Inferring Rankings Using Constrained Sensing

Srikanth Jagabathula and Devavrat Shah

***Abstract*—We consider the problem of recovering a function over the space of permutations (or, the symmetric group) over n elements from given partial information; the partial information we consider is related to the group theoretic Fourier Transform of the function. This problem naturally arises in several settings such as ranked elections, multi-object tracking, ranking systems, and recommendation systems. Inspired by the work of Donoho and Stark in the context of discrete-time functions, we focus on non-negative functions with a sparse support (support size $\ll$ domain size). Our recovery method is based on finding the sparsest solution (through ℓ_0 optimization) that is consistent with the available information. As the main result, we derive sufficient conditions for functions that can be recovered exactly from partial information through ℓ_0 optimization. Under a natural random model for the generation of functions, we quantify the recoverability conditions by deriving bounds on the sparsity (support size) for which the function satisfies the sufficient conditions with a high probability as $n \to \infty$. ℓ_0 optimization is computationally hard. Therefore, the popular compressive sensing literature considers solving the convex relaxation, ℓ_1 optimization, to find the sparsest solution. However, we show that ℓ_1 optimization fails to recover a function (even with constant sparsity) generated using the random model with a high probability as $n \to \infty$. In order to overcome this problem, we propose a novel iterative algorithm for the recovery of functions that satisfy the sufficient conditions. Finally, using an Information Theoretic framework, we study necessary conditions for exact recovery to be possible.**

***Index Terms*—Compressive sensing, Fourier analysis over symmetric group, functions over permutations, sparsest-fit.**

I. Introduction

FUNCTIONS over permutations serve as rich tools for modeling uncertainty in several important practical applications; they correspond to a general model class, where each model has a factorial number of parameters. However, in many practical applications, only partial information is available about the underlying functions; this is because either the problem setting naturally makes only partial information available, or memory constraints allow only partial information to be maintained as opposed to the entire function—which requires storing a factorial number of parameters in general. In either case, the following important question arises: which "types" of functions can be recovered from access to only partial information? Intuitively, one expects a characterization that relates the "complexity" of the functions that can be recovered to the "amount" of partial information one has access to. One of the main goals of this paper is to formalize this statement. More specifically, this paper considers the problem of *exact* recovery of a function over the space of permutations given only partial information. When the function is a probability distribution, the partial information we consider can be thought of as lower-order marginals; more generally, the types of partial information we consider are related to the group theoretic Fourier Transform of the function, which provides a general way to represent varying "amounts" of partial information. In this context, our goal is to (a) characterize a class of functions that can be recovered exactly from the given partial information, and (b) design a procedure for their recovery. We restrict ourselves to non-negative functions, which span many of the useful practical applications. Due to the generality of the setting we consider, a thorough understanding of this problem impacts a wide-ranging set of applications. Before we present the precise problem formulation and give an overview of our approach, we provide below a few motivating applications that can be modeled effectively using functions over permutations.

A popular application where functions over permutations naturally arise is the problem of *rank aggregation*. This problem arises in various contexts. The classical setting is that of *ranked election*, which has been studied in the area of *Social Choice Theory* for the past several decades. In the ranked election problem, the goal is to determine a "socially preferred" ranking of n candidates contesting an election using the individual preference lists (permutations of candidates) of the voters. Since the "socially preferred" outcome should be independent of the identities of voters, the available information can be summarized as a function over permutations that maps each permutation σ to the fraction of voters that have the preference list σ. While described in the context of elections, the ranked election setting is more general and also applies to aggregating through polls the population preferences on global issues, movies, movie stars, etc. Similarly, rank aggregation has also been studied in the context of aggregating webpage rankings [2], where one has to aggregate rankings over a large number of webpages. Bulk of the work done on the ranked election problem deals with the question of aggregation *given* access to the entire function over permutations that summarizes population preferences. In many practical settings, however, determining the function itself is nontrivial—even for reasonably small values of n. Like in the setting of polling, one typically can gather only partial information about population preferences. Therefore, our ability to recover functions over permutations from available partial information impacts our ability to aggregate rankings. Interestingly, in the context of ranked elections, Diaconis [3] showed through spectral analysis that a partial set of Fourier coefficients of the function possesses

Manuscript received August 06, 2010; revised January 13, 2011; accepted June 08, 2011. Date of current version November 11, 2011. This work was supported in part by NSF CAREER CNS 0546590 and NSF CCF 0728554.

