+} a_{ij}d_j^\top x}{{w'_i}^\top x}\right)\\ \text{s.t.} \quad & x \in \{0,1\}^m,\ 1 \leq e^\top x \leq k.\end{aligned} \tag{$\mathsf{P}'(I)$}$$

## C.4 Step 2: Enumeration of Partial Solutions

Our algorithm enumerates a set of partial solutions (where a "partial solution" fixes the values of a subset of variables), and then for each partial solution, solves the remaining problem near-optimally. In this step we bound the total number of partial solutions, and in the next steps we show that for each partial solution, the remaining problem can be solved sufficiently fast.

Let

$$\lambda = \lceil (2r_+ + 2)(Vu)_{\max}/\epsilon \rceil. \tag{5}$$

Each partial solution that our algorithm considers is specified by a tuple $(X_1, \ldots, X_{r_+})$, where each $X_j \subset [m]$ is an index set such that $1 \leq |X_1| = \cdots = |X_{r_+}| \leq \lambda$. For each $j \in [r_+]$, let

$$\hat{X}_j = \{i \in [m] \setminus X_j \mid d_{ji} > \min_{i' \in X_j}\{d_{ji'}\}\}. \tag{6}$$

In words, $\hat{X}_j$ consists of the indices outside of $X_j$ whose coefficients in $d_j$ are strictly greater than the minimum coefficient in $d_j$ across indices in $X_j$.

We say a tuple $(X_1, \ldots, X_{r_+})$, with corresponding index sets $\hat{X}_1, \ldots, \hat{X}_{r_+}$ defined according to (6), is **valid** if $|\cup_j X_j| \leq k$ and $(\cup_j X_j)\bigcap(\cup_j \hat{X}_j) = \emptyset$. Then every feasible solution $z$ to $\mathsf{P}'(I)$ **corresponds** to a valid tuple $(X_1, \ldots, X_{r_+})$ in the following way: Let $Z = \{i \in [m] \mid z_i = 1\}$. For each $j \in [r_+]$, we define $X_j \subset Z$ to be the set of indices $i \in Z$ such that $d_{ji}$ is among the $\min\{\lambda, |Z|\}$ highest values in $Z$. That is, let $\pi_j : [|Z|] \to Z$ be a sorting of $Z$ according to $d_j$ such that $d_{j,\pi_j(1)} \geq \cdots \geq d_{j,\pi_j(|Z|)}$. Then $X_j = \{\pi_j(1), \ldots, \pi_j(\min\{\lambda, |Z|\})\}$. Notice that $|Z| \leq k$, so $|\cup_j X_j| \leq k$. Also, we claim that $Z \cap \hat{X}_j = \emptyset$ for each $j \in [r_+]$. Supposing otherwise that $i \in Z \cap \hat{X}_j$, then by construction $d_{ji} > d_{j,\pi_j(\min\{\lambda,|Z|\})}$. Because $i \in Z$, we have that $i$ is in the image of $\pi_j$, so there exists $i'$ such that $\pi_j(i') = i$. Therefore $\pi_j(i') \geq \pi_j(\min\{\lambda, |Z|\})$, which shows $i \in X_j$. This contradicts $X_j \cap \hat{X}_j = \emptyset$. Therefore $Z \cap \hat{X}_j = \emptyset$ for each $j \in [r_+]$. Hence we have $(\cup_j X_j)\bigcap(\cup_j \hat{X}_j) = \emptyset$. Therefore $(X_1, \ldots, X_{r_+})$ is indeed a valid tuple.

The notion of correspondence to valid tuples forms a partition of the set of feasible solutions to $\mathsf{P}'(I)$, so it suffices to solve $\mathsf{P}'(I)$ separately for each subset of this partition. This is formally stated in the following result:

**Lemma 4** *Suppose we are given an oracle* $\mathrm{ALG}'$ *that takes* $\mathsf{P}'(I)$*, any valid tuple* $(X_1, \dots, X_{r_+})$*, and any* $\delta > 0$ *as inputs, and outputs a solution* $x'_{(X_1,\dots,X_{r+})}$ *of* $\mathsf{P}'(I)$ *that satisfies*

1. $x'_{(X_1,\dots,X_{r+})}$ *corresponds to* $(X_1, \dots, X_{r_+})$*, and*
2. *for any* $x_{(X_1,\dots,X_{r+})}$ *that corresponds to* $(X_1, \dots, X_{r_+})$*, we have*
$$f_{\mathsf{P}'(I)}(x'_{(X_1,\dots,X_{r+})}) \geq (1 - g'(\delta))f_{\mathsf{P}'(I)}(x_{(X_1,\dots,X_{r+})}) - h'(\delta),$$
*where* $0 \leq g'(\delta) \leq 1$ *and* $h'(\delta) \geq 0$*,*

*with runtime* $T(\delta)$*. Then there exists an algorithm* $\mathrm{ALG}$ *that takes* $\mathsf{P}'(I)$ *and any* $\delta > 0$ *as inputs, and outputs a solution of* $\mathsf{P}'(I)$ *that satisfies*
$$\mathrm{ALG}_{\mathsf{P}'(I)} \geq (1 - g'(\delta))\mathrm{OPT}_{\mathsf{P}'(I)} - h'(\delta)$$
*with runtime*
$$r_+ m \log_2 m + \lambda m^\lambda + \lambda r_+^2 m^{\lambda r_+} + m^{\lambda r_+} T(\delta).$$

**Proof of Lemma 4.** We will construct ALG explicitly. Now because every feasible solution $z$ to $\mathsf{P}'(I)$ corresponds to exactly one valid tuple, we can solve $\mathsf{P}'(I)$ by enumerating all valid tuples, and applying $\mathrm{ALG}'$ to each valid tuple. It turns out that pre-sorting the vectors $d_j$ allows for more-efficient enumeration. Let ALG take the following steps:

1. **Sort $d_j$ for each $j \in [r_+]$.** This takes runtime $r_+ m \log_2 m$.
2. **Enumerate all valid tuples with $|X_1| < \lambda$, and record the unique corresponding feasible solutions.** We will show momentarily that when $|X_1| < \lambda$, there is a unique corresponding feasible solution, and as a result this step takes runtime $\lambda m^\lambda$.
3. **Enumerate all valid tuples with $|X_1| = \lambda$, and record the solution output by $\mathrm{ALG}'_{\mathsf{P}'(I)}$ for each such valid tuple.** We will show that it takes runtime $\lambda r_+^2 m^{\lambda r_+}$ to enumerate all such valid tuples, and then because there are at most $m^{\lambda r_+}$ such valid tuples, the total runtime of this step is $\lambda r_+^2 m^{\lambda r_+} + m^{\lambda r_+} T(\delta)$.
4. **Output a solution that is recorded with the highest objective value in $\mathsf{P}'(I)$.**

Therefore the total runtime of ALG is
$$r_+ m \log_2 m + \lambda m^\lambda + \lambda r_+^2 m^{\lambda r_+} + m^{\lambda r_+} T(\delta).$$
Finally, let $x^*_{(X_1^*,\dots,X_{r+}^*)}$ be an optimal solution of $\mathsf{P}'(I)$ where $(X_1^*, \dots, X_{r+}^*)$ is the valid tuple that it corresponds to. Let $x'_{(X_1^*,\dots,X_{r+}^*)}$ be the solution that $\mathrm{ALG}'_{\mathsf{P}'(I)}$ outputs with input $\mathsf{P}'(I)$, valid tuple $(X_1^*, \dots, X_{r_+}^*)$, and $\delta > 0$. Then
$$\begin{aligned}\mathrm{ALG}_{\mathsf{P}'(I)} &\geq f_{\mathsf{P}'(I)}(x'_{(X_1^*,\dots,X_{r+}^*)}) \\ &\geq (1 - g'(\delta))f_{\mathsf{P}'(I)}(x^*_{(X_1^*,\dots,X_{r+}^*)}) - h'(\delta) \\ &= (1 - g'(\delta))\mathrm{OPT}_{\mathsf{P}'(I)} - h'(\delta).\end{aligned}$$

It remains to analyze Steps 2 and 3 of ALG.

