NORTHWESTERN UNIVERSITY

On MAP Inference of Ferromagnetic Potts Models and Nonsymmetric Determinantal Point Processes

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Computer Science

By

Aravind Reddy Talla

EVANSTON, ILLINOIS

June 2023

Abstract

On MAP Inference of Ferromagnetic Potts Models and
Nonsymmetric Determinantal Point Processes

Aravind Reddy Talla

In the Maximum-a-Posteriori (MAP) Inference problem, for any given probability distribution, the goal is to find the point in the support of that distribution with the highest probability. Potts models and Determinantal Point Processes (DPPs) are probabilistic models that were introduced in the context of statistical physics several decades ago. They have been extensively used in several computer science applications like computer vision, recommender systems, and document summarization. Exact MAP Inference in these models correspond to NP-hard combinatorial optimization problems and so approximate inference algorithms have been extensively studied. For MAP Inference in Ferromagnetic Potts models, we provide a strong justification for the excellent performance of a linear programming relaxation approach by going beyond worst-case analysis. For MAP Inference in Nonsymmetric Determinantal Point Processes, we provide the first one-pass streaming and online algorithms.

Acknowledgements

This is going to be a long one. Take a seat and grab some popcorn...

> Master Shifu: "*But a peach cannot defeat Tai Lung!*"
>
> Master Oogway: "*Maybe it* ***can****... If you are willing to guide it, to nurture it, to believe in it.*" - Kung Fu Panda

They say it takes a village to raise a child. Similarly, it took the care and patience of a lot of people in my journey from a student who was clueless about what research was or how to go about it to what I am now. First and foremost, I would like to thank my wonderful advisors Konstantin Makarychev and Aravindan Vijayaraghavan. It would take too many words to express my entire gratitude to both of them. But there are a couple of moments which come to mind, which I hope might bring solace to future generations of PhD students who might stumble upon my thesis.

My very first paper (Makarychev, Reddy, and Shan, 2020) was with Kostya and Liren, which appeared in NeurIPS 2020. I can't thank them enough for being involved in this project! It provided me confidence that I can contribute in some meaningful way to publishable research. I learned a lot from working on this paper, not only about technical material related to the paper, but also about how research papers are written, and how rejection is a part and parcel of academic writing. We had first submitted the paper to ICML 2020 where it was rejected by the meta-reviewer even after getting positive reviews from all the reviewers. That had shattered my confidence as I had expected the paper to be accepted without much fuss and I went from being an author on 0 papers to 0 papers even after having spent several months working on a project. But Kostya kept encouraging us that it was a good paper. It is in moments like these where I think it is the most important to have a senior researcher guide you.

Another moment that comes to mind is a discussion with Aravindan close to the end of my first year, when I had to decide my advisor. He had asked me if I was interested in working on a project for which he had an NSF grant. Although I had read and presented a few papers related to that project, I was a bit worried if I could contribute in any research related to that. And I remember Aravindan telling me that he would do the worrying and I only needed to do the working :P . At least a little bit of that optimism has rubbed off on me!

There are a lot of other such stories which unfortunately I don't think I can share here due to space constraints... But the overall takeaway is that I needed a lot of encouragement from Aravindan, Kostya, and lot of my other collaborators to get going on this path of research. I would also like to thank Samir Khuller and David Sontag for being on my qualifying exam committee and also for taking out extra time to meet with me and discussing some papers and book chapters after the exam. Also thanks to Aditya Bhaskara for serving on my thesis committee.

Thanks to Ryan Rossi and Ritwik Sinha for being my hosts at Adobe Research for the summers of 2021 and 2022, respectively.

Thanks to Zhao Song for being an enthusiastic collaborator and mentor on multiple projects (Reddy et al., 2022a;b; Qin et al., 2022), encouraging me to write proposals for gift grants from Adobe Research and providing feedback for them, and for writing recommendation letters for my postdoc applications. I was pleasantly surprised to find out that a grant application we had written ended up being funded by Adobe for $\$20,000$! Might not seem like much money when compared to NSF grants but it is a massive amount of money from my perspective as a grad student :P . Again, similar to my first paper with Kostya and Liren, the value of the proposal is far more than the money for me. It gave me confidence that I can indeed write successful grant proposals which can compete even with faculty from the very best institutions across this Land of the Free and the Home of the Brave.

In this section, I would like to thank my collaborators (in no particular order, please forgive me if I forget to include your name): Konstantin Makarychev, Liren Shan, Hunter Lang, Aravindan Vijayaraghavan, David Sontag, Ryan A. Rossi, Zhao Song, Anup Rao, Tung Mai, Nedim Lipka, Gang Wu, Eunyee Koh, Nesreen K. Ahmed, Lichen Zhang, Ritwik Sinha, David Arbour, Raghav Addanki, Nikos Vlassis, Lianke Qin, Zhaozhuo Xu, and Danyang Zhuo. Special thanks to my coauthors of the papers (Lang, Reddy, Sontag, and Vijayaraghavan, 2021a) and (Reddy, Rossi, Song, Rao, Mai, Lipka, Wu, Koh, and Ahmed, 2022a) on which chapters 2 and 3 are respectively based on, for letting me include those results in this thesis.

I would also like to thank all members of the Northwestern CS Theory group, who provided a very warm and welcoming atmosphere for the while I had been there, and also for teaching me in courses or being my project/homework partners (in no particular order, please forgive me if I forget to include your name): Jason Hartline, Anindya De, Annie Liang, Ben Golub, Xiao Wang, Lev Reyzin, Xue Chen, Huck Bennett, Shravas Rao, Hedyeh Beyhaghi, Vaggos Chatziafratis, Sami Davies, Jinshuo Dong, Quanquan C. Liu, Don Stull, Sheng Yang, Randall Berry, Yingkai Li, Yiding Feng, Abhratanu Dutta, Aleck Johnsen, Paula Kayongo, Leif Rasmussen, Aidao Chen, Charles Cui, Nirmit Joshi, Sanchit Kalhan, Yiduo Ke, Sheng Long, Michail Mamakos, Anant Shah, Vaidehi Srinivas, Pattara Sukprasert, Alex Tang, Matthew VonAllmen, Yifan Wu, Chenhao Zhang, Shirley Zhang, Liam O'Carroll, and Nathan White.

I also want to thank the staff at the Northwestern CS department, several of whom helped me directly over the years regarding various issues (especially reimbursements), including but not limited to: Xiaolin Wang, Julia Blend, James Devine, Tatiana Meza, Colleen Gallagher, and Iliana Vargas.

> "*It's been a long day*
> *Without you, my friend*
> *And I'll tell you all about it when I see you again*
> *We've come a long way*
> *From where we began*
> *Oh, I'll tell you all about it when I see you again*
> *When I see you again...*" - Charlie Puth, See You Again

Here, I would like to thank my friends in the Chicago area, for providing some much needed warmth in the frigid winters, and also for lots of board games/hiking/biking sessions (in no particular order, please forgive me if I forget to include your name) : Tushant, Kshitij, Lavanya, Pallav, Susho, Sheetal, Vasundhara, Ashlee, Khadija, Jafar, Amruta, Ishika, Nikhil, Ayushi, Himani, Aniket, Sudarshan, Palak, Lakshman, Anmol, Naren, Goutham, Antares, Iffat, Joo Seung, Sky, Carmanie, Fizza, Narasimhan, and Omar.

As we reach close to the end of the acknowledgments, I would also like to take this oppurtunity to thank all my teachers and mentors, all the way from IIT Kanpur to my primary school. Special thanks to my research mentors in undergrad and also my undergrad summer internships: Rajat Mittal, Iordanis Kerenidis, Eleni Diamanti, Hartmut Klauck, Rahul Jain, Prasanta K. Panigrahi. Thanks a lot to Rajat and Iordanis for writing recommendation letters for my PhD applications. Also thanks to Manindra Agrawal for teaching amazing and inspiring courses on Abstract Algebra and Cryptography, and also being one of my recommendation letter writers for PhD applications.

Of course, I have only reached the tip of the iceberg in terms of all people who have helped and supported me through the years. You know who you are! Thank you :)

> "*How could we not talk about family when family's all that we got?*
> *Everything I went through, you were standing there by my side*
> *And now you gon' be with me for the last ride*"
>
> - Wiz Khalifa, See You Again

Last, but of course not the least, I would like to thank my family for always being there through my ups and downs and always supporting me - My Mom: Vani, My Dad: Venkat Ramana, and My Annayya (elder brother): Harsha Vardhan. Special thanks also to some of my extended family who live(/d) in the Chicago area: Babu mamayya, Swetha attamma, Nayan, and Ninni.

$$$: I would like to thank NSF for providing financial support through grants CCF-1955351, CCF-1637585, CCF-1652491, HDR TRIPODS CCF-1934931. I would also like to thank Adobe Research for hosting me for the summers of 2021 and 2022, and also for providing a gift grant to support my research. I would also like to thank Kostya,

Aravindan, and Shravas for supporting me as a Teaching Assistant for Algorithms (336) and Mathematical Foundations of Computer Science (212). Thanks to their guidance, I also received the Peter and Adrienne Barris Outstanding Teaching Assistant award for the spring 2022 quarter for COMP_SCI 212: Mathematical Foundations of Computer Science.

At the very very end, I would like to thank all the reviewers of my conference submissions for doing the much needed but not as much appreciated reviewing work. Thank you!!

Although it might be a bit unconventional, I would also like to thank some great film-makers : Rajamouli, Peter Jackson, James Cameron, Russo brothers, and many others, for making awesome films, which kept me sane even after being cooped up for months on end in my studio apartment. I would also like to thank some amazing writers (most of whom are also scientists) for writing amazing and inspiring books: Judea Pearl, Siddhartha Mukherjee, Atul Gawande, David Sinclair, Satchin Panda, Nassim Nicholas Taleb, George R.R. Martin, J.R.R. Tolkien and many others.

For historical reference, let me also add that majority of my PhD took place during the COVID-19 pandemic. I joined Northwestern in September of 2018 and the pandemic hit the Chicago area in January 2020. This led to a tremendous upheaval of my (and everyone else's) life. Fortunately, neither I, nor my immediate family were affected by any unforeseeable tragedy. But my heart goes out to all who were impacted in any direct way. I would also like to thank all front-line workers who risked their lives to help us all keep functioning in at least some way that resembled normalcy.

Dedication

Dedicated to my parents Vani and Venkat Ramana, and my brother Harsha

Table of Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

"*All models are wrong, but some are useful*"

- George Box

In this chapter, we will introduce the MAP Inference problem, Ferromagnetic Potts models, and Nonsymmetric Determinantal Point Processes, and provide a brief overview of the results which will be discussed more extensively in the rest of the thesis. Since this is a thesis submitted in the Computer Science *theory* group at Northwestern, we will go over the specific applications later but first discuss why these topics are theoretically fascinating to study.

The first major part of this thesis studies Ferromagnetic Potts models, which are a class of *Probabilistic Graphical Models* (Wainwright & Jordan, 2008; Koller & Friedman, 2009), which are often referred to as PGMs. Unstructured probability distributions over n binary variables need $2^n - 1$ parameters to describe, which is prohibitive except in the case of very small models. One common way to impose structure on the probability distribution is by using *graphs* to model dependencies between variables. In this section, we will not discuss in detail the relationship between the graphs used and the corresponding distributions but will give a simple example first: If a weighted graph is used to represent a probability distribution, we can think of the weights between edges as corresponding to the correlation between the two corresponding variables in the distribution. PGMs are very interesting to study because they combine two extremely interesting mathematical areas, i.e., probability theory and graph theory, in very fascinating and exciting ways. Often, the complexity of different tasks we care about related to the distribution, like learning and inference, are intricately related to the properties of the corresponding graph, like connectivity and tree-width.

The second major part of this thesis studies Determinantal Point Processes (Kulesza & Taskar, 2012), often referred to as DPPs. Unlike PGMs, where a *graph* is used to represent the probability distribution, DPPs use a *matrix*. Properties of the distribution correspond to properties of the matrix and so DPPs offer an exciting way to study probability theory and linear algebra together.

Overall, the algorithmic problems we study correspond to exciting discrete optimization problems which involve solutions using various algorithmic techniques like linear programming, greedy, local search, and other combinatorial algorithms.

1.1. MAP Inference

In this thesis, we are concerned with the Maximum a Posteriori (MAP) Inference problem for two classes of probabilistic models: Ferromagnetic Potts models and Non-symmetric Determinantal Point Processes. For a given probabilistic model, the MAP Inference problem corresponds to computing the *mode* of the distribution corresponding to that model. In other words, it corresponds to finding the *most likely assignment* to the variables under the probabilistic model. This problem is referred to as Maximum a *Posteriori* Inference because the distribution we are concerned with is the *posterior* *distribution*, which incorporates both the *prior distribution* and also information from the observed samples.

1.2. Ferromagnetic Potts models

The *Potts model* is an extremely well studied mathematical model in statistical mechanics, which was introduced more than half a century ago by Renfrey Potts towards the end of his PhD thesis (Potts, 1952). It is a generalization of the classic *Ising model*. In the Ising model, all states are constrained to be binary, whereas the Potts model allows the states to have a value from k discrete choices, where $k \geq 2$. In this section, we will present a very brief introduction to the Potts model as is relevant for the rest of the thesis (chapter 2 in particular). We refer the reader to the survey monograph by Savchynskyy (2019) for a more detailed introduction.

Given an undirected graph $G(V, E)$ with edge weights $w : E \mapsto \mathbb{R}_{\geq 0}$ and node costs $c : V \times [k] \mapsto \mathbb{R}$, for any labeling $g : V \mapsto \{1, 2, \ldots, k\}$ for integer $k \geq 2$, the *energy* $Q(g)$ is defined as

$$Q(g) \coloneqq \sum_{u \in V} c(u, g(u)) + \sum_{(u,v) \in E} w(u, v) \mathbb{1}[g(u) \neq g(v)]$$

Note that the first term in the above expression takes into account the individual node preferences for each label and the second term leads to a “smoothening” effect i.e., leads to adjacent nodes preferring the same label. In other words, the second term corresponds to the weight of the *cut* edges. The reason why we call the specific Potts models we study as “Ferromagnetic” is because the choice of the non-negativity of the edge-weights leads to an “attraction” between neighboring nodes i.e., neighboring nodes preferring the same state, similar to ferromagnets.

The term θ is often used in the graphical models literature to represent the edge weights and node costs stacked together i.e., $\theta \in \mathbb{R}^{k|V|+k^2|E|}$ where $\theta = (\theta_u, \theta_{uv})$ are the unary cost functions and pairwise cost functions respectively. In Potts models, we have $\theta_u(i) = c(u, i)$ and $\theta_{uv}(i, j) = w(u, v)\mathbb{1}[i \neq j]$.

The probability distribution corresponding to the Potts model parameterized by θ is

$$\mathbb{P}(g, \theta) = \frac{1}{Z(\theta)} \exp(-Q(g, \theta))$$

for all labelings $g : V \mapsto \{1, 2, \ldots, k\}$.

The normalizing constant $Z(\theta)$[*] is also known as the *Partition function.* Computing the partition function for the Potts model is a hard and interesting algorithmic question with a flurry of work in the TCS community (Goldberg and Jerrum, 2012; Galanis, Stefankovic, Vigoda, and Yang, 2016; Carlson, Davies, Fraiman, Kolla, Potukuchi, and Yap, 2022). But note that, since $Z(\theta)$ is independent of g, we don't need to compute Z as far as the MAP inference question is concerned i.e., we only need to find $\arg\max_{g:V\mapsto[k]} \mathbb{P}(g, \theta)$, which is the same as $\arg\max_{g:V\mapsto[k]} \exp(-Q(g, \theta))$, which is equivalent to

$$\arg\min_{g:V\mapsto[k]} Q(g, \theta) = \arg\min_{g:V\mapsto[k]} \sum_{u\in V} c(u, g(u)) + \sum_{uv\in E} w(u, v)\mathbb{1}g(u) \neq g(v)$$

Indeed, for the rest of the thesis, we will exclusively consider this above problem, which is also known as the *energy minimization problem* for graphical models. In particular, this problem in the case of Ferromagnetic Potts models is also known as the *Metric Labeling*

[*] The letter Z is used because it stands for the German word *Zustandssumme*, “sum over states”.

problem (Kleinberg & Tardos, 2002), which is a generalization of the minimum multi-way cut problem (Vazirani, 2001), which is further a generalization of the classic min s-t cut problem. Although min s-t cut can be solved exactly in polynomial time, it turns out that multi-way cut (and so Metric Labeling) is NP-hard (infact APX-hard) (Dahlhaus, Johnson, Papadimitriou, Seymour, and Yannakakis, 1992) when $k \geq 3$.

We will now provide a brief overview of our contributions related to MAP Inference of Ferromagnetic Potts models (Lang, Reddy, Sontag, and Vijayaraghavan, 2021a):

Prior works (Lang, Sontag, and Vijayaraghavan, 2018; 2019) have shown that *perturbation stable* (Bilu & Linial, 2012; Makarychev & Makarychev, 2021) instances of the MAP inference problem in Potts models can be solved exactly using a natural linear programming (LP) relaxation. However, these works give few (or no) guarantees for the LP solutions on instances that do not satisfy the relatively strict perturbation stability definitions. In this work, we go beyond these stability results by showing that the LP approximately recovers the MAP solution of a stable instance even after the instance is corrupted by noise. This "noisy stable" model realistically fits with practical MAP inference problems: we design an algorithm for finding "close" stable instances, and show that several real-world instances from computer vision have nearby instances that are perturbation stable. These results suggest a new theoretical explanation for the excellent performance of this LP relaxation in practice.

1.3. Nonsymmetric Determinantal Point Processes

Determinantal Point Processes (DPPs) were first introduced in the context of quantum mechanics (Macchi, 1975) and have subsequently been extensively studied with applications in several areas of pure and applied mathematics like graph theory, combinatorics, random matrix theory (Hough, Krishnapur, Peres, and Virág, 2006; Borodin, 2009), and randomized numerical linear algebra (Derezinski & Mahoney, 2021). Discrete DPPs have gained widespread adoption in machine learning following the seminal work of Kulesza & Taskar (2012), to which we refer the reader for a more comprehensive introduction.

A discrete DPP is a probability distribution over all subsets of items $[n]$ characterized by a kernel matrix $\boldsymbol{L} \in \mathbb{R}^{n \times n}$ such that the probability of sampling any subset $S \subseteq [n]$ i.e., $\mathbb{P}[S] \propto \det(\boldsymbol{L}_S)$ where $\boldsymbol{L}_S$ is the principal submatrix of $\boldsymbol{L}$ obtained by keeping only the rows and columns corresponding to indices in S. For the DPP corresponding to $\boldsymbol{L}$ to be a valid probability distribution, we need $\det(\boldsymbol{L}_S) \geq 0$ for all $S \subseteq [n]$ since $\mathbb{P}[S] \geq 0$ for all $S \subseteq [n]$. Matrices which satisfy this property are known as P_0-matrices (Fiedler & Pták, 1966). For any symmetric matrix $\boldsymbol{L}$, $\det(\boldsymbol{L}_S) \geq 0$ for all $S \subseteq [n]$ if and only if $\boldsymbol{L}$ is positive semi-definite (PSD) i.e., $\boldsymbol{x}^T \boldsymbol{L} \boldsymbol{x} \geq 0$ for all $\boldsymbol{x} \in \mathbb{R}^n$. Therefore, all symmetric matrices which correspond to valid DPPs are PSD.

Until very recently, most prior work on DPPs focused on the setting where the kernel matrix is symmetric. Due to this constraint, DPPs could only model negative correlations between items. But there are P_0-matrices which are not necessarily symmetric (or even positive semi-definite). For example, $\boldsymbol{L} = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$ is a nonsymmetric P_0 matrix. Recent work has shown that allowing the kernel matrix to be nonsymmetric can greatly increase the expressive power of DPPs and allows them to model *compatible* sets of items (Gartrell, Brunel, Dohmatob, and Krichene, 2019; Brunel, 2018). To differentiate this line of work from prior literature on symmetric DPPs, the term Nonsymmetric DPPs (NDPPs) has often been used. Modeling positive correlations can be useful in many practical scenarios.

For instance, an E-commerce company trying to build a product recommendation system would want the system to increase the probability of suggesting a router if a customer adds a modem to a shopping cart.

State-of-the-art algorithms for MAP inference of NDPPs (Gartrell, Han, Dohmatob, Gillenwater, and Brunel, 2021; Anari and Vuong, 2022) require storing the full data in memory and take multiple passes over the complete dataset. Therefore, these algorithms take too much memory to be useful for industry scale datasets, where the size of the entire dataset can be much larger than the random-access memory available. In the paper (Reddy, Rossi, Song, Rao, Mai, Lipka, Wu, Koh, and Ahmed, 2022a) on which Chapter 3 is based on, we initiate the study of one-pass algorithms for solving the maximum-a-posteriori (MAP) inference problem for Non-symmetric Determinantal Point Processes (NDPPs). In particular, we formulate streaming and online versions of the problem and provide one-pass algorithms for solving these problems. In our streaming setting, data points arrive in an arbitrary order and the algorithms are constrained to use a single-pass over the data as well as sub-linear memory, and only need to output a valid solution at the end of the stream. Our online setting has an additional requirement of maintaining a valid solution at any point in time. We design new one-pass algorithms for these problems and show that they perform comparably to (or even better than) the offline greedy algorithm while using substantially lower memory.

CHAPTER 2

Beyond Perturbation Stability: LP Recovery Guarantees for MAP Inference on Noisy Stable Instances

Note: This chapter is (mostly) a reprint of Lang, Reddy, Sontag, and Vijayaraghavan (2021a).

2.1. Introduction

In this chapter, we study the MAP inference problem in the *Ferromagnetic Potts model*, which is also known as uniform metric labeling (Kleinberg & Tardos, 2002). Given a graph $G = (V, E)$, the problem is to:

$$\underset{x:V\to[k]}{\text{minimize}} \sum_{u\in V} c(u, x(u)) + \sum_{(u,v)\in E} w(u,v)\mathbb{1}[x(u) \neq x(v)].$$

Here we are optimizing over *labelings* $x : V \to [k]$ where $[k] = \{1, 2, \dots, k\}$. The objective is comprised of "node costs" $c : V \times [k] \to \mathbb{R}$, and "edge weights" $w : E \to \mathbb{R}_{>0}$; a labeling x pays the cost $c(u, i)$ when it labels node u with label i and pays $w(u, v)$ on edge (u, v) when it labels u and v differently. This problem is NP-hard for variable $k \geq 3$ (Kleinberg & Tardos, 2002) even when the graph G is planar (Dahlhaus, Johnson, Papadimitriou, Seymour, and Yannakakis, 1992). However, there are several efficient and empirically successful approximation algorithms for the MAP inference problem—such as TRW (Wainwright, Jaakkola, and Willsky, 2005) and MPLP (Globerson and Jaakkola, 2008)—that are related in some way to the *local LP relaxation*, which is also sometimes called the *pairwise LP* (Wainwright and Jordan, 2008; Chekuri, Khanna, Naor, and Zosin, 2001). This LP relaxation returns an approximate MAP solution for most problem instances. However, when the parameters of these models are learned so as to enable good structured prediction, often the LP relaxation exactly or almost exactly recovers the MAP solution (Meshi, London, Weller, and Sontag, 2019). The connection between the LP relaxation and commonly used approximate MAP inference algorithms then leads to the following compelling question, which is of great practical relevance for understanding the "tightness" of the LP solution (informally, how close the LP solution is to the MAP solution):

Can we explain the exceptional performance of the local LP relaxation in recovering the MAP solution in practice?

Several works have studied different conditions that imply the local relaxation or related relaxations are tight (e.g., Kolmogorov and Wainwright, 2005; Wainwright and Jordan, 2008; Thapper, 2012; Weller, Rowland, and Sontag, 2016; Rowland, Pacchiano, and Weller, 2017). Recent work on tightness of the local relaxation has focused on a class of several related conditions known as *perturbation stability.* Intuitively, an instance is perturbation stable if the solution x^* to the MAP inference problem is unique, and moreover, x^* is the unique solution even when the edge weights w are multiplicatively perturbed by a certain adversarial amount (Bilu & Linial, 2012; Makarychev & Makarychev, 2021). This structural assumption about the instance (G, c, w) captures the intuition that, on "real-world" instances, the ground-truth solution is stable and does not change much when the weights are slightly perturbed.

For constants $\beta, \gamma \geq 1$, we say that w' is a (β, γ)-perturbation of the weights w if $\frac{1}{\beta} \cdot w(u,v) \leq w'(u,v) \leq \gamma \cdot w(u,v)$ for all $(u,v) \in E$. Suppose x^* is the unique MAP solution to the instance (G,c,w). Then, we say (G,c,w) is a (β,γ)-stable instance if x^* is also the unique MAP solution to every instance (G,c,w') where w' is a (β,γ)-perturbation of w. Lang, Sontag, and Vijayaraghavan (2018) showed that when (G,c,w) is $(2,1)$-stable, the solution to the local LP relaxation is *persistent* i.e., the LP solution exactly recovers the MAP solution x^*.

While theoretically interesting, $(2,1)$-stability is a strict condition that is unlikely to be satisfied in practice: the solution x^* is not allowed to change *at all* when the weights are perturbed. In practice, it is often the case that real-world instances are not exactly $(2,1)$-stable (Lang, Sontag, and Vijayaraghavan, 2019) and furthermore the LP relaxation is also not exactly persistent on most of those instances. However, the solution of the local LP relaxation is still *nearly persistent* i.e., the LP solution is very close to the MAP solution x^* (see Definition 2.3.1 for a formal definition). Those examples made it clear that theory must go beyond perturbation stability to explain this phenomenon of near-persistence that is prevalent in practice (see e.g., Sontag, 2010; Shekhovtsov, Swoboda, and Savchynskyy, 2017).

Why is the LP relaxation nearly persistent on MAP inference instances in practice?

There are several theoretical frameworks to explain exactness or tightness of LP relaxations, such as total unimodularity, submodularity (Kolmogorov and Wainwright, 2005), and perturbation stability (Lang, Sontag, and Vijayaraghavan, 2018; 2019), as well as structural assumptions about the graph G (Wainwright and Jordan, 2008), or combined assumptions about G and the form of the objective function (Weller, Rowland, and Sontag,

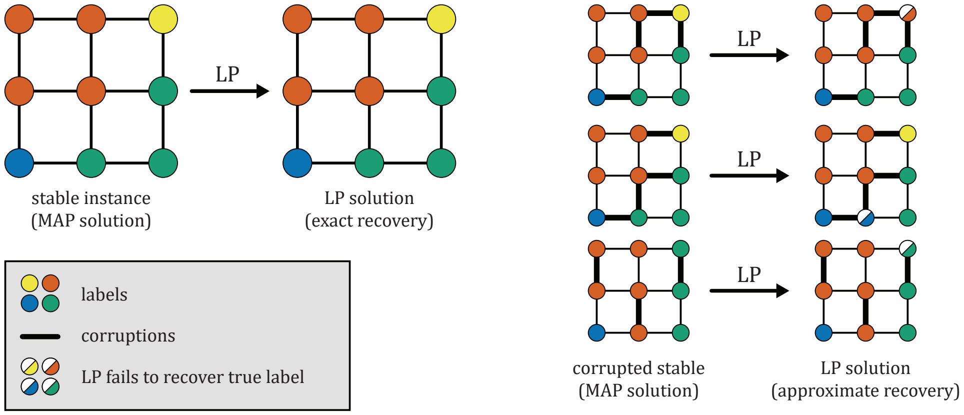


Figure 2.1. Left: prior work (Lang, Sontag, and Vijayaraghavan, 2018) showed that a *stable instance* can be exactly solved efficiently. Colors indicate the label of each vertex in the MAP solution x^*. On stable instances, solving the LP relaxation (represented by the arrow) recovers the MAP solution. However, real-world instances are *not* suitably stable for this result to apply in practice (Lang, Sontag, and Vijayaraghavan, 2019). Right: in this work, we show that solving the LP relaxation on a (slightly) *corrupted* stable instance (corruptions shown as bold edges) *approximately* recovers the original MAP solution. This is true even if the corruption changes the MAP solution (as in the bottom example). In other words, we prove that "easy" instances are still approximately easy even after small perturbations.

2016; Rowland, Pacchiano, and Weller, 2017). However, these frameworks can not be used to prove near-persistence.

Figure 2.1 (informally) shows our main result. The left side depicts the previous result of Lang, Sontag, and Vijayaraghavan (2018): if the instance is $(2, 1)$-stable (a fairly strong structural assumption), the LP relaxation exactly recovers the full solution x^*. This result is limited because real-world instances have been shown to not satisfy $(2, 1)$-stability (Lang, Sontag, and Vijayaraghavan, 2019). The right side shows our main result: if the instance is a *slightly corrupted* $(2, 1)$-stable instance, the LP relaxation still *approximately* recovers the solution x^* to the stable instance.

Intuitively, we may expect a real-world instance to be "close" to a stable instance (i.e., to be a "corrupted stable" instance, as in Figure 2.1) even if the instance itself is not stable. We design an algorithm to check whether this is the case. We find that on several real examples, sparse and small-norm changes to the instance make it appropriately stable for our theorems to apply. In other words, we certify that these real instances are close to stable instances. For these instances, our theoretical results explain why the LP relaxation approximately recovers the MAP solution.