S. Jagabathula is with the Stern School of Business, New York University, New York, NY 10012 USA (e-mail: sjagabat@stern.nyu.edu).

D. Shah is with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02140 USA (e-mail: devavrat@mit.edu).

Communicated by J. Romberg, Associate Editor for Signal Processing.

Digital Object Identifier 10.1109/TIT.2011.2165827

"rich" information about the underlying function. This hints to the possibility that, in relevant applications, limited partial information can still capture a lot of structure of the underlying function.

Another important problem, which has received a lot of attention recently, is the *Identity Management Problem* or the *Multi-object tracking problem*. This problem is motivated by applications in air traffic control and sensor networks, where the goal is to track the identities of n objects from noisy measurements of identities and positions. Specifically, consider an area with sensors deployed that can identify the unique signature and the position associated with each object when it passes close to it. Let the objects be labeled $1, 2, \ldots, n$ and let $x(t) = (x_1(t), x_2(t), \ldots, x_n(t))$ denote the vector of positions of the n objects at time t. Whenever a sensor registers the signature of an object the vector $x(t)$ is updated. A problem, however, arises when two objects, say i, j, pass close to a sensor simultaneously. Because the sensors are inexpensive, they tend to confuse the signatures of the two objects; thus, after the two objects pass, the sensor has information about the positions of the objects, but it only has beliefs about which position belongs to which object. This problem is typically modeled as a probability distribution over permutations, where, given a position vector $x(t)$, a permutation σ of $1, 2, \ldots, n$ describes the assignment of the positions to objects. Because the measurements are noisy, to each position vector $x(t)$, we assign, not a single permutation, but a distribution over permutations. Since we now have a distribution over permutations, the factorial blow-up makes it challenging to maintain it. Thus, it is often approximated using a partial set of Fourier coefficients. Recent work by [4], [5] deals with updating the distribution with new observations in the Fourier domain. In order to obtain the final beliefs one has to recover the distribution over permutations from a partial set of Fourier coefficients.

Finally, consider the task of coming up with rankings for teams in a sports league, for example, the "Formula-one" car racing or American football, given the outcomes of various games. In this context, one approach is to model the final ranking of the teams using, not just one permutation, but a distribution over permutations. A similar approach has been taken in ranking players in online games (cf. Microsoft's TrueSkill solution [6]), where the authors, instead of maintaining scores, maintain a distribution over scores for each player. In this context, clearly, we can gather only partial information and the goal is to fit a model to this partial information. Similar questions arise in recommendation systems in cases where rankings, instead of ratings, are available or are preferred.

In summary, all the examples discussed above relate to inferring a function over permutations using partial information. To fix ideas, let S_n denote the permutation group of order n and $f: S_n \rightarrow \mathbb{R}_+$ denote a non-negative function defined over the permutations. We assume we have access to partial information about $f(\cdot)$ that, as discussed subsequently, corresponds to a subset of coefficients of the group theoretic Fourier Transform of $f(\cdot)$. We note here that a partial set of Fourier coefficients not only provides a rigorous way to compress the high-dimensional function $f(\cdot)$ (as used in [4], [5]), but also have natural interpretations, which makes it easy to gather in practice. Under this setup, our goal is to characterize the functions f that can be recovered. The problem of exact recovery of functions from a partial information has been widely studied in the context of discrete-time functions; however, the existing approaches don't naturally extend to our setup. One of the classical approaches for recovery is to find the function with the minimum "energy" consistent with the given partial information. This approach was extended to functions over permutations in [7], where the authors obtain lower bounds on the energy contained in subsets of Fourier Transform coefficients to obtain better ℓ_2 guarantees when using the function the minimum "energy." This approach, however, does not naturally extend to the case of exact recovery. In another approach, which recently gained immense popularity, the function is assumed to have a sparse support and conditions are derived for the size of the support for which exact recovery is possible. This work was pioneered by Donoho; in [1], Donoho and Stark use generalized uncertainty principles to recover a discrete-time function with sparse support from a limited set of Fourier coefficients. Inspired by this, we restrict our attention to functions with a sparse support.