**Step 2:** Fix any valid tuple $(X_1, \dots, X_{r_+})$ such that $|X_1| < \lambda$. Assume there exists a feasible solution $z$ to $\mathsf{P}'(I)$ that corresponds to the valid tuple, and let $Z = \{i \in [m] \mid z_i = 1\}$. Then because $|X_1| = \min\{\lambda, |Z|\} = |Z|$ and $X_1 \subset Z$, we have $X_1 = Z$. Similarly, $X_j = Z$ for all $j \in [r_+]$. Therefore, there exists a feasible solution to $\mathsf{P}'(I)$ that corresponds to $(X_1, \dots, X_{r_+})$ only if $X_1 = \cdots = X_{r+}$. There are at most $\sum_{i=1}^{\lambda-1} \binom{m}{i} \leq \lambda m^\lambda$ such tuples. Moreover, there is a unique feasible solution $z$ that corresponds to $(X_1, \dots, X_{r_+})$, namely $z_i = 1$ for every $i \in X_1$ and $z_i = 0$ for every $i \notin X_1$. Thus, the runtime of enumerating all corresponding feasible solutions is bounded by $\lambda m^\lambda$.

**Step 3:** Fix any valid tuple $(X_1, \dots, X_{r_+})$ such that $|X_1| = \lambda$. Then by construction we must have $z_i = 1$ for every $i \in \cup_j X_j$, and $z_i = 0$ for every $i \in \cup_j \hat{X}_j$. Therefore every feasible solution $z$ that corresponds to $(X_1, \dots, X_{r_+})$ must lie in the following set:

$$\begin{aligned}\{z \in \{0,1\}^m \,|& z_i = 1 \,\forall i \in \cup_j X_j,\\ & z_i = 0 \,\forall i \in \cup_j \hat{X}_j,\\ & 1 \le e^\top z \le k\}.\end{aligned}$$

There are $\binom{m}{\lambda}^{r_+} \le m^{\lambda r_+}$ tuples such that $|X_1| = \lambda$. We enumerate all such tuples, and check each for validity according to the following procedure:

0. From Step 1 of ALG, let $\pi_j : [m] \to [m]$ be a sorting of $[m]$ according to $d_j$ such that $d_{j,\pi_j(1)} \ge \dots \ge d_{j,\pi_j(m)}$.

1. Fix any given tuple $(X_1, \dots, X_{r_+})$. For each $j \in [r_+]$, let $i(j) \in [m]$ be an index such that $d_{j,i(j)} \in X_j$ and $d_{j,i(j)} \le d_{ji}$ for all $d_{ji} \in X_j$. Without loss of generality, if $d_{j,i(j)} = d_{ji'}$ and $d_{ji'} \notin X_j$ for some index $i'$, we let $\pi_j(i(j)) > \pi_j(i')$ for tie-breaking in the sorting. Also, if $d_{j,i(j)} = d_{ji'}$ and $d_{ji'} \in X_j$ for some index $i'$, we let $\pi_j(i(j)) < \pi_j(i')$ for tie-breaking in the sorting. Then by our construction

$$\begin{aligned}\hat{X}_j &= \{i \in [m] \setminus X_j \mid d_{ji} > \min_{i' \in X_j}\{d_{ji'}\}\}\\ &= \{i \in [m] \setminus X_j \mid d_{ji} > d_{j,i(j)}\}\}\\ &= \{i \in [m] \setminus X_j \mid \pi_j(i) > \pi_j(i(j))\}.\end{aligned}$$

   Therefore $X_j \cup \hat{X}_j = \{i \in [m] \mid \pi_j(i) \ge \pi_j(i(j))\}$.

2. Note that $\sum_{j=1}^{r_+} |X_j| = \lambda r_+$. Therefore, in order to check whether $|\cup_j X_j| \le k$, we just need to count the number of overlaps among all $X_j$'s. Set a counter $c = 0$ to count the overlaps. For each $j \in [r_+]$ and for each $i \in X_j$, we check if $\pi_{j'}(i) \ge \pi_{j'}(i(j'))$, and do the following:

   - If $\pi_{j'}(i) < \pi_{j'}(i(j'))$, then we have $i \notin X_{j'} \cup \hat{X}_{j'}$. We do nothing in this case.
   - If $\pi_{j'}(i) \ge \pi_{j'}(i(j'))$ and $\pi_{j'}(i) \in X_{j'}$, then $i$ appears in both $X_j$ and $X_{j'}$. Therefore we increase $c$ by 1.
   - If $\pi_{j'}(i) \ge \pi_{j'}(i(j'))$ and $\pi_{j'}(i) \notin X_{j'}$, then we must have $\pi_{j'}(i) \in \hat{X}_{j'}$. Therefore $(\cup_j X_j) \bigcap (\cup_j \hat{X}_j) \ne \emptyset$, so we can terminate the process and declare that $(X_1, \dots, X_{r_+})$ is not a valid tuple.

   We iterate all $j \in [r_+]$ and $i \in X_j$. Notice that $c$ counts the number of overlaps (with multiplicity) of elements in $X_j$, we have $|\cup_j X_j| = \sum_{j=1}^{r_+} |X_j| - c = \lambda r_+ - c$. Therefore we can check if $|\cup_j X_j| \le k$. Also, if the above procedure never encounters $\pi_{j'}(i) \in \hat{X}_{j'}$, then we have $(\cup_j X_j) \bigcap (\cup_j \hat{X}_j) = \emptyset$. Therefore this procedure allows us to check the validity of $(X_1, \dots, X_{r_+})$.

Because each $d_{j'}$ is sorted, the above procedure takes a constant runtime for each $j' \in [r_+]$, so the runtime for each fixed $j \in [r_+]$ and $i \in X_j$ is $r_+$. Because $\sum_{j=1}^{r_+} |X_j| = \lambda r_+$, there are $\lambda r_+$ combinations of $j \in [r_+]$ and $i \in X_j$. Therefore the runtime to check the validity is $\lambda r_+^2$ for any given tuple $(X_1, \dots, X_{r_+})$. Because there are at most $m^{\lambda r_+}$ such tuples, the runtime of enumerating all such tuples is $\lambda r_+^2 m^{\lambda r_+}$.

The consequence of this result is that we have reduced to the task of, for each valid tuple $X = (X_1, \dots, X_{r_+})$ with $|X_1| = \lambda$, solving $\mathsf{P}'(I)$ with the additional constraint that the solution must correspond to the valid tuple, as stated in Problem $\mathsf{P}(X)$ below:

$$\begin{aligned} \max \quad & f_{\mathsf{P}(X)}(x) = \sum_{i=1}^{m} x_i f_i \left( \frac{\sum_{j=1}^{r_+} a_{ij} d_j^\top x}{w_i'^\top x} \right) \\ \text{s.t.} \quad & x \in \{0,1\}^m, \\ & 1 \le e^\top x \le k, \\ & x_i = 1 \quad \forall i \in \cup_j X_j, \\ & x_i = 0 \quad \forall i \in \cup_j \hat{X}_j. \end{aligned} \tag{$\mathsf{P}(X)$}$$

## C.5 Step 3: Linearization

In order to solve $\mathsf{P}(X)$, we define the following auxiliary problem $\mathsf{P}(X,t)$ for each $t \in \mathbb{R}_+^m$:

$$\begin{aligned} \max \quad & f_{\mathsf{P}(X,t)}(x) = \sum_{i=1}^{m} x_i f_i \left( \frac{\sum_{j=1}^{r_+} a_{ij} d_j^\top x}{t_i} \right) \\ \text{s.t.} \quad & x \in \{0,1\}^m, \\ & e^\top x \le k, \\ & w_i'^\top x \le t_i \quad \forall i \in [m], \\ & x_i = 1 \quad \forall i \in \cup_j X_j, \\ & x_i = 0 \quad \forall i \in \cup_j \hat{X}_j. \end{aligned} \tag{$\mathsf{P}(X,t)$}$$

Note that we have dropped the constraint $1 \le e^\top x$ in $\mathsf{P}(X,t)$. This is inconsequential: because we assume $\text{OPT}_\mathsf{P}$ is positive, the solution $x$ with all entries equal to zero is not an optimal solution to $\mathsf{P}$. Indeed, the only reason we have maintained the $1 \le e^\top x$ constraint until now has been to rule out notational edge cases (such as dividing by zero).