More formally, we assume that there is some *latent* stable instance $(G, \bar{c}, \bar{w})$, and that the observed instance $(G, \hat{c}, \hat{w})$ is a noisy version of $(G, \bar{c}, \bar{w})$ that is close to it. Let $\hat{x}$ be the solution to the local LP on the observed instance $(G, \hat{c}, \hat{w})$, and let $\bar{x}$ be the (unknown) MAP solution on the unseen stable instance $(G, \bar{c}, \bar{w})$. We prove that under certain conditions, the LP solution $\hat{x}$ is *nearly persistent* i.e., the Hamming error $\|\hat{x} - \bar{x}\|_1$ is small (see Definition 2.3.1). In other words, the local LP solution to the observed instance approximately recovers the latent integral solution $\bar{x}$.

We complement this by studying a natural generative model that generates noisy stable instances which, with high probability, satisfy the above conditions for *near persistency*. In other words, the observed instance $(G, \hat{c}, \hat{w})$ is obtained by random perturbations to the latent stable instance $(G, \bar{c}, \bar{w})$, and the LP relaxation approximately recovers the MAP solution to the latent instance with high probability.

Our theoretical results imply that the local LP approximately recovers the MAP solution when the observed instance is close to a stable instance. Our empirical results suggest that real-world instances are very close to stable instances. These results together suggest a new explanation for the near-persistency of the solution of the local LP relaxation for MAP inference in practice. To prove these results and derive our algorithm for finding a "close-by" stable instance, we make several novel technical contributions, which we outline below.

Technical contributions.

- In Section 2.4, we generalize the $(2, 1)$-stability result of Lang, Sontag, and Vijayaraghavan (2018) to work under a much weaker assumption, which we call $(2, 1)$-expansion stability. That is, we prove the local LP is tight on $(2, 1)$-expansion stable instances. Additionally, given the instance's MAP solution, $(2, 1)$-expansion stability is efficiently checkable. To the best of our knowledge, most other perturbation stability assumptions are not known to satisfy this desirable property. This generalization is crucial for the efficiency of our algorithm for finding stable instances that are close to a given observed instance.
- In Section 2.5, we give a simple extension of $(2, 1)$-expansion stability called $(2, 1, \psi)$-expansion stability. We prove it implies a "curvature" result around the MAP solution $\bar{x}$. On instances that satisfy this condition, if a labeling $\hat{x}$ is close in objective value to $\bar{x}$, it must also be close in the solution space. This result lets us translate between objective gap and Hamming distance.
- In Section 2.6, we study a natural generative model where the observed instance is generated from an arbitrary *latent stable* instance by random (sub-Gaussian) perturbations to the costs and weights. We prove that, with high probability, every feasible LP solution takes close objective values on the latent and observed instances. The proof uses a rounding algorithm for metric labeling in a novel way to obtain stronger guarantees. When combined with our other results, this

proves that when the latent instance is $(2, 1, \psi)$-expansion stable, the LP solution is nearly persistent on the observed instance with high probability. These results suggest a theoretical explanation for the phenomenon of near-persistence of the LP solution in practice.

- We design an efficient algorithm for finding $(2, 1, \psi)$-expansion stable instances that are "close" to a given instance $(G, \hat{c}, \hat{w})$ in Section 2.7. To the best of our knowledge, this is the first algorithm for finding close-by stable instances, and is also an efficient algorithm for checking $(2, 1, \psi)$-expansion stability. This algorithm allows us to check whether real-world instances can plausibly be considered "corrupted stable" instances as shown in Figure 2.1.
- We run our algorithm on several real-world instances of MAP inference in Section 2.8, and find that the observed instances $(G, \hat{c}, \hat{w})$ often admit close-by $(2, 1, \psi)$-stable instances $(G, \bar{c}, \bar{w})$. Moreover, we find that the local LP solution $\hat{x}$ typically has very close objective to $\bar{x}$ in $(G, \bar{c}, \bar{w})$. Our curvature result for $(2, 1, \psi)$-stable instances thus gives an explanation for the tightness of the local LP relaxation on $(G, \hat{c}, \hat{w})$.

2.2. Related work

Perturbation stability. Several works have given recovery guarantees for the local LP relaxation on perturbation stable instances of uniform metric labeling (Lang, Sontag, and Vijayaraghavan, 2018; 2019) and for similar problems (Makarychev, Makarychev, and Vijayaraghavan, 2014; Angelidakis, Makarychev, and Makarychev, 2017). Lang, Sontag, and Vijayaraghavan (2019) give partial recovery guarantees for the local LP when parts (*blocks*) of the observed instance satisfy a stability-like condition, and they showed that practical instances have blocks that satisfy their condition. However, the required *block stability* condition in turn depends on certain quantities related to the LP dual. This is unsatisfactory since this does not explain when and why such instances are likely to arise in practice. For a more extensive treatment of Perturbation Stability and its applications in other problems, we refer the reader to Makarychev and Makarychev (2021).

Easy instances corrupted with noise. Our random noise model is similar to several planted average-case models like stochastic block models (SBMs) considered in the context of problems like community detection, correlation clustering and partitioning (see e.g., McSherry, 2001; Abbe, 2018; Globerson, Roughgarden, Sontag, and Yildirim, 2015). Instances generated from these models can also be seen as the result of random noise injected into an instance with a nice block structure that is easy to solve. Several works give exact recovery and approximate recovery guarantees for semidefinite programming (SDP) relaxations for such models in different parameter regimes (Abbe, 2018; Guédon and Vershynin, 2016). In our model however, we start with an *arbitrary* stable instance as opposed to an instance with a block structure.

Partial optimality algorithms. Several works have developed fast algorithms for identifying parts of the MAP assignment. These algorithms output an approximate solution $\hat{x}$ and a set of vertices where $\hat{x}$ provably agrees with the MAP solution x^* (e.g., Kovtun, 2003; Shekhovtsov, 2013; Swoboda, Shekhovtsov, Kappes, Schnörr, and Savchynskyy, 2016; Shekhovtsov, Swoboda, and Savchynskyy, 2017). Like these works, our results also prove that an approximate solution $\hat{x}$ has small error $|\hat{x} - x^*|$. However, these previous approaches are more concerned with designing fast algorithms for finding such $\hat{x}$. In contrast, we focus on giving structural conditions that explain why a *particular* $\hat{x}$ (the solution to the local LP relaxation) often approximately recovers x^*. Our algorithm in Section 2.7 is thus not meant as an efficient method for certifying that $|\hat{x} - x^*|$ is small, but rather as a method for checking whether our structural condition (that the observed instance is close to a stable instance) is satisfied in practice.

2.3. Preliminaries

In this section we introduce our notation, define the local LP relaxation for MAP inference, and give more details on perturbation stability. As in the previous section, the MAP inference problem in the ferromagnetic Potts model on the instance (G, c, w) can be written in *energy minimization* form as:

$$\underset{x:V\to[k]}{\text{minimize}} \sum_{u\in V} c(u, x(u)) + \sum_{(u,v)\in E} w(u,v)\mathbb{1}[x(u) \neq x(v)]. \tag{2.1}$$

Here x is an *assignment* (or labeling) of vertices to labels i.e., $x : V \to \{1, 2, \ldots, k\}$. We can identify each labeling x with a point $(x_u : u \in V; x_{uv} : (u,v) \in E)$, where each $x_u \in \{0,1\}^k$ and each $x_{uv} \in \{0,1\}^{k\times k}$.

In this chapter, we consider all node costs $c(u,i) \in \mathbb{R}$ and all edge weights $w(u,v) > 0$. We note that this is equivalent to the formulation where all node costs and edge weights are non-negative (Kleinberg & Tardos, 2002). See Appendix A.1 for a proof of this equivalence.

We encode the node costs and the edge weights in a vector $\theta \in \mathbb{R}^{nk+mk^2}$ where $n = |V|$ and $m = |E|$ s.t. $\theta(u,i) = c(u,i), \theta(u,v,i,j) = w(u,v)\mathbb{1}[i \neq j]$. Then the objective can be written as $\langle \theta, x \rangle$. We set $x_u(i) = 1$ when $x(u) = i$, and 0 otherwise. Similarly, we set $x_{uv}(i,j) = 1$ when $x(u) = i$ and $x(v) = j$, and 0 otherwise. Where convenient, we use x to refer to this point rather than the labeling $x : V \to [k]$. We can then rewrite equation 2.1 as:

$$\begin{aligned}
\min_{x}.\ & \sum_{u \in V} \sum_{i=1}^{k} c(u,i) x_u(i) + \sum_{(u,v) \in E} w(u,v) \sum_{i \neq j} x_{uv}(i,j) \\
\text{subject to: } & \sum_{i=1}^{k} x_u(i) = 1 && \forall\ u \in V \\
& \sum_{i=1}^{k} x_{uv}(i,j) = x_v(j) && \forall\ (u,v) \in E,\ j \in [k] \\
& \sum_{j=1}^{k} x_{uv}(i,j) = x_u(i) && \forall\ (u,v) \in E,\ i \in [k] \\
& x_{uv}(i,j) \in \{0,1\} && \forall\ (u,v),\ (i,j) \\
& x_u(i) \in \{0,1\} && \forall\ u,\ i.
\end{aligned}$$

This is equivalent to equation 2.1, and is an integer linear program (ILP). By relaxing the integrality constraints from $\{0,1\}$ to $[0,1]$, we obtain the *local LP relaxation*:

$$\min_{x \in L(G)}.\ \sum_{u \in V} \sum_{i=1}^{k} c(u,i) x_u(i) + \sum_{(u,v) \in E} w(u,v) \sum_{i \neq j} x_{uv}(i,j),$$

where $L(G)$ is the polytope defined by the same constraints as above, with $x \in \{0,1\}$ replaced with $x \in [0,1]$. This is known as the *local polytope* (Wainwright & Jordan, 2008). The vertices of $L(G)$ are either *integral*, meaning all x_u and x_{uv} take values in $\{0,1\}$, or *fractional*, when some variables take values in $(0,1)$. Integral vertices of this polytope correspond to labelings $x : V \to [k]$, so if the LP solution is obtained at an integral vertex, then it is also a MAP assignment.

If the solution x^* of this relaxation on an instance (G, c, w) is obtained at an integral vertex, we say the LP is *tight* on the instance, because the LP has exactly recovered a MAP assignment. If the LP is not tight, there may still be some vertices u where x^*_u takes integral values. In this case, if $x^*_u(i) = 1$ and $\bar{x}(u) = i$, i.e., the LP solution agrees with the MAP assignment $\bar{x}$ at vertex u, the LP is said to be *persistent* at u. $x^*_u(i) \in \{0, 1\}$ does not imply the LP is persistent at u, in general. The LP solution x^* is said to be persistent if it agrees with $\bar{x}$ at every vertex $u \in V$.

Recovery error: In practice, the local LP relaxation is often not tight, but is *nearly persistent*. We will measure the recovery error of our LP solution in terms of the "Hamming error" between the LP solution and the MAP assignment.

Definition 2.3.1 (Recovery error)**.** Given an instance (G, c, w) of equation 2.1, let $\bar{x}$ be a MAP assignment, and let x^* be a solution to the local LP relaxation. The recovery error is given by (with some abuse of notation)

$$\begin{aligned}\frac{1}{2}\|x^* - \bar{x}\|_1 &:= \frac{1}{2}\|x^*_V - \bar{x}_V\|_1 \\ &= \frac{1}{2}\sum_{u \in V}\sum_{i \in [k]}\Big|x^*_u(i) - \mathbb{1}[\bar{x}(u) = i]\Big|.\end{aligned}$$

$x_V \in \mathbb{R}^{nk}$ denotes the portion of x restricted to the vertex set V. If x^* is integral, the recovery error measures the number of vertices where x^* disagrees with $\bar{x}$. When the recovery error of x^* is 0, the solution x^* is *persistent*. We will say that the LP solution x^* is *nearly persistent* when the recovery error of solution x^* is a small fraction of n.

In our analysis, we will consider the following subset $L^*(G)$ of $L(G)$ which is easier to work with, and which contains all points we are interested in.

Definition 2.3.2 ($L^*(G)$)**.** We define $L^*(G) \subseteq L(G)$ to be the set of points $x \in L(G)$ which further satisfy the constraint that $x_{uv}(i,i) = \min(x_u(i), x_v(i))$ for all $(u,v) \in E$ and $i \in [k]$.

Claim 2.3.3. *For a given graph G, every solution $x \in L(G)$ that minimizes $\langle \theta, x \rangle$ for some valid objective vector $\theta = (c,w)$ also belongs to $L^*(G)$. Further, all integer solutions in $L(G)$ also belong to $L^*(G)$.*

We prove this claim in Appendix A.1.

Our new stability result relies on the set of *expansions* of a labeling x.

Definition 2.3.4 (Expansion)**.** Let $x : V \to [k]$ be a labeling of V. For any label $\alpha \in [k]$, we say that x' is an α-expansion of x if $x' \neq x$ and the following hold for all $u \in V$:

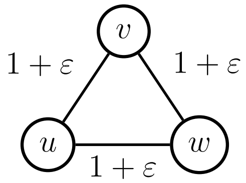
$$x(u) = \alpha \implies x'(u) = \alpha,$$

$$x'(u) \neq \alpha \implies x'(u) = x(u).$$

That is, x' may only expand the set of points labeled α, and cannot make other changes to x.

2.4. Expansion Stability

In this section, we generalize the stability result of Lang, Sontag, and Vijayaraghavan (2018) to a much broader class of instances. This generalization allows us to efficiently check whether a real-world instance could plausibly have the structure shown in Figure 2.1 (that is, whether the instance is close to a suitably stable instance).



Node	Costs		
u	.5	∞	∞
v	1	0	∞
w	1	∞	0

Figure 2.2. $(2,1)$-expansion stable instance that is not $(2,1)$-stable. In the original instance (shown left), the optimal solution labels each vertex with label 1, for an objective of 2.5. This instance is not $(2,1)$-stable: consider the $(2,1)$-perturbation that multiples all edge weights by $1/2$. In this perturbed instance, the original solution still has objective 2.5, and the new optimal solution labels $(u,v,w) \to (1,2,3)$. This has a node cost of 0.5 and an edge cost of $(3+3\varepsilon)/2$, for a total of $2+3\varepsilon/2 < 2.5$. Since the original solution is not optimal in the perturbed instance, this instance is not $(2,1)$-perturbation stable. However, note that the only expansions of the original solution (which had all label 1) that have non-infinite objective are $(u,v,w) \to (1,2,1)$ and $(u,v,w) \to (1,1,3)$. These each have objective $2.5+\varepsilon$, which is strictly greater than the perturbed objective of the original solution. In fact, checking this single perturbation, known as the *adversarial perturbation* is enough to verify expansion stability: this instance is $(2,1)$-expansion stable. We include the full details in Appendix A.2.

Consider a fixed instance (G,c,w) with a unique MAP solution $\bar{x}$. Theorem 1 of Lang, Sontag, and Vijayaraghavan (2018) requires that for all $\theta' \in \{(c,w') \mid w' \in \{(2,1)\text{-perturbations of } w\}\}$, $\langle\theta',x\rangle > \langle\theta',\bar{x}\rangle$ for *all* labelings $x \neq \bar{x}$. That is, that result requires $\bar{x}$ to be the unique optimal solution in any $(2,1)$-perturbation of the instance. By contrast, our result only requires $\bar{x}$ to have better perturbed objective than the set of *expansions* of $\bar{x}$ (c.f. Definition 2.3.4).

Definition 2.4.1 ((2,1)-expansion stability)**.** Let $\bar{x}$ be the unique MAP solution for (G,c,w), and let $\mathcal{E}_{\bar{x}}$ be the set of expansions of $\bar{x}$ (see Definition 2.3.4). Let

$$\Theta = \{(c,w') \mid w' \in \{(2,1)\text{-perturbations of } w\}\}$$

be the set of all objective vectors within a $(2,1)$-perturbation of $\theta = (c, w)$. We say the instance (G, c, w) is $(2,1)$-expansion stable if the following holds for all $\theta' \in \Theta$ and all $x \in \mathcal{E}_{\bar{x}}$:

$$\langle \theta', x \rangle > \langle \theta', \bar{x} \rangle.$$

That is, $\bar{x}$ is better than all of its expansions $x \neq \bar{x}$ in every $(2,1)$-perturbation of the instance.

Theorem 2.4.2 (Local LP is tight on $(2,1)$-expansion stable instances)**.** *Let $\bar{x}$ and $\hat{x}$ be the MAP and local LP solutions to a $(2,1)$-expansion stable instance (G, c, w), respectively. Then $\bar{x} = \hat{x}$ i.e., the local LP is tight on (G, c, w).*

We defer the proof of this theorem to Appendix A.2 as it is similar to the proof of Theorem 1 from Lang, Sontag, and Vijayaraghavan (2018). The $(2,1)$-expansion stability assumption is much weaker than $(2,1)$-stability because the former only compares $\bar{x}$ to its expansions, whereas the latter compares $\bar{x}$ to *all* labelings. While the rest of our results can also be adapted to the $(2,1)$-stability definition, this relaxed assumption gives better empirical results. Figure 2.2 shows an example of a $(2,1)$-expansion stable instance that is not $(2,1)$-stable. This shows that our new stability condition is less restrictive.

2.5. Curvature around MAP solution and near persistence of the LP solution

In this section, we show that a condition related to $(2,1)$-expansion stability, called $(2,1,\psi)$-expansion stability, implies a "curvature" result for the objective function around the MAP solution $\bar{x}$. On instances satisfying this condition, any point $\hat{x} \in L(G)$ with objective close to $\bar{x}$ also has small $\|\hat{x} - \bar{x}\|_1$, so $\hat{x}$ and $\bar{x}$ are close in solution space. In other words, if the LP solution $\hat{x}$ to a "corrupted" $(2,1,\psi)$-expansion stable instance is near-optimal in the original $(2,1,\psi)$-expansion stable instance (whose solution is $\bar{x}$), then the result in this section implies $\|\hat{x}-\bar{x}\|_1$ is small. This immediately gives a version of the result in the right panel of Figure 2.1: suppose we define an instance to be *close* to a $(2,1,\psi)$-expansion stable $(G,\bar{c},\bar{w})$ if its LP solution $\hat{x}$ is approximately optimal in $(G,\bar{c},\bar{w})$. Then the curvature result implies that the LP approximately recovers the stable instance's MAP solution $\bar{x}$ for all close instances. In Section 2.6, we give a generative model where the generated instances are "close" according to this definition with high probability.

The $(2,1,\psi)$-expansion stability condition, for $\psi > 0$, says that the instance is $(2,1)$-expansion stable even if we allow all node costs $c(u,i)$ to be additively perturbed by up to ψ. This extra additive stability will allow us to prove the curvature result. This is related to the use of additive stability in Lang, Sontag, and Vijayaraghavan (2019) to give persistency guarantees.

Definition 2.5.1 ($(2,1,\psi)$-expansion stable)**.** For $\psi > 0$, we say an instance (G,c,w) is $(2,1,\psi)$-expansion stable if (G,c',w) is $(2,1)$-expansion stable for all c' with $c \le c' \le c+\psi\cdot\mathbf{1}$ where $\mathbf{1}$ is the all-ones vector.

The following theorem shows low recovery error i.e., near persistence of the LP solution on $(2,1,\psi)$ expansion stable instances in terms of the gap in objective value.

Theorem 2.5.2. *Let (G,c,w) be a $(2,1,\psi)$-expansion stable instance with MAP solution $\bar{x}$. Let $\theta = (c,w)$. Then for any $x \in L^*(G)$, the recovery error (see Def. 2.3.1)* *satisfies*

$$\frac{1}{2}\|x-\bar{x}\|_1 := \frac{1}{2}\|x_V-\bar{x}_V\|_1 \leq \frac{1}{\psi}|\langle\theta,x\rangle-\langle\theta,\bar{x}\rangle|. \tag{2.2}$$

Proof (sketch). For any $x \in L^*(G)$, we construct a feasible solution $\hat{x}$ which is a strict convex combination of x and $\bar{x}$ that is very close to $\bar{x}$. Then, we apply a rounding algorithm to $\hat{x}$ to get a random integer solution h. Let $\hat{\theta}$ represent the worst $(2,1)$-perturbation for $\bar{x}$. This is the instance where all the edges not cut by $\bar{x}$ have their weights multiplied by $1/2$. We define the objective difference using $\hat{\theta}$ as $A_h = \langle\hat{\theta},h\rangle - \langle\hat{\theta},\bar{x}\rangle$. First we show an upper bound for $\mathbb{E}[A_h]$ using properties of the rounding algorithm. Then we show that for any solution h in the support of our rounding algorithm, $A_h \geq \psi \cdot B_h$ where B_h is the Hamming error of h (when compared to $\bar{x}$). On the other hand, one can also use the properties of the rounding algorithm to get a lower bound on $\mathbb{E}[B_h]$ in terms of the recovery error (i.e., Hamming error) of the LP solution. These bounds together imply the required upper bound on the recovery error of the LP solution. ■

We defer the complete proof and an alternate dual-based proof to Appendix A.3.

Theorem 2.5.2 shows that on a $(2,1,\psi)$-expansion stable instance, small objective gap $\langle\theta,x\rangle-\langle\theta,\bar{x}\rangle$ implies small distance $||x_V-\bar{x}_V||_1$ in solution space. Although this holds for any $x \in L^*(G)$, we will be interested in x that are LP solutions to an observed, corrupted version of the stable instance.

We now show that if the observed instance has a nearby stable instance, then the LP solution for the observed instance has small Hamming error. For any two instances $\hat{\theta} = (\hat{c},\hat{w})$ and $\bar{\theta} = (\bar{c},\bar{w})$ on the same graph G, the metric between them $d(\hat{\theta},\bar{\theta}) := \sup_{x\in L^*(G)} |\langle\hat{\theta},x\rangle - \langle\bar{\theta},x\rangle|$.

Corollary 2.5.3 (LP solution is good if there is a nearby stable instance)**.** *Let* $\hat{x}^{MAP}$ *and* $\hat{x}$ *be the MAP and local LP solutions to an observed instance* $(G, \hat{c}, \hat{w})$*. Also, let* $\bar{x}$ *be the MAP solution for a latent* $(2, 1, \psi)$*-expansion stable instance* $(G, \bar{c}, \bar{w})$*. If* $\hat{\theta} = (\hat{c}, \hat{w})$ *and* $\bar{\theta} = (\bar{c}, \bar{w})$*,*

$$\frac{1}{2}\|\hat{x}_V - \hat{x}_V^{MAP}\|_1 \leq \frac{2d(\hat{\theta}, \bar{\theta})}{\psi} + \frac{1}{2}\|\hat{x}_V^{MAP} - \bar{x}_V\|_1.$$

We defer the proof of this corollary to Appendix A.3.

2.6. Generative model for noisy stable instances

In the previous section, we showed that if an instance $(G, \hat{c}, \hat{w})$ is "close" to a $(2, 1, \psi)$-expansion stable instance $(G, \bar{c}, \bar{w})$ (i.e., the LP solution $\hat{x}$ to $(G, \hat{c}, \hat{w})$ has good objective in $(G, \bar{c}, \bar{w})$), then $\|\hat{x} - \bar{x}\|$ is small, where $\bar{x}$ is the MAP solution to the stable instance. In this section, we give a natural generative model for $(G, \hat{c}, \hat{w})$, based on randomly corrupting $(G, \bar{c}, \bar{w})$, in which $\hat{x}$ has good objective in $(G, \bar{c}, \bar{w})$ with high probability. Together with the curvature result from the previous section (Theorem 2.5.2), this implies that the LP relaxation, run on the noisy instances $(G, \hat{c}, \hat{w})$, approximately recovers $\bar{x}$ with high probability.

We now describe our generative model for the problem instances, which starts with an arbitrary stable instance and perturbs it with random additive perturbations to the edge costs and node costs (of potentially varying magnitudes). The random perturbations reflect possible uncertainty in the edge costs and node costs of the Markov random field. We will assume the random noise comes from any distribution that is sub-Gaussian*. However, there is a small technicality: the edge costs need to be positive (node costs can

*All of the results that follow can also be generalized to sub-exponential random variables; however for convenience, we restrict our attention to sub-Gaussians.

be negative). For this reason we will consider truncated sub-Gaussian random variables for the noise for the edge weights. We define sub-Gaussian and truncated sub-Gaussian random variables in Appendix A.4.

Generative Model: We start with an instance $(G, \bar{c}, \bar{w})$ that is $(2, 1, \psi)$-expansion stable, and perturb the edge costs and node costs independently. Given any instance $(G, \bar{c}, \bar{w})$, an instance $(G, \hat{c}, \hat{w})$ from the model is generated as follows:

(1) For all node-label pairs (u, i), $\hat{c}(u, i) = \bar{c}(u, i) + \tilde{c}(u, i)$, where $\tilde{c}(u, i)$ is sub-Gaussian with mean 0 and parameter $\sigma_{u,i}$.

(2) For all edges (u, v), $\hat{w}(u, v) = \bar{w}(u, v) + \tilde{w}(u, v)$, where $\tilde{w}(u, v)$ is an independent r.v. that is $(-w(u, v), \gamma_{u,v})$-truncated sub-Gaussian with mean 0.

(3) $(G, \hat{c}, \hat{w})$ is the observed instance.

By the definition of our model, the edge weights $\hat{w}(u, v) \geq 0$ for all $(u, v) \in E$. The parameters of the model are the unperturbed instance $(G, \bar{c}, \bar{w})$, and the noise parameters $\{\, \gamma_{u,v}, \sigma_{u,i} \,\}_{u,v \in V, i \in [k]}$. On the one hand, the above model captures a natural average-case model for the problem. For a fixed ground-truth solution $x^* : V \to [k]$, consider the stable instance (H, c, w) where $w^*_{uv} = 2$ for all u, v in the same cluster (i.e., $x^*(u) = x^*(v)$) and $w^*_{uv} = 1$ otherwise; and with $c^*(u, i) = 1$ if $x^*(u) = i$, and $c^*(u, i) = 1 + \psi$ otherwise. The above noisy stable model with stable instance (H, c, w) generates instances that are reminiscent of (stochastic) block models, with additional node costs. On the other hand, the above model is much more general, since we can start with *any* stable instance (G, c, w).

With high probability over the random corruptions of our stable instance, the local LP on the corrupted instance approximately recovers the MAP solution $\bar{x}$ of the stable instance. The key step in the proof of this theorem is showing that, with high probability, the observed instance is close to the latent stable instance in the metric we defined earlier.

Lemma 2.6.1 ($d(\hat{\theta}, \bar{\theta})$ is small w.h.p.)**.** *There exists a universal constant* $c < 1$ *such that for any instance in the above model, with probability at least* $1 - o(1)$*,*

$$\sup_{x \in L^*(G)} |\langle \hat{\theta}, x \rangle - \langle \bar{\theta}, x \rangle| \leq c\sqrt{nk}\sqrt{\sum_{u,i} \sigma_{u,i}^2 + \frac{k^2}{4}\sum_{uv} \gamma_{u,v}^2}$$

Proof (sketch). For any *fixed* $x \in L^*(G)$, we can show that $|\langle \hat{\theta}, x \rangle - \langle \bar{\theta}, x \rangle|$ is small w.h.p. using a standard large deviations bound for sums of sub-Gaussian random variables. The main technical challenge is in showing that the supremum over all feasible solutions is small w.h.p. The standard approach is to perform a union bound over an ε-net of feasible LP solutions in L^*. However, this gives a loose bound. Instead, we upper bound the supremum by using a rounding algorithm for LP solutions in $L^*(G)$, and union bound only over the discrete solutions output by the rounding algorithm. This gives significant improvements over the standard approach; for example, in a d-regular graph with equal variance parameter γ_{uv}, this saves a factor of $\sqrt{d}$ apart from logarithmic factors in n. ■

We defer the details to Appendix A.4. The above proof technique that uses a rounding algorithm to provide a deviation bound for a continuous relaxation is similar to the analysis of SDP relaxations for average-case problems (see e.g., Makarychev, Makarychev, and Vijayaraghavan, 2013; Guédon and Vershynin, 2016). The above lemma, when combined with Theorem 2.5.2 gives the following guarantee.