Assuming that the function is sparse, our approach to performing exact recovery is to find the function with the sparsest support that is consistent with the given partial information, henceforth referred to as ℓ_0 optimization. This approach is often justified by the philosophy of *Occam's razor*. We derive sufficient conditions in terms of sparsity (support size) for functions that can be recovered through ℓ_0 optimization. Furthermore, finding a function with the sparsest support through ℓ_0 minimization is in general computationally hard. This problem is typically overcome by considering the convex relaxation of the ℓ_0 optimization problem. However, as we show in Theorem III.2, such a convex relaxation does not yield exact recovery in our case. Thus, we propose a simple iterative algorithm called the 'sparsest-fit' algorithm and prove that the algorithm performs exact recovery of functions that satisfy the sufficient conditions.

It is worth noting that our work has important connections to the work done in the recently popular area of *compressive sensing*. Broadly speaking, this work derives sufficient conditions under which the sparsest function that is consistent with the given information can be found by solving the corresponding ℓ_1 relaxation problem. However, as discussed below in the section on relevant work, the sufficient conditions derived in this work do not apply to our setting. Therefore, our work may be viewed as presenting an alternate set of conditions under which the ℓ_0 optimization problem can be solved efficiently.

A. Related Work

Fitting sparse models to observed data has been a classical approach used in statistics for model recovery and is inspired by the philosophy of *Occam's Razor*. Motivated by this, sufficient conditions for learnability based on sparsity have been of great interest over years in the context of communication, signal processing and statistics, cf. [8], [9]. In recent years, this approach has become of particular interest due to exciting developments and wide ranging applications including:

- In signal processing (see [10]–[14]) where the goal is to estimate a 'signal' by means of minimal number of measurements. This is referred to as compressive sensing.

- In coding theory through the design of low-density parity check codes [15]–[17] or in the design of Reed Solomon codes [18] where the aim is to design a coding scheme with maximal communication rate.
- In the context of streaming algorithms through the design of 'sketches' (see [19]–[23]) for the purpose of maintaining a minimal 'memory state' for the streaming algorithm's operation.

In all of the above work, the basic question (see [24]) pertains to the design of an $m \times n$ "measurement" matrix A so that x can be recovered efficiently from measurements $y = Ax$ (or its noisy version) using the "fewest" possible number measurements m. The setup of interest is when x is sparse and when $m < n$ or $m \ll n$. The type of interesting results (such as those cited above) pertain to characterization of the sparsity K of x that can be recovered for a given number of measurements m. The usual tension is between the ability to recover x with large k using a sensing matrix A with minimal m.

The sparsest recovery approach of this paper is similar (in flavor) to the above stated work; in fact, as is shown subsequently, the partial information we consider can be written as a linear transform of the function $f(\cdot)$. However, the methods or approaches of the prior work do not apply. Specifically, the work considers finding the sparsest function consistent with the given partial information by solving the corresponding ℓ_1 relaxation problem. The work derives a necessary and sufficient condition, called the *Restricted Nullspace Property*, on the structure of the matrix A that guarantees that the solutions to the ℓ_0 and ℓ_1 relaxation problems are the same (see [11], [21]). However, such sufficient conditions trivially fail in our setup (see [25]). Therefore, our work provides an alternate set of conditions that guarantee efficient recovery of the sparsest function.

B. Our Contributions

Recovery of a function over permutations from only partial information is clearly a hard problem both from a theoretical and computational standpoint. We make several contributions in this paper to advance our understanding of the problem in both these respects. As the main result, we obtain sufficient conditions—in terms of sparsity—for functions that can be recovered exactly from partial information. Specifically, our result establishes a relation between the "complexity" (as measured in sparsity) of the function that can be recovered and the "amount" of partial information available.