We prove an important property regarding the relationship between optimal solutions of $\mathsf{P}(X)$ and those of $\mathsf{P}(X,t)$.

**Lemma 5** *Fix any valid tuple $X$. Let $t^* \in \arg\max_{t \in \mathbb{R}_+^m} \text{OPT}_{\mathsf{P}(X,t)}$, and let $x^*$ be an optimal solution to $\mathsf{P}(X,t^*)$. Then $W'x^* = t^*$, and $x^*$ is an optimal solution to $\mathsf{P}(X)$.*

**Proof of Lemma 5.** First, we show that $W'x^* = t^*$. Suppose otherwise, and let $t' = W'x^*$. Then $t_i' \le t_i^*$ for every $i \in [m]$ and $t_i' < t_i^*$ for some $i$. Therefore

$$\text{OPT}_{\mathsf{P}(X,t^*)} = \sum_{i=1}^{m} x_i^* f_i \left( \frac{\sum_{j=1}^{r_+} a_{ij} d_j^\top x^*}{t_i^*} \right) < \sum_{i=1}^{m} x_i^* f_i \left( \frac{\sum_{j=1}^{r_+} a_{ij} d_j^\top x^*}{t_i'} \right) \le \text{OPT}_{\mathsf{P}(X,t')},$$

contradicting the definition of $t^*$.

Now we show that $x^*$ is an optimal solution to the $\mathsf{P}(X)$. For the sake of contradiction, suppose that $x'$ gives a higher objective value than $x^*$ to $\mathsf{P}(X)$, that is,

$$\sum_{i=1}^{m} x_i' f_i \left( \frac{\sum_{j=1}^{r_+} a_{ij} d_j^\top x'}{w_i'^\top x'} \right) > \sum_{i=1}^{m} x_i^* f_i \left( \frac{\sum_{j=1}^{r_+} a_{ij} d_j^\top x^*}{w_i'^\top x^*} \right) = \text{OPT}_{\mathsf{P}(X,t^*)}.$$

Let $t' = Wx'$. Then $x'$ is a feasible solution to $P(X,t')$. Therefore, we have

$$\text{OPT}_{\mathsf{P}(X,t')} \ge \sum_{i=1}^{m} x_i' f_i \left( \frac{\sum_{j=1}^{r_+} a_{ij} d_j^\top x'}{w_i'^\top x'} \right) > \text{OPT}_{\mathsf{P}(X,t^*)},$$

contradicting the definition of $t^*$.

Because $t^* = W'x^*$ and $\sum_{j=1}^{m} w_{ij} \le 1 + \gamma$ for each $i \in [m]$, we have $t^* \in [W'_{\min}, 1+\gamma]^m$. Thus, from here on we will only consider the problems $\mathsf{P}(X,t)$ with $t \in [W'_{\min}, 1+\gamma]^m$.

## C.6 Step 4: Dimensionality Reduction and Discretization of Auxiliary Problems:

Lemma 5 shows that, in order to solve $\mathsf{P}(X)$, it is enough to solve for $\arg\max_{t\in\mathbb{R}_+^m} \mathrm{OPT}_{\mathsf{P}(X,t)}$. Below we show that it suffices to solve the auxiliary problem $\mathsf{P}(X,t)$ for a smaller, discretized set of $t$'s.

**Lemma 6** *Suppose we are given an oracle* $\mathrm{ALG}'$ *that takes* $\mathsf{P}(X,t)$ *with any* $t \in [W'_{\min}, 1+\gamma]^m$ *and any* $\delta > 0$ *as inputs, and outputs a solution of* $\mathsf{P}(X,t)$ *that satisfies*

$$\mathrm{ALG}'_{\mathsf{P}(X,t)} \geq (1-g'(\delta))\mathrm{OPT}_{\mathsf{P}(X,t)} - h'(\delta)$$

*with runtime* $T(\delta)$*, where* $0 \leq g'(\delta) \leq 1$ *and* $h'(\delta) \geq 0$*. Then there exists an algorithm* ALG *that takes* $\mathsf{P}(X)$ *and any* $\delta > 0$ *as inputs, and outputs a solution of* $\mathsf{P}(X)$ *that satisfies*

$$\mathrm{ALG}_{\mathsf{P}(X)} \geq (1-g'(\delta))\left(\left(1-g\left(\frac{(1+\gamma)\delta}{W'_{\min}}\right)\right)\mathrm{OPT}_{\mathsf{P}(X)} + kh\left(\frac{(1+\gamma)\delta}{W'_{\min}}\right)\right) - h'(\delta)$$

*with runtime*

$$\left\lceil\left(\frac{4(1+\gamma)k}{\epsilon W'_{\min}}\right)^{r_+}\right\rceil T(\delta).$$

**Proof of Lemma 6.**

Recall that we have known non-negative factorization $W' = AB^\top$, where $A, B \in \mathbb{R}_{\geq 0}^{m\times r_+}$. First, we make the following observation on the scale of $A$ and $B$:

**Observation 1** *There exists* $A', B' \in \mathbb{R}_{\geq 0}^{m\times r_+}$ *where* $W' = A'B'^\top$*, such that* $\|a'_i\|_1 \leq 1+\gamma$ *for every* $i \in [m]$ *and* $\|b'_j\|_1 = 1$ *for every* $j \in [r_+]$.

**Proof of Observation 1.** We construct $A'$ and $B'$ explicitly. Let the rows of $B'$ be the rows of $B$ that are rescaled so that $\|b_j\|_1 = 1$. That is, let $b'_{jk} = b_{jk}/\sum_{k'=1}^m b_{jk'}$ for every $j \in [r_+]$ and $k \in [m]$. Let the columns of $A'$ be the columns of $A$ that are rescaled accordingly. That is, let $a'_{ij} = a_{ij}\sum_{k'=1}^m b_{jk'}$ for every $i \in [m]$ and $j \in [r_+]$. Then we have $a'_{ij}b'_{jk} = a_{ij}b_{jk}$. Therefore $W' = A'B'^\top$. Finally, since $1-\gamma \leq W_{ij}/W'_{ij} \leq 1+\gamma$ and $\sum_{j=1}^m w_{ij} = 1$ for every $i \in [m]$, we have $\sum_{j=1}^m w'_{ij} \leq 1+\gamma$ for every $i \in [m]$. Therefore for every $i \in [m]$, we have

$$1+\gamma \geq \sum_{j=1}^m w'_{ij} = \sum_{j=1}^{r_+} a'_{ij}\sum_{k=1}^m b'_{jk} = \sum_{j=1}^{r_+} a'_{ij} = \|a'_i\|_1.$$

By Observation 1, we may assume $\|a_i\|_1 \leq 1+\gamma$ for every $i \in [m]$ and $\|b_j\|_1 = 1$ for every $j \in [r_+]$ from now on. Let

$$Y = \{B^\top x \mid x \text{ is a feasible solution to } \mathsf{P}'(I)\} \subset \mathbb{R}^{r_+}. \tag{7}$$

We will partition the $t$-space $[W'_{\min}, 1+\gamma]^m$ by partitioning $Y$. Let $\delta'$ be the quantity

$$\delta' = \frac{W'_{\min}\delta}{(1+\gamma)\sqrt{k}}, \tag{8}$$

where the reason for this choice will be specified momentarily. Notice that $\|y\|_2 \leq \sqrt{k}$ for every $y \in Y$. As seen in the proof of Proposition 5, we can create a cover of a ball with radius $\sqrt{k}$ in $\mathbb{R}^{r_+}$ using $\lceil(4\sqrt{k}/\delta')^{r_+}\rceil$ number of balls with radius $\delta'/2$. Therefore we can create a partition $\mathcal{Y} = \{Y_1, \dots, Y_\ell\}$ of $Y$ such that $\ell = \lceil(4\sqrt{k}/\delta')^{r_+}\rceil$, and $\|y - y'\|_2 \leq \delta'$ for every $\ell' \in [\ell]$ and $y, y' \in Y_{\ell'}$.