Theorem 2.6.2 (LP solution is nearly persistent)**.** *Let* $\hat{x}$ *be the local LP solution to the observed instance* $(G, \hat{c}, \hat{w})$ *and* $\bar{x}$ *be the MAP solution to the latent* $(2, 1, \psi)$*-expansion* *stable instance $(G, \bar{c}, \bar{w})$. With high probability over the random noise,*

$$\frac{1}{2}\|\hat{x}_V - \bar{x}_V\|_1 \leq \frac{2}{\psi} \cdot c\sqrt{nk} \cdot \sqrt{\sum_{u,i} \sigma_{u,i}^2 + k^2 \sum_{uv} \gamma_{u,v}^2}$$

Proof. We know that for any feasible solution $x \in L(G)$, $\langle \bar{\theta}, x\rangle \geq \langle \bar{\theta}, \bar{x}\rangle$. Therefore, $\langle \bar{\theta}, \hat{x}\rangle \geq \langle \bar{\theta}, \bar{x}\rangle$. Remember that we defined $d(\hat{\theta}, \bar{\theta})$ as $\sup_{x \in L^*(G)} |\langle \hat{\theta}, x\rangle - \langle \bar{\theta}, x\rangle|$. Since $\hat{x}$ and $\bar{x}$ are both points in $L^*(G)$,

$$\begin{aligned}\langle \bar{\theta}, \hat{x}\rangle \leq \langle \hat{\theta}, \hat{x}\rangle + d(\hat{\theta}, \bar{\theta}) &\leq \langle \hat{\theta}, \bar{x}\rangle + d(\hat{\theta}, \bar{\theta}) \\ &\leq \langle \bar{\theta}, \bar{x}\rangle + 2d(\hat{\theta}, \bar{\theta})\end{aligned}$$

The first and third inequalities follow from the definition of $d(\hat{\theta}, \bar{\theta})$. The second inequality follows from the fact that $\hat{x}$ is the minimizer of $\langle \bar{\theta}, x\rangle$ over all $x \in L(G)$. Therefore,$0 \leq \langle \bar{\theta}, \hat{x}\rangle - \langle \bar{\theta}, \bar{x}\rangle \leq 2d(\hat{\theta}, \bar{\theta})$. Using this in Theorem 2.5.2, we get $\frac{1}{2}\|\hat{x} - \bar{x}\|_1 \leq \frac{2d(\hat{\theta}, \bar{\theta})}{\psi}$. Lemma 2.6.1 then gives an upper bound on $d(\hat{\theta}, \bar{\theta})$ that holds w.h.p. ■

For a d-regular graph in the uniform setting, we get the following useful corollary:

Corollary 2.6.3 (MAP solution recovery for regular graphs). *Suppose we have a d-regular graph G with $\gamma_{u,v}^2 = \gamma^2$ for all edges (u, v), and $\sigma_{u,i}^2 = \sigma^2$ for all vertex-label pairs (u, i). Also, suppose only a fraction ρ of the vertices and η of the edges are subject to the noise. With high probability over the random noise,*

$$\frac{\|\hat{x}_V - \bar{x_V}\|_1}{2n} \leq \frac{2ck\sqrt{\rho\sigma^2 + \frac{\eta dk}{8}\gamma^2}}{\psi}$$

Note that when $\hat{x}$ is an integer solution, the left-hand-side is the fraction of vertices misclassified by $\hat{x}$.

2.7. Finding nearby stable instances

In this section, we describe an algorithm for efficiently finding $(2, 1, \psi)$-expansion stable instances that are "close" to an observed instance $(G, \hat{c}, \hat{w})$. This algorithm allows us to check whether we can plausibly model real-world instances as "corrupted" versions of a $(2, 1, \psi)$-expansion stable instance.

In addition to an observed instance $(G, \hat{c}, \hat{w})$, the algorithm takes as input a "target" labeling x^t. For example, x^t could be a MAP solution of the observed instance. Surprisingly, once given a target solution, this algorithm is efficient.

We want to search over costs c and weights w. The broad goal to solve the following optimization problem:

$$\begin{aligned} \underset{c \geq 0, w \geq 0}{\text{minimize}} \quad & f(c, w) \\ \text{subject to} \quad & (G, c, w) \text{ is } (2, 1, \psi)\text{-expansion stable} \\ & \text{with MAP solution } x^t, \end{aligned} \tag{2.3}$$

where $f(c, w)$ is any convex function of c and w. In particular, we will use $f_1(c, w) = ||(c, w) - (\hat{c}, \hat{w})||_1$ and $f_2(c, w) = \frac{1}{2}||(c, w) - (\hat{c}, \hat{w})||_2^2$ for minimizing the L1 and L2 distances between to the observed instance. The output of this optimization problem will give the closest objective vector $(\bar{c}, \bar{w})$ for which the instance $(G, \bar{c}, \bar{w})$ is $(2, 1, \psi)$-expansion stable. If the optimal objective value of this optimization is 0, the observed instance $(G, \hat{c}, \hat{w})$ is $(2, 1, \psi)$-expansion stable.

There is always a feasible (c, w) for equation 2.3 (see Appendix A.5 for a proof), but it may change many weights and costs. Next we derive an efficiently-solvable reformulation of equation 2.3. In the next section, we find that the changes to $\hat{c}$ and $\hat{w}$ required to find a $(2, 1, \psi)$-expansion stable instance are relatively sparse in practice.

Theorem 2.7.1. *The optimization problem equation 2.3 can be expressed as a convex minimization problem over a polytope described by poly(n, m, k) constraints. When $f(c, w) = ||(c, w) - (\hat{c}, \hat{w})||_1$, equation 2.3 can be expressed as a linear program.*

The instance (G, c, w) is $(2, 1, \psi)$-expansion stable if x^t is better than every expansion y of x^t in every $(2, 1, \psi)$-perturbation of (c, w). Let $\mathcal{E}$ be the set of all expansions of the target solution x^t. Then for all θ' within a $(2, 1, \psi)$-perturbation of $\theta = (c, w)$, we should have that $\langle \theta', x^t \rangle \leq \min_{y \in \mathcal{E}} \langle \theta', x^t \rangle$. It is enough to check the *adversarial* $(2, 1, \psi)$-perturbation θ_{adv}. This perturbation makes the target solution x^t as bad as possible. If x^t is better than all the expansions $y \in \mathcal{E}$ in this perturbation, it is better than all $y \in \mathcal{E}$ in every $(2, 1, \psi)$-perturbation (see Appendix A.5 for a proof). We set $\theta_{adv} = (c_{adv}, w_{adv})$ as:

$$c_{adv}(u, i) = \begin{cases} c(u, i) + \psi & i = x^t(u), \\ c(u, i) & \text{otherwise.} \end{cases}$$

$$w_{adv}(u, v) = \begin{cases} w(u, v) & x^t(u) \neq x^t(u), \\ \frac{1}{2} w(u, v) & \text{otherwise.} \end{cases}$$

The target solution x^t is fixed, so $\langle \theta_{adv}, x^t \rangle$ is a linear function of the optimization variables c and w. For $\alpha \in [k]$, let $\mathcal{E}^\alpha$ be the set of α-expansions of x^t. Because $\mathcal{E} = \cup_{\alpha \in [k]} \mathcal{E}^\alpha$, we have $\langle \theta', x^t \rangle \leq \min_{y \in \mathcal{E}} \langle \theta', x^t \rangle$ if and only if $\langle \theta', x^t \rangle \leq \min_{y \in \mathcal{E}^\alpha} \langle \theta', x^t \rangle$ for all $\alpha \in [k]$. We can simplify the original optimization problem to:

$$\begin{aligned} \underset{c \geq 0, w \geq 0}{\text{minimize}} \quad & f(c, w) \\ \text{subject to} \quad & \langle \theta_{adv}, x^t \rangle \leq \min_{y \in \mathcal{E}^\alpha} \langle \theta_{adv}, y \rangle \qquad \forall\ \alpha \in [k], \end{aligned}$$

θ_{adv} is a linear function of c, w as defined above. We now use the structure of the sets $\mathcal{E}^\alpha$ to simplify the constraints. The optimal value of $\min_{y \in \mathcal{E}^\alpha} \langle \theta_{adv}, y \rangle$ is the objective value of the best (w.r.t. θ_{adv}) α-expansion of x^t. The best α-expansion of a fixed solution x^t can be found by solving a minimum cut problem in an auxiliary graph $G^{x^t}_{aux}(\alpha)$ whose weights depend on linearly on the objective θ_{adv}, and therefore depend linearly on our optimization variables (c, w) (Boykov, Veksler, and Zabih, 2001, Section 4). Therefore, the optimization problem $\min_{y \in \mathcal{E}^\alpha} \langle \theta_{adv}, y \rangle$ can be expressed as a minimum cut problem. Because this minimum cut problem can be written as a linear program, we can rewrite each constraint as

$$\langle \theta_{adv}, x^t \rangle \leq \min_{z : A(\alpha) z = b(\alpha), z \geq 0} \langle \theta_{adv}, z \rangle, \tag{2.4}$$

where $\{A(\alpha) z = b(\alpha),\ z \geq 0\}$ is the feasible region of the standard metric LP corresponding to the minimum cut problem in $G^{x^t}_{aux}(\alpha)$. The number of vertices in $G^{x^t}_{aux}(\alpha)$ and the number of constraints in $A(\alpha) z = b(\alpha)$ is $\text{poly}(m, n, k)$ for all α. We now derive an equivalent linear formulation of equation 2.4 using a careful application of strong duality.

The dual to the LP on the RHS is:

$$\underset{\nu}{\text{maximize}} \;\; \langle b(\alpha), \nu \rangle, \;\text{s.t.}\; A(\alpha)^T \nu \leq \theta_{adv}.$$

Because strong duality holds for this linear program, we have that equation 2.4 holds if and only if there exists ν with $A(\alpha)^T \nu \leq \theta_{adv}$ such that $\langle \theta_{adv}, x^t \rangle \leq \langle b(\alpha), \nu \rangle$.

This is a linear constraint in (c, w, ν). By using this dual witness trick for each label $\alpha \in [k]$, we obtain:

$$\begin{aligned} &\underset{c\geq 0, w\geq 0, \{\nu_\alpha\}}{\text{minimize}} && f(c, w) && \\ &\text{subject to} && \langle \theta_{adv}, x^t \rangle \leq \langle b(\alpha), \nu_\alpha \rangle && \forall\ \alpha \\ & && A(\alpha)^T \nu_\alpha \leq \theta_{adv} && \forall\ \alpha. \end{aligned} \tag{2.5}$$

The constraints of equation 2.5 are linear in the optimization variables (c, w) and ν_α. The dimension of θ_{adv} is $nk + mk^2$, so there are $k(nk + mk^2 + 1)$ constraints and $nk + m + \sum_\alpha |b(\alpha)| = poly(m, n, k)$ variables. Because minimization of the L1 distance $f_1(c, w)$ can be encoded using a linear function and linear constraints, equation 2.5 is a linear program in this case. It is clear from the derivation of equation 2.5 that it is equivalent to equation 2.3. This proves Theorem 2.7.1. This formulation equation 2.5 can easily be input into “off-the-shelf” convex programming packages such as Gurobi (Gurobi Optimization, 2020).

Table 2.1. Results from the output of equation 2.5 on three stereo vision instances. The "Error bound" column has the normalized recovery error bound. More details in Appendix A.6.

Instance	Costs changed	Weights changed	Error bound	$\|\|\hat{x}_V - \hat{x}_V^{MAP}\|\|_1/2n$
`tsukuba`	4.9%	2.3%	0.0518	0.0027
`venus`	22.5%	1.3%	0.0214	0.0016
`cones`	1.2%	2.1%	0.0437	0.0022

2.8. Numerical results

Table 2.1 shows the results of running equation 2.5 on real-world instances of MAP inference to find nearby $(2, 1, \psi)$-expansion stable instances. We study *stereo vision* models using images from the Middlebury stereo dataset (Scharstein & Szeliski, 2002) and Potts models from Tappen & Freeman (2003). Please see Appendix A.6 for more details about the models and experiments.

We find, surprisingly, that only relatively sparse changes are required to make the observed instances $(2, 1, \psi)$-expansion stable with $\psi = 1$. On these instances, we evaluate the recovery guarantees by our bound from Theorem 2.5.2 and compare it to the actual value of the recovery error $||\hat{x} - \hat{x}^{MAP}||_1/2n$. In all of our experiments, we choose the *target* solution x^t for equation 2.5 to be equal to the MAP solution $\hat{x}^{MAP}$ of the observed instance. Therefore, we find a $(2, 1, \psi)$-expansion stable instance that has the same MAP solution as our observed instance. The recovery error bound given by Theorem 2.5.2 is then also a bound for the recovery error between $\hat{x}$ and $\hat{x}^{MAP}$, because $\hat{x}^{MAP} = x^t$. On these instances, the bounds from our curvature result (Theorem 2.5.2) are reasonably close to the actual recovery value. However, this bound uses the property that $\hat{x}$ has good objective in the stable instance and so it is still a “data-dependent” bound in the sense that it uses an empirically observed property of the LP solution $\hat{x}$. In Appendix A.6, we show how to refine Corollary 2.5.3 to give non-vacuous recovery bounds that do not depend on $\hat{x}$.

2.9. Conclusion

We studied the phenomenon of near persistence of the local LP relaxation on instances of MAP inference in ferromagnetic Potts model. We gave theoretical results, algorithms (for finding nearby stable instances) and empirical results to demonstrate that even after a $(2, 1, \psi)$-perturbation stable instance is corrupted with noise, the solution to the LP relaxation is nearly persistent i.e., it approximately recovers the MAP solution. Our theoretical results imply that the local LP approximately recovers the MAP solution when the observed instance is close to a stable instance. Our empirical results suggest that real-world instances are very close to stable instances. These results together suggest a new explanation for the near-persistency of the solution of the local LP relaxation for MAP inference in practice.

CHAPTER 3

One-Pass algorithms for MAP Inference of Nonsymmetric Determinantal Point Processes

Note: This chapter is (mostly) a reprint of Reddy, Rossi, Song, Rao, Mai, Lipka, Wu, Koh, and Ahmed (2022a).

3.1. Introduction

Determinantal Point Processes (DPPs) were first introduced in the context of quantum mechanics (Macchi, 1975) and have subsequently been extensively studied with applications in several areas of pure and applied mathematics like graph theory, combinatorics, random matrix theory (Hough, Krishnapur, Peres, and Virág, 2006; Borodin, 2009), and randomized numerical linear algebra (Derezinski & Mahoney, 2021). Discrete DPPs have gained widespread adoption in machine learning following the seminal work of (Kulesza & Taskar, 2012) and have also seen a recent explosion of interest in the machine learning community. For instance, some of the very recent uses of DPPs include automation of deep neural network design (Nguyen, Le, Yamada, and Osborne, 2021), deep generative models (Chen and Ahmed, 2021), document and video summarization (Perez-Beltrachini and Lapata, 2021), image processing (Launay, Desolneux, and Galerne, 2021), and learning in games (Perez-Nieves, Yang, Slumbers, Mguni, Wen, and Wang, 2021).

A DPP is a probability distribution over subsets of items and is characterized by some kernel matrix such that the probability of sampling any particular subset is proportional to the determinant of the submatrix corresponding to that subset in the kernel. Until very recently, most prior work on DPPs focused on the setting where the kernel matrix is symmetric. Due to this constraint, DPPs can only model negative correlations between items. Recent work has shown that allowing the kernel matrix to be nonsymmetric can greatly increase the expressive power of DPPs and allows them to model *compatible* sets of items (Gartrell, Brunel, Dohmatob, and Krichene, 2019; Brunel, 2018). To differentiate this line of work from prior literature on symmetric DPPs, the term Nonsymmetric DPPs (NDPPs) has often been used. Modeling positive correlations can be useful in many practical scenarios. For instance, an E-commerce company trying to build a product recommendation system would want the system to increase the probability of suggesting a router if a customer adds a modem to a shopping cart.

State-of-the-art algorithms for MAP inference of NDPPs (Gartrell, Han, Dohmatob, Gillenwater, and Brunel, 2021; Anari and Vuong, 2022) require storing the full data in memory and take multiple passes over the complete dataset. Therefore, these algorithms take too much memory to be useful for industry scale datasets, where the size of the entire dataset can be much larger than the random-access memory available.

Technical contributions.

- We give the first problem formulations for streaming and online versions of MAP Inference of low-rank-NDPPs. Our formulations provide a good structure on how to store NDPP models which are so large that they cannot fit by themselves in the memory (RAM) of any single machine (this is an extremely important practical problem due to the massive scale of industry datasets) i.e., store the $\boldsymbol{C}$ matrix separately and all the $\boldsymbol{v}_i$ and $\boldsymbol{b}_i$ as $(\boldsymbol{v}_i, \boldsymbol{b}_i)$ pairs (the straight-forward way to store the data would be to store $\boldsymbol{V}, \boldsymbol{C}, \boldsymbol{B}$ separately (as in the open-source code provided by (Gartrell, Han, Dohmatob, Gillenwater, and Brunel, 2021)).

- In Section 3.4, we provide our first streaming algorithm, Partition Greedy, which is a streaming version of the standard greedy algorithm for submodular maximization (Nemhauser, Wolsey, and Fisher, 1978), and provide bounds for the approximation ratio, space used, and time taken.
- In Section 3.5, we provide Online-LSS and Online-2-Neighbour, our online algorithms based on local search with a stash, which are generalizations of the online local search algorithm for MAP Inference of symmetric DPPs by Bhaskara, Karbasi, Lattanzi, and Zadimoghaddam (2020), and provide bounds for the space used and time taken.
- In Section 3.6, we provide a hard instance for one-pass sublinear-space MAP Inference of NDPPs on which all of our algorithms fail to output solutions with a bounded approximation ratio. This illustrates that it might even be impossible to prove approximation factor guarantees for our algorithms without additional strong assumptions. The hard instance uses properties of NDPPs that differ from symmetric DPPs illustrating some of the divergence between them. We also provide some additional comparison between MAP Inference of nonsymmetric DPPs and symmetric DPPs in this section.
- In Section 3.7, we evaluate our proposed online NDPP MAP Inference algorithms on several datasets and show that they show that they perform comparably to (or even better than) the offline greedy algorithm which takes multiple passes over the data and also uses substantially more memory (linear in number of items).

3.2. Related Work

Nonsymmetric DPPs. A special subset of NDPPs called signed DPPs were the first class of NDPPs that were studied by Brunel, Moitra, Rigollet, and Urschel (2017). Gartrell, Brunel, Dohmatob, and Krichene (2019) studied a more general class of NDPPs and provided learning and MAP Inference algorithms, and also showed that NDPPs have additional expressiveness over symmetric DPPs and can better model certain problems. This was improved by Gartrell, Han, Dohmatob, Gillenwater, and Brunel (2021) in which they provided a new decomposition which enabled linear time learning and MAP Inference for NDPPs. More recently, Anari and Vuong (2022) proposed the first algorithm with a $k^{O(k)}$ approximation factor for MAP Inference on NDPPs where k is the number of items to be selected.

DPP MAP Inference. MAP Inference in DPPs is a very well studied NP-hard problem (Ko, Lee, and Queyranne, 1995) with numerous applications in machine learning (Gillenwater, Kulesza, and Taskar, 2012; Han, Kambadur, Park, and Shin, 2017; Chen, Zhang, and Zhou, 2018; Han and Gillenwater, 2020). Since matrix log-determinant is a submodular function, several offline algorithms for submodular function maximization (Krause and Golovin, 2014) have been applied to the problem. For instance, Civril and Magdon-Ismail (2009) showed that the standard greedy algorithm of Nemhauser, Wolsey, and Fisher (1978) provides an $O(k!)$ factor approximation. Nevertheless, even in the case of symmetric DPPs, the study of streaming and online algorithms is in a nascent stage. In particular, Indyk, Mahabadi, Oveis Gharan, and Rezaei (2019; 2020); Mahabadi, Razenshteyn, Woodruff, and Zhou (2020) provided streaming algorithms for MAP inference of DPPs and Bhaskara, Karbasi, Lattanzi, and Zadimoghaddam (2020) were the first to propose online algorithms for MAP inference of DPPs. Also, Liu, Soni, Kang, Wang, and Parsana (2021) designed the first streaming algorithms for the maximum induced cardinality objective proposed by Gillenwater, Kulesza, Mariet, and Vassilvitskii (2018). However, there has not yet been any work other than ours which has focused on either streaming or online algorithms for NDPPs.

Streaming and Online Algorithms. Streaming (Alon, Matias, and Szegedy, 1999; Muthukrishnan, 2005) and online (Karp, Vazirani, and Vazirani, 1990; Karp, 1992; Borodin and El-Yaniv, 2005) algorithms have been extensively studied in theoretical computer science.

3.3. Preliminaries

3.3.1. Notation

Throughout the paper, we use uppercase bold letters ($\boldsymbol{A}$) to denote matrices and lowercase bold letters ($\boldsymbol{a}$) to denote vectors. Letters in normal font (a) will be used for scalars. For any positive integer n, we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. A matrix $\boldsymbol{M}$ is said to be skew-symmetric if $\boldsymbol{M} = -\boldsymbol{M}^\top$ where $\top$ is used to represent matrix transposition.

3.3.2. Background on DPPs

A DPP is a probability distribution on all subsets of $[n]$ characterized by a matrix $\boldsymbol{L} \in \mathbb{R}^{n \times n}$. The probability of sampling any subset $S \subseteq [n]$ i.e., $\mathbb{P}[S] \propto \det(\boldsymbol{L}_S)$ where $\boldsymbol{L}_S$ is the submatrix of $\boldsymbol{L}$ obtained by keeping only the rows and columns corresponding to indices in S. The normalization constant for this distribution can be computed efficiently since we know that $\sum_{S \subseteq [n]} \det(\boldsymbol{L}_S) = \det(\boldsymbol{L} + \boldsymbol{I}_n)$ (Kulesza & Taskar, 2012, Theorem 2.1).

Therefore, $\mathbb{P}[S] = \frac{\det(\boldsymbol{L}_S)}{\det(\boldsymbol{L}+\boldsymbol{I}_n)}$. For the DPP corresponding to $\boldsymbol{L}$ to be a valid probability distribution, we need $\det(\boldsymbol{L}_S) \geq 0$ for all $S \subseteq [n]$ since $\mathbb{P}[S] \geq 0$ for all $S \subseteq [n]$. Matrices which satisfy this property are known as P_0-matrices (Fiedler & Pták, 1966). For any symmetric matrix $\boldsymbol{L}$, $\det(\boldsymbol{L}_S) \geq 0$ for all $S \subseteq [n]$ if and only if $\boldsymbol{L}$ is positive semi-definite (PSD) i.e., $\boldsymbol{x}^T\boldsymbol{L}\boldsymbol{x} \geq 0$ for all $\boldsymbol{x} \in \mathbb{R}^n$. Therefore, all symmetric matrices which correspond to valid DPPs are PSD. But there are P_0-matrices which are not necessarily symmetric (or even positive semi-definite). For example, $\boldsymbol{L} = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$ is a nonsymmetric P_0 matrix.

Any matrix $\boldsymbol{L}$ can be uniquely written as the sum of a symmetric and skew-symmetric matrix: $\boldsymbol{L} = (\boldsymbol{L} + \boldsymbol{L}^\top)/2 + (\boldsymbol{L} - \boldsymbol{L}^\top)/2$. For the DPP characterized by $\boldsymbol{L}$, the symmetric part of the decomposition can be thought of as encoding negative correlations between items and the skew-symmetric part as encoding positive correlations. Gartrell, Brunel, Dohmatob, and Krichene (2019) proposed a decomposition which covers the set of all nonsymmetric PSD matrices (a subset of P_0 matrices) which allowed them to provide a cubic time algorithm (in the ground set size) for NDPP learning. This decomposition is $\boldsymbol{L} = \boldsymbol{V}^\top\boldsymbol{V} + (\boldsymbol{B}\boldsymbol{C}^\top - \boldsymbol{C}\boldsymbol{B}^\top)$. Gartrell, Han, Dohmatob, Gillenwater, and Brunel (2021) provided more efficient (linear time) algorithms for learning and MAP inference using a new decomposition $\boldsymbol{L} = \boldsymbol{V}^\top\boldsymbol{V} + \boldsymbol{B}^\top\boldsymbol{C}\boldsymbol{B}$. Although both these decompositions only cover a subset of P_0 matrices, it turns out that they are quite useful for modeling real world instances and provide improved results when compared to (symmetric) DPPs.

For the decomposition $\boldsymbol{L} = \boldsymbol{V}^\top\boldsymbol{V} + \boldsymbol{B}^\top\boldsymbol{C}\boldsymbol{B}$, we have $\boldsymbol{V}, \boldsymbol{B} \in \mathbb{R}^{d\times n}, \boldsymbol{C} \in \mathbb{R}^{d\times d}$ and $\boldsymbol{C}$ is skew-symmetric. Here we can think of the items having having a latent low-dimensional representation $(\boldsymbol{v}_i, \boldsymbol{b}_i)$ where $\boldsymbol{v}_i, \boldsymbol{b}_i \in \mathbb{R}^d$. Intuitively, a low-dimensional representation (when compared to n) is sufficient for representing items because any particular item only interacts with a small number of other items in real-world datasets, as evidenced by the fact that the maximum basket size encountered in real-world data is much smaller than n.

3.3.3. Background on Streaming and Online Algorithms

The main difference between streaming and online algorithms are their parameters of interest. In streaming algorithms (Muthukrishnan, 2005), the main focus is on the solution quality at the end of the stream and memory used by the algorithm throughout the stream. Instead, in online algorithms (Karp, Vazirani, and Vazirani, 1990), the main focus is on the solution quality at every time step and update time after seeing a new input. In the streaming and online models we will define, the online setting is a more restrictive version of the streaming setting. Therefore, for us, any algorithms which are valid online algorithms are also valid streaming algorithms. However, not all streaming algorithms are online algorithms. For instance, our main streaming algorithm (Algorithm 1) is not a valid online algorithm.

3.4. Streaming MAP Inference

In this section, we formulate the streaming MAP inference problem for NDPPs and design an algorithm for this problem with guarantees on the solution quality, space, and time.

3.4.1. Streaming MAP Inference Problem

We study the MAP Inference problem in low-rank NDPPs in the streaming setting where we see columns of a $2d \times n$ matrix in order (column-arrival model). Given some fixed skew-symmetric matrix $C \in \mathbb{R}^{d\times d}$, consider a stream of $2d$-dimensional vectors (which can be viewed as pairs of d-dimensional vectors) arriving in order:

$$(\boldsymbol{v}_1, \boldsymbol{b}_2), (\boldsymbol{v}_2, \boldsymbol{b}_2), \dots, (\boldsymbol{v}_n, \boldsymbol{b}_n) \text{ where } \boldsymbol{v}_t, \boldsymbol{b}_t \in \mathbb{R}^d, \forall\, t \in [n]$$

The main goal in the streaming setting is to output the maximum likelihood subset $S \subseteq [n]$ of cardinality k at the end of the stream assuming that S is drawn from the NDPP characterized by $\boldsymbol{L} = \boldsymbol{V}^\top \boldsymbol{V} + \boldsymbol{B}^\top \boldsymbol{C}\boldsymbol{B}$ i.e.,

$$\begin{aligned}(3.1) \qquad S &= \underset{S\subseteq[n],|S|=k}{\operatorname{argmax}} \det(\boldsymbol{L}_S) \\ &= \underset{S\subseteq[n],|S|=k}{\operatorname{argmax}} \det(\boldsymbol{V}_S^\top \boldsymbol{V}_S + \boldsymbol{B}_S^\top \boldsymbol{C}\boldsymbol{B}_S)\end{aligned}$$

For any $S \subseteq [n], \boldsymbol{V}_S \in \mathbb{R}^{d\times|S|}$ is the matrix whose each column corresponds to $\{\boldsymbol{v}_i, i \in S\}$. Similarly, $\boldsymbol{B}_S \in \mathbb{R}^{d\times|S|}$ is the matrix whose columns correspond to $\{\boldsymbol{b}_i, i \in S\}$. In the case of symmetric DPPs, this maximization problem in the non-streaming setting corresponds to MAP Inference in cardinality constrained DPPs, also known as k-DPPs (Kulesza & Taskar, 2011).

Definition 3.4.1. Given three matrices $\boldsymbol{V} \in \mathbb{R}^{d\times k}, \boldsymbol{B} \in \mathbb{R}^{d\times k}$ and $\boldsymbol{C} \in \mathbb{R}^{d\times d}$, let $\mathcal{T}_{\text{det}}(k, d)$ denote the running time of computing $\det(\boldsymbol{V}^\top \boldsymbol{V} + \boldsymbol{B}^\top \boldsymbol{C}\boldsymbol{B})$. We can take $\mathcal{T}_{\text{det}}(k, d)$ being $O(kd^2)$ as a crude estimate.

Note that $\mathcal{T}_{\text{det}}(k, d) = 2\mathcal{T}_{\text{mat}}(d, k, d) + \mathcal{T}_{\text{mat}}(d, d, k) + \mathcal{T}_{\text{mat}}(k, k, k)$ where $\mathcal{T}_{\text{mat}}(a, b, c)$ is the time required to multiply two matrices of dimensions $a \times b$ and $b \times c$. We have the last $\mathcal{T}_{\text{mat}}(k, k, k)$ term because computing the determinant of a $k \times k$ matrix can be done

Algorithm 1 Streaming Partition Greedy MAP Inference for low-rank NDPPs

```
Input: Length of the stream $n$ and a stream of data points $\{(\boldsymbol{v}_1, \boldsymbol{b}_1), (\boldsymbol{v}_2, \boldsymbol{b}_2), \ldots, (\boldsymbol{v}_n, \boldsymbol{b}_n)\}$
Output: A solution set $S$ of cardinality $k$ at the end of the stream.
$S_0 \leftarrow \emptyset,\ s_0 \leftarrow \emptyset$
while new data $(\boldsymbol{v}_t, \boldsymbol{b}_t)$ arrives in stream at time $t$ do
    $i \leftarrow \lceil \frac{tk}{n} \rceil$
    if $f(S_{i-1} \cup \{t\}) > f(S_{i-1} \cup \{s_i\})$ then
        $s_i \leftarrow t$
    if $t$ is a multiple of $\frac{n}{k}$ then
        $S_i \leftarrow S_{i-1} \cup s_i$
        $s_i \leftarrow \emptyset$
return $S_k$
```

(essentially) in the same time as computing the product of two matrices of dimension $k \times k$ (Aho et al., 1974, Theorem 6.6).

We will now describe a streaming algorithm for MAP inference in NDPPs, which we call the *Streaming Partition Greedy* algorithm.