Our recovery scheme consists of finding the sparsest solution consistent with the given partial information through ℓ_0 optimization. We derive sufficient conditions under which a function can be recovered through ℓ_0 optimization. First, we state the sufficient conditions for recovery through ℓ_0 optimization in terms of the structural properties of the functions. To understand the strength of the sufficient conditions, we propose a random generative model for functions with a given support size; we then obtain bounds on the size of the support for which a function generated according to the random generative model satisfies the sufficient conditions with a high probability. To our surprise, it is indeed possible to recover, with high probability, functions with seemingly large sparsity for given partial information (see precise statement of Theorems III.3–III.6 for details).

Finding the sparsest solution through ℓ_0 optimization is computationally hard. This problem is typically overcome by considering the ℓ_1 convex relaxation of the ℓ_0 optimization problem. However, as we show in Example II-C.1, ℓ_1 relaxation does not always result in exact recovery, even when the the sparsity of the underlying function is only 4. In fact, a necessary and sufficient condition for ℓ_1 relaxation to yield the sparsest solution x that satisfies the constraints $y = Ax$ is the so called Restricted Nullspace Condition (RNC) on the measurement matrix A; interestingly, the more popular Restricted Isoperimetric Property (RIP) on the measurement matrix A is a sufficient condition. However, as shown below, the types of partial information we consider can be written as a linear transform of $f(\cdot)$. Therefore, Example II-C.1 shows that in our setting, the measurement matrix does not satisfy RNC. It is natural to wonder if Example II-C.1 is anomalous. We show that this is indeed not the case. Specifically, we show in Theorem III.2 that, with a high probability, ℓ_1 relaxation fails to recover a function generated according to the random generative model.

Since convex relaxations fail in recovery, we exploit the structural property of permutations to design a simple iterative algorithm called the 'sparsest-fit' algorithm to perform recovery. We prove that the algorithm recovers a function from a partial set of its Fourier coefficients as long as the function satisfies the sufficient conditions.

We also study the limitation of *any* recovery algorithm to recover a function exactly from a given form of partial information. Through an application of classical information theoretic Fano's inequality, we obtain a bound on the sparsity beyond which recovery is not *asymptotically reliable*; a recovery scheme is called asymptotically reliable if the probability of error asymptotically goes to 0.

In summary, we obtain an intuitive characterization of the "complexity" (as measured in sparsity) of the functions that can be recovered from the given partial information. We show how ℓ_1 relaxation fails in recovery in this setting. Hence, the sufficient conditions we derive correspond to an alternate set of conditions that guarantee efficient recovery of the sparsest function.

C. Organization

Section II introduces the model, useful notations and the precise formulation of the problem. In Section III, we provide the statements of our results. Section IV describes our iterative algorithm that can recover f from $\hat{f}(\lambda)$ when certain conditions (see Condition 1) are satisfied. Sections V to XI provide detailed proofs. Conclusions are presented Section XII.

II. Problem Statement

In this section, we introduce the necessary notations, definitions and provide the formal problem statement.

A. Notations

Let n be the number of elements and S_n be set of all possible $n!$ permutations or rankings of these of n elements. Our interest

is in learning non-negative valued functions f defined on S_n, i.e., $f : S_n \rightarrow \mathbb{R}_+$, where $\mathbb{R}_+ = \{x \in \mathbb{R} : x \geq 0\}$. The support of f is defined as

$$\text{supp}(f) = \{\sigma \in S_n : f(\sigma) \neq 0\}.$$

The cardinality of support, $|\text{supp}(f)|$ will be called the *sparsity* of f and will be denoted by K. We will also call it the ℓ_0 norm of f, denoted by $|f|_0$.