Fix any row $a_i^\top$ of $A$. For every $\ell' \in [\ell]$ and $i, i' \in I_{\ell'}$, we have

$$\begin{aligned}\left|\frac{a_i^\top y}{a_i^\top y'} - 1\right| &= \left|\frac{a_i^\top(y-y')}{a_i^\top y}\right| \\ &\leq \frac{\|a_i\|_2\|y-y'\|_2}{|a_i^\top y|} \\ &\leq \frac{\delta'(1+\gamma)\sqrt{k}}{W'_{\min}}.\end{aligned}$$

Thus, for our particular choice of $\delta'$, we have that $|a_i^\top y/a_i^\top y' - 1| \le \delta$ for every $\ell' \in [\ell]$ and $y, y' \in Y_{\ell'}$.

For every $\ell' \in [\ell]$, let $T_{\ell'} = \{Ay \mid y \in Y_{\ell'}\} \subset \mathbb{R}^m$. Then for every $\ell' \in [\ell]$ and $t, t' \in T_{\ell'}$, we have that $|(t_i/t'_i) - 1| \le \delta$ for every $i \in [m]$. Fix any $t_1 \in T_1, \ldots, t_\ell \in T_\ell$. The algorithm ALG will use the given oracle ALG$'$ to obtain a solution for each $\mathsf{P}(X, t_{\ell'})$, and then output the solution that has the highest objective value of $\mathsf{P}(X, t_{\ell'})$. That is,

$$\text{ALG}_{\mathsf{P}(X)} = \max_{\ell' \in [\ell]} \text{ALG}'_{\mathsf{P}(X,t_{\ell'})}.$$

Because $\ell = \lceil (4\sqrt{k}/\delta)^{r_+} \rceil = \lceil (4(1+\gamma)k/\epsilon W'_{\min})^{r_+} \rceil$, the runtime of ALG is

$$\lceil (4(1+\gamma)k/\epsilon W'_{\min})^{r_+} \rceil T(\delta).$$

Finally, we prove the performance guarantee for ALG. Let $t^* \in \arg\max_{t \in \mathbb{R}_+^m} \text{OPT}_{\mathsf{P}(X,t)}$. Assume $t^* \in T_{\ell'}$. Then $|(t_i^*/(t_{\ell'})_i) - 1| \le \delta$. Let $x^*$ be the corresponding optimal solution to $\mathsf{P}(X, t^*)$. Then by Lemma 5, $x^*$ is an optimal solution to $\mathsf{P}(X)$. Let $x'$ be an optimal solution of $\mathsf{P}(X, t_{\ell'})$. Because $t_{\ell'} \in [W'_{\min}, 1+\gamma]^m$, we have $\sum_{j=1}^{r_+} a_{ij} d_j^\top x^*/(t_{\ell'})_i = {w'_i}^\top x^*/(t_{\ell'})_i \le (1+\gamma)/W'_{\min}$. Also, since $f_i(x - \epsilon) \ge (1 - g(\epsilon))f_i(x) - h(\epsilon)$ for all $x$, we have $f_i(x+\epsilon) \le (f_i(x) + h(\epsilon))/(1 - g(\epsilon))$ for all $x$. As a consequence of Lemma 5, we have $\text{OPT}_{\mathsf{P}(X)} = \text{OPT}_{\mathsf{P}(X,t^*)}$. Then

$$\begin{aligned}
\text{OPT}_{\mathsf{P}(X)} &= \text{OPT}_{\mathsf{P}(X,t^*)} \\
&= \sum_{i=1}^m x_i^* f_i\left(\frac{\sum_{j=1}^{r_+} a_{ij} d_j^\top x^*}{t_i^*}\right) \\
&\le \sum_{i=1}^m x_i^* f_i\left((1+\delta)\frac{\sum_{j=1}^{r_+} a_{ij} d_j^\top x^*}{(t_{\ell'})_i}\right) \\
&\le \frac{\sum_{i=1}^m x_i^* f_i\left(\frac{\sum_{j=1}^{r_+} a_{ij} d_j^\top x^*}{(t_{\ell'})_i}\right) + kh(\delta(1+\gamma)/W'_{\min})}{1 - g(\delta(1+\gamma)/W'_{\min})} \\
&\le \frac{\text{OPT}_{\mathsf{P}(X,t_{\ell'})} + kh(\delta(1+\gamma)/W'_{\min})}{1 - g(\delta(1+\gamma)/W'_{\min})}.
\end{aligned}$$

Rearranging the above, we have

$$\text{OPT}_{\mathsf{P}(X,t_{\ell'})} \ge (1 - g(\delta(1+\gamma)/W'_{\min}))\text{OPT}_{\mathsf{P}(X)} - kh(\delta(1+\gamma)/W'_{\min}).$$

Therefore,

$$\begin{aligned}
\text{ALG}_{\mathsf{P}(X)} &\ge \text{ALG}'_{\mathsf{P}(X,t_{\ell'})} \\
&\ge (1 - g'(\delta))\text{OPT}_{\mathsf{P}(X,t_{\ell'})} - h'(\delta) \\
&\ge (1 - g'(\delta))\left(\left(1 - g\left(\frac{(1+\gamma)\delta}{W'_{\min}}\right)\right)\text{OPT}_{\mathsf{P}(X)} - kh\left(\frac{(1+\gamma)\delta}{W'_{\min}}\right)\right) - h'(\delta).
\end{aligned}$$

## C.7 Step 5: Complete Linearization of Auxiliary Problems

Lemma 6 shows that, to approximately solve $\mathsf{P}(X)$, it suffices to construct an oracle that approximately solves $\mathsf{P}(X, t)$ for any given $t \in [W'_{\min}, 1+\gamma]^m$. Below we give such an oracle. Let $c_i = a_i/t_i \in \mathbb{R}_{\ge 0}^{r_+}$ for each $i \in [m]$. Then $\mathsf{P}(X, t)$ can be equivalently formulated as

$$\begin{aligned}
\max \quad & f_{\mathsf{P}(X,t)}(x) = \sum_{i=1}^m x_i f_i\left(\sum_{j=1}^{r_+} c_{ij} d_j^\top x\right) && (\mathsf{P}(X,t)) \\
\text{s.t.} \quad & x \in \{0,1\}^m, \\
& e^\top x \le k, \\
& {w'_i}^\top x \le t_i \quad \forall i \in [m], \\
& x_i = 1 \quad \forall i \in \cup_j X_j, \\
& x_i = 0 \quad \forall i \in \cup_j \hat{X}_j.
\end{aligned}$$

To solve $\mathsf{P}(X,t)$, we partition the space of possible values $(d_1^\top x, \ldots, d_{r_+}^\top x) \in \mathbb{R}^{r_+}$, as well as the space of the objective value $f_{\mathsf{P}(X,t)}(x)$.