3.4.2. Streaming Partition Greedy

Outline of Algorithm 1: Our algorithm picks the first element of the solution greedily from the first seen $\frac{n}{k}$ elements, the second element from the next sequence of $\frac{n}{k}$ elements and so on. As described in Algorithm 1, let us use $S_0, S_1, \ldots, S_k$ to denote the solution sets maintained by the algorithm, where S_i represents the solution set of size i. In particular, we have that $S_i = S_{i-1} \cup \{s_i\}$ where $s_i = \arg\max_{j \in \mathcal{P}_i} f(S \cup \{j\})$ and $\mathcal{P}_i$ denotes the i'th partition of the data i.e., $\mathcal{P}_i := \{\frac{(i-1)\cdot n}{k} + 1, \frac{(i-1)\cdot n}{k} + 2, \ldots, \frac{i \cdot n}{k}\}$.

$$\underbrace{(\boldsymbol{v}_1, \boldsymbol{b}_1), (\boldsymbol{v}_2, \boldsymbol{b}_2), ..., (\boldsymbol{v}_{n/k}, \boldsymbol{b}_{n/k})}_{\mathcal{P}_1}, \ldots, \underbrace{(\boldsymbol{v}_{n-(n/k)+1}, \boldsymbol{b}_{n-(n/k)+1}), ..., (\boldsymbol{v}_n, \boldsymbol{b}_n)}_{\mathcal{P}_k} \tag{3.2}$$

Theorem 3.4.2. *For a random-order arrival stream, if S is the solution output by Algorithm 1 at the end of the stream and $\sigma_{\min} > 1$ where $\sigma_{\min}$ and $\sigma_{\max}$ denote the smallest and largest singular values of $\boldsymbol{L}_S$ among all $S \subseteq [n]$ and $|S| \leq 2k$, then*

$$\frac{\mathbb{E}[\log\det(\boldsymbol{L}_S)]}{\log(\mathrm{OPT})} \geq \left(1 - \frac{1}{\sigma_{\min}^{(1-\frac{1}{e})\cdot(2\log\sigma_{\max} - \log\sigma_{\min})}}\right)$$

where $\boldsymbol{L}_S = \boldsymbol{V}_S^\top \boldsymbol{V}_S + \boldsymbol{B}_S^\top \boldsymbol{C} \boldsymbol{B}_S$ *and* $\mathrm{OPT} = \max_{R \subseteq [n],\ |R|=k} \det(\boldsymbol{V}_R^\top \boldsymbol{V}_R + \boldsymbol{B}_R^\top \boldsymbol{C} \boldsymbol{B}_R)$.

We will first give a high-level proof sketch for this theorem and defer the full proof to Appendix B.1.

Proof sketch. For a random-order arrival stream, the distribution of any set of consecutive n/k elements is the same as the distribution of n/k elements picked uniformly at random (without replacement) from $[n]$. If the objective function f is submodular, then this algorithm has an approximation guarantee of $(1 - 2/e)$ by Mirzasoleiman, Badanidiyuru, Karbasi, Vondrák, and Krause (2015). But neither $\det(\boldsymbol{L}_S)$ nor $\log\det(\boldsymbol{L}_S)$ are submodular. Instead, Gartrell, Han, Dohmatob, Gillenwater, and Brunel (2021) [Equation 45] showed that $\log\det(\boldsymbol{L}_S)$ is “close” to submodular when $\sigma_{\min} > 1$ where this closeness is measured using a parameter known as “submodularity ratio” (Bian, Buhmann, Krause, and Tschiatschek, 2017). Using this parameter, we can prove a guarantee for our algorithm. ■

Theorem 3.4.3. *For any length-n stream $(\boldsymbol{v}_1, \boldsymbol{b}_1), \ldots, (\boldsymbol{v}_n, \boldsymbol{b}_n)$ where $(\boldsymbol{v}_t, \boldsymbol{b}_t) \in \mathbb{R}^d \times \mathbb{R}^d\ \forall\ t \in [n]$, the space used is $O(k^2 + d^2)$ and the total time taken is $O(n \cdot \mathcal{T}_{\det}(k, d))$ where $\mathcal{T}_{\det}(k, d)$ is the time taken to compute $f(S) = \det(\boldsymbol{V}_S^\top \boldsymbol{V} + \boldsymbol{B}_S^\top \boldsymbol{C}\boldsymbol{B})$ for $|S| = k$.*

Algorithm 2 Online-LSS: Online MAP Inference for low-rank NDPPs with Stash.

```
Input: A stream of data points $\{(\boldsymbol{v}_1, \boldsymbol{b}_1), \ldots, (\boldsymbol{v}_n, \boldsymbol{b}_n)\}$, and a constant $\alpha \geq 1$
Output: A solution set $S$ of cardinality $k$ at the end of the stream.
$S, T \leftarrow \emptyset$
while new data point $(\boldsymbol{v}_t, \boldsymbol{b}_t)$ arrives in stream at time $t$ do
    if $|S| < k$ and $f(S \cup \{t\}) \neq 0$ then
        $S \leftarrow S \cup \{t\}$
    else
        $i \leftarrow \arg\max_{j \in S} f(S \cup \{t\} \setminus \{j\})$
        if $f(S \cup \{t\} \setminus \{i\}) > \alpha \cdot f(S)$ then
            $S \leftarrow S \cup \{t\} \setminus \{i\}$
            $T \leftarrow T \cup \{i\}$
            while $\exists\ a \in S,\ b \in T : f(S \cup \{b\} \setminus \{a\}) > \alpha \cdot f(S)$ do
                $S \leftarrow S \cup \{b\} \setminus \{a\}$
                $T \leftarrow T \cup \{a\} \setminus \{b\}$
return $S$
```

Proof. For any particular data-point $(\boldsymbol{v}_t, \boldsymbol{b}_t)$, Algorithm 1 needs to compute $f(S_{i-1} \cup \{t\})$, where $f(S) = \det(\boldsymbol{V}_S^\top \boldsymbol{V} + \boldsymbol{B}_S^\top C \boldsymbol{B}_S)$ and $S = S_{i-1} \cup \{t\}$. This takes at most $O(k^2 + d^2)$ space and $\mathcal{T}_{\det}(k, d)$ time. The algorithm also needs to store S_{i-1}, s_i and $f(S_{i-1} \cup \{s_i\})$ but all of these are dominated by $O(k^2 + d^2)$ space needed to compute the determinant. All the other comparison and update steps are also much faster and so the total time is $O(n \cdot \mathcal{T}_{\det}(k, d))$ ■

3.5. Online MAP Inference for NDPPs

We now consider the online MAP inference problem for NDPPs, which is natural in settings where data is generated on the fly. In addition to the constraints of the streaming setting (Section 3.4.1), our online setting requires us to maintain a valid solution at every time step. In this section, we provide two algorithms for solving this problem.

3.5.1. Online Local Search with a Stash

Outline of Algorithm 2: On a high-level, our algorithm is a generalization of the Online-LS algorithm for DPPs from (Bhaskara, Karbasi, Lattanzi, and Zadimoghaddam, 2020). At each time step $t \in [n]$ (after $t \geq k$), our algorithm maintains a candidate solution subset of indices S of cardinality k from the data seen so far i.e., $S \subseteq [t]$ s.t. $|S| = k$ in a streaming fashion. Additionally, it also maintains two matrices $\boldsymbol{V}_S, \boldsymbol{B}_S \in \mathbb{R}^{d \times |S|}$ where the columns of $\boldsymbol{V}_S$ are $\{\boldsymbol{v}_i, i \in S\}$ and the columns of $\boldsymbol{B}_S$ are $\{\boldsymbol{b}_i, i \in S\}$. Whenever the algorithm sees a new data point $(\boldsymbol{v}_t, \boldsymbol{b}_t)$, it replaces an existing index from S with the newly arrived index if doing so increases $f(S)$ at-least by a factor of $\alpha \geq 1$ where α is a parameter to be chosen (we can think of α being 2 for understanding the algorithm). Instead of just deleting the index replaced from S, it is stored in an auxiliary set T called the "stash" (and also maintains corresponding matrices $\boldsymbol{V}_T, \boldsymbol{B}_T$), which the algorithm then uses to performs a local search over to find a locally optimal solution.

We now define a data-dependent parameter Δ which we will need to describe the time and space used by Algorithm 2.

Definition 3.5.1. Let the first non-zero value of $f(S)$ with $|S| = k$ that can be achieved in the stream without any swaps be $\mathrm{val}_{\mathrm{nz}}$ i.e., till S reaches a size k, any index seen is added to S if $f(S)$ remains non-zero even after adding it. Let us define $\Delta := \frac{\mathrm{OPT}_k}{\mathrm{val}_{\mathrm{nz}}}$ where $\mathrm{OPT}_k = \max_{S \subseteq [n], |S| = k} \det(\boldsymbol{V}_S^\top \boldsymbol{V}_S + \boldsymbol{B}_S^\top C \boldsymbol{B}_S)$.

Note that Δ is a data-dependent parameter which can potentially be unbounded. This happens in the hard-instance we will describe in Section 3.6. However, for practical datasets, Δ doesn't seem to be too bad (see the experiments section for a more detailed discussion).

Theorem 3.5.2. *For any length-n stream* $(\boldsymbol{v}_1, \boldsymbol{b}_1), \ldots, (\boldsymbol{v}_n, \boldsymbol{b}_n)$ *where* $(\boldsymbol{v}_t, \boldsymbol{b}_t) \in \mathbb{R}^d \times \mathbb{R}^d \ \forall\ t \in [n]$*, the worst case update time of Algorithm 2 is* $O(\mathcal{T}_{\det}(k,d) \cdot k \log^2(\Delta))$ *where* $\mathcal{T}_{\det}(k,d)$ *is the time taken to compute* $f(S) = \det(\boldsymbol{V}_S^\top \boldsymbol{V} + \boldsymbol{B}_S^\top \boldsymbol{C}\boldsymbol{B})$ *for* $|S| = k$*. Furthermore, the amortized update time is* $O(\mathcal{T}_{\det}(k,d) \cdot (k + \frac{k \log^2(\Delta)}{n}))$ *and the space used at any time step is at most* $O(k^2 + d^2 + d \log_\alpha(\Delta))$*.*

Proof. For every iteration of the while loop in line 4: It takes at most $\mathcal{T}_{\det}(k,d)$ time for checking the first if condition (lines 5-6). The $\text{argmax}_{j \in S}\, f(S \cup \{t\} \setminus \{j\}$ step takes at most $k \cdot \mathcal{T}_{\det}(k,d)$ time. The while loop in line 12 takes time at most $|S| \cdot |T| \cdot \mathcal{T}_{\det}(k,d)$ for every instance of an increase in $f(S)$. Note that $f(S)$ can increase at most $\log_\alpha(\Delta)$ times since the value of $f(S)$ cannot exceed OPT_k. Therefore, the update time of Algorithm 2 is at most $\mathcal{T}_{\det}(k,d) + k \cdot \mathcal{T}_{\det}(k,d) + \log_\alpha(\Delta) \cdot (|S| \cdot |T| \cdot \mathcal{T}_{\det}(k,d)) \leq \mathcal{T}_{\det}(k,d) \cdot \left(k + 1 + k \log_\alpha^2(\Delta)\right)$ since $|S| \leq k$ and $|T| \leq \log_\alpha(\Delta)$. Notice that the cardinality of T can increase by 1 only when the value of $f(S)$ increases at least by a factor of α and so $|T| \leq \log_\alpha(\Delta)$.

During any time step t, the algorithm needs to store the indices in S, T and the corresponding matrices $\boldsymbol{V}_S, \boldsymbol{B}_S, \boldsymbol{V}_T, \boldsymbol{B}_T$. Since $|S| \leq k, |T| \leq \log_\alpha(\Delta)$ and it takes d words to store every $\boldsymbol{v}_i$ and $\boldsymbol{b}_i$, we need at most $k + \log_\alpha(\Delta) + 2dk + 2d \log_\alpha(\Delta)$ words to store all these in memory. The space needed to compute $\det(\boldsymbol{V}_S^\top \boldsymbol{V}_S + \boldsymbol{B}_S^\top \boldsymbol{C}\boldsymbol{B}_S)$ is at most $O(k^2 + d^2)$. We compute all such determinants one after the other in our algorithm. So the algorithm only needs space for one such computation during it's run. Therefore, the space required by Algorithm 2 is $O(k^2 + d^2 + d \log_\alpha(\Delta))$. ■

3.5.2. Online 2-neighborhood Local Search Algorithm with a Stash

Algorithm 3 ONLINE-2-NEIGHBOR: Local Search over 2-neighborhoods with Stash for Online NDPP MAP Inference.

```
Input: A stream of data points $\{(\boldsymbol{v}_1, \boldsymbol{b}_1), (\boldsymbol{v}_2, \boldsymbol{b}_2), \ldots, (\boldsymbol{v}_n, \boldsymbol{b}_n)\}$, and a constant $\alpha \geq 1$
Output: A solution set $S$ of cardinality $k$ at the end of the stream.
$S, T \leftarrow \emptyset$
while new data $(\boldsymbol{v}_t, \boldsymbol{b}_t)$ arrives in stream at time $t$ do
    if $|S| < k$ and $f(S \cup \{t\}) \neq 0$ then
        $S \leftarrow S \cup \{t\}$
    else
        $\{i, j\} \leftarrow \arg\max_{a,b \in S} (f(S \cup \{t\} \setminus \{a\}), f(S \cup \{t-1, t\} \setminus \{a, b\}))$
        $f_{\max} \leftarrow \max_{a,b \in S} (f(S \cup \{t\} \setminus \{a\}), f(S \cup \{t-1, t\} \setminus \{a, b\}))$
        if $f_{\max} > \alpha \cdot f(S)$ then
            if two items are chosen to be replaced: then
                $S \leftarrow S \cup \{t-1, t\} \setminus \{i, j\}$
                $T \leftarrow T \cup \{i, j\}$
            else
                $S \leftarrow S \cup \{t\} \setminus \{i\}$
                $T \leftarrow T \cup \{i\}$
            while $\exists\, a, b \in S,\ c, d \in T : f(S \cup \{c, d\} \setminus \{a, b\}) > \alpha \cdot f(S)$ do
                $S \leftarrow S \cup \{c, d\} \setminus \{a, b\}$
                $T \leftarrow T \cup \{a, b\} \setminus \{c, d\}$
return $S$
```

Before we describe our algorithm, we will define a *neighborhood* of any solution, which will be useful for describing the local search part of our algorithm.

Definition 3.5.3 ($\mathcal{N}_r(S, T)$)**.** For any natural number $r \geq 1$ and any sets S, T we define the r-neighborhood of S with respect to T

$$\mathcal{N}_r(S, T) \coloneqq \{S' \subseteq S \cup T \mid |S'| = |S| \text{ and } |S' \setminus S| \leq r\}$$

Outline of Algorithm 3: Similar to Algorithm 2, our new algorithm also maintains two subsets of indices S and T, and corresponding data matrices $\boldsymbol{V}_S, \boldsymbol{B}_S, \boldsymbol{V}_T, \boldsymbol{B}_T$. Whenever the algorithm sees a new data-point $(\boldsymbol{v}_t, \boldsymbol{b}_t)$, it checks if the solution quality $f(S)$ can be improved by a factor of α by replacing any element in S with the newly seen data-point. Additionally, it also checks if the solution quality can be made better by including both the points $(\boldsymbol{v}_t, \boldsymbol{b}_t)$ and the data-point $(\boldsymbol{v}_{t-1}, \boldsymbol{b}_{t-1})$. Further, the algorithm tries to improve the solution quality by performing a local search on $\mathcal{N}_2(S, T)$ i.e., the neighborhood of the candidate solution S using the stash T by replacing at most two elements of S. There might be interactions captured by *pairs* of items which are much stronger than single items in NDPPs (see example 5 from (Anari and Vuong, 2022)).

Theorem 3.5.4. *For any length-n stream $(\boldsymbol{v}_1, \boldsymbol{b}_1), \ldots, (\boldsymbol{v}_n, \boldsymbol{b}_n)$ where $(\boldsymbol{v}_t, \boldsymbol{b}_t) \in \mathbb{R}^d \times \mathbb{R}^d \;\forall\; t \in [n]$, the worst case update time of Algorithm 3 is $O(\mathcal{T}_{\det}(k, d) \cdot k^2 \log^3(\Delta))$ where $\mathcal{T}_{\det}(k, d)$ is the time taken to compute $f(S) = \det(\boldsymbol{V}_S^\top \boldsymbol{V} + \boldsymbol{B}_S^\top \boldsymbol{C}\boldsymbol{B})$ for $|S| = k$. The amortized update time is $O\left(\mathcal{T}_{\det}(k, d) \cdot \left(k^2 + \frac{k^2 \log^3(\Delta)}{n}\right)\right)$ and the space used at any time step is at most $O(k^2 + d^2 + d \log_\alpha(\Delta))$.*

Proof. It takes at most $\mathcal{T}_{\det}(k, d)$ time for lines 5-6 (same as in LSS).
The $\arg\max_{a,b \in S} \left(f(S \cup \{t\} \setminus \{a\}), f(S \cup \{t-1, t\} \setminus \{a, b\})\right)$ step takes at most $k^2 \cdot \mathcal{T}_{\det}(k, d)$ time. The while loop in line 18 takes time at most $|S|^2 \cdot |T|^2 \cdot \mathcal{T}_{\det}(k, d)$ for every instance of an increase in $f(S)$. Similar to LSS, $f(S)$ can increase at most by a factor of $\log_\alpha(\Delta)$ since the value of $f(S)$ cannot exceed OPT_k. Therefore, the update time of Algorithm 3 is at most $\mathcal{T}_{\det}(k, d) + k^2 \cdot \mathcal{T}_{\det}(k, d) + \log_\alpha(\Delta) \cdot (|S|^2 \cdot |T|^2 \cdot \mathcal{T}_{\det}(k, d)) \leq \mathcal{T}_{\det}(k, d) \cdot \left(k^2 + 1 + k^2 \log_\alpha^3(\Delta)\right)$ since $|S| \leq k$ and $|T| \leq \log_\alpha(\Delta)$.

Although Algorithm 3 executes more number of determinant computations than Algorithm 2, all of them are done sequentially and only the maximum value among all the previously computed values in any specific iteration needs to be stored in memory. Therefore, the space needed is (nearly) the same for both the algorithms. ■

3.6. Hard Instance for One-Pass MAP Inference of NDPPs

We will now give a high-level description of a hard instance for one-pass MAP inference of NDPPs with sub-linear memory (inspired by (Anari & Vuong, 2022, Example 5)) on which all of our algorithms output solutions with zero objective value whereas the optimal solution has non-zero value. Due to this, we believe it might be impossible to prove any bounded approximation factor guarantees for our algorithms without any strong additional assumptions.

Sketch of the hard instance. Suppose we have a total of $2d$ items consisting of **pairs** of complementary items like modem-router, printer-ink cartridge, pencil-eraser etc. Let us use $\{1, 1^{\mathsf{c}}, 2, 2^{\mathsf{c}}, \ldots, d, d^{\mathsf{c}}\}$ to denote them. Any item i is independent of every item other than it's complement i^{c}. Individually, $\mathbb{P}[\{i\}] = \mathbb{P}[\{i^{\mathsf{c}}\}] = 0$. And $\mathbb{P}[\{i, i^{\mathsf{c}}\}] = x_i^2$ with $x_i > 0$ for all $i \in [d]$. Also, we have $\mathbb{P}[\{i, j\}] = 0$ for any $i \neq j$. Suppose any of our online algorithms are given the sequence $\{1, 2, 3, \ldots, d, r^{\mathsf{c}}\}$ where $r \in [d]$ is some arbitrary item and the algorithm needs to pick 2 items i.e., $k = 2$. Then, OPT > 0 whereas all of our online algorithms (Online LSS, Online 2-neighbor, Online-Greedy) will fail to output a valid solution. We provide a more complete description with the instantiations of the matrices $\boldsymbol{B}, \boldsymbol{C}, \boldsymbol{V}$ in Appendix B.2.

Comparison between MAP Inference for NDPPs and symmetric DPPs. The hard instance described above necessarily uses the skew-symmetric component of the kernel matrix to form such *complementary* pairs and this illustrates some of the divergence between NDPPs and symmetric DPPs. NDPPs are significantly more general (and complex) than DPPs and unlike the case for DPPs where the objective function corresponds to a nice geometric notion i.e., volume, the objective function for MAP Inference on NDPPs doesn't have a corresponding clean notion. This is a core issue because of which it is unclear how the proofs of approximation factors for similar algorithms for DPPs would generalize to NDPPs (the proof techniques for DPPs from (Bhaskara, Karbasi, Lattanzi, and Zadimoghaddam, 2020) for Online-LS heavily use the fact that the objective function is a volume and thus use properties of coresets developed for related geometric problems).

3.7. Experiments

We first learn all the matrices $\boldsymbol{B}, \boldsymbol{C}$, and $\boldsymbol{V}$ by applying the learning algorithm of (Gartrell, Han, Dohmatob, Gillenwater, and Brunel, 2021) on several real-world datasets. For example, some datasets consist of carts of items bought by Amazon customers. Then, we run our inference algorithms on the learned $\boldsymbol{B}, \boldsymbol{C}$, and $\boldsymbol{V}$. More details about the experiments and the datasets used can be found in Appendix B.3.

As a point of comparison, we also use the offline greedy algorithm from (Gartrell, Han, Dohmatob, Gillenwater, and Brunel, 2021). This algorithm stores all data in memory and makes k passes over the dataset and in each round, picks the data point which gives the maximum marginal gain in solution value. Online-Greedy (Algorithm 4) is a simple online variant of the greedy algorithm which replaces a point in the current solution set with the observed point if doing so increases the objective.

First, we want to mention that the performance of our streaming algorithm (Algorithm 1) on the datasets we consider is (unfortunately) pretty bad (the objective function value

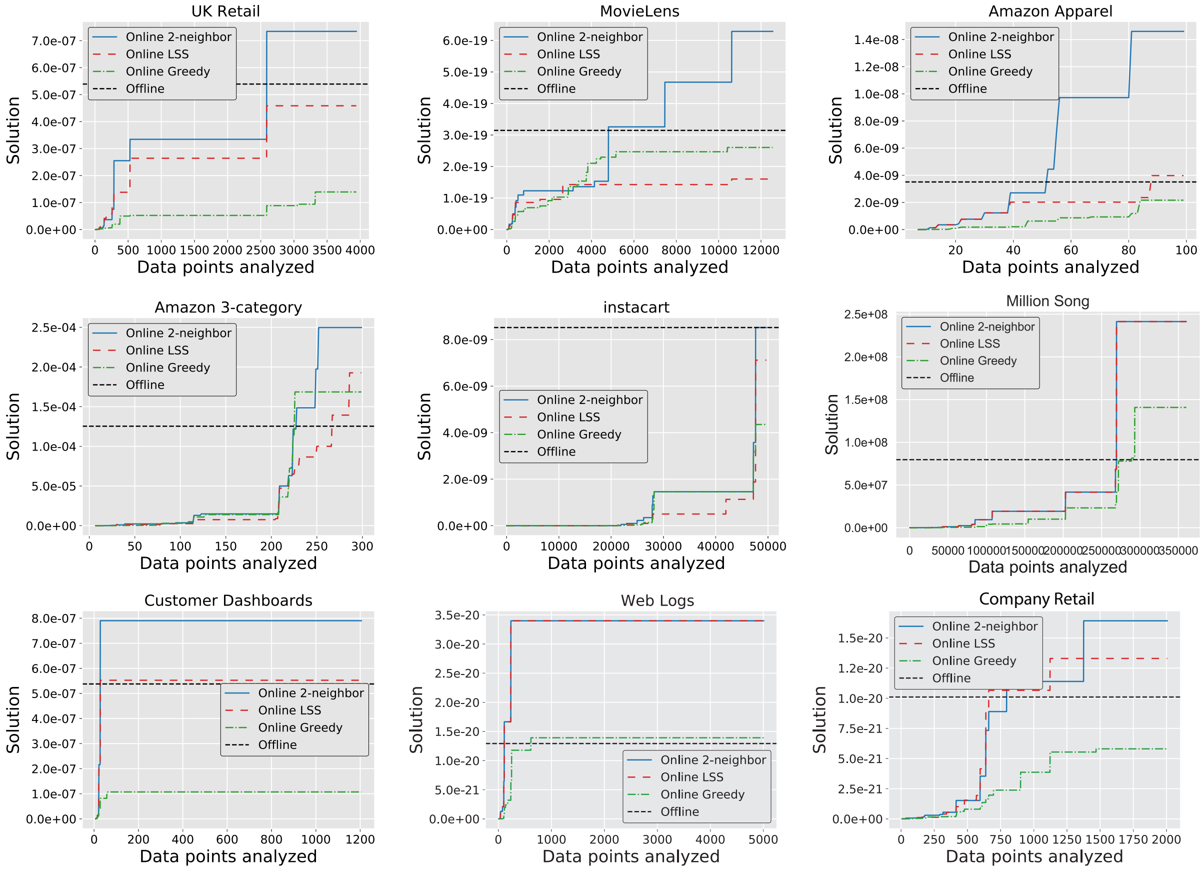


Figure 3.1. Solution quality i.e., objective function value as a function of the number of data points analyzed for all our online algorithms and also the offline greedy algorithm. All our online algorithms give comparable (or even better) performance to offline greedy using only a single pass and a small fraction of the memory.

is close to zero in most cases). This is because in real-world datasets, adjacent pairs of items have a very high likelihood of occurring together when compared to pairs of items far from each other. For example, white socks and grey socks might be adjacent to each other in numbering. And customers tend to buy both of them in a basket. Our streaming algorithm is forced to ignore most such pairs.

Results for a various datasets for our online algorithms are provided in Figure 3.1. Surprisingly, the solution quality of our online algorithms compare favorably with the

Algorithm 4 Online-Greedy: Online Greedy MAP Inference for NDPPs

```
Input: A stream of data points $\{(\boldsymbol{v}_1, \boldsymbol{b}_1), (\boldsymbol{v}_2, \boldsymbol{b}_2), \ldots, (\boldsymbol{v}_n, \boldsymbol{b}_n)\}$
Output: A solution set $S$ of cardinality $k$ at the end of the stream.
$S \leftarrow \emptyset$
while new data $(\boldsymbol{v}_t, \boldsymbol{b}_t)$ arrives in stream at time $t$ do
    if $|S| < k$ and $f(S \cup \{t\}) \neq 0$ then
        $S \leftarrow S \cup \{t\}$
    else
        $i \leftarrow \arg\max_{j \in S} f(S \cup \{t\} \setminus \{j\})$
        if $f(S \cup \{t\} \setminus \{i\}) > f(S)$ then
            $S \leftarrow S \cup \{t\} \setminus \{i\}$
return $S$
```

offline greedy algorithm while using only a single-pass over the data, and a tiny fraction of memory. In most cases, Online-2-neighbor (Algorithm 3) performs better than Online-LSS (Algorithm 2) which in turn performs better than the online greedy algorithm (Algorithm 4). Strikingly, our online-2-neighbor algorithm performs even better than offline greedy in many cases.

We also perform several experiments comparing the number of determinant computations (as a system-independent proxy for time) and the number of swaps (as a measure of solution consistency) of all our online algorithms. Results for determinant computations (Figure B.1) and swaps (Figure B.2) can be found in Appendix B.4. We summarize the main findings here. The number of determinant computations of Online-LSS is comparable to that of Online Greedy but the number of swaps performed is significantly smaller. Online-2-neighbor is the most time-consuming but superior performing algorithm in terms of solution quality.

Our experimental results in Appendix B.4 demonstrate that our online algorithms use substantially lower memory than any offline algorithms. Note that the main memory bottleneck for offline inference algorithms (Gartrell et al., 2021; Anari & Vuong, 2022) is the need to store the entire data-set in memory. We can consider other factors (like the memory needed for computing $\det(k, d)$ i.e., $O(k^2 + d^2)$ (which can be re-used every-time) to be essentially free because the regime of interest is $n \gg d \geq k$. The memory usage by our online algorithms is also primarily dominated by the size of the stash, which is upper bounded by the number of swaps for which we have plots in Appendix B.4.2. Similarly, the update times also depend only on the size of the stash (which are quite small and so we have very fast update times).

We also investigate the performance of our algorithms under the random stream paradigm, where we consider a random permutation of some of the datasets used earlier. Results for the solution quality (Figure B.3), number of determinant computations and swaps (Figure B.4) can be found in Appendix B.4.3. In this setting, we see that Online-LSS and Online-2-neighbor have nearly identical performance and are always better than Online-Greedy in terms of solution quality and number of swaps.

We study the effect of varying α in Online-LSS (Algorithm 2) for various values of set sizes k in Appendices B.4.4 and B.4.5. We notice that, in general, the solution quality, number of determinant computations, and the number of swaps increase as α decreases (Figure B.5). We also see that as k increases, the solution value decreases across all values of α (Figure B.6). This is in accordance with our intuition that the probability of larger sets should be smaller.

3.8. Conclusion & Future Directions

In this paper, we formulate and study the streaming and online MAP inference problems for Nonsymmetric Determinantal Point Processes. To the best of our knowledge, this is the first work to study these problems in these practical settings. We design new one-pass algorithms for these problems, prove theoretical guarantees for them in terms of space required, time taken, and solution quality for our algorithms, and empirically show that they perform comparably or sometimes even better than state-of-the-art offline algorithms while using substantially lower memory.