In this paper, we wish to learn f from a partial set of Fourier coefficients. To define the Fourier transform of a function over the permutation group, we need some notations. To this end, consider a partition of n, i.e., an ordered tuple $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_r)$, such that $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_r \geq 1$, and $n = \lambda_1 + \lambda_2 + \ldots + \lambda_r$. For example, $\lambda = (n-1, 1)$ is a partition of n. Now consider a partition of the n elements, $\{1, \ldots, n\}$, as per the λ partition, i.e., divide n elements into r bins with ith bin having λ_i elements. It is easy to see that n elements can be divided as per the λ partition in D_λ distinct ways, with

$$D_\lambda = \frac{n!}{\prod_{i=1}^{r} \lambda_i!}.$$

Let the distinct partitions be denoted by $t_i, 1 \leq i \leq D_\lambda$.[1] For example, for $\lambda = (n-1, 1)$ there are $D_\lambda = n!/(n-1)! = n$ distinct ways given by

$$t_i \equiv \{1, \ldots, i-1, i+1, \ldots, n\}\{i\}, 1 \leq i \leq n.$$

Given a permutation $\sigma \in S_n$, its action on t_i is defined through its action on the n elements of t_i, resulting in a λ partition with the n elements permuted. In the above example with $\lambda = (n-1, 1)$, σ acts on t_i to give $t_{\sigma(i)}$, i.e.,

$$\sigma : t_i \rightarrow t_{\sigma(i)}, \text{ where } t_i \equiv \{1, \ldots, i-1, i+1, \ldots, n\}\{i\} \text{ and } t_{\sigma(i)} \equiv \{1, \ldots, \sigma(i)-1, \sigma(i)+1, \ldots, n\}\{\sigma(i)\}.$$

Now, for a given partition λ and a permutation $\sigma \in S_n$, define a $0/1$ valued $D_\lambda \times D_\lambda$ matrix $M^\lambda(\sigma)$ as

$$M^\lambda{}_{ij}(\sigma) = \begin{cases} 1, & \text{if } \sigma(t_j) = t_i \\ 0, & \text{otherwise.} \end{cases} \quad \text{for all } 1 \leq i, j \leq D_\lambda$$

This matrix $M^\lambda(\sigma)$ corresponds to a degree D_λ representation of the permutation group.

B. Partial Information as a Fourier Coefficient

The partial information we consider in this paper is the Fourier transform coefficient of f at the representation M^λ, for each λ. The motivation for considering Fourier coefficients at representations M^λ is two fold: first, they provide a rigorous way to compress the high-dimensional function $f(\cdot)$ (as used in [4], [5]), and second, as we shall see, Fourier coefficients at representations M^λ have natural interpretations, which makes it easy to gather in practice. In addition, each representation M^λ contains a subset of the lower-order irreducible representations; thus, for each λ, M^λ conveniently captures the information contained in a subset of the lower-order Fourier coefficients up to λ. We now define the Fourier coefficient of f at the representation M^λ, which we call λ-partial information.

Definition II.1. (λ-Partial Information): Given a function $f : S_n \rightarrow \mathbb{R}_+$ and partition λ. The Fourier Transform coefficient at representation M^λ, which we call the λ-partial information, is denoted by $\hat{f}(\lambda)$ and is defined as

$$\hat{f}(\lambda) = \sum_{\sigma \in S_n} f(\sigma) M^\lambda(\sigma).$$

Recall the example of $\lambda = (n-1, 1)$ with f as a probability distribution on S_n. Then, $\hat{f}(\lambda)$ is an $n \times n$ matrix with the (i, j)th entry being the probability of element j mapped to element i under f. That is, $\hat{f}(\lambda)$ corresponds to the *first order* marginal of f in this case.