**Lemma 7** *Fix any $t \in [W'_{\min}, 1+\gamma]^m$. Suppose we are given an oracle with runtime $T(\delta_1, \delta_2)$ that takes $\mathsf{P}(X,t)$, any $\theta = (\theta_1, \ldots, \theta_{r_+}) \in \mathbb{R}^{r_+}$, any $\zeta \geq 0$, and any $\delta_1, \delta_2 > 0$ as inputs, and either*

1. *Scenario one: correctly declares that there is no feasible $x$ to $\mathsf{P}(X,t)$ such that $d_j^\top x \geq \theta_j$ for every $j \in [r_+]$ and $\sum_{i=1}^m x_i f_i(c_i^\top \theta) \geq \zeta$, or*
2. *Scenario two: outputs a feasible $x$ to $\mathsf{P}(X,t)$ such that $d_j^\top x + \delta_1 \geq \theta_j$ for every $j \in [r_+]$ and $\sum_{i=1}^m x_i f_i(c_i^\top \theta) + \delta_2 \geq \zeta$.*

*Then there exists an algorithm that satisfies*

$$\mathrm{ALG}_{\mathsf{P}(X,t)} \geq \left(1 - g\left(\frac{\delta_1(1+\gamma)}{W'_{\min}}\right)\right)(\mathrm{OPT}_{\mathsf{P}(X,t)} - 2\delta_2) - kh\left(\frac{\delta_1(1+\gamma)}{W'_{\min}}\right)$$

*with runtime*

$$\lceil ((Vu)_{\max} - \min\{0, (Vu)_{\min}\})/\delta_1 \rceil^{r_+} \cdot \left\lceil k \max_{i \in [m]} \{f_i((Vu)_{\max})\}/\delta_2 \right\rceil T(\delta_1, \delta_2).$$

**Proof of Lemma 7.** Let $(Vu)_{\min}$ be the minimum entry of $Vu$ (possibly negative). Because $\|b_j\|_1 = 1$ and $d_j = b_j \odot (Vu)$ for every $j \in [r_+]$, we have $d_j^\top x^* \in [\min\{0, (Vu)_{\min}\}, (Vu)_{\max}]$ for every $j \in [r_+]$. Also, because each $f_i$ is non-decreasing, $\mathrm{OPT}_{\mathsf{P}(X,t)} \in [0, k \max_{i\in[m]}\{f_i((Vu)_{\max})\}]$. Similar to the proof of Lemma 6, we will create a partition of the space of possible values $(d_1^\top x, \ldots, d_{r_+}^\top x) \in \mathbb{R}^{r_+}$, as well as the space of the objective value $f_{\mathsf{P}(X,t)}(x)$. We then show that it is sufficient to solve $\mathsf{P}(X,t)$ in each subset of the partition. Let $\Delta_\ell = \min\{0, (Vu)_{\min}\} + \delta_1(\ell - 1)$ for $\ell = 1, \ldots, \lceil ((Vu)_{\max} - \min\{0, (Vu)_{\min}\})/\delta_1 \rceil$. Let $\Delta'_s = \delta_2(s-1)$ for $s = 1, \ldots, \lceil k \max_{i\in[m]}\{f_i((Vu)_{\max})\}/\delta_2 \rceil$. Consider all tuples $(\ell_1, \ldots, \ell_{r_+}, s)$. There are in total

$$\lceil ((Vu)_{\max} - \min\{0, (Vu)_{\min}\})/\delta_1 \rceil^{r_+} \cdot \left\lceil k \max_{i \in [m]} \{f_i((Vu)_{\max})\}/\delta_2 \right\rceil$$

such tuples. Moreover, there exists a tuple $(\ell_1^*, \ldots, \ell_{r_+}^*, s^*)$ such that $\Delta_{\ell_i^*} \leq x_i^* \leq \Delta_{\ell_i^*+1}$ for each $i \in [m]$ and $\Delta'_{s^*} \leq \mathrm{OPT}_{\mathsf{P}(X,t)} \leq \Delta'_{s^*+1}$.

For each tuple $(\ell_1, \ldots, \ell_{r_+}, s)$, our desired algorithm uses the given oracle to determine whether there exists a feasible $x$ to $\mathsf{P}(X,t)$ that satisfies the conditions in scenario two with $\theta_i = \Delta_{\ell_i}$ for each $i \in [r_+]$ and $\zeta = \Delta'_s$. Then our desired algorithm returns the $x$ with the highest objective value of $\mathsf{P}(X,t)$ among all tuples. Note that $x^*$ satisfies the conditions in scenario two on the tuple $(\ell_1^*, \ldots, \ell_{r_+}^*, s^*)$. Therefore the given oracle would return some feasible $x'$ to $\mathsf{P}(X,t)$ that satisfies the conditions in scenario two with $\theta_i = \Delta_{\ell_i^*}$ for each $i \in [r_+]$ and $\zeta = \Delta'_{s^*}$. Notice that $c_{ij} = a_{ij}/t_i \leq (1+\gamma)/W'_{\min}$. Then by the conditions in scenario two we have

$$\begin{aligned}
\mathrm{ALG}_{\mathsf{P}(X,t)} &\geq \sum_{i=1}^m x'_i f_i\left(\sum_{j=1}^{r_+} c_{ij} d_j^\top x'\right) \\
&\geq \sum_{i=1}^m x'_i f_i\left(\sum_{j=1}^{r_+} c_{ij}(\theta_j - \delta_1)\right) \\
&\geq \sum_{i=1}^m x'_i f_i\left(\sum_{j=1}^{r_+} c_{ij}\theta_j - \delta_1(1+\gamma)/W'_{\min}\right) \\
&\geq (1 - g(\delta_1(1+\gamma)/W'_{\min})) \sum_{i=1}^m x'_i f_i\left(\sum_{j=1}^{r_+} c_{ij}\theta_j\right) - kh(\delta_1(1+\gamma)/W'_{\min}) \\
&\geq (1 - g(\delta_1(1+\gamma)/W'_{\min}))(\Delta'_{s^*} - \delta_2) - kh(\delta_1(1+\gamma)/W'_{\min}) \\
&\geq (1 - g(\delta_1(1+\gamma)/W'_{\min}))(\mathrm{OPT}_{\mathsf{P}(X,t)} - 2\delta_2) - kh(\delta_1(1+\gamma)/W'_{\min}),
\end{aligned}$$

where the second and the fifth inequalities follow from the conditions in scenario two, and the last inequality follows since $\Delta'_{s^*} \leq \mathsf{OPT}_{\mathsf{P}(X,t)} \leq \Delta'_{s^*+1}$.

Because there are in total

$$\lceil ((Vu)_{\max} - \min\{0, (Vu)_{\min}\})/\delta_1 \rceil^{r_+} \cdot \left\lceil k \max_{i\in[m]} \{f_i((Vu)_{\max})\}/\delta_2 \right\rceil$$

number of tuples $(\ell_1, \ldots, \ell_{r_+}, s)$, the runtime of our algorithm is

$$\lceil ((Vu)_{\max} - \min\{0, (Vu)_{\min}\})/\delta_1 \rceil^{r_+} \cdot \left\lceil k \max_{i\in[m]} \{f_i((Vu)_{\max})\}/\delta_2 \right\rceil T(\delta_1, \delta_2).$$

## C.8 Step 6: Approximation of Linearized Auxiliary Problems via LP Rounding

Lemma 7 shows that, to solve $P(X,t)$ for any given $t \in [W'_{\min}, 1+\gamma]^m$, it is enough to give an oracle described in Lemma 7. Similar to the idea of enumerating partial solutions by constructing valid tuples based on the values of each $d_j$, we further construct index sets based on the values of $f_i(c_i^\top \theta)$ and enumerate all possible index sets. Recall that via the valid tuple $(X_1, \ldots, X_{r_+})$, we have already fixed at least $\lambda$ indices of any feasible solution to $\mathsf{P}(X,t)$ to be equal to 1, namely the indices in $\cup_j X_j$. Fix

$$\lambda' = \lceil (2r_+ + 2) \max_{i\in[m]} \{f_i((Vu)_{\max})\}/\epsilon \rceil. \tag{9}$$

Let $X' \subset [m] \setminus (\cup_j X_j) \cup (\cup_j \hat{X}_j)$ be an index set such that $0 \leq |X'| \leq \lambda'$. Let

$$\hat{X}' = \{i \in [m] \setminus X' \cup (\cup_j X_j) \cup (\cup_j \hat{X}_j) \mid f_i(c_i^\top \theta) > \min_{i'\in X'} \{f_{i'}(c_{i'}^\top \theta)\}\}. \tag{10}$$

In words, $\hat{X}'$ consists of the indices outside of $X' \cup (\cup_j X_j) \cup (\cup_j \hat{X}_j)$ whose corresponding values of $f_i(c_{i'}^\top \theta)$ are strictly greater than the minimum value of $f_i(c_i^\top \theta)$ across indices in $X'$. Then every feasible solution $z$ to $\mathsf{P}(X,t)$ **corresponds** to an index set $X'$ in the following way: let $Z = \{i \in [m] \setminus (\cup_j X_j) \cup (\cup_j \hat{X}_j) \mid z_i = 1\}$. We define $X' \subset Z$ to be the set of indices $i \in Z$ such that $f_i(c_i^\top \theta)$ is among the $\min\{\lambda', |Z|\}$ highest values in $Z$. That is, let $\pi' : [|Z|] \to Z$ be a sorting of $Z$ according to $f_i(c_i^\top \theta)$ such that $f_{\pi'(1)}(c_{\pi'(1)}^\top \theta) \geq \cdots \geq f_{\pi'(|Z|)}(c_{\pi'(|Z|)}^\top \theta)$. Then $X' = \{\pi'(1), \ldots, \pi'(\min\{\lambda', |Z|\})\}$. Similar to before, if $z$ corresponds to $X'$, we must have $z_i = 1$ for $i \in X'$ and $z_i = 0$ for $i \in \hat{X}'$.