As we have discussed in the experiments section, the empirical performance of our partition greedy algorithm is quite bad. The main reason we have chosen to include it in this paper is because it is the only one for which we have a provable guarantee on the approximation quality (albeit with strong assumptions). This also leads us to an important open direction from our work, i.e., gaining a better theoretical understanding of our online algorithms, potentially by proving approximation bounds going beyond worst-case analysis (Roughgarden, 2021). For instance, by assuming that the learned NDPP model satisfies natural assumptions like perturbation stability (Bilu & Linial, 2012; Makarychev et al., 2014; Angelidakis et al., 2017). For example, in the line of prior work (Lang et al., 2018; 2019; 2021a;b) studying MAP inference for Potts models. Another interesting direction is in providing parallelizable algorithms which use a *small* number of passes (greater than one but less than k) - similar to the k-means$\|$ (read as "k-means parallel") algorithm (Bahmani et al., 2012; Makarychev et al., 2020)

We have only studied 1-neighbor and 2-neighbor online local search algorithms in our paper. Extending them to arbitrary sizes of subsets (3-neighbor, etc.) is also another interesting open direction. Understanding at which point the degree of interactions cease to provide benefits that are worth the increase in memory/time constraints would be of interest to the DPP research community. Are pairwise interactions, as in 2-neighbor, sufficient to characterize most of the necessary NDPP properties?

Bibliography

Abbe, E. Community detection and stochastic block models. *Foundations and Trends® in Communications and Information Theory*, 14(1-2):1–162, 2018. ISSN 1567-2190. doi: 10.1561/0100000067.

Aho, A., Hopcroft, J., and Ullman, J. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

Alon, N., Matias, Y., and Szegedy, M. The space complexity of approximating the frequency moments. *Journal of Computer and system sciences*, 58(1):137–147, 1999.

Anari, N. and Vuong, T.-D. From Sampling to Optimization on Discrete Domains with Applications to Determinant Maximization. In *Conference on Learning Theory (COLT)*, 2022.

Angelidakis, H., Makarychev, K., and Makarychev, Y. Algorithms for stable and perturbation-resilient problems. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pp. 438–451, 2017.

Archer, A., Fakcharoenphol, J., Harrelson, C., Krauthgamer, R., Talwar, K., and Tardos, É. Approximate classification via earthmover metrics. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1079–1087. Society for Industrial and Applied Mathematics, 2004.

Bahmani, B., Moseley, B., Vattani, A., Kumar, R., and Vassilvitskii, S. Scalable k-means++. *Proceedings of the VLDB Endowment*, 5(7):622–633, 2012.

Bhaskara, A., Karbasi, A., Lattanzi, S., and Zadimoghaddam, M. Online MAP Inference of Determinantal Point Processes. In *Neural Information Processing Systems (NeurIPS)*, 2020.

Bian, A. A., Buhmann, J. M., Krause, A., and Tschiatschek, S. Guarantees for Greedy Maximization of Non-submodular Functions with Applications. In *International Conference on Machine Learning (ICML)*, 2017.

Bilu, Y. and Linial, N. Are stable instances easy? *Combinatorics, Probability and Computing*, 21(5):643–660, 2012.

Birchfield, S. and Tomasi, C. A pixel dissimilarity measure that is insensitive to image sampling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(4): 401–406, 1998.

Borodin, A. Determinantal point processes. In *The Oxford Handbook of Random Matrix Theory*. Oxford University Press, 2009.

Borodin, A. and El-Yaniv, R. *Online Computation and Competitive Analysis*. Cambridge University Press, 2005.

Boykov, Y., Veksler, O., and Zabih, R. Fast approximate energy minimization via graph cuts. *IEEE Transactions on pattern analysis and machine intelligence*, 23(11):1222–1239, 2001.

Brunel, V.-E. Learning Signed Determinantal Point Processes through the Principal Minor Assignment Problem. In *Neural Information Processing Systems (NeurIPS)*, 2018.

Brunel, V.-E., Moitra, A., Rigollet, P., and Urschel, J. Rates of estimation for determinantal point processes. In *Conference on Learning Theory (COLT)*, 2017.

Carlson, C., Davies, E., Fraiman, N., Kolla, A., Potukuchi, A., and Yap, C. Algorithms for the ferromagnetic potts model on expanders. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 344–355. IEEE, 2022.

Chekuri, C., Khanna, S., Naor, J. S., and Zosin, L. Approximation algorithms for the metric labeling problem via a new linear programming formulation. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '01, pp. 109–118, USA, 2001. ISBN 0898714907.

Chen, D., Sain, S. L., and Guo, K. Data mining for the online retail industry: A case study of RFM model-based customer segmentation using data mining. *Journal of Database Marketing & Customer Strategy Management*, 2012.

Chen, L., Zhang, G., and Zhou, H. Fast Greedy MAP Inference for Determinantal Point Process to Improve Recommendation Diversity. In *Neural Information Processing Systems (NeurIPS)*, 2018.

Chen, W. and Ahmed, F. PaDGAN: A Generative Adversarial Network for Performance Augmented Diverse Designs. *Journal of Mechanical Design*, 143(3):031703, 2021.

Civril, A. and Magdon-Ismail, M. On selecting a maximum volume sub-matrix of a matrix and related problems. *Theoretical Computer Science*, 410(47-49):4801–4811, 2009.

Dahlhaus, E., Johnson, D. S., Papadimitriou, C. H., Seymour, P. D., and Yannakakis, M. The complexity of multiway cuts. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pp. 241–251, 1992.

Derezinski, M. and Mahoney, M. W. Determinantal point processes in randomized numerical linear algebra. *Notices of the American Mathematical Society*, 68(1):34–45, 2021.

Fiedler, M. and Pták, V. Some generalizations of positive definiteness and monotonicity. *Numerische Mathematik*, 9(2):163–172, 1966.

Galanis, A., Stefankovic, D., Vigoda, E., and Yang, L. Ferromagnetic potts model: Refined# bis-hardness and related results. *SIAM Journal on Computing*, 45(6):2004–2065, 2016.

Gartrell, M., Brunel, V.-E., Dohmatob, E., and Krichene, S. Learning Nonsymmetric Determinantal Point Processes. In *Neural Information Processing Systems (NeurIPS)*, 2019.

Gartrell, M., Han, I., Dohmatob, E., Gillenwater, J., and Brunel, V.-E. Scalable Learning and MAP Inference for Nonsymmetric Determinantal Point Processes. In *International Conference on Learning Representations (ICLR)*, 2021.

Gillenwater, J., Kulesza, A., and Taskar, B. Near-Optimal MAP Inference for Determinantal Point Processes. In *Neural Information Processing Systems (NIPS)*, 2012.

Gillenwater, J., Kulesza, A., Mariet, Z., and Vassilvitskii, S. Maximizing Induced Cardinality Under a Determinantal Point Process. In *Neural Information Processing Systems (NeurIPS)*, 2018.

Globerson, A. and Jaakkola, T. S. Fixing max-product: Convergent message passing algorithms for map lp-relaxations. In *Advances in neural information processing systems*, pp. 553–560, 2008.

Globerson, A., Roughgarden, T., Sontag, D., and Yildirim, C. How hard is inference for structured prediction? In Bach, F. and Blei, D. (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 2181–2190, Lille, France, 07–09 Jul 2015. PMLR. URL `https://proceedings.mlr.press/v37/globerson15.html`.

Goldberg, L. A. and Jerrum, M. Approximating the partition function of the ferromagnetic potts model. *Journal of the ACM (JACM)*, 59(5):1–31, 2012.

Guédon, O. and Vershynin, R. Community detection in sparse networks via grothendieck's inequality. *Probability Theory and Related Fields*, 165(3-4):1025–1049, 2016.

Gurobi Optimization, L. Gurobi optimizer reference manual, 2020. URL `http://www.gurobi.com`.

Han, I. and Gillenwater, J. MAP Inference for Customized Determinantal Point Processes via Maximum Inner Product Search. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

Han, I., Kambadur, P., Park, K., and Shin, J. Faster Greedy MAP Inference for Determinantal Point Processes. In *International Conference on Machine Learning*, 2017.

Hough, J. B., Krishnapur, M., Peres, Y., and Virág, B. Determinantal Processes and Independence. *Probability surveys*, 3:206–229, 2006.

Indyk, P., Mahabadi, S., Oveis Gharan, S., and Rezaei, A. Composable Core-sets for Determinant Maximization: A Simple Near-Optimal Algorithm. In *International Conference on Machine Learning (ICML)*, 2019.

Indyk, P., Mahabadi, S., Oveis Gharan, S., and Rezaei, A. Composable Core-sets for Determinant Maximization Problems via Spectral Spanners. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2020.

Instacart. The Instacart Online Grocery Shopping Dataset, 2017. URL `https://www.instacart.com/datasets/grocery-shopping-2017`. Accessed August 2021.

Karp, R. M. On-Line Algorithms Versus Off-Line Algorithms: How Much is It Worth to Know the Future? In *Proceedings of the IFIP 12th World Computer Congress on Algorithms, Software, Architecture - Information Processing*, 1992.

Karp, R. M., Vazirani, U. V., and Vazirani, V. V. An Optimal Algorithm for On-Line Bipartite Matching. In *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing (STOC)*, 1990.

Kleinberg, J. and Tardos, E. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and markov random fields. *Journal of the ACM (JACM)*, 49(5):616–639, 2002.

Ko, C.-W., Lee, J., and Queyranne, M. An Exact Algorithm for Maximum Entropy Sampling. *Operations Research*, 1995.

Koller, D. and Friedman, N. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Kolmogorov, V. and Wainwright, M. J. On the optimality of tree-reweighted max-product message-passing. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pp. 316–323, 2005.

Kovtun, I. Partial optimal labeling search for a np-hard subclass of (max,+) problems. In *Joint Pattern Recognition Symposium*, pp. 402–409. Springer, 2003.

Krause, A. and Golovin, D. Submodular Function Maximization. In *Tractability: Practical Approaches to Hard Problems*. Cambridge University Press, February 2014.

Kulesza, A. and Taskar, B. k-DPPs: Fixed-size Determinantal Point Processes. In *International Conference on Machine Learning (ICML)*, 2011.

Kulesza, A. and Taskar, B. Determinantal Point Processes for Machine Learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286, 2012.

Lang, H., Sontag, D., and Vijayaraghavan, A. Optimality of Approximate Inference Algorithms on Stable Instances. In *International Conference on Artificial Intelligence and Statistics*, pp. 1157–1166, 2018.

Lang, H., Sontag, D., and Vijayaraghavan, A. Block Stability for MAP Inference. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 216–225, 2019.

Lang, H., Reddy, A., Sontag, D., and Vijayaraghavan, A. Beyond Perturbation Stability: LP Recovery Guarantees for MAP Inference on Noisy Stable Instances. In *International Conference on Artificial Intelligence and Statistics*, pp. 3043–3051. PMLR, 2021a.

Lang, H., Sontag, D., and Vijayaraghavan, A. Graph cuts always find a global optimum for Potts models (with a catch). In *International Conference on Machine Learning*, pp. 5990–5999. PMLR, 2021b.

Launay, C., Desolneux, A., and Galerne, B. Determinantal point processes for image processing. *SIAM Journal on Imaging Sciences*, 14(1):304–348, 2021.

Liu, P., Soni, A., Kang, E. Y., Wang, Y., and Parsana, M. Diversity on the Go! Streaming Determinantal Point Processes under a Maximum Induced Cardinality Objective. In *Proceedings of the Web Conference (WWW)*, 2021.

Macchi, O. The Coincidence Approach to Stochastic Point Processes. *Advances in Applied Probability*, 7(1):83–122, 1975.

Mahabadi, S., Razenshteyn, I., Woodruff, D. P., and Zhou, S. Non-Adaptive Adaptive Sampling on Turnstile Streams. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, 2020.

Makarychev, K. and Makarychev, Y. Perturbation resilience. In Roughgarden, T. (ed.), *Beyond the Worst-Case Analysis of Algorithms*, pp. 95–119. Cambridge University Press, 2021. doi: 10.1017/9781108637435.008. URL https://doi.org/10.1017/9781108637435.008.

Makarychev, K., Makarychev, Y., and Vijayaraghavan, A. Sorting noisy data with partial information. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pp. 515–528. ACM, 2013.

Makarychev, K., Makarychev, Y., and Vijayaraghavan, A. Bilu–Linial stable instances of max cut and minimum multiway cut. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pp. 890–906. SIAM, 2014.

Makarychev, K., Reddy, A., and Shan, L. Improved guarantees for k-means++ and k-means++ parallel. *Advances in Neural Information Processing Systems*, 33:16142–16152, 2020.

McFee, B., Bertin-Mahieux, T., Ellis, D. P., and Lanckriet, G. R. The Million Song Dataset Challenge. In *Proceedings of the International Conference on World Wide Web (WWW)*, 2012.

McSherry, F. Spectral partitioning of random graphs. In *Proceedings of the 42nd IEEE symposium on Foundations of Computer Science*, FOCS '01, pp. 529–, Washington, DC, USA, 2001. IEEE Computer Society. ISBN 0-7695-1390-5. URL http://dl.acm.org/citation.cfm?id=874063.875554.

Meshi, O., London, B., Weller, A., and Sontag, D. Train and test tightness of lp relaxations in structured prediction. *The Journal of Machine Learning Research*, 20(1):480–513, 2019.

Mirzasoleiman, B., Badanidiyuru, A., Karbasi, A., Vondrák, J., and Krause, A. Lazier Than Lazy Greedy. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2015.

Muthukrishnan, S. *Data streams: Algorithms and applications*. Now Publishers Inc, 2005.

Nemhauser, G., Wolsey, L., and Fisher, M. An Analysis of Approximations for Maximizing Submodular Set Functions I. *Mathematical Programming*, 14(1), 1978.

Nguyen, V., Le, T., Yamada, M., and Osborne, M. A. Optimal Transport Kernels for Sequential and Parallel Neural Architecture Search. In *International Conference on Machine Learning (ICML)*, 2021.

Perez-Beltrachini, L. and Lapata, M. Multi-Document Summarization with Determinantal Point Process Attention. *Journal of Artificial Intelligence Research (JAIR)*, 71:371–399, 2021.

Perez-Nieves, N., Yang, Y., Slumbers, O., Mguni, D. H., Wen, Y., and Wang, J. Modelling Behavioural Diversity for Learning in Open-Ended Games. In *International Conference on Machine Learning (ICML)*, 2021.

Potts, R. B. Some generalized order-disorder transformations. *Mathematical Proceedings of the Cambridge Philosophical Society*, 48(1):106–109, 1952. doi: 10.1017/S0305004100027419.

Qian, X., Rossi, R. A., Du, F., Kim, S., Koh, E., Malik, S., Lee, T. Y., and Chan, J. Learning to Recommend Visualizations from Data. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 1359–1369, 2021.

Qin, L., Reddy, A., Song, Z., Xu, Z., and Zhuo, D. Adaptive and dynamic multi-resolution hashing for pairwise summations. In *2022 IEEE International Conference on Big Data (Big Data)*, pp. 115–120, 2022. doi: 10.1109/BigData55660.2022.10020385.

Reddy, A., Rossi, R. A., Song, Z., Rao, A., Mai, T., Lipka, N., Wu, G., Koh, E., and Ahmed, N. One-pass algorithms for map inference of nonsymmetric determinantal point processes. In *International Conference on Machine Learning*, pp. 18463–18482. PMLR, 2022a.

Reddy, A., Song, Z., and Zhang, L. Dynamic tensor product regression. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022b. URL `https://openreview.net/forum?id=hUjMhflYvGc`.

Roughgarden, T. *Beyond the Worst-Case Analysis of Algorithms*. Cambridge University Press, 2021.

Rowland, M., Pacchiano, A., and Weller, A. Conditions beyond treewidth for tightness of higher-order lp relaxations. In *Artificial Intelligence and Statistics*, pp. 10–18, 2017.

Savchynskyy, B. Discrete graphical models—an optimization perspective. *Foundations and Trends® in Computer Graphics and Vision*, 11(3-4):160–429, 2019.

Scharstein, D. and Szeliski, R. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1):7–42, 2002.

Sharma, M., Harper, F. M., and Karypis, G. Learning from Sets of Items in Recommender Systems. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 9(4):1–26, 2019.

Shekhovtsov, A. *Exact and partial energy minimization in computer vision*. PhD thesis, Czech Technical University, 2013.

Shekhovtsov, A., Swoboda, P., and Savchynskyy, B. Maximum persistency via iterative relaxed inference in graphical models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(7):1668–1682, 2017.

Sontag, D. *Approximate Inference in Graphical Models using LP Relaxations*. PhD thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 2010.

Sontag, D., Globerson, A., and Jaakkola, T. Introduction to dual decomposition for inference. In *Optimization for Machine Learning*. MIT Press, 2011.

Swoboda, P., Shekhovtsov, A., Kappes, J. H., Schnörr, C., and Savchynskyy, B. Partial optimality by pruning for map-inference with general graphical models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(7), 2016.

Tappen and Freeman. Comparison of graph cuts with belief propagation for stereo, using identical mrf parameters. In *Proceedings Ninth IEEE International Conference on*

Computer Vision, pp. 900–906 vol.2, 2003.

Thapper, J. The power of linear programming for valued csps. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pp. 669–678. IEEE, 2012.

Vazirani, V. V. *Approximation algorithms*, volume 1. Springer, 2001.

Vershynin, R. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

Wainwright, M. J. and Jordan, M. I. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

Wainwright, M. J., Jaakkola, T. S., and Willsky, A. S. Map estimation via agreement on trees: message-passing and linear programming. *IEEE transactions on information theory*, 51(11):3697–3717, 2005.

Weller, A., Rowland, M., and Sontag, D. Tightness of lp relaxations for almost balanced models. In *Artificial Intelligence and Statistics*, pp. 47–55, 2016.

APPENDIX A

Appendix to Chapter 2

A.1. Preliminaries Details

Claim A.1.1. *For Uniform Metric Labeling, we can assume* $c(u,i) \geq 0$ *and* $w(u,v) > 0$ *without loss of generality.*

Proof. For problem instances where some node costs are strictly negative, let $c_{\min}$ be the minimum value among all the node costs. Consider a new problem instance where we keep the edge costs the same, but set $c'(u,i) = c(u,i) + |c_{\min}|$ for all $u \in V$ and $i \in [k]$. This new problem instance has all non-negative node costs, and the optimization problem is equivalent, because we added the same constant for all solutions. This reformulation also does not affect the $(2,1)$-expansion stability or $(2,1,\psi)$-expansion stability of the instance.

Likewise, for problem instances where some edge weights are 0, let E_0 be the set of all edges with 0 edge weight. Consider a new problem instance with $E' = E \setminus E_0$, with $w(u,v)$ unchanged for $(u,v) \in E \setminus E_0$, and identical node costs. The MAP optimization problem remains the same, and the new instance $((V,E'),c,w)$ is equivalent: it has the same MAP solution, and satisfies the stability definitions if and only if the original instance does as well. ■

Claim 2.3.3. *For a given graph G, every solution $x \in L(G)$ that minimizes $\langle \theta, x \rangle$ for some valid objective vector $\theta = (c, w)$ also belongs to $L^*(G)$. Further, all integer solutions in $L(G)$ also belong to $L^*(G)$.*

Proof of Claim 2.3.3. Recall the local LP:

$$\min_x . \sum_{u \in V} \sum_i c(u,i) x_u(i) + \sum_{(u,v) \in E} w(u,v) \sum_{i \neq j} x_{uv}(i,j) \tag{A.1}$$

$$\text{subject to:} \sum_i x_u(i) = 1 \qquad \forall\ u \in V \tag{A.2}$$

$$\sum_i x_{uv}(i,j) = x_v(j) \qquad \forall\ (u,v) \in E,\ j \in [k] \tag{A.3}$$

$$\sum_j x_{uv}(i,j) = x_u(i) \qquad \forall\ (u,v) \in E,\ i \in [k] \tag{A.4}$$

$$x_{uv}(i,j) \in [0,1] \qquad \forall\ (u,v),\ (i,j) \tag{A.5}$$

$$x_u(i) \in [0,1] \qquad \forall\ u,\ i. \tag{A.6}$$

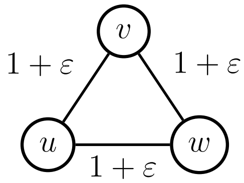
The feasible region defined by the above constraints is $L(G)$. $L^*(G) \subseteq L(G)$ is the set of points that satisfy the additional constraint that $x_{uv}(i,i) = \min(x_u(i), x_v(i))$ for all $(u,v) \in E$ and $i \in [k]$. For any feasible node variable assignments $\{x_u\}$, $L^*(G)$ is not empty: a simple flow argument[*] implies that the constraints equation A.3, equation A.4,

[*] For an edge (u,v), consider the bipartite graph $\tilde{G} = ((U,V),E)$, where $|U| = |V| = k$. We let $x_u(i)$ represent the supply at node i in U, and let $x_v(i)$ represent the demand at node j in V. Because x_u and x_v are both feasible, the total supply equals the total demand. E contains all edges between U and V, so we can send flow from $i \in U$ to $j \in V$ for any (i,j) pair. Let $x_{uv}(i,j)$ represent this flow, and set $x_{uv}(i,i) = \min(x_u(i), x_v(i))$. For every i, this either satisfies the demand at node V_i or exhausts the supply at node U_i. In each case, we can remove that satisfied/exhausted node from the graph. After this choice of $x_u(i,i)$, the total remaining supply equals the total remaining demand ($\sum_i x_u(i) - \min(x_u(i), x_v(i)) = \sum_i x_v(i) - \min(x_u(i), x_v(i))$), all supplies and demands are nonnegative, and the remaining graph $\tilde{G}'$ is a complete bipartite graph (over fewer nodes). This implies that the flow constraints equation A.3, equation A.4, equation A.5 are still feasible.

and equation A.5 are always satisfiable even when we set $x_{uv}(i,i) = \min(x_u(i), x_v(i))$. For all integer feasible solutions in $L(G)$, notice that $x_{uv}(i,j) = 1$ if $x_u(i) = 1$ and $x_v(j) = 1$ or 0 otherwise. Therefore, all integer solutions satisfy this additional constraint. Consider a θ where all edge weights are strictly positive. If x minimizes $\langle \theta, x \rangle$, x must pay the minimum edge cost consistent with its node variables $x_u(i)$. So if we fix the $x_u(i)$ portion of x, we know that the edge variables x_{uv} of x are a solution to:

$$\min_{x \in L(G)} \sum_{(u,v) \in E} w(u,v) \sum_{i \neq j} x_{uv}(i,j).$$

Notice that since we have fixed the node variables $x_u(i)$, there is no interaction between the x_{uv} variables across different edges. So we can minimize this objective by minimizing each individual term $w(u,v) \sum_{i \neq j} x_{uv}(i,j)$. Since $w_{uv} > 0$ for all edges, we need to minimize $\sum_{i \neq j} x_{uv}(i,j)$. Notice that for every edge $(u,v) \in E$, we get that $\sum_i \sum_j x_{uv}(i,j) = 1$ by substituting $x_u(i)$ in constraint A.2 with $\sum_j x_{uv}(i,j)$ from constraint A.4. Therefore $\sum_{i \neq j} x_{uv}(i,j) = 1 - \sum_i x_{uv}(i,i)$. Thus, minimizing $\sum_{i \neq j} x_{uv}(i,j)$ is the same as maximizing $\sum_i x_{uv}(i,i)$. And the maximizing choice for $x_{uv}(i,i) = \min(x_u(i), x_v(i))$ due to constraints A.3 and A.4. ■



Node	**Costs**		
u	.5	∞	∞
v	1	0	∞
w	1	∞	0

Figure A.1. $(2,1)$-expansion stable instance that is not $(2,1)$-stable. In the original instance (shown left), the optimal solution labels each vertex with label 1, for an objective of 2.5. The adversarial $(2,1)$-perturbation for this instance replaces all the edge weights of $1+\varepsilon$ with $(1+\varepsilon)/2$. In this perturbed instance, the optimal solution labels $(u,v,w) \to (1,2,3)$. This has a node cost of 0.5 and an edge cost of $(3+3\varepsilon)/2$, for a total of $2+3\varepsilon/2 < 2.5$. Since the original solution is not optimal in the perturbed instance, this instance is not $(2,1)$-perturbation stable. However, note that the only expansions of the original solution (which had all label 1) that have non-infinite objective are $(u,v,w) \to (1,2,1)$ and $(u,v,w) \to (1,1,3)$. These each have objective $2.5+\varepsilon$, which is strictly greater than the perturbed objective of the original solution. Therefore, this instance is $(2,1)$-expansion stable.

A.2. Expansion Stability details

Claim A.2.1. *An instance (G,w,c) is $(2,1)$-expansion stable iff the MAP solution $\bar{x}$ is strictly better than all its expansions in the adversarial perturbation θ_{adv}. That is, for all $x \in \mathcal{E}_{\bar{x}}$, $\langle \theta_{adv}, x\rangle > \langle \theta_{adv}, \bar{x}\rangle$ where θ_{adv} has the same node costs c but has weights*

$$w_{adv}(u,v) = \begin{cases} \frac{1}{2}w(u,v) & \bar{x}(u) = \bar{x}(v) \\ \\ w(u,v) & \bar{x}(u) \neq \bar{x}(v). \end{cases}$$

Proof. Consider $\theta' = (c,w')$, any valid $(2,1)$-perturbation of $\theta = (c,w)$ i.e., for every edge $(u,v) \in E$, $\frac{w(u,v)}{2} \leq w'(u,v) \leq w(u,v)$. For any valid labeling x, let E_x represent the edges cut by x. Then, for any x which is an expansion of $\bar{x}$ i.e., $x \in \mathcal{E}_{\bar{x}}$,

$$\begin{aligned}
\langle\theta', x\rangle - \langle\theta', \bar{x}\rangle &= \sum_{u \in V} c(u, x(u)) - c(u, \bar{x}(u)) + \sum_{(u,v)\in E_x} w'(u,v) - \sum_{(u,v)\in E_{\bar{x}}} w'(u,v) \\
&= \sum_{u \in V} c(u, x(u)) - c(u, \bar{x}(u)) + \sum_{(u,v)\in E_x \setminus E_{\bar{x}}} w'(u,v) - \sum_{(u,v)\in E_{\bar{x}} \setminus E_x} w'(u,v) \\
&= \langle\theta_{adv}, x\rangle - \langle\theta_{adv}, \bar{x}\rangle + \sum_{(u,v)\in E_x \setminus E_{\bar{x}}} w'(u,v) - w_{adv}(u,v) \\
&\quad + \sum_{(u,v)\in E_{\bar{x}} \setminus E_x} w_{adv} - w'(u,v) \\
&= \langle\theta_{adv}, x\rangle - \langle\theta_{adv}, \bar{x}\rangle + \sum_{(u,v)\in E_x \setminus E_{\bar{x}}} w'(u,v) - \frac{w(u,v)}{2} \\
&\quad + \sum_{(u,v)\in E_{\bar{x}} \setminus E_x} w(u,v) - w'(u,v)
\end{aligned}$$

Since w' is a valid $(2,1)$-perturbation, $w'(u,v) \geq w(u,v)/2$ and $w'(u,v) \leq w(u,v)$. Therefore, for any valid $(2,1)$-perturbation θ', we have

$$\langle\theta', x\rangle - \langle\theta', \bar{x}\rangle \geq \langle\theta_{adv}, x\rangle - \langle\theta_{adv}, \bar{x}\rangle.$$

If the instance is $(2,1)$-expansion stable, then certainly $\langle\theta_{adv}, x\rangle > \langle\theta_{adv}, \bar{x}\rangle$ for all $x \in \mathcal{E}^{\bar{x}}$, since θ_{adv} is a valid $(2,1)$-perturbation of θ. If the instance is not $(2,1)$-expansion stable, there exists a θ' and an $x \in \mathcal{E}^{\bar{x}}$ for which $\langle\theta', x\rangle - \langle\theta', \bar{x}\rangle \leq 0$. But the above inequality then implies that $\langle\theta_{adv}, x\rangle - \langle\theta_{adv}, \bar{x}\rangle \leq 0$ as well. This gives both directions. ■

This claim shows that to check whether an instance is $(2,1)$-expansion stable, it is sufficient to check that the MAP solution is strictly better than all its expansions in the adversarial perturbation θ_{adv}. We don't need to verify that this condition is satisfied in *every* valid $(2,1)$-perturbation. Because the optimal expansion of $\bar{x}$ in the instance with objective θ_{adv} can be computed efficiently, this claim also implies that $(2,1)$-expansion stability can be efficiently checked once the MAP solution $\bar{x}$ is known.