C. Problem Formulation

We wish to recover a function f based on its partial information $\hat{f}(\lambda)$ based from its partition λ. As noted earlier, the classical approach based on Occam's razor suggests recovering the function as a solution of the following ℓ_0 optimization problem:

$$\begin{aligned} &\text{minimize } ||g||_0 \quad \text{over} \quad g : S_n \rightarrow \mathbb{R}_+ \\ &\text{subject to } \hat{g}(\lambda) = \hat{f}(\lambda). \end{aligned} \tag{1}$$

We note that the question of recovering f from $\hat{f}(\lambda)$ is very similar to the question studied in the context of compressed sensing, i.e., recover x from $y = Ax$. To see this, with an abuse of notation imagine $\hat{f}(\lambda)$ as a D_λ^2 dimensional vector and f as an $n!$ dimensional vector. Then, $\hat{f}(\lambda) = Af$ where each column of A corresponds to $M^\lambda(\sigma)$ for certain permutation σ. The key difference from the compressed sensing literature is that A is given in our setup rather than being a design choice.

Question One: As the first question of interest, we wish to identify precise conditions under which ℓ_0 optimization problem (1) recovers the original function f as its unique solution.

Unlike the popular literature (cf. compressed sensing), such conditions can not be based on sparsity only. This is well explained by the following (counter-)example. In addition, the example also shows that linear independence of the support of f does not guarantee uniqueness of the solution to the ℓ_0 optimization problem.

Example II-C.1: For any $n \geq 4$, consider the four permutations $\sigma_1 = (1,2), \sigma_2 = (3,4), \sigma_3 = (1,2)(3,4)$ and $\sigma_4 = \text{id}$, where id is the identity permutation. In addition, consider the partition $\lambda = (n-1, 1)$. Then, it is easy to see that

$$M^\lambda(\sigma_1) + M^\lambda(\sigma_2) = M^\lambda(\sigma_3) + M^\lambda(\sigma_4).$$

We now consider three cases where a bound on sparsity is not sufficient to guarantee the existence of a unique solution to (1).

1) This example shows that a sparsity bound (even 4) on f is not sufficient to guarantee that f will indeed be the sparsest solution. Specifically, suppose that $f(\sigma_i) = p_i$, where $p_i \in$

[1] To keep notation simple, we use t_i instead of t_i^λ that takes explicit dependence on λ into account.

$\mathbb{R}_+$ for $1 \le i \le 4$, and $f(\sigma) = 0$ for all other $\sigma \in S_n$. Without loss of generality, let $p_1 \le p_2$. Then

$$\begin{aligned}&\hat{f}(\lambda)\\ =&p_1 M^\lambda(\sigma_1) + p_2 M^\lambda(\sigma_2) + p_3 M^\lambda(\sigma_3) + p_4 M^\lambda(\sigma_4)\\ =&(p_2 - p_1) M^\lambda(\sigma_2) + (p_3 + p_1) M^\lambda(\sigma_3)\\ &+ (p_4 + p_1) M^\lambda(\sigma_4).\end{aligned}$$

Thus, function g with $g(\sigma_2) = p_2 - p_1, g(\sigma_3) = p_3 + p_1, g(\sigma_4) = p_4 + p_1$ and $g(\sigma) = 0$ for all other $\sigma \in S_n$ is such that $\hat{g}(\lambda) = \hat{f}(\lambda)$ but $\|g\|_0 = 3 < 4 = \|f\|_0$. That is, f can not be recovered as the solution of ℓ_0 optimization problem (1) even when support of f is only 4.

2) This example shows that although f might be a sparsest solution, it may not be unique. In particular, suppose that $f(\sigma_1) = f(\sigma_2) = p$ and $f(\sigma) = 0$ for all other $\sigma \in S_n$. Then, $\hat{f}(\lambda) = pM^\lambda(\sigma_1) + pM^\lambda(\sigma_2) = pM^\lambda(\sigma_3) + pM^\lambda(\sigma_4)$. Thus, (1) does not have a unique solution.
3) Finally, this example shows that even though the support of f corresponds to a linearly independent set of columns, the sparsest solution may not be unique. Now suppose that $f(\sigma_i) = p_i$, where $p_i \in \mathbb{R}_+$ for $1 \le i \le 3$, and $f(\sigma) = 0$ for all other $\sigma \in S_n$. Without loss of generality, let $p_1 \le p_2$. Then

$$\begin{aligned}&\hat{f}(\lambda)\\ =&p_1 M^\lambda(\sigma_1) + p_2 M^\lambda(\sigma_2) + p_3 M^\lambda(\sigma_3)\\ =&(p_2 - p_1) M^\lambda(\sigma_2) + (p_3 + p_1) M^\lambda(\sigma_3) + p_1 M^\lambda(\sigma_4).\end{aligned}$$

Here, note that $\{M^\lambda(\sigma_1), M^\lambda(\sigma_2), M^\lambda(\sigma_3)\}$ is linearly independent, yet the solution to (1) is not unique.