As in Lemma 4, the notion of correspondence to index sets forms a partition of the set of feasible solutions to $\mathsf{P}(X,t)$. Therefore, in order to give an oracle described in Lemma 7, it suffices to give an oracle described in Lemma 7 separately for each subset of this partition. This is formally stated in the following result:

**Observation 2** *Suppose we are given an oracle with runtime $T(\delta_1, \delta_2)$ that takes $\mathsf{P}(X,t)$, any $\theta = (\theta_1, \ldots, \theta_{r_+}) \in \mathbb{R}^{r_+}$, any $\zeta \geq 0$, any $\delta_1, \delta_2 > 0$, and any index set $X' \subset [m] \setminus (\cup_j X_j) \cup (\cup_j \hat{X}_j)$ such that $0 \leq |X'| \leq \lambda'$ as inputs, and either*

1. *Scenario one: correctly declares that there is no feasible $x$ to $\mathsf{P}(X,t)$ that corresponds to $X'$ such that $d_j^\top x \geq \theta_j$ for every $j \in [r_+]$ and $\sum_{i=1}^m x_i f_i(c_i^\top \theta) \geq \zeta$, or*
2. *Scenario two: outputs a feasible $x$ to $\mathsf{P}(X,t)$ that corresponds to $X'$ such that $d_j^\top x + \delta_1 \geq \theta_j$ for every $j \in [r_+]$ and $\sum_{i=1}^m x_i f_i(c_i^\top \theta) + \delta_2 \geq \zeta$.*

*Then there exists an oracle described in Lemma 7 with runtime $\lambda' m^{\lambda'} T(\delta_1, \delta_2)$.*

**Proof of Observation 2.** Because every feasible solution $x$ to $\mathsf{P}(X,t)$ corresponds to exactly one index set $X' \subset [m] \setminus (\cup_j X_j) \cup (\cup_j \hat{X}_j)$ such that $0 \leq |X'| \leq \lambda'$, we can give an oracle described in Lemma 7 by enumerating all such index sets $X'$ and applying the oracle in Observation 2.

More specifically, for the oracle described in Lemma 7 with inputs $\theta, \zeta, \delta_1, \delta_2$:

- If the oracle in Observation 2 outputs an $x$ in scenario two with inputs $\theta, \zeta, \delta_1, \delta_2, X'$ for some $X'$, then the oracle described in Lemma 7 also outputs this $x$ in scenario two.

- If the oracle in Observation 2 declares scenario one with inputs $\theta, \zeta, \delta_1, \delta_2, X'$ for all $X'$, then the oracle described in Lemma 7 also declares scenario one.

We can enumerate all index sets $X' \subset [m] \setminus (\cup_j X_j) \cup (\cup_j \hat{X}_j)$ such that $0 \leq |X'| \leq \lambda'$ by simply enumerating all combinations of $|X'|$ indices from $[m] \setminus (\cup_j X_j) \cup (\cup_j \hat{X}_j)$. Because there are at most $\sum_{k=0}^{\lambda'} \binom{m}{\lambda'} \leq \lambda' m^{\lambda'}$ such index sets $X'$, the runtime of the oracle described in Lemma 7 is $\lambda' m^{\lambda'} T(\delta_1, \delta_2)$.

By Observation 2, it suffices to give an oracle as described. Below we give such an oracle.

First, suppose $|X'| < \lambda'$. Assume there exists a feasible solution $z$ to $\mathsf{P}(X, t)$ that corresponds to $X'$. Let $Z = \{i \in [m] \setminus (\cup_j X_j) \cup (\cup_j \hat{X}_j) \mid z_i = 1\}$. Then because $|X'| = \min\{\lambda', |Z|\} = |Z|$ and $X' \subset Z$, we have $X' = Z$. Therefore, there is a unique feasible solution $z$ of $\mathsf{P}(X, t)$ that corresponds to $X'$, namely $z_i = 1$ for every $i \in X' \cup (\cup_j X_j)$ and $z_i = 0$ for every $i \notin X' \cup (\cup_j X_j)$. Therefore, if $|X'| < \lambda'$, the oracle in Observation 2 can directly check this unique feasible solution of $\mathsf{P}(X, t)$ that corresponds to $X'$, and outputs the correct scenario accordingly.

From now on, we assume $|X'| = \lambda'$. Fix any $\theta \in \mathbb{R}^{r_+}$, any $\zeta \in \mathbb{R}_+$, any $\delta_1, \delta_2 > 0$, and any $X'$. The oracle essentially needs to determine the existence of a feasible binary solution to a system of linear inequalities, which is NP-hard in general. However, the linear constraints of $\mathsf{P}(X, t)$ lie in a lower-dimensional subspace, a structure we can exploit by solving a relaxation of the system, obtained by replacing binary variables with continuous ones, and rounding its solution back to a binary solution. Because of the rounding, it is possible that the values $(d_1^\top x, \ldots, d_{r_+}^\top x)$ and $\sum_{i=1}^m x_i f_i(c_i^\top \theta)$ of the rounded solution are out of the desired ranges. However, the valid tuple $X$ and the index set $X'$ we fixed before ensures that the gaps between the values and the desired ranges are within small constants.

We define a polyhedron $PH \subset \mathbb{R}^m$ as follows:

$$\begin{aligned}
\sum_{j=1}^{r_+} a_{ij}\, b_j^\top x \leq t_i & \qquad \text{for } i = 1, \ldots, m, \\
e^\top x \leq k, & \\
d_j^\top x \geq \theta_j & \qquad \text{for } j = 1, \ldots, r_+, \\
\sum_{i=1}^m x_i\, f_i(c_i^\top \theta) \geq \zeta, & \\
x_i = 1 & \qquad \text{for } i \in \Big(\bigcup_j X_j\Big) \cup X', \\
x_i = 0 & \qquad \text{for } i \in \Big(\bigcup_j \hat{X}_j\Big) \cup \hat{X}', \\
x_i \in [0, 1] & \qquad \text{for } i \notin \Big(\bigcup_j X_j\Big) \cup \Big(\bigcup_j \hat{X}_j\Big) \cup X' \cup \hat{X}'.
\end{aligned} \tag{$PH$}$$

Then $PH$ is a polyhedron in $\mathbb{R}^m$ defined by at most $3m + r_+ + 2$ inequalities. Let LP$(m, n)$ be the runtime of solving a linear program with $m$ variables and $n$ constraints. Then checking whether $PH$ is non-empty and return a point in $PH$ if $PH$ is non-empty can be done in runtime LP$(m, 3m + r_+ + 2)$. For more on the runtime of solving a linear program, we refer the readers to e.g. [22, 18] (ellipsoid methods) and [32, 45] (interior point methods). In what follows we assume that $PH \neq \emptyset$, otherwise the oracle outputs scenario one.

**Lemma 8** *If $PH \neq \emptyset$, then we can find a point $z \in PH$ with at most $2r_+ + 2$ fractional components in runtime* LP$(m, 3m + r_+ + 2) +$ LP$(m, 2m + 2r_+ + 2)$.