Claim A.2.2. $(2,1)$*-expansion stability is strictly weaker than* $(2,1)$*-perturbation stability.*

Proof. Figure 2.2 gives an instance of uniform metric labeling that is $(2,1)$-expansion stable but not $(2,1)$-perturbation stable. Here, $0 < \varepsilon < 1/3$. ■

Theorem 2.4.2 (Local LP is tight on $(2,1)$-expansion stable instances)**.** *Let $\bar{x}$ and $\hat{x}$ be the MAP and local LP solutions to a* $(2,1)$*-expansion stable instance* (G,c,w)*, respectively. Then $\bar{x} = \hat{x}$ i.e., the local LP is tight on* (G,c,w)*.*

Proof. First, we note that for any $x \in L^*(G)$, the objective value of the local LP can be written in a form that depends only on the node variables x_V. The objective term corresponding to the edges

$$\begin{aligned}\sum_{(u,v)\in E} w(u,v)\sum_{i\neq j} x_{uv}(i,j) &= \sum_{(u,v)\in E} w(u,v)\left(\sum_{i,j} x_{uv}(i,j) - \sum_i x_{uv}(i,i)\right)\\ &= \sum_{(u,v)\in E} w(u,v)\left(1 - \sum_i x_{uv}(i,i)\right)\\ &= \sum_{(u,v)\in E} w(u,v)\left(1 - \sum_i \min(x_u(i), x_v(i))\right)\\ &= \sum_{(u,v)\in E} w(u,v)\left(1 - \sum_i\left(\frac{x_u(i)+x_v(i)}{2} - \frac{|x_u(i)-x_v(i)|}{2}\right)\right)\\ &= \sum_{(u,v)\in E} w(u,v)\left(\frac{1}{2}\sum_i |x_u(i)-x_v(i)|\right)\end{aligned}$$

Here we used the definition of $L^*(G)$ and the facts that $\sum_i x_u(i) = 1$ for all (u,i) and $\sum_j x_{uv}(i,j) = x_u(i)$ for all $(u,v) \in E,\ i \in [k]$.

Thus, for any $x \in L^*(G)$, the objective of the local LP can be written as

$$\sum_{u \in V} \sum_i c(u,i) x_u(i) + \sum_{(u,v) \in E} w(u,v) d(u,v)$$

where $d(u,v) \coloneqq \frac{1}{2} \sum_i |x_u(i) - x_v(i)|$. This is the objective function of another LP relaxation for uniform metric labeling called the "metric LP", which is equivalent to the local LP (Archer, Fakcharoenphol, Harrelson, Krauthgamer, Talwar, and Tardos, 2004). Note that both $\bar{x}$ and $\hat{x}$ are in $L^*(G)$ by Claim 2.3.3. Therefore, the objective function can be written in the above form for both of them.

In the next section, we introduce a rounding algorithm and prove some guarantees for the random solutions h output by it. We then use these guarantees to show an upper bound on the expected cost of these random solutions in a perturbed instance of the problem. Finally, we use this upper bound to prove that $\hat{x} = \bar{x}$.

A.2.1. ε-close rounding:

Given any feasible solution $x \in L(G)$ and a valid labeling $\bar{x}$, we construct a related feasible solution x' which is ε-close to $\bar{x}$ in the ℓ_∞-norm i.e., $\|x' - \bar{x}\|_\infty \leq \varepsilon$:

$$x' = \varepsilon x + (1 - \varepsilon)\bar{x}, \tag{A.7}$$

where $\varepsilon < 1/k$ and we have identified the labeling $\bar{x} : V \to \mathcal{L}$ with its corresponding vertex of the marginal polytope (a vector in $\{0,1\}^{nk+mk^2}$). We consider the following rounding algorithm applied to x', which is a modified version of the ε-close rounding algorithm used in Lang, Sontag, and Vijayaraghavan (2018):

Algorithm 5 ε-close rounding

```
Choose $i \in \{1, \ldots, k\}$ uniformly at random.
Choose $r \in (0, 1/k)$ uniformly at random.
Initialize labeling $h : V \to [k]$.
for each $u \in V$ do
    if $x'_u(i) > r$ then
        Set $h(u) = i$.
    else
        Set $h(u) = \bar{x}(u)$
Return $h$
```

Lemma A.2.3 (Rounding guarantees)**.** *Given any x' constructed using equation A.7, the labeling h output by Algorithm 5 satisfies the following guarantees:*

$$\begin{aligned} \mathbb{P}\{\, h(u) = i \,\} &= x'_u(i) && \forall\ u \in V, i \in [k] \\ \mathbb{P}\{\, h(u) \neq h(v) \,\} &\leq 2d(u, v) && \forall\ (u, v) \in E : \bar{x}(u) = \bar{x}(v) \\ \mathbb{P}\{\, h(u) = h(v) \,\} &= (1 - d(u, v)) && \forall\ (u, v) \in E : \bar{x}(u) \neq \bar{x}(v), \end{aligned}$$

where $d(u, v) = \frac{1}{2} \sum_i |x'_u(i) - x'_v(i)|$ is the edge separation of the constructed feasible point x'.

Proof of Lemma A.2.3 (rounding guarantees). First, fix $u \in V$ and a label $i \neq \bar{x}(u)$. We output $h(u) = i$ precisely when i is chosen and $0 < r < x'_u(i)$, which occurs with probability $\frac{1}{k}\frac{x'_u(i)}{1/k} = x'_u(i)$ (we used here that $x'_u(i) \leq \varepsilon < 1/k$ for all $i \neq \bar{x}(u)$). Now we output $h(u) = \bar{x}(u)$ with probability $1 - \sum_{j \neq \bar{x}(u)} \mathbb{P}\{\, h(u) = j \,\} = 1 - \sum_{j \neq \bar{x}(u)} x'_u(j) = x'_u(\bar{x}(u))$, since $\sum_i x'_u(i) = 1$. This proves the first guarantee.

For the second, consider an edge (u, v) not cut by $\bar{x}$, so $\bar{x}(u) = \bar{x}(v)$. Then (u, v) is cut by h when some $i \neq \bar{x}(u)$ is chosen and r falls between $x'_u(i)$ and $x'_v(i)$. This occurs with probability

$$\frac{1}{k}\sum_{i \neq \bar{x}(u)} \frac{\max(x'_u(i), x'_v(i)) - \min(x'_u(i), x'_v(i))}{1/k} = \sum_{i \neq \bar{x}(u)} |x'_u(i) - x'_v(i)| \leq 2d(u, v).$$

Finally, consider an edge (u, v) cut by $\bar{x}$, so that $\bar{x}(u) \neq \bar{x}(v)$. Here $h(u) = h(v)$ if some i, r are chosen with $r < \min(x'_u(i), x'_v(i))$. We have $r < \min(x'_u(i), x'_v(i))$ with probability $\frac{\min(x'_u(i), x'_v(i))}{1/k}$. Note that this is still valid if $i = \bar{x}(u)$ or $i = \bar{x}(v)$, since only one of those equalities can hold. So we get

$$\frac{1}{k}\sum_i \frac{\min(x'_u(i), x'_v(i))}{1/k} = \frac{1}{2}\left(\sum_i x'_u(i) + x'_v(i) - |x'_u(i) - x'_v(i)|\right) = 1 - d(u, v),$$

where we used again that $\sum_i x'_u(i) = 1$. ■

Given these rounding guarantees, we can relate the expected cost difference between h and $\bar{x}$ in a perturbation of the original instance to the cost difference between x and $\bar{x}$ in the original instance. We are only interested in the case when $x \in L^*(G)$ and so the objective function $f(x) = \sum_{u \in V}\sum_i c(u, i)x_u(i) + \sum_{(u,v) \in E} w(u, v)d(u, v)$.

A.2.2. Using the rounding guarantees

Lemma A.2.4. *Given an integer solution $\bar{x}$, a feasible LP solution $x \in L^*(G)$, and a random output h of Algorithm 5 on an input $x' = \varepsilon x + (1-\varepsilon)\bar{x}$, define*

$$w'(u,v) = \begin{cases} \frac{1}{2} w(u,v) & \bar{x}(u) = \bar{x}(v) \\ w(u,v) & \bar{x}(u) \neq \bar{x}(v) \end{cases}$$

and let $f'(y) = \sum_{u \in V} \sum_i c(u,i) y_u(i) + \sum_{(u,v) \in E} w'(u,v) d(y,u,v)$ be the objective in the instance with the original costs, but using weights w'. Here $d(y,u,v) = \frac{1}{2} \sum_i |y_u(i) - y_v(i)|$. Let $A_h \coloneqq f'(h) - f'(\bar{x})$ be the difference in this perturbed objective between h and $\bar{x}$. Then,

$$\mathbb{E}[A_h] = \mathbb{E}[f'(h) - f'(\bar{x})] \leq \varepsilon \cdot (f(x) - f(\bar{x}))\,.$$

Proof.

$$\begin{aligned}
\mathbb{E}[f'(h) - f'(\bar{x})] &= \sum_{u \in V} \sum_{i} c(u,i)\mathbb{P}\{\, h(u) = i \,\} - \sum_{u \in V} c(u, \bar{x}(u)) \\
&\quad + \sum_{uv : \bar{x}(u) = \bar{x}(v)} w'(u,v)\mathbb{P}\{\, h(u) \neq h(v) \,\} \\
&\quad - \sum_{uv : \bar{x}(u) \neq \bar{x}(v)} w'(u,v)\mathbb{P}\{\, h(u) = h(v) \,\} \\
&= \sum_{u} \sum_{i} c(u,i)x'_u(i) - \sum_{u} c(u, \bar{x}(u)) + \sum_{uv : \bar{x}(u) = \bar{x}(v)} 2w'(u,v)d(x', u, v) \\
&\quad - \sum_{uv : \bar{x}(u) \neq \bar{x}(v)} w'(u,v)(1 - d(x', u, v)) \\
&= \sum_{u} \sum_{i} c(u,i)x'_u(i) - \sum_{u} c(u, \bar{x}(u)) + \sum_{uv \in E} w(u,v)d(x', u, v) \\
&\quad - \sum_{uv : \bar{x}(u) \neq \bar{x}(v)} w(u,v) \\
&= f(x') - f(\bar{x}).
\end{aligned}$$

where the second-to-last equality used the definition of w' (note that w' is identical to the worst-case perturbation w_{adv} for $\bar{x}$). Because f is convex (in particular, $d(x, u, v)$ is convex in x), we have $f(x') \leq \varepsilon f(x) + (1 - \varepsilon) f(\bar{x})$. Therefore,

$$\mathbb{E}[f'(h) - f'(\bar{x})] \leq \varepsilon f(x) + (1 - \varepsilon) f(\bar{x}) - f(\bar{x}) = \varepsilon(f(x) - f(\bar{x})),$$

which is what we wanted. ■

A.2.3. Final proof of Theorem 2.4.2:

Suppose the local LP solution $\hat{x}$ is not the same as the MAP solution $\bar{x}$ i.e., $\hat{x} \neq \bar{x}$. Consider $x' = \varepsilon\hat{x} + (1-\varepsilon)\bar{x}$ where $0 < \varepsilon < 1/k$ (see equation equation A.7). Let h be the random integer solution output by using Algorithm 5 on x'. By Lemma A.2.4, we have

$$\mathbb{E}[f'(h) - f'(\bar{x})] \leq \varepsilon \cdot (f(\hat{x}) - f(\bar{x}))$$

We note that any solution h that we get from rounding x' is either $\bar{x}$ or an expansion move of $\bar{x}$. This is because we pick only a single label i in step 1 of Algorithm 5 and label all vertices u either i or $\bar{x}(u)$. Therefore, for the i picked in step 1, h is an i-expansion of $\bar{x}$ if $h \neq \bar{x}$.

$$\begin{aligned}\mathbb{E}[f'(h) - f'(\bar{x})] &= \mathbb{E}[f'(h) - f'(\bar{x}) | h \neq \bar{x}]\ \mathbb{P}[h \neq \bar{x}] + \mathbb{E}[f'(h) - f'(\bar{x}) | h = \bar{x}]\ \mathbb{P}[h = \bar{x}] \\ &= \mathbb{E}[f'(h) - f'(\bar{x}) | h \neq \bar{x}]\ \mathbb{P}[h \neq \bar{x}]\end{aligned}$$

Since (G, c, w) is a $(2,1)$-expansion stable instance, we know that $f'(h) > f'(\bar{x})$ when $h \neq \bar{x}$ since all h in the support of the rounding (other than $\bar{x}$) are expansion moves of $\bar{x}$ and we get f' by a valid $(2,1)$-perturbation of (G, c, w). Therefore, $\mathbb{E}[f'(h) - f'(\bar{x}) | h \neq \bar{x}] > 0$. We also have that $\mathbb{P}[h \neq \bar{x}] > 0$ since we assumed that $\hat{x} \neq \bar{x}$. Therefore, $\mathbb{E}[f'(h) - f'(\bar{x})] > 0$. But we know that $f(\hat{x}) - f(\bar{x}) \leq 0$ since $\hat{x}$ is the minimizer of $f(x)$ among all feasible $x \in L(G)$. So Lemma A.2.4 implies $\mathbb{E}[f'(h) - f'(\bar{x})] \leq 0$. Thus we have a contradiction and so the local LP solution $\hat{x}$ has to be the same as the MAP solution $\bar{x}$.

■

A.3. Stability and Curvature around MAP solution details

Theorem 2.5.2. *Let (G, c, w) be a $(2, 1, \psi)$-expansion stable instance with MAP solution $\bar{x}$. Let $\theta = (c, w)$. Then for any $x \in L^*(G)$, the recovery error (see Def. 2.3.1) satisfies*

$$\frac{1}{2}\|x - \bar{x}\|_1 := \frac{1}{2}\|x_V - \bar{x}_V\|_1 \leq \frac{1}{\psi}|\langle\theta, x\rangle - \langle\theta, \bar{x}\rangle|. \tag{2.2}$$

Here, we provide two proofs for this theorem, one deals directly with the local LP relaxation and the other uses the dual of the relaxation. The dual proof is more general than the primal proof as it works for all $x \in L(G)$, not just for those in $L^*(G)$.

A.3.1. Primal-based proof

Proof. For any $x \in L^*(G)$, consider a feasible solution x' which is ε-close to $\bar{x}$ constructed using Equation A.7 i.e., $x' = \varepsilon x + (1 - \varepsilon)\bar{x}$. Let h be the random solution output by Algorithm 5 on x'.

Lemma A.3.1 (Bound for $\mathbb{E}[B_h]$)**.** *For any h in the support of the rounding of $x' = \varepsilon x + (1 - \varepsilon)\bar{x}$, let us define B_h to be the number of vertices which it labels differently from $\bar{x}$. In other words, it is the number of vertices which are misclassified by h i.e., $B_h := \sum_{u \in V} \mathbb{1}[h(u) \neq \bar{x}(u)]$. Then,*

$$\mathbb{E}[B_h] = \varepsilon \sum_{u \in V} \frac{1}{2}\|x_u - \bar{x}_u\|_1$$

Proof.

$$\begin{aligned}\mathbb{E}[B_h] &= \sum_{u\in V} \mathbb{E}[\mathbb{1}[h(u) \neq \bar{x}(u)]] = \sum_{u\in V} \mathbb{P}\left\{ h(u) \neq \bar{x}(u) \right\} = \sum_{u\in V} 1 - \mathbb{P}\left\{ h(u) = \bar{x}(u) \right\} \\ &= \sum_{u\in V} 1 - x'_u(\bar{x}(u)) = \sum_{u\in V} 1 - (\varepsilon x_u(\bar{x}(u)) + (1-\varepsilon)) = \sum_{u\in V} \varepsilon\,(1 - x_u(\bar{x}(u))) \\ &= \varepsilon \sum_{u\in V} \frac{1}{2}\left(1 - x_u(\bar{x}(u)) + \sum_{i\neq\bar{x}(u)} x_u(i)\right) = \varepsilon \sum_{u\in V} \frac{1}{2}\|x_u - \bar{x}_u\|_1\end{aligned}$$

Here, we used the fact that for all $u \in V, \bar{x}_u(\bar{x}(u)) = 1$ and $\bar{x}_u(i) = 0\ \forall\ i \neq \bar{x}(u)$ and since x is a feasible solution to the LP, it satisfies $\sum_{i\neq\bar{x}(u)} x_u(i) = 1 - x_u(\bar{x}(u))$ for all $u \in V$. ■

Lemma A.3.2 (Lower bound for A_h using $(2,1,\psi)$-expansion stability)**.** *If (G, w, c) is a $(2,1,\psi)$-expansion stable instance, then for any h in the support of the rounding of $x' = \varepsilon x + (1-\varepsilon)\bar{x}$,*

$$A_h \geq \psi \cdot B_h$$

where $A_h = f'(h) - f'(\bar{x})$ and f' is the objective in the instance (G, c, w') where w' is the worst $(2,1)$ perturbation for $\bar{x}$ i.e.,

$$w'(u,v) = \begin{cases} \frac{1}{2}w(u,v) & \bar{x}(u) = \bar{x}(v) \\ w(u,v) & \bar{x}(u) \neq \bar{x}(v) \end{cases}$$

Proof. Note that A_h here is the same as the one defined for Lemma A.2.4. Since the instance (G, c, w) is $(2,1,\psi)$-expansion stable, we know that (G, c', w) should be $(2,1)$-expansion stable for all c' such that $c \leq c' \leq c + \psi \cdot \mathbf{1}$. Consider the worst c' for $\bar{x}$ i.e.,

$c'(u,i) = \begin{cases} c(u,i) + \psi & i = \bar{x}(u) \\ c(u,i) & i \neq \bar{x}(u) \end{cases}$. Let f'' be the objective in the instance (G, c', w'). As discussed in section A.2.3, we know that any $h \neq \bar{x}$ in the support of the rounding is an expansion move of $\bar{x}$. Therefore, for any $h \neq \bar{x}$ in the support of the rounding of x',

$$
\begin{aligned}
& f''(h) - f''(\bar{x}) > 0 \\
\implies & \sum_{u \in V} c'(u, h(u)) - c'(u, \bar{x}(u)) + \sum_{(u,v):h(u) \neq h(v)} w'(u,v) - \sum_{(u,v):\bar{x}(u) \neq \bar{x}(v)} w'(u,v) > 0 \\
\implies & \sum_{u \in V} c(u, h(u)) - (c(u, \bar{x}(u)) + \psi \cdot \mathbb{1}[h(u) \neq \bar{x}(u)]) + \sum_{(u,v):h(u) \neq h(v)} w'(u,v) \\
& \quad - \sum_{(u,v):\bar{x}(u) \neq \bar{x}(v)} w'(u,v) > 0 \\
\implies & f'(h) - f'(g) > \psi \cdot \sum_{u \in V} \mathbb{1}[h(u) \neq \bar{x}(u)] \implies A_h > \psi \cdot B_h.
\end{aligned}
$$

This is true for all $h \neq \bar{x}$ in the support of the rounding of x'. When $h = \bar{x}$, we have $A_h = B_h = 0$. Therefore for all h in the support of the rounding of x', we have that $A_h \geq \psi \cdot B_h$. ■

A.3.2. Final proof of Theorem 2.5.2:

We use the Lemmas A.2.4(upper bound for $\mathbb{E}[A_h]$), A.3.2(lower bound for A_h), and A.3.1(bound for $\mathbb{E}[B_h]$) to prove Theorem 2.5.2. For all h in the support of rounding of x, $A_h \geq \psi \cdot B_h$. Also,

$$\mathbb{E}[A_h] \leq \varepsilon \left(f(x) - f(\bar{x}) \right), \ \mathbb{E}[B_h] = \varepsilon \sum_{u \in V} \frac{1}{2} \|x_u - \bar{x}_u\|_1$$

Suppose that $\|x - \bar{x}\|_1 > \tau \cdot (f(x) - f(\bar{x}))$. Then,

$$\frac{\mathbb{E}[A_h]}{\mathbb{E}[B_h]} \leq \frac{f(x) - f(\bar{x})}{\sum_{u \in V} \frac{1}{2}\|x_u - \bar{x}_u\|_1} < \frac{2}{\tau}$$

But since $A_h \geq \psi \cdot B_h$ for every h in the rounding of x, we get that $\frac{\mathbb{E}[A_h]}{\mathbb{E}[B_h]} \geq \psi$. Setting $\tau = \frac{2}{\psi}$, we get a contradiction and thus we get,

$$\frac{1}{2}\|x - \bar{x}\|_1 \leq \frac{1}{\psi} \cdot (f(x) - f(\bar{x})) = \frac{1}{\psi} \cdot (\langle \theta, x \rangle - \langle \theta, \bar{x} \rangle)$$

■

Corollary 2.5.3 (LP solution is good if there is a nearby stable instance)**.** *Let $\hat{x}^{MAP}$ and $\hat{x}$ be the MAP and local LP solutions to an observed instance $(G, \hat{c}, \hat{w})$. Also, let $\bar{x}$ be the MAP solution for a latent $(2, 1, \psi)$-expansion stable instance $(G, \bar{c}, \bar{w})$. If $\hat{\theta} = (\hat{c}, \hat{w})$ and $\bar{\theta} = (\bar{c}, \bar{w})$,*

$$\frac{1}{2}\|\hat{x}_V - \hat{x}_V^{MAP}\|_1 \leq \frac{2d(\hat{\theta}, \bar{\theta})}{\psi} + \frac{1}{2}\|\hat{x}_V^{MAP} - \bar{x}_V\|_1.$$

Proof of Corollary 2.5.3. First, we note that for the nearby stable instance, the MAP and the local LP solutions are the same due to Theorem 2.4.2. Therefore, for any feasible solution $x \in L(G)$, $\langle \bar{\theta}, x \rangle \geq \langle \bar{\theta}, \bar{x} \rangle$. In particular, this implies that $\langle \bar{\theta}, \hat{x} \rangle \geq \langle \bar{\theta}, \bar{x} \rangle$ and $\langle \bar{\theta}, \hat{x}^{MAP} \rangle \geq \langle \bar{\theta}, \bar{x} \rangle$ since $\hat{x}, \hat{x}^{MAP}$ are also feasible solutions. Remember that we defined $d(\hat{\theta}, \bar{\theta}) := \sup_{x \in L^*(G)} |\langle \hat{\theta}, x \rangle - \langle \bar{\theta}, x \rangle|$. Therefore,

$$\langle \bar{\theta}, \hat{x} \rangle \leq \langle \hat{\theta}, \hat{x} \rangle + d(\hat{\theta}, \bar{\theta}) \leq \langle \hat{\theta}, \bar{x} \rangle + d(\hat{\theta}, \bar{\theta}) \leq \langle \bar{\theta}, \bar{x} \rangle + 2d(\hat{\theta}, \bar{\theta}).$$

The first and third inequalities hold due to the definition of $d(\hat{\theta},\bar{\theta})$. The second inequality follows from the fact that $\hat{x}$ is the minimizer for $\langle\hat{\theta},x\rangle$ among $x\in L(G)$. Thus, $0\leq\langle\bar{\theta},\hat{x}\rangle-\langle\bar{\theta},\bar{x}\rangle\leq 2d(\hat{\theta},\bar{\theta})$. From Theorem 2.5.2, we get $\frac{1}{2}\|\hat{x}_V-\bar{x}_V\|_1\leq\frac{2d(\hat{\theta},\bar{\theta})}{\psi}$. Thus,

$$\begin{aligned}\frac{1}{2}\|\hat{x}_V-\hat{x}_V^{MAP}\|_1 &\leq \frac{1}{2}\|\hat{x}_V-\bar{x}_V\|_1+\frac{1}{2}\|\hat{x}_V^{MAP}-\bar{x}_V\|_1\\ &\leq \frac{2d(\hat{\theta},\bar{\theta})}{\psi}+\frac{1}{2}\|\hat{x}_V^{MAP}-\bar{x}_V\|_1.\end{aligned}$$

■

A.3.3. Dual-based proof

Here we provide an alternate proof of the curvature result using the dual of the local LP relaxation. First, we show that the curvature bound is related to the *dual margin* of the instance. Then we show that $(2,1,\psi)$-expansion stability implies that the dual margin is at least ψ. Throughout this section, we assume the local LP solution $\hat{x}$ is unique and integral (as guaranteed, for example, by $(2,1)$-expansion stability), so $\hat{x}=\bar{x}$.

Relaxing the local LP's marginalization constraints in both directions for each edge, we obtain the following Lagrangian for the local LP:

$$L(\delta,x)=\sum_u\sum_i\left(\theta_u(i)+\sum_{v\in N(u)}\delta_{uv}(i)\right)x_u(i)+\sum_{uv}\sum_{ij}\left(\theta_{uv}(i,j)-\delta_{uv}(i)-\delta_{vu}(j)\right)x_{uv}(i,j)$$

where each x_u is constrained to be in the $(k-1)$-dimensional simplex, and each x_{uv} the k^2-1-dimensional simplex (i.e., the normalization constraints remain). There are no constraints on the dual variables δ. Observe that for any δ and any primal-feasible x, $L(\delta,x)=\langle\theta,x\rangle$. This gives rise to the *reparametrization* view: for a fixed δ, define $\theta_u^{\delta}(i)=$

$\theta_u(i) + \sum_{v \in N(u)} \delta_{uv}(i)$, and $\theta^\delta_{uv}(i,j) = \theta_{uv}(i,j) - \delta_{uv}(i) - \delta_{vu}(j)$. Then $L(\delta, x) = \langle \theta^\delta, x \rangle$. This will allow us to define equivalent primal problems with simpler structure than the original. $L(\delta, x)$ also gives the dual function:

$$D(\delta) = \min_x L(\delta, x) = \sum_u \min_i \left(\theta_u(i) + \sum_{v \in N(u)} \delta_{uv}(i) \right) + \sum_{uv} \min_{i,j} \left(\theta_{uv}(i,j) - \delta_{uv}(i) - \delta_{vu}(j) \right).$$

A dual point δ is a dual *solution* if $\delta \in \operatorname{argmax}_{\delta'} D(\delta')$. Theorem 2.4.2 implies that the local LP has a unique, integral solution when the instance is $(2, 1, \psi)$-expansion stable. Sontag, Globerson, and Jaakkola (2011, Theorem 1.3) show that this implies the existence of a dual solution δ^* that is *locally decodable* at all nodes u: for each u, $\operatorname{argmin}_i \theta^{\delta^*}_u(i)$ is unique, and moreover, the edge and node dual subproblems agree:

$$\left(\operatorname*{argmin}_i \theta^{\delta^*}_u(i), \operatorname*{argmin}_j \theta^{\delta^*}_v(j) \right) \in \operatorname*{argmin}_{i,j} \theta^{\delta^*}_{uv}(i,j). \tag{A.8}$$

In this case, the primal solution defined by "decoding" δ^*, $x(u) = \operatorname{argmin}_i \theta^{\delta^*}_u(i)$, is the MAP solution (Sontag, Globerson, and Jaakkola, 2011).

For locally decodable δ^*, we define the *node margin* $\psi_u(\delta^*)$ at a node u as:

$$\psi_u(\delta^*) = \min_{i \neq \operatorname{argmin}_j \theta^{\delta^*}(j)} \theta^{\delta^*}(i) - \min_j \theta^{\delta^*}(j).$$

This is the difference between the optimal reparametrized node cost at u and the next-smallest cost. Local decodability of δ is the property that $\psi_u(\delta) > 0$ for every u.

Together with equation A.8, the following lemma implies that we need only consider locally decodable dual solutions where the optimal primal solution pays zero edge cost.

Lemma A.3.3 (Dual edge "removal")**.** *Given a locally decodable dual solution δ, we can transform it to a locally decodable dual solution δ' that satisfies $\min_{i,j} \theta^{\delta'}_{uv}(i,j) = 0$ and has the same (additive) margin at every node.*

Proof. Fix an edge (u, v), and consider any pair i^*, j^* in $\text{argmin}_{i,j}\, \theta^{\delta}_{uv}(i,j)$. Put $\theta^{\delta}_{uv}(i^*, j^*) = \theta_{uv}(i^*, j^*) + \varepsilon$ for $\varepsilon \in \mathbb{R}$. Now define $\delta'_{uv}(i) = \delta_{uv}(i) - \varepsilon$ for all i (or, equivalently, $\delta'_{vu}(j) = \delta_{vu}(j) - \varepsilon$ for all j). Because we changed $\theta^{\delta}_{u}(i)$ by a constant for each i, local decodability is preserved and the additive *margin of local decodability* is not changed. We incurred a change of $+\varepsilon$ in the dual objective of δ from the edge term $\min_{i,j} \theta^{\delta'}_{uv}(i,j)$, and a $-\varepsilon$ in the objective from the decrease in the node term $\min_i \theta^{\delta'}_{u}(i)$, so δ' is still optimal. We can repeat this process for every edge (u, v). ■

Lemma A.3.3 implies that when (x^*, δ^*) is a pair of primal/dual optima and δ^* is locally decodable, we can assume that $L(x^*, \delta^*) = \sum_u \theta^{\delta^*}_u(x^*_u)$, where we overload notation to define x^*_u to be the label for which $x^*_u(i) = 1$. That is, the primal optimum pays no edge cost in the problem reparametrized by the dual opt δ^*. Finally, Lemma A.3.3 implies that we can always assume that $\theta^{\delta^*}_{uv}(i,j) \geq 0$ for all $(u, v),\ (i, j)$. Therefore, if there is any locally decodable dual solution, and the primal LP solution is integral and unique, we may assume there exists a locally decodable dual solution δ such that $\bar{x}(u) = \text{argmin}_i\, \theta^{\delta}(i)$, $(\bar{x}(u), \bar{x}(v)) \in \text{argmin}_{i,j}\, \theta^{\delta}_{uv}(i,j)$, $\theta^{\delta}_{uv}(\bar{x}(u), \bar{x}(v)) = 0$, and $\theta^{\delta}_{uv}(i,j) \geq 0$.

Lemma A.3.4 (Dual margin implies curvature around $\bar{x}$)**.** *For an instance with objective θ and MAP solution $\bar{x}$, assume there exists a locally decodable dual solution δ such that $\bar{x}(u) = \text{argmin}_i\, \theta^{\delta}(i)$, $(\bar{x}(u), \bar{x}(v)) \in \text{argmin}_{i,j}\, \theta^{\delta}_{uv}(i,j)$, $\theta^{\delta}_{uv}(\bar{x}(u), \bar{x}(v)) = 0$, and $\theta^{\delta}_{uv}(i,j) \geq 0$. Additionally, let $\psi(\delta) = \min_u \psi_u(\delta)$ be the smallest node margin. Note that*

$\psi(\delta) > 0$ *because* δ *is locally decodable. Then for any* $x \in L(G)$*,*

$$\frac{1}{2}||x_V - \bar{x}_V||_1 \leq \frac{\langle \theta, x - \bar{x} \rangle}{\psi(\delta)}$$

Proof. Let $\Delta = \langle \theta, x - \bar{x} \rangle$. Since x and $\bar{x}$ are both primal-feasible, we have $L(x, \delta) = \langle \theta, x \rangle$ and $L(\bar{x}, \delta) = \langle \theta, \bar{x} \rangle$. Therefore,

$$L(x, \delta) = L(\bar{x}, \delta) + \Delta. \tag{A.9}$$

Because $\theta^\delta(\bar{x}(u), \bar{x}(v)) = 0$ for all (u, v), we have

$$L(\bar{x}, \delta) = \sum_u \theta_u^\delta(\bar{x}(u)).$$

Additionally, because $\theta_{uv}^\delta(i, j) \geq 0$,

$$L(x, \delta) = \sum_u \sum_i \theta_u^\delta(i) x_u(i) + \sum_{uv} \sum_{ij} \theta_{uv}^\delta(i, j) x_{uv}(i, j) \geq \sum_u \sum_i \theta_u^\delta(i) x_u(i).$$

Combining the above two inequalities with equation A.9 gives:

$$\sum_u \sum_i \theta_u^\delta(i) x_u(i) \leq \sum_u \theta_u^\delta(\bar{x}(u)) + \Delta \tag{A.10}$$

Because δ is locally decodable to $\bar{x}$, and the smallest node margin is equal to $\psi(\delta)$, we have that for every u, $\theta_u^\delta(\bar{x}(u)) + \psi(\delta) \leq \theta_u^\delta(i)$ for all $i \neq \bar{x}(u)$. The margin condition implies:

$$\sum_u \theta_u^\delta(\bar{x}(u)) x_u(\bar{x}(u)) + \sum_u \sum_{i \neq \bar{x}(u)} (\theta_u^\delta(\bar{x}(u)) + \psi(\delta)) x_u(i) < \sum_u \sum_i \theta_u^\delta(i) x_u(i),$$

and simplifying using $\sum_i x_u(i) = 1$ gives:

$$\sum_u \theta_u^\delta(\bar{x}(u)) + \psi(\delta) \sum_u \sum_{i \neq \bar{x}(u)} x_u(i) < \sum_u \sum_i \theta_u^\delta(i) x_u(i).$$

Plugging in to equation A.10 gives:

$$\sum_u \sum_{i \neq \bar{x}(u)} x_u(i) < \frac{\Delta}{\psi(\delta)}.$$

The left-hand-side is precisely $||x_V - \bar{x}_V||_1/2$. ■

Now we show that $(2, 1, \psi)$-expansion stability implies that there exists a locally decodable dual solution δ with dual margin $\psi(\delta) \geq \psi$.

Lemma A.3.5 ($(2, 1, \psi)$-expansion stability gives a lower bound on dual margin)**.** *Let (G, c, w) be a $(2, 1, \psi)$-expansion stable instance with $\psi > 0$. Then there exists a locally decodable dual solution δ with dual margin $\psi(\delta) \geq \psi$.*

Proof. Define new costs c_ψ as

$$c_\psi(u, i) = \begin{cases} c(u, i) + \psi & \bar{x}(u) = i \\ c(u, i) & \text{otherwise.} \end{cases}$$

By definition, the instance (G, c_ψ, w) is $(2, 1)$-expansion stable (see Definition 2.4.1). Theorem 2.4.2 implies the pairwise LP solution is unique and integral on (G, c_ψ, w). This implies there exists a dual solution δ^0 that is locally decodable to $\bar{x}$. The only guarantee on the dual margin of δ^0 is that $\psi(\delta^0) > 0$. But note that δ^0 is also an optimal dual solution for (G, c, w), since its objective in that instance is the same as the objective of $\bar{x}$. But in that instance, the dual margin at every node is at least ψ, because $c_\psi(u, \bar{x}(u)) - c(u, \bar{x}(u)) = \psi$. So $\psi(\delta) \geq \psi$. ■

These two lemmas directly imply Theorem 2.5.2. This dual proof is slightly more general than the primal proof, since the curvature result applies to any $x \in L(G)$.

A.4. Generative model details

Definition A.4.1 (sub-Gaussians and (b, σ)-truncated sub-Gaussians)**.** Suppose $b \in \mathbb{R}, \sigma \in \mathbb{R}_+$. A random variable X with mean μ is sub-Gaussian with parameter σ if and only if $\mathbb{E}[e^{\lambda(X-\mu)}] \leq \exp(\lambda^2\sigma^2/2)$ for all $\lambda \in \mathbb{R}$. The random variable X is (b, σ)-truncated sub-Gaussian if and only if X is supported in (b, ∞) and X is sub-Gaussian with parameter σ.

We remark that the above definition captures many well-studied families of bounded random variables e.g., Rademacher distributions, uniform distributions on an interval etc. We remark that a bounded random variable supported on $[-M, M]$ is also sub-Gaussian with parameter M. However in our setting, it needs to be truncated only on negative side, and the bound M will be much larger than the variance parameter σ; the bound is solely to ensure non-negativity of edge costs. A canonical example to keep in mind is a truncated Gaussian distribution.We use the following standard large deviations bound for sums of sub-Gaussian random variables (for details, refer to Thm 2.6.2 from Vershynin (2018)). Given independent r.v.s $X_1, X_2, \ldots, X_n$, with X_i drawn from a sub-Gaussian with parameter σ_i we have for $\mu = \sum_{i=1}^n \mathbb{E}[X_i]$ and $\sigma^2 = \sum_{i=1}^n \sigma_i^2$,

$$\mathbb{P}\Big[\Big|\sum_{i=1}^n X_i - \mu\Big| \geq t\Big] \leq 2\exp\Big(-\frac{t^2}{2\sigma^2}\Big). \tag{A.11}$$

Lemma 2.6.1 ($d(\hat{\theta}, \bar{\theta})$ is small w.h.p.)**.** *There exists a universal constant $c < 1$ such that for any instance in the above model, with probability at least $1 - o(1)$,*

$$\sup_{x \in L^*(G)} |\langle \hat{\theta}, x \rangle - \langle \bar{\theta}, x \rangle| \leq c\sqrt{nk}\sqrt{\sum_{u,i} \sigma_{u,i}^2 + \frac{k^2}{4} \sum_{uv} \gamma_{u,v}^2}$$

Proof. As discussed in section A.2, for any $x \in L^*(G)$, the objective of the local LP can be written as

$$\sum_{u \in V} \sum_i c(u,i) x_u(i) + \sum_{(u,v) \in E} w(u,v) d(u,v)$$

where $d(u,v) \coloneqq \frac{1}{2} \sum_i |x_u(i) - x_v(i)|$. Let $\hat{f}(x) \coloneqq \langle \hat{\theta}, x \rangle$, $\bar{f}(x) \coloneqq \langle \bar{\theta}, x \rangle$. Then,

$$|\langle \hat{\theta}, x \rangle - \langle \bar{\theta}, x \rangle| = |\hat{f}(x) - \bar{f}(x)| = \left| \sum_{u \in V} \sum_{i \in L} \tilde{c}(u,i) x_u(i) + \sum_{(u,v) \in E} \tilde{w}(u,v) d(u,v) \right|$$

For any feasible LP solution x, consider the following rounding algorithm $\mathcal{R}$:

Then, we have

$$\begin{aligned}
&\mathbb{E}[\hat{f}(\mathcal{R}(x)) - \bar{f}(\mathcal{R}(x))] \\
&= \sum_{u\in V}\sum_{i\in L}\tilde{c}(u,i)\mathbb{E}[\mathbb{1}[\mathcal{R}(x)_u(i)=1]] + \sum_{(u,v)\in E}\frac{\tilde{w}(u,v)}{2}\sum_i \mathbb{E}[\mathbb{1}[\mathcal{R}(x)_u(i) \neq \mathcal{R}(x)_v(i)]] \\
&= \sum_{u\in V}\sum_{i\in L}\tilde{c}(u,i)\mathbb{P}[x_u(i) > r_i] \\
&\quad + \sum_{(u,v)\in E}\frac{\tilde{w}(u,v)}{2}\sum_i \mathbb{P}[\min(x_u(i),x_v(i)) \le r_i < \max(x_u(i),x_v(i))] \\
&= \sum_{u\in V}\sum_{i\in L}\tilde{c}(u,i)x_u(i) + \sum_{(u,v)\in E}\frac{\tilde{w}(u,v)}{2}\sum_i |x_u(i)-x_v(i)| \\
&= \sum_{u\in V}\sum_{i\in L}\tilde{c}(u,i)x_u(i) + \sum_{(u,v)\in E}\tilde{w}(u,v)d(u,v) = \hat{f}(x) - \bar{f}(x)
\end{aligned}$$

Therefore,

$$\sup_{x\in L^*(G)}|\hat{f}(x)-\bar{f}(x)| = \sup_{x\in L^*(G)}|\mathbb{E}[\hat{f}(\mathcal{R}(x))-\bar{f}(\mathcal{R}(x))]| \le \sup_{\hat{x}_V\in\{0,1\}^{nk}}|\hat{f}(\hat{x}_V)-\bar{f}(\hat{x}_V)|$$

Note that for all $x \in L^*(G)$, $\hat{f}(x)$ and $\bar{f}(x)$ only depend on the portion of x restricted to the vertices i.e., x_V. This is why we only need to look at $\hat{x}_v \in \{0,1\}^{nk}$ for the last inequality.

For any fixed $\hat{x}_V \in \{0,1\}^{nk}$, since $\tilde{w}(u,v), \tilde{c}(u,i)$ are all mean 0 and sub-Gaussian with parameters $\gamma_{u,v}, \sigma_{u,i}$, we have for any $t > 0$,

$$\mathbb{P}\left[|\hat{f}(\hat{x}_V) - \bar{f}(\hat{x})| > t\right] \le 2\exp\left(\frac{-t^2}{2\left(\sum_{u,i}\sigma_{u,i}^2 + k^2/4\sum_{uv}\gamma_{u,v}^2\right)}\right)$$

Taking $t = c\sqrt{nk}\sqrt{\sum_{u,i}\sigma_{u,i}^2 + k^2/4\sum_{uv}\gamma_{u,v}^2}$, we get that for any fixed $\hat{x}_V \in \{0,1\}^{nk}$,

$$\mathbb{P}\left[|\hat{f}(\hat{x}) - \bar{f}(\hat{x})| > t\right] \leq 2\exp\left(-c^2 nk\right)$$

Taking a union bound over $\{0,1\}^{nk}$, we get that

$$\mathbb{P}\left[\sup_{\hat{x}_V \in \{0,1\}^{nk}} |\hat{f}(\hat{x}) - \bar{f}(\hat{x})| > t\right] \leq 2\exp\left(nk\left(\log 2 - c^2\right)\right)$$

■

Here, c needs to be greater than $\sqrt{\ln 2} \approx 0.83$ to get a high probability guarantee.

Corollary 2.6.3 (MAP solution recovery for regular graphs)**.** *Suppose we have a d-regular graph G with $\gamma_{u,v}^2 = \gamma^2$ for all edges (u,v), and $\sigma_{u,i}^2 = \sigma^2$ for all vertex-label pairs (u,i). Also, suppose only a fraction ρ of the vertices and η of the edges are subject to the noise. With high probability over the random noise,*

$$\frac{\|\hat{x}_V - \bar{x_V}\|_1}{2n} \leq \frac{2ck\sqrt{\rho\sigma^2 + \frac{\eta dk}{8}\gamma^2}}{\psi}$$

Proof. From Theorem 2.6.2, we have that, with high probability over the random noise

$$\frac{1}{2}\|\hat{x}_V - \bar{x}_V\|_1 \leq \frac{2}{\psi} \cdot c\sqrt{nk} \cdot \sqrt{\sum_{u,i} \sigma_{u,i}^2 + \frac{k^2}{4}\sum_{uv} \gamma_{u,v}^2}$$

In this setting, this leads to

$$\frac{2}{\psi} \cdot c\sqrt{nk} \cdot \sqrt{\sum_{u,i} \sigma_{u,i}^2 + \frac{k^2}{4}\sum_{uv} \gamma_{u,v}^2} = \frac{2}{\psi} \cdot c\sqrt{nk} \cdot \sqrt{\rho nk\sigma^2 + \eta\frac{k^2}{4}\frac{nd}{2}\gamma^2} = \frac{2cnk\sqrt{\rho\sigma^2 + \frac{\eta dk}{8}\gamma^2}}{\psi}$$

since $|V| = n, |L| = k,$ and $|E| = \frac{nd}{2}$.

■

A.5. Algorithm for finding nearby stable instances details

Let $\bar{x}$ be a MAP solution, and let $\mathcal{E}^{\bar{x}}$ be the set of expansions of $\bar{x}$. We prove that an instance is $(2,1,\psi)$-expansion stable if and only if $\langle \theta_{adv}, \bar{x} \rangle \leq \langle \theta_{adv}, x \rangle$ for all $x \in \mathcal{E}^{\bar{x}}$. In other words, it is sufficient to check for stability in the *adversarial* perturbation for $\bar{x}$. This proves that we need not check every possible perturbation when finding a $(2,1,\psi)$-expansion stable instance.

Claim A.5.1. *Let (G, c, w) be an instance of uniform metric labeling with MAP solution $\bar{x}$. Define:*

$$w_{adv}(u,v) = \begin{cases} \frac{1}{2} w_{uv} & \bar{x}(u) = \bar{x}(v) \\ w(u,v) & \bar{x}(u) \neq \bar{x}(v) \end{cases}$$

$$c_{adv}(u,i) = \begin{cases} c(u,i) + \psi & \bar{x}(u) = i \\ c(u,i) & \textit{otherwise.} \end{cases}$$

Let θ_{adv} be the objective vector in the instance (G, c_{adv}, w_{adv}). Then

$$\langle \theta', \bar{x} \rangle \leq \langle \theta', x \rangle$$

for all $(2,1,\psi)$-perturbations θ' of θ and all $x \in \mathcal{E}^{\bar{x}}$ if and only if:

$$\langle \theta_{adv}, \bar{x} \rangle \leq \langle \theta_{adv}, x \rangle$$

for all $x \in \mathcal{E}^{\bar{x}}$.

Proof. The proof is analogous to that of Claim A.2.1. If the instance is $(2,1,\psi)$-expansion stable, then $\langle \theta', x \rangle - \langle \theta', \bar{x} \rangle \geq 0$ for all $(2,1,\psi)$-perturbations θ' and all expansions

x of $\bar{x}$. Because θ_{adv} is a valid $(2, 1, \psi)$-perturbation, this gives one direction. For the other, note that if the instance is not $(2, 1, \psi)$-expansion stable, there exists a θ' and an $x \in \mathcal{E}^{\bar{x}}$ for which $\langle \theta', x \rangle - \langle \theta', \bar{x} \rangle < 0$. A direct computation shows that $\langle \theta', x \rangle - \langle \theta', \bar{x} \rangle \geq \langle \theta_{adv}, x \rangle - \langle \theta_{adv}, \bar{x} \rangle$ for all $(2, 1, \psi)$-perturbations θ' of θ. Then we have $\langle \theta_{adv}, x \rangle - \langle \theta_{adv}, \bar{x} \rangle < 0$. ■

This claim justifies equation 2.5, which only enforces that $\bar{x}$ is at least as good as all of its expansions in θ_{adv}. The following claim implies that there is always a feasible point of equation 2.5 that makes modifications of bounded size to c and w.

Claim A.5.2. *Consider an instance (G, c, w) with a unique MAP solution $\bar{x}$. Let w' be defined as*

$$w'(u, v) = \begin{cases} w(u, v) & \bar{x}(u) \neq \bar{x}(v) \\ 2w(u, v) & \bar{x}(u) = \bar{x}(v), \end{cases}$$

and let c' be defined as

$$c'(u, i) = \begin{cases} c(u, i) - \psi & \bar{x}(u) = i \\ c(u, i) & \bar{x}(u) \neq i. \end{cases}$$

Then the instance (G, c', w') is $(2, 1, \psi)$-expansion stable with MAP solution $\bar{x}$.

Proof (sketch). The original MAP solution $\bar{x}$ is also the MAP solution to (G, c', w'). Then the original instance (G, c, w) is obtained from (G, c', w') by performing the adversarial $(2, 1, \psi)$-perturbation for $\bar{x}$ (see Claim A.5.1). Because $\bar{x}$ was the unique MAP solution to this instance, it has better objective than all of its expansions. Therefore, (G, c', w') is $(2, 1, \psi)$-expansion stable, by Claim A.5.1. ■

(G, c', w') is a "nearby" stable instance to (G, c, w), but it requires changes to quite a few edges—every edge that is not cut by $\bar{x}$—and changes the node costs of every vertex. Surprisingly, the stable instances we found in Section 2.8 were much closer than (G, c', w')—that is, only sparse changes were required to transform the observed instance (G, c, w) to a $(2, 1, \psi)$-expansion stable instance.

A.6. Experiment details

In this section, we give more details for the numerical examples for which we evaluate our curvature bound from Theorem 2.5.2. We studied instances for stereo vision, where the input is two images taken from slightly offset locations, and the desired output is the disparity of each pixel location between the two images (this disparity is inversely proportional to the depth of that pixel). We used the models from Tappen & Freeman (2003) on three images from the Middlebury stereo dataset (Scharstein & Szeliski, 2002). In this model, G is a grid graph with one node corresponding to each pixel in one of the images (say, the one taken from the left), the costs $c(u, i)$ are set using the Birchfield-Tomasi matching costs (Birchfield & Tomasi, 1998), and the edge weights $w(u, v)$ are set as:

$$w(u, v) = \begin{cases} P \times s & |I(u) - I(v)| < T \\ s & \text{otherwise.} \end{cases}$$

Here $I(u)$ is the intensity of one of the images (again, say the left image) at pixel location u, and we set $(P, T, s) = (2, 50, 4)$. This is a Potts model. The `tsukuba`, `cones`, and `venus` images were `120 x 150`, `125 x 150`, and `383 x 434`, respectively. These models had $k = 7$, $k = 5$, and $k = 5$, respectively.

To generate Table 2.1, we ran the algorithm in equation 2.5 using Gurobi (Gurobi Optimization, 2020) for the L1 distance. For each observed instance $(G, \hat{c}, \hat{w})$, this output a nearby $(2, 1, \psi)$-stable instance $(G, \bar{c}, \bar{w})$. In all of our experiments, we used $\psi = 1$. Additionally, we always set the target MAP solution x^t in equation 2.5 to be equal to the observed MAP solution $\hat{x}^{MAP}$. To evaluate our recovery bound, we compared the objective of the observed LP solution $\hat{x}$ to the $\hat{x}^{MAP}$ in $(G, \bar{c}, \bar{w})$. That is, if $\bar{\theta}$ is the objective for $(G, \bar{c}, \bar{w})$, we computed $\langle \bar{\theta}, \hat{x} \rangle - \langle \bar{\theta}, \hat{x}^{MAP} \rangle = \langle \bar{\theta}, \hat{x} \rangle - \langle \bar{\theta}, \bar{x} \rangle$, where the second equality is because we set the target solution x^t to be equal to $\hat{x}^{MAP}$, so $\hat{x}^{MAP} = \bar{x}$. Because $\psi = 1$, the difference between these two objectives is precisely the value of our curvature bound. In particular, Theorem 2.5.2 guarantees that

$$\frac{1}{2n} ||\hat{x}_V - \hat{x}_V^{MAP}||_1 \leq \frac{1}{n} \left(\langle \bar{\theta}, \hat{x} \rangle - \langle \bar{\theta}, \hat{x}^{MAP} \rangle \right).$$

The right-hand-side is shown for these instances in the "Recovery error bound" column of Table 2.1, and the true value of $\frac{1}{2n} ||\hat{x}_V - \hat{x}_V^{MAP}||_1$ (i.e., the true recovery error) is shown in the identically titled column of Table 2.1. On these instances, $\frac{1}{n} \left(\langle \bar{\theta}, \hat{x} \rangle - \langle \bar{\theta}, \hat{x}^{MAP} \rangle \right)$ is close to 0, so our curvature bound "explains" a large portion of $\hat{x}$'s recovery of $\hat{x}^{MAP}$. These instances are close to $(2, 1, \psi)$-stable instances where $\hat{x}$ and $\hat{x}^{MAP}$ have close objective, and this implies by Theorem 2.5.2 that $\hat{x}$ approximately recovers $\hat{x}^{MAP}$.

However, this result relies on a property of the LP solution $\hat{x}$: that it has good objective in the stable instance discovered by the procedure equation 2.5. Compare this to Corollary 2.5.3, which only depends on properties of the observed instance $\hat{\theta}$ and the stable instance $\bar{\theta}$ (in particular, some notion of "distance" between them). Given an observed instance $\hat{\theta}$ and stable instance $\bar{\theta}$, we can try to compute $d(\bar{\theta}, \hat{\theta})$ from Corollary 2.5.3 to give a bound that does not depend on $\hat{x}$. Unfortunately, this distance can be large, leading to a bound that can be vacuous (i.e., normalized Hamming recovery > 1). The following refinement of Corollary 2.5.3 gives much tighter bounds.

Corollary A.6.1 (LP solution is good if there is a nearby stable instance, refined)**.** *Let $\hat{x}^{MAP}$ and $\hat{x}$ be the MAP and local LP solutions to an observed instance $(G, \hat{c}, \hat{w})$. Also, let $\bar{x}$ be the MAP solution for a latent $(2, 1, \psi)$-expansion stable instance $(G, \bar{c}, \bar{w})$. If $\hat{\theta} = (\hat{c}, \hat{w})$ and $\bar{\theta} = (\bar{c}, \bar{w})$, define*

$$d(\bar{\theta}, \hat{\theta}) \coloneqq \sup_{x \in L^*(G)} \langle \bar{\theta}, x \rangle - \langle \bar{\theta}, \bar{x} \rangle - (\langle \hat{\theta}, x \rangle - \langle \hat{\theta}, \bar{x} \rangle).$$

Note that while we still use the name $d(\cdot, \cdot)$, evoking a metric, d is not symmetric. Then

$$\frac{1}{2}\|\hat{x}_V - \hat{x}_V^{MAP}\|_1 \leq \frac{d(\bar{\theta}, \hat{\theta})}{\psi} + \frac{1}{2}\|\hat{x}_V^{MAP} - \bar{x}_V\|_1.$$

Proof. Note:

$$\frac{1}{2}\|\hat{x}_V - \hat{x}_V^{MAP}\|_1 \leq \frac{1}{2}\|\hat{x}_V - \bar{x}_V\|_1 + \frac{1}{2}\|\hat{x}_V^{MAP} - \bar{x}_V\|_1.$$

By the definition of d, for any $x \in L^*(G)$,

$$\langle \bar{\theta}, x \rangle - \langle \bar{\theta}, \bar{x} \rangle \leq d(\bar{\theta}, \hat{\theta}) + (\langle \hat{\theta}, x \rangle - \langle \hat{\theta}, \bar{x} \rangle).$$

Now if we set $x = \hat{x}$, the LP solution to the observed instance, we have $\langle \hat{\theta}, \hat{x} \rangle - \langle \hat{\theta}, \bar{x} \rangle) \leq 0$, so

$$\langle \bar{\theta}, \hat{x} \rangle - \langle \bar{\theta}, \bar{x} \rangle \leq d(\bar{\theta}, \hat{\theta}).$$

Theorem 2.5.2 then implies $\frac{1}{2}\|\hat{x}_V - \bar{x}_V\|_1 \leq d(\bar{\theta}, \hat{\theta})/\psi$, which gives the claim. ■

Given an observed instance $\hat{\theta}$ and a $(2, 1, \psi)$-expansion stable instance $\bar{\theta}$ output by equation 2.5 with $\bar{x}^{MAP} = \hat{x}^{MAP}$, we can upper bound $d(\bar{\theta}, \hat{\theta})$ by computing

$$d_{up}(\bar{\theta}, \hat{\theta}) := \sup_{x \in L(G)} \langle \bar{\theta}, x \rangle - \langle \bar{\theta}, \bar{x} \rangle - (\langle \hat{\theta}, x \rangle - \langle \hat{\theta}, \bar{x} \rangle),$$

which is a linear program in x because we relaxed $L^*(G)$ to $L(G)$. Corollary A.6.1 then implies that the recovery error of $\hat{x}$ is at most $d_{up}(\bar{\theta}, \hat{\theta})/\psi$, which we can compute. Table A.1 shows the results of this procedure on two of the same instances from Table 2.1 in the "Unconditional bound" column. While the values of this bound are much larger than the "Curvature bound" of Theorem 5.2, they are much more theoretically appealing, since they only depend on the difference between $\hat{\theta}$ and $\bar{\theta}$ rather than on a property of the LP solution $\hat{x}$ to $\hat{\theta}$. For Table A.1, we did a grid search for ψ over $\{1, \ldots, 10\}$; $\psi = 4$ gave the optimal unconditional bound for both instances. The difference in ψ explains the slight differences between the other columns of Tables 2.1 and A.1.

Table A.1. Results from the output of equation 2.5 on two stereo vision instances. Curvature bound shows the bound obtained from Theorem 2.5.2, which depends on the observed LP solution $\hat{x}$. Unconditional bound shows the bound from the refined version of Corollary 2.5.3, which depends *only* on the observed instance and the stable instance. This "unconditional" bound explains a reasonably large fraction of the LP solution's recovery for these instances: because the instance is close to a stable instance, the LP solution approximate recovers the MAP solution.

Instance	Costs changed	Weights changed	Curvature bound	Unconditional bound	$\frac{\|\|\hat{x}_V - \hat{x}_V^{MAP}\|\|_1}{2n}$
`tsukuba`	4.9%	2.8%	0.0173	0.4878	0.0027
`cones`	2.81%	2.31%	0.0137	0.2819	0.0022

APPENDIX B

Appendix to Chapter 3

B.1. Streaming MAP Inference Details

Theorem 3.4.2. *For a random-order arrival stream, if S is the solution output by Algorithm 1 at the end of the stream and $\sigma_{\min} > 1$ where $\sigma_{\min}$ and $\sigma_{\max}$ denote the smallest and largest singular values of $\boldsymbol{L}_S$ among all $S \subseteq [n]$ and $|S| \leq 2k$, then*

$$\frac{\mathbb{E}[\log\det(\boldsymbol{L}_S)]}{\log(\mathrm{OPT})} \geq \left(1 - \frac{1}{\sigma_{\min}^{(1-\frac{1}{e})\cdot(2\log\sigma_{\max} - \log\sigma_{\min})}}\right)$$

where $\boldsymbol{L}_S = \boldsymbol{V}_S^\top \boldsymbol{V}_S + \boldsymbol{B}_S^\top \boldsymbol{C} \boldsymbol{B}_S$ *and* $\mathrm{OPT} = \max_{R\subseteq[n],\ |R|=k} \det(\boldsymbol{V}_R^\top \boldsymbol{V}_R + \boldsymbol{B}_R^\top \boldsymbol{C}\boldsymbol{B}_R)$.

Proof. As described in Algorithm 1, we will use $S_0, S_1, \ldots, S_k$ to denote the solution sets maintained by the algorithm, where S_i represents the solution set of size i. In particular, we have that $S_i = S_{i-1} \cup \{s_i\}$ where $s_i = \arg\max_{j\in B_i} f(S \cup \{j\})$ and B_i denotes the i'th partition i.e., $B_i \coloneqq \{\frac{(i-1)\cdot n}{k} + 1, \frac{(i-1)\cdot n}{k} + 2, \ldots, \frac{i\cdot n}{k}\}$.

For $i \in [k]$, let us use $X_i \coloneqq [B_i \cap (S_* \setminus S_{i-1}) \neq \emptyset]$ to denote the event that there is at least one element of the optimal solution which has not already been picked by the algorithm in the batch B_i and $\lambda_i \coloneqq |S_* \setminus S_{i-1}|$. Then,

$$\begin{aligned}
\mathbb{P}[X_i] &= 1 - \mathbb{P}[X_i^{\mathsf{c}}] \\
&= 1 - (1 - \frac{\lambda_i}{n})(1 - \frac{\lambda_i}{n-1}) \dots (1 - \frac{\lambda_i}{n - \frac{n}{k} + 1}) \\
&\geq 1 - \left(1 - \frac{\lambda_i}{n}\right)^{\frac{n}{k}} \\
&\geq 1 - e^{-\frac{\lambda_i}{k}} \\
&\geq \frac{\lambda_i}{k} \cdot \left(1 - \frac{1}{e}\right)
\end{aligned}$$

Here we use the facts: $e^x \geq 1 + x$ for all $x \in \mathbb{R}$, $1 - e^{-\frac{\lambda}{k}}$ is concave as a function of λ, and $\lambda \in [0, k]$.

For any element $s \in [n]$ and set $S \subseteq [n]$, let us use $f(s \mid S) \coloneqq f(S \cup \{s\}) - f(S)$ to denote the marginal gain in f obtained by adding the element s to the set S. For any round $i \in [k]$, we then have that $f(S_i) - f(S_{i-1}) = f(s_i \mid S_{i-1})$.