**Proof of Lemma 8.** Let $z^* \in PH$ be an (arbitrary) point found in runtime LP$(m, 3m + r_+ + 2)$. Let $PH(z^*) \subset \mathbb{R}^{m - |(\cup_j X_j) \cup (\cup_j \hat{X}_j) \cup X' \cup \hat{X}'|}$ be the polyhedron on variable $y$, where the index set of

$y$ is taken to be $I_y = [m] \setminus (\cup_j X_j) \cup (\cup_j \hat{X}_j) \cup X' \cup \hat{X}'$, with the following constraints:

$$\begin{aligned}
\sum_{i\in I_y} b_{ji}\, y_i &\le \sum_{i\in I_y} b_{ji}\, z_i^* && \text{for } j = 1,\dots,r_+,\\
\sum_{i\in I_y} y_i &\le \sum_{i\in I_y} z_i^*,\\
\sum_{i\in I_y} d_{ji}\, y_i &\ge \theta_j - \sum_{i\in(\bigcup_j X_j)\cup X'} d_{ji} && \text{for } j = 1,\dots,r_+, \qquad (PH(z^*))\\
\sum_{i\in I_y} y_i\, f_i(c_i^\top\theta) &\ge \zeta - \sum_{i\in(\bigcup_j X_j)\cup X'} f_i(c_i^\top\theta),\\
y_i &\in [0,1] && \text{for } i\in I_y.
\end{aligned}$$

Note that $PH(z^*) \neq \emptyset$ since the projection of $z^*$ on $\mathbb{R}^{|I_y|}$ is in $PH(z^*)$. Because $PH(z^*)$ has $2r_+ + 2$ linear inequalities other than the inequalities $y_i \in [0,1]$ for $i \in I_y$, we can compute a vertex $y^*$ of $PH(z^*)$ with at most $2r_+ + 2$ fractional components with runtime LP$(m, 2m + 2r_+ + 2)$ (see a standard textbook on linear programming, e.g., [41]).

Let $z \in [0,1]^m$ where

$$z_i = \begin{cases} 1 & \text{if } i \in (\cup_j X_j) \cup X' \\ 0 & \text{if } i \in (\cup_j \hat{X}_j) \cup \hat{X}' \\ y_i^* & \text{if } i \in I_y \end{cases}.$$

Then $z$ has at most $2r_+ + 2$ fractional components. We show that $z \in PH$. Because $z_i = z_i^* = 1$ for $i \in (\cup_j X_j) \cup X'$ and $z_i = z_i^* = 0$ for $i \in (\cup_j \hat{X}_j) \cup \hat{X}'$, the last three sets of constraints of $PH$ is satisfied. By the first set of constraints of $PH(z^*)$ we have $\sum_{i\in I_y} b_{ji} z_i \le \sum_{i\in I_y} b_{ji} z_i^*$. Because $a_{ij} \ge 0$ for every $i, j$, the first set of constraints of $PH$ is satisfied. Similarly the second constraint of $PH$ is also satisfied. By the third set of constraints of $PH(z^*)$ we have

$$d_j^\top z = \sum_{i\in(\cup_j X_j)\cup X'} d_{ji} + \sum_{i\in I_y} d_{ji} z_i \ge \sum_{i\in(\cup_j X_j)\cup X'} d_{ji} + \left(\theta_j - \sum_{i\in(\cup_j X_j)\cup X'} d_{ji}\right) = \theta_j,$$

so the third set of constraints of $PH$ is satisfied. Similarly the fourth constraint of $PH$ is also satisfied. Therefore $z \in PH$ is the desired point.

Let $z$ be the point obtained in Lemma 8. Then $z$ satisfies all the constraints of $\mathsf{P}(X,t)$ except the integrality constraints. We round $z$ down to obtain a feasible solution: let $\bar{z} \in \{0,1\}^m$ where $\bar{z}_i = \lfloor z_i \rfloor$ for each $i$. Notice that since $w'_{ij} > 0$ for every $i, j$, we have $e^\top \bar{z} \le e^\top z \le k$ and ${w'_i}^\top \bar{z} \le {w'_i}^\top z \le t_i$ for every $i \in [m]$. Therefore $\bar{z}$ is feasible to $\mathsf{P}(X,t)$. Moreover, since $\bar{z} = 1$ for $i \in X'$ and $\bar{z} = 0$ for $i \in \hat{X}'$, we have that $\bar{z}$ corresponds to $X'$.

In the final step, we show that by setting $\lambda, \lambda'$ appropriately, $\bar{z}$ satisfies the conditions in scenario two of Observation 2, hence completing the oracle in Observation 2.

**Lemma 9** *Set* $\lambda = \lceil (2r_+ + 2)(Vu)_{\max}/\delta_1 \rceil$ *and* $\lambda' = \lceil (2r_+ + 2)k \max_{i\in[m]}\{f_i((Vu)_{\max})\}/\delta_2 \rceil$. *Then* $d_j^\top \bar{z} + \delta_1 \ge \theta_j$ *for every* $j \in [r_+]$*, and* $\sum_{i=1}^m \bar{z}_i f_i(c_i^\top\theta) + \delta_2 \ge \zeta$.

**Proof of Lemma 9.** Because $z \in PH$, we have $d_j^\top z \ge \theta_j$ for every $j \in [r_+]$ and $\sum_{i=1}^m z_i f_i(c_i^\top\theta) \ge \zeta$. Fix $j \in [r_+]$ and let $\ell \in X_j$ be an index where $d_{j\ell} = \min_{\ell'\in X_j}\{d_{j\ell'}\}$. Then since $|X_j| = \lambda$, we have

$$d_j^\top z \ge \sum_{\ell'\in X_j} d_{j\ell'} z_{\ell'} \ge \lambda d_{j\ell}.$$

On the other hand, $\bar{z}$ is obtained by rounding $z$ down. Notice that $z_{\ell'} \in \{0,1\}$ for all $\ell' \in X_j \cup \hat{X}_j$, that is, for all $\ell'$ such that $d_{j\ell'} > d_{j\ell}$. Therefore for all $\ell'$ such that $d_{j\ell'} > d_{j\ell}$, we have $z_{\ell'} = \bar{z}_{\ell'}$. By Lemma 8, $z$ has at most $2r_+ + 2$ fractional components. Therefore, because $\theta_j \le d_j^\top z \le (Vu)_{\max}$, we have

$$\begin{aligned} d_j^\top \bar{z} &\geq d_j^\top z - (2r_+ + 2)d_{j\ell} \\ &\geq d_j^\top z - (2r_+ + 2)d_j^\top z/\lambda \\ &\geq \theta_j - (2r_+ + 2)(Vu)_{\max}/\lambda \\ &\geq \theta_j - \delta_1. \end{aligned}$$

Similarly, let $p \in X'$ be an index where $f_p(c_p^\top \theta) = \min_{p' \in X'}\{f_{p'}(c_{p'}^\top \theta)\}$. Then since $|X'| = \lambda'$, we have

$$\sum_{i=1}^m z_i f_i(c_i^\top \theta) \geq \sum_{p' \in X'} z_{p'} f_{p'}(c_{p'}^\top \theta) \geq \lambda' f_p(c_p^\top \theta).$$

On the other hand, $\bar{z}$ is obtained by rounding $z$ down. Notice that $z_{p'} \in \{0,1\}$ for all $p' \in X' \cup \hat{X}'$, that is, for all $p'$ such that $f_{p'}(c_{p'}^\top \theta) > f_p(c_p^\top \theta)$. Therefore for all $p'$ such that $f_{p'}(c_{p'}^\top \theta) > f_p(c_p^\top \theta)$, we have $z_{\ell'} = \bar{z}_{\ell'}$. By Lemma 8 $z$ has at most $2r_+ + 2$ fractional components. Therefore, because $\zeta \leq \sum_{i=1}^m z_i f_i(c_i^\top \theta) \leq k \max_{i \in [m]}\{f_i((Vu)_{\max})\}$,

$$\begin{aligned} \sum_{i=1}^m \bar{z}_i f_i(c_i^\top \theta) &\geq \sum_{i=1}^m z_i f_i(c_i^\top \theta) - (2r_+ + 2) f_p(c_p^\top \theta) \\ &\geq \sum_{i=1}^m z_i f_i(c_i^\top \theta) - (2r_+ + 2)\sum_{i=1}^m z_i f_i(c_i^\top \theta)/\lambda' \\ &\geq \zeta - (2r_+ + 2)k \max_{i \in [m]}\{f_i((Vu)_{\max})\}/\lambda' \\ &\geq \zeta - \delta_2. \end{aligned}$$

## C.9 Completing the Proof

Finally, we analyze our algorithm's overall performance and runtime. Let $\delta_1 = \epsilon$ and $\delta_2 = k\epsilon$, and in order to apply Lemma 9, we set $\lambda = (2r_+ + 2)(Vu)_{\max}/\epsilon$ and $\lambda' = (2r_+ + 2)\max_{i \in [m]}\{f_i((Vu)_{\max})\}/\epsilon$. We will treat the performance guarantee and runtime analysis separately.

**Performance Guarantee:** Let

$$c_{\epsilon,\gamma,W'_{\min}} = \frac{(1+\gamma)\epsilon}{W'_{\min}}.$$

The algorithm ALG (for solving $\mathsf{P}(X,t)$) in Lemma 7 satisfies

$$\begin{aligned} \mathrm{ALG}_{\mathsf{P}(X,t)} &\geq \left(1 - g\left(\frac{\delta_1(1+\gamma)}{W'_{\min}}\right)\right)(\mathrm{OPT}_{\mathsf{P}(X,t)} - 2\delta_2) - kh\left(\frac{\delta_1(1+\gamma)}{W'_{\min}}\right) \\ &= (1 - g(c_{\epsilon,\gamma,W'_{\min}}))\mathrm{OPT}_{\mathsf{P}(X,t)} - k(2\epsilon(1 - g(c_{\epsilon,\gamma,W'_{\min}})) + h(c_{\epsilon,\gamma,W'_{\min}})). \end{aligned}$$

This gives the ALG$'$ (for solving $\mathsf{P}(X,t)$) in Lemma 6 with

$$g'(\epsilon) = g(c_{\epsilon,\gamma,W'_{\min}})$$

and

$$h'(\epsilon) = k(2\epsilon(1 - g(c_{\epsilon,\gamma,W'_{\min}})) + h(c_{\epsilon,\gamma,W'_{\min}})).$$

Therefore, the algorithm ALG (for solving $\mathsf{P}(X)$) in Lemma 6 satisfies

$$\begin{aligned} \mathrm{ALG}_{\mathsf{P}(X)} &\geq (1 - g'(\epsilon))((1 - g(c_{\epsilon,\gamma,W'_{\min}}))\mathrm{OPT}_{\mathsf{P}(X)} + kh(c_{\epsilon,\gamma,W'_{\min}})) - h'(\epsilon) \\ &= (1 - g(c_{\epsilon,\gamma,W'_{\min}}))((1 - g(c_{\epsilon,\gamma,W'_{\min}}))\mathrm{OPT}_{\mathsf{P}(X)} + kh(c_{\epsilon,\gamma,W'_{\min}})) \\ &\quad - k(2\epsilon(1 - g(c_{\epsilon,\gamma,W'_{\min}})) + h(c_{\epsilon,\gamma,W'_{\min}})) \\ &= (1 - g(c_{\epsilon,\gamma,W'_{\min}}))^2\mathrm{OPT}_{\mathsf{P}(X)} - k(2\epsilon(1 - g(c_{\epsilon,\gamma,W'_{\min}})) + g(c_{\epsilon,\gamma,W'_{\min}})h(c_{\epsilon,\gamma,W'_{\min}})). \end{aligned}$$

Therefore, the algorithm ALG (for solving $\mathsf{P}'(I)$) in Lemma 4 satisfies

$$\mathrm{ALG}_{\mathsf{P}'(I)} \geq (1 - g(c_{\epsilon,\gamma,W'_{\min}}))^2\mathrm{OPT}_{\mathsf{P}'(I)} - k(2\epsilon(1 - g(c_{\epsilon,\gamma,W'_{\min}})) + g(c_{\epsilon,\gamma,W'_{\min}})h(c_{\epsilon,\gamma,W'_{\min}})).$$

Finally, we apply Lemma 3 by plugging in $1-\alpha = (1-g(c_{\epsilon,\gamma,W'_{\min}}))^2$ and $\beta = -k(2\epsilon(1-g(c_{\epsilon,\gamma,W'_{\min}})) + g(c_{\epsilon,\gamma,W'_{\min}})h(c_{\epsilon,\gamma,W'_{\min}}))$. This gives

$$\begin{aligned}\mathrm{ALG}_{\mathsf{P}(I)} \geq (1-g(2\gamma(Vu)_{\max}))^2(1-g(c_{\epsilon,\gamma,W'_{\min}}))^2\mathsf{OPT}_{\mathsf{P}(I)}\\ - k(1-g(2\gamma(Vu)_{\max}))(2\epsilon(1-g(c_{\epsilon,\gamma,W'_{\min}})) + g(c_{\epsilon,\gamma,W'_{\min}})h(c_{\epsilon,\gamma,W'_{\min}})\\ + (1-g(c_{\epsilon,\gamma,W'_{\min}}))^2h(2\gamma(Vu)_{\max}) + h(2\gamma(Vu)_{\max})).\end{aligned}$$

**Runtime Analysis:** By Lemma 8, we give an oracle described in Observation 2 with runtime

$$T_{\mathsf{LP}} := \mathsf{LP}(m, 3m+r_+ +2) + \mathsf{LP}(m, 2m+2r_+ +2).$$

Therefore, by Observation 2, we give an oracle described in Lemma 7 with runtime

$$\lambda' m^{\lambda'} T_{\mathsf{LP}}.$$

Therefore, the algorithm ALG$'$ (for solving $\mathsf{P}(X,t)$) in Lemma 6 has runtime

$$\begin{aligned}&\lceil((Vu)_{\max} - \min\{0,(Vu)_{\min}\})/\delta_1\rceil^{r_+} \cdot \left\lceil k\max_{i\in[m]}\{f_i((Vu)_{\max})\}/\delta_2\right\rceil \cdot \lambda' m^{\lambda'} T_{\mathsf{LP}}\\ &= \lceil((Vu)_{\max} - \min\{0,(Vu)_{\min}\})/\epsilon\rceil^{r_+} \cdot \left\lceil \max_{i\in[m]}\{f_i((Vu)_{\max})\}/\epsilon\right\rceil \cdot \lambda' m^{\lambda'} T_{\mathsf{LP}}.\end{aligned}$$

Therefore, the algorithm ALG$'$ (for solving $\mathsf{P}(X)$) in Lemma 4 has runtime

$$\left\lceil\left(\frac{4(1+\gamma)k}{\epsilon W'_{\min}}\right)^{r_+}\right\rceil \cdot \lceil((Vu)_{\max} - \min\{0,(Vu)_{\min}\})/\epsilon\rceil^{r_+} \cdot \left\lceil \max_{i\in[m]}\{f_i((Vu)_{\max})\}/\epsilon\right\rceil \cdot \lambda' m^{\lambda'} T_{\mathsf{LP}}.$$

Finally, by Lemma 4, our algorithm's runtime is

$$\begin{aligned}&r_+ m\log_2 m + \lambda m^{\lambda}\\ &+ \lambda r_+^2 m^{\lambda r_+} + \left\lceil\left(\frac{4(1+\gamma)k}{\epsilon W'_{\min}}\right)^{r_+}\right\rceil \cdot \left\lceil\frac{(Vu)_{\max} - \min\{0,(Vu)_{\min}\}}{\epsilon}\right\rceil^{r_+}\\ &\cdot \left\lceil\frac{\max_{i\in[m]}\{f_i((Vu)_{\max})\}}{\epsilon}\right\rceil \cdot \lambda' m^{\lambda r_+ + \lambda'} T_{\mathsf{LP}}.\end{aligned}$$