Note that

$$\mathbb{E}[f(s_i \mid S_{i-1}) \mid X_i] \geq \frac{\sum_{\omega \in \mathrm{OPT} \setminus S_{i-1}} f(\omega \mid S_{i-1})}{|\mathrm{OPT} \setminus S_{i-1}|}.$$

This happens due to the fact that conditioned on X_i, every element in $S_* \setminus S_{i-1}$ is equally likely to be present in B_i and the algorithm picks s_i such that $f(s_i \mid S_{i-1}) \geq f(s \mid S_{i-1})$

for all $s \in B_i$.

$$\begin{aligned}
\mathbb{E}[f(s_i \mid S_{i-1}) \mid S_{i-1}] &= \mathbb{E}[f(s_i \mid S_{i-1}) \mid S_{i-1}, X_i]\mathbb{P}[X_i] \\
&+ \mathbb{E}[f(s_i \mid S_{i-1}) \mid S_{i-1}, X_i^{\mathsf{c}}]\mathbb{P}[X_i^{\mathsf{c}}] \\
&\geq \mathbb{E}[f(s_i \mid S_{i-1}) \mid S_{i-1}, X_i]\mathbb{P}[X_i] \\
&\geq \frac{\lambda_i}{k}\left(1 - \frac{1}{e}\right) \cdot \frac{\sum_{\omega \in S_* \setminus S_{i-1}} f(\omega \mid S_{i-1})}{|S_* \setminus S_{i-1}|} \\
&= \frac{\lambda_i}{|S_* \setminus S_{i-1}|}\left(1 - \frac{1}{e}\right) \cdot \frac{1}{k} \cdot \sum_{\omega \in S_* \setminus S_{i-1}} f(\omega \mid S_{i-1}) \\
&= \left(1 - \frac{1}{e}\right) \cdot \frac{1}{k} \cdot \sum_{\omega \in S_* \setminus S_{i-1}} f(\omega \mid S_{i-1}) \\
&\geq \left(1 - \frac{1}{e}\right) \cdot \frac{1}{k} \cdot \gamma \cdot (f(S_{i-1} \cup S_*) - f(S_{i-1})) \\
&\geq \left(1 - \frac{1}{e}\right) \cdot \frac{1}{k} \cdot \gamma \cdot (\mathrm{OPT} - f(S_{i-1}))
\end{aligned}$$

For the last 2 inequalities, we use the fact that $f(S) = \log\det(\boldsymbol{L}_S)$ is monotone non-decreasing and has a submodularity ratio of $\gamma = \left(2\frac{\log\sigma_{\max}}{\log\sigma_{\min}} - 1\right)^{-1}$ when $\sigma_{\min} > 1$ (Gartrell et al., 2021)[Eq. 45].

Taking expectation over all random draws of S_{i-1}, we get

$$\mathbb{E}[f(s_i \mid S_{i-1})] \geq \left(1 - \frac{1}{e}\right) \cdot \frac{\gamma}{k}(\mathrm{OPT} - \mathbb{E}[f(S_{i-1})])$$

Combining the above equation with $f(s_i|S_{i-1}) = f(S_i) - f(S_{i-1})$, we have

$$\mathbb{E}[f(S_i)] - \mathbb{E}[f(S_{i-1})] \geq \left(1 - \frac{1}{e}\right) \cdot \frac{\gamma}{k} \cdot (\mathrm{OPT} - \mathbb{E}[f(S_{i-1})])$$

Next we have

$$-(\mathrm{OPT} - \mathbb{E}[f(S_i)]) + (\mathrm{OPT} - \mathbb{E}[f(S_{i-1})]) \geq \left(1 - \frac{1}{e}\right) \cdot \frac{\gamma}{k} \cdot (\mathrm{OPT} - \mathbb{E}[f(S_{i-1})])$$

Re-organizing the above equation, we obtain

$$\mathrm{OPT} - \mathbb{E}[f(S_i)] \leq \left(1 - \left(1 - \frac{1}{e}\right) \cdot \frac{\gamma}{k}\right) (\mathrm{OPT} - \mathbb{E}[f(S_{i-1})])$$

Applying the above equation recursively k times,

$$\begin{aligned} \mathrm{OPT} - \mathbb{E}[f(S_k)] &\leq \left(1 - \left(1 - \frac{1}{e}\right) \cdot \frac{\gamma}{k}\right)^k (\mathrm{OPT} - \mathbb{E}[f(S_0)]) \\ &= \left(1 - \left(1 - \frac{1}{e}\right) \cdot \frac{\gamma}{k}\right)^k \mathrm{OPT} \end{aligned}$$

where the last step follows from $f(S_0) = 0$.

Re-organized the terms again, we have

$$\begin{aligned} \mathbb{E}[f(S_k)] &\geq \left(1 - \left(1 - \left(1 - \frac{1}{e}\right) \cdot \frac{\gamma}{k}\right)^k\right) \mathrm{OPT} \\ &\geq \left(1 - e^{-\gamma(1-\frac{1}{e})}\right) \mathrm{OPT} \end{aligned}$$

When we substitute $\gamma = \left(2\frac{\log \sigma_{\max}}{\log \sigma_{\min}} - 1\right)^{-1}$, we get our final inequality:

$$\mathbb{E}[f(S_k)] \geq \left(1 - \frac{1}{\sigma_{\min}^{(1-\frac{1}{e})\cdot(2\log\sigma_{\max} - \log\sigma_{\min})}}\right) \mathrm{OPT}$$

■

B.2. Hard instance for One-Pass MAP Inference of NDPPs

Outline: We will now give a high-level description of a hard instance for online MAP inference of NDPPs (this is inspired by (Anari & Vuong, 2022, Example 5)) . Suppose we have a total of $2d$ items consisting of **pairs** of complementary items like modem-router, printer-ink cartridge, pencil-eraser etc. Let us use $\{1, 1^{\mathsf{c}}, 2, 2^{\mathsf{c}}, \ldots, d, d^{\mathsf{c}}\}$ to denote them. Any item i is independent of every item other than it's complement i^{c}. Individually, $\mathbb{P}[\{i\}] = \mathbb{P}[\{i^{\mathsf{c}}\}] = 0$. And $\mathbb{P}[\{i, i^{\mathsf{c}}\}] = x_i^2$ with $x_i > 0$ for all $i \in [d]$. Also, we have $\mathbb{P}[\{i, j\}] = 0$ for any $i \neq j$. Suppose any of our online algorithms are given the sequence $\{1, 2, 3, \ldots, d, r^{\mathsf{c}}\}$ where $r \in [d]$ is some arbitrary item and the algorithm needs to pick 2 items i.e., $k = 2$. Then, OPT > 0 whereas all of our online algorithms (Online LSS, Online 2-neighbor, Online-Greedy) will fail to output a valid solution.

Details: Let $0 < x_1 < x_2 < \cdots < x_d$. Suppose

$$\boldsymbol{C} = \begin{bmatrix} 0 & x_1 & & & & & \\ -x_1 & 0 & & & & & \\ & & 0 & x_2 & & & \\ & & -x_2 & 0 & & & \\ & & & & \ddots & & \\ & & & & & 0 & x_d \\ & & & & & -x_d & 0 \end{bmatrix}$$

$\boldsymbol{C} \in \mathbb{R}^{2d \times 2d}$ is a skew-symmetric (i.e., $\boldsymbol{C} = -\boldsymbol{C}^\top$) block diagonal matrix where the blocks are of the form $\begin{bmatrix} 0 & x_i \\ -x_i & 0 \end{bmatrix}$. Suppose we have a total of $2d$ items consisting of

d pairs of complementary items. We use $\{1, 1^{\mathsf{c}}, 2, 2^{\mathsf{c}}, \ldots, d, d^{\mathsf{c}}\}$ to denote them. Let $\boldsymbol{v}_i = \boldsymbol{v}_{i^{\mathsf{c}}} = \boldsymbol{0} \; \forall \; i \in [d]$ and $\boldsymbol{b}_1 = \boldsymbol{e}_1, \boldsymbol{b}_{1^{\mathsf{c}}} = \boldsymbol{e}_2, \ldots, \boldsymbol{b}_{d^{\mathsf{c}}} = \boldsymbol{e}_{2d}$ where $\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_{2d}$ are the standard unit vectors in $\mathbb{R}^{2d}$ i.e., $B = \boldsymbol{I}_{2d}$.

For a pair of complementary items $S = \{i, i^{\mathsf{c}}\}, f(S) = x_i^2$. Without loss of generality, consider $S = \{1, 1^{\mathsf{c}}\}$. Then we can compute $\boldsymbol{B}_S^\top \boldsymbol{C} \boldsymbol{B}_S$ as follows:

$$\begin{aligned}
\boldsymbol{B}_S^\top \boldsymbol{C} \boldsymbol{B}_S &= \begin{bmatrix} e_1 & e_2 \end{bmatrix}^\top \boldsymbol{C} \begin{bmatrix} e_1 & e_2 \end{bmatrix} \\
&= \begin{bmatrix} 0 & x_1 & 0 & 0 & \cdots & 0 \\ -x_1 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \cdot \begin{bmatrix} e_1 & e_2 \end{bmatrix} \\
&= \begin{bmatrix} 0 & x_1 \\ -x_1 & 0 \end{bmatrix}
\end{aligned}$$

In this case, we have $f(S) = x_1^2$.

For any pair of non-complementary items $S = \{i_1, i_2\}$ where the indices are distinct, $f(S) = 0$. Without loss of generality, we can consider $S = \{1, 2\}$. Then,

$$\begin{aligned}
\boldsymbol{B}_S^\top \boldsymbol{C} \boldsymbol{B}_S &= \begin{bmatrix} e_1 & e_3 \end{bmatrix}^\top \boldsymbol{C} \begin{bmatrix} e_1 & e_3 \end{bmatrix} \\
&= \begin{bmatrix} 0 & x_1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & x_2 & \cdots & 0 \end{bmatrix} \cdot \begin{bmatrix} e_1 & e_3 \end{bmatrix} \\
&= \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}
\end{aligned}$$

And so, we have that $f(S) = 0$.

B.3. Experiments and Datasets details

All experiments were performed using a standard desktop computer (Quad-Core Intel Core i7, 16 GB RAM) using many real-world datasets composed of sets (or baskets) of items from some ground set of items:

- **UK Retail:** This is an online retail dataset consisting of sets of items all purchased together by users (in a single transaction) (Chen et al., 2012). There are 19,762 transactions (sets of items purchased together) that consist of 3,941 items. Transactions with more than 100 items are discarded.
- **MovieLens:** This dataset contains sets of movies that users watched (Sharma et al., 2019). There are 29,516 sets consisting of 12,549 movies.
- **Amazon Apparel**: This dataset consists of 14,970 registries (sets) from the apparel category of the Amazon Baby Registries dataset, which is a public dataset that has been used in prior work on NDPPs (Gartrell et al., 2021; 2019). These apparel registries are drawn from 100 items in the apparel category.
- **Amazon 3-category**: We also use a dataset composed of the apparel, diaper, and feeding categories from Amazon Baby Registries, which are the most popular categories, giving us 31,218 registries made up of 300 items (Gartrell et al., 2019).
- **Instacart:** This dataset represents sets of items purchased by users on Instacart (Instacart, 2017). Sets with more than 100 items are ignored. This gives 3.2 million total item-sets from 49,677 unique items.
- **Million Song:** This is a dataset of song playlists put together by users where every playlist is a set (basket) of songs played by Echo Nest users (McFee et al.,

2012). Playlists with more than 150 songs are discarded. This gives 968,674 playlists from 371,410 songs.

- **Customer Dashboards:** This dataset consists of dashboards or baskets of visualizations created by users (Qian et al., 2021). Each dashboard represents a set of visualizations selected by a user. There are 63436 dashboards (baskets/sets) consisting of 1206 visualizations.
- **Web Logs:** This dataset consists of sets of webpages (baskets) that were all visited in the same session. There are 2.8 million baskets (sets of webpages) drawn from 2 million webpages.
- **Company Retail:** This dataset contains the set of items viewed (or purchased) by a user in a given session. Sets (baskets) with more than 100 items are discarded. This results in 2.5 million baskets consisting of 107,349 items.

The last two datasets are proprietary Adobe data. The learning algorithm of (Gartrell et al., 2021) takes as input a parameter d, which is the embedding size for $\boldsymbol{V}$, $\boldsymbol{B}$, $\boldsymbol{C}$. We use $d = 10$ for all datasets other than Instacart, Customer Dashboards, Company Retail where $d = 50$ is used and Million Song, where $d = 100$ is used. For all of our results in 3.7, we set $k = 8$ and choose $\alpha = 1.1$.

B.4. Additional Experimental Results

B.4.1. Number of Determinant Computations

We perform several experiments comparing the number of determinant computations (as a system-independent proxy for time) of all our online algorithms. We do not compare with offline greedy here because that algorithm doesn't explicitly compute all the determinants.

Results comparing the number of determinant computations as a function of the number of data points analyzed for a variety of datasets are provided in Figure B.1. Online-2-neighbor requires the most number of determinant computations but also gives the best results in terms of solution value. Online-LSS and Online-Greedy use very similar number of determinant computations.

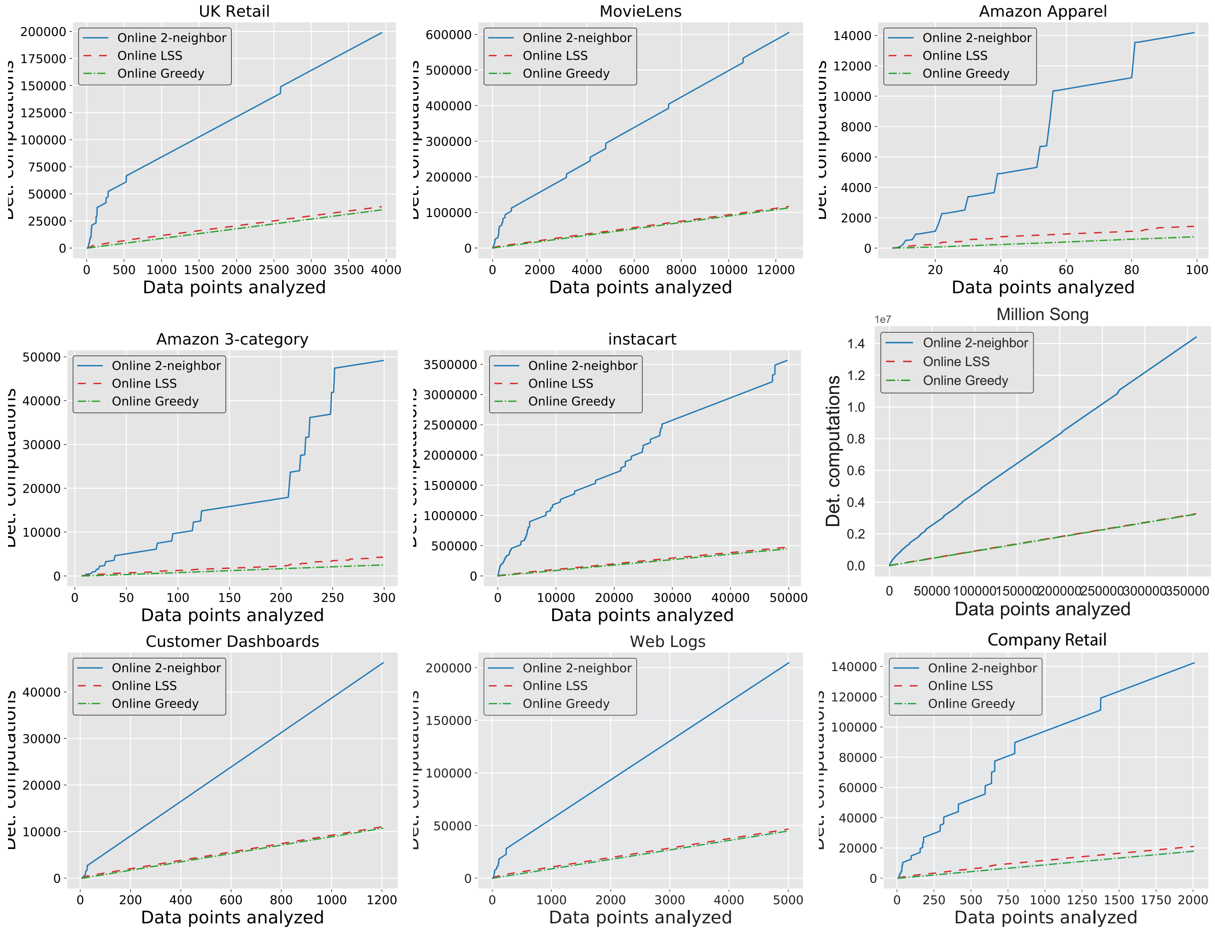


Figure B.1. Results comparing the number of determinant computations as a function of the number of data points analyzed for all our online algorithms. Online-2-neighbor requires the most number of determinant computations but also gives the best results in terms of solution value. Online-LSS and Online-Greedy use very similar number of determinant computations.

B.4.2. Number of Swaps

Results comparing the number of swaps (as a measure of solution consistency) of all our online algorithms can be found in Figure B.2. Online-Greedy has the most number of swaps and therefore the least consistent solution set. On most datasets, the number of swaps by Online-2-neighbor is very similar to Online-LSS.

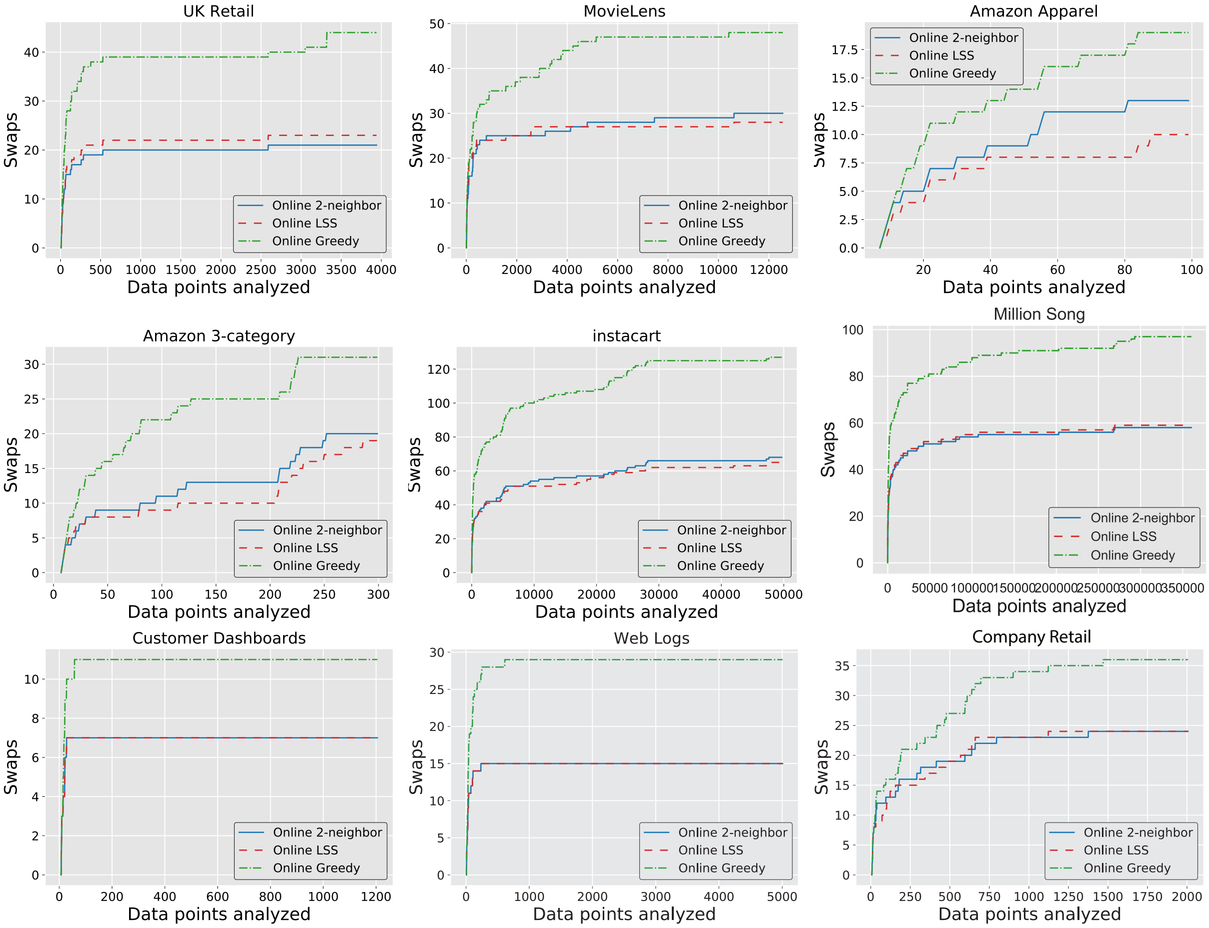


Figure B.2. Results comparing the number of swaps of all our online algorithms. Online-Greedy does the most number of swaps and therefore has the least consistent solution set. On most datasets, the number of swaps by Online-2-neighbor is very similar to Online-LSS.

B.4.3. Random Streams

We also investigate our algorithms under the random stream paradigm. For this setting, we use some of the previous real-world datasets, and randomly permute the order in which the data appears in the stream. We do this 100 times and report the average of solution values in Figure B.3 and the average of number of determinant computations and swaps in Figure B.4. We observe that Online-2-neighbor and Online-LSS give very similar performance in this regime and they are always better than Online-Greedy.

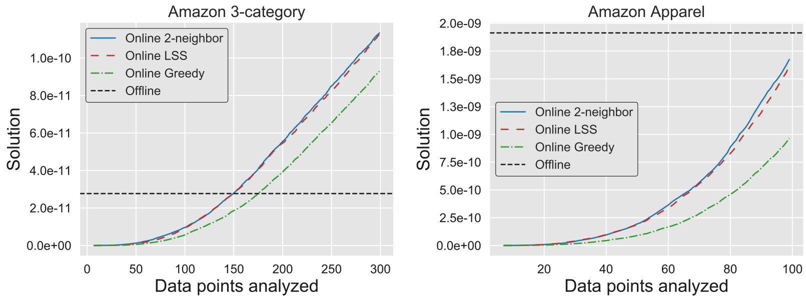


Figure B.3. Solution quality as a function of the number of data points analyzed in the random stream paradigm. Online-2-neighbor and Online-LSS give very similar performance in this setting and they are always better than Online-Greedy.

B.4.4. Ablation study varying ε

To study the effect of ε in Online-LSS (Algorithm 2), we vary $\varepsilon \in \{0.05, 0.1, 0.3, 0.5\}$ and analyze the value of the obtained solutions, number of determinant computations, and number of swaps. We notice that, in general, the solution quality, number of determinant computations, and the number of swaps increase as ε decreases. Results are provided in Figure B.5.

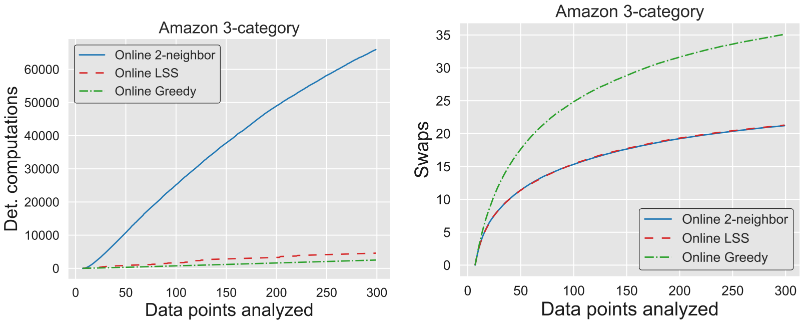


Figure B.4. Number of determinant computations and swaps as a function of the number of data points analyzed in the random stream setting. Online-2-Neighbor needs more determinant computations than Online-LSS but has very similar number of swaps in this setting. Note that $\varepsilon = \alpha - 1$

.

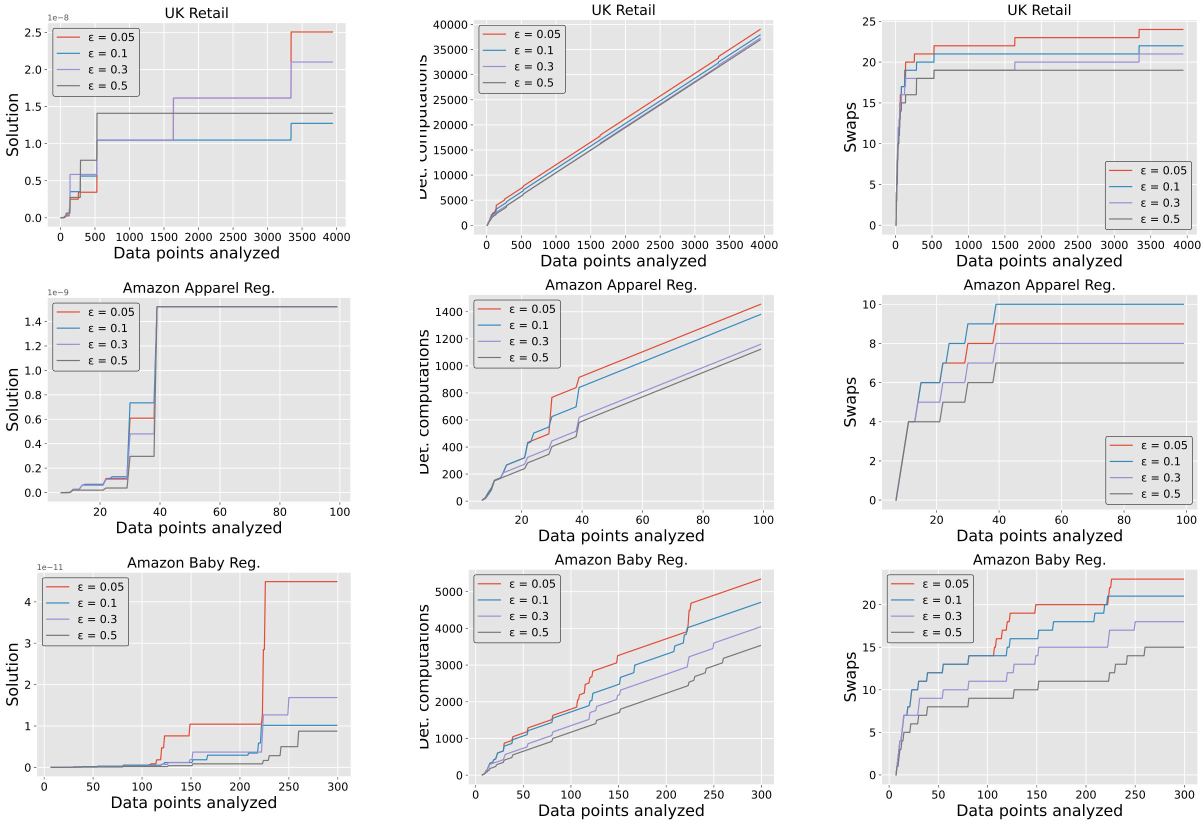


Figure B.5. Performance of Online-LSS varying ε for $k = 8$. Solution quality, number of determinant computations, and number of swaps seem to increase with decreasing ε.

B.4.5. Ablation study varying set size k and ε

In this set of experiments on the Amazon-Apparel dataset using Online-LSS, we study the choice of set size k and ε on the solution quality, number of determinant computations, and number of swaps while fixing all other settings to be same as those used previously in Figure B.3. We can see that as k increases, the solution value decreases across all values of ε. This is in accordance with our intuition that the probability of larger sets should be smaller. In general, the number of determinant computations and swaps increases for increasing k across different values of ε.

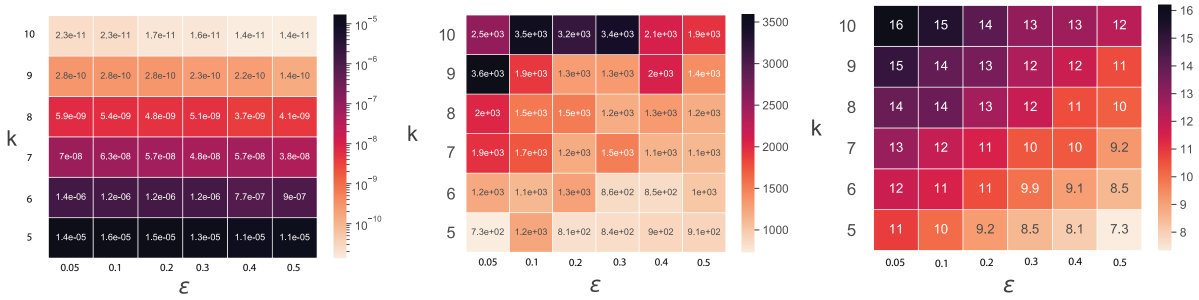


Figure B.6. Comparing the effect of set size k and ε on the solution value, number of determinant computations, and number of swaps for Online-lss.

Vita

Aravind Reddy Talla was born in the bustling Indian metropolis of Hyderabad in 1996 and grew up there eating its world-renowned biryani. He received a Bachelor of Technology (B.Tech.) in Computer Science & Engineering with distinction, along with a minor in Physics, from the Indian Institute of Technology Kanpur in June 2018. He also subsequently received a Master of Science (M.S.) in Computer Science from Northwestern University in March 2021.

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346 USA