Question Two: The resolution of the first question will provide a way to recover f by means of solving the ℓ_0 optimization problem in (1). However, in general, it is computationally a hard problem. Therefore, we wish to obtain a simple and possibly iterative algorithm to recover f [and hence for solving (1)].

Question Three: Once we identify the conditions for exact recovery of f, the next natural question to ask is "how restrictive are the conditions we imposed on f for exact recovery?" In other words, as mentioned above, we know that the sufficient conditions don't translate to a simple sparsity bound on functions, however, can we find a sparsity bound such that "most," if not all, functions that satisfy the sparsity bound can be recovered? We make the notion of "most" functions precise by proposing a natural random generative model for functions with a given sparsity. Then, for a given partition λ, we want to obtain $K(\lambda)$ so that if $K < K(\lambda)$ then recovery of f generated according to the generative model from $\hat{f}(\lambda)$ is possible with high probability.

This question is essentially an inquiry into whether the situation demonstrated by Example II.C.1 is contrived or not. In other words, it is an inquiry into whether such examples happen with vanishingly low probability for a randomly chosen function. To this end, we describe a natural random function generation model.

Definition II.2 (Random Model): Given $K \in \mathbb{Z}_+$ and an interval $\mathscr{C} = [a, b], 0 < a < b$, a random function f with sparsity K and values in $\mathscr{C}$ is generated as follows: choose K permutations from S_n independently and uniformly at random,[2] say $\sigma_1, \ldots, \sigma_K$; select K values from $\mathscr{C}$ uniformly at random, say $p_1, \ldots, p_K$; then function f is defined as

$$f(\sigma) = \begin{cases} p_i, & \text{if} \quad \sigma = \sigma_i, 1 \le i \le K \\ 0, & \text{otherwise.} \end{cases}$$

We will denote this model as $R(K, \mathscr{C})$.

Question Four: Can we characterize a limitation on the ability of *any* algorithm to recover f from $\hat{f}(\lambda)$?

III. Main Results

As the main result of this paper, we provide answers to the four questions stated in Section II-C. We start with recalling some notations. Let $\lambda = (\lambda_1, \ldots, \lambda_r)$ be the given partition of n. We wish to recover function $f : S_n \to \mathbb{R}_+$ from available information $\hat{f}(\lambda)$. Let the sparsity of f be K

$$\text{supp}(f) = \{\sigma_1, \ldots, \sigma_K\}, \quad \text{and} \quad f(\sigma_k) = p_k, 1 \le k \le K.$$

Answers One & Two: To answer the first two questions, we need to find sufficiency conditions for recovering f through ℓ_0 optimization (1) and a simple algorithm to recover the function. For that, we first try to gain a qualitative understanding of the conditions that f must satisfy. Note that a necessary condition for ℓ_0 optimization to recover f is that (1) must have a *unique* solution; otherwise, without any additional information, we wouldn't know which of the multiple solutions is the true solution. Clearly, since $\hat{f}(\lambda) = \sum_{\sigma \in S_n} f(\sigma) M^\lambda(\sigma)$, (1) will have a unique solution only if $\{M^\lambda(\sigma)\}_{\sigma \in \text{supp}(f)}$ is linearly independent. However, this linear independence condition is, in general, not sufficient to guarantee a unique solution; in particular, even if $\{M^\lambda(\sigma)\}_{\sigma \in \text{supp}(f)}$ is linearly independent, there could exist $\{M^\lambda(\sigma')\}_{\sigma' \in \mathcal{H}}$ such that $\hat{f}(\lambda) = \sum_{\sigma' \in \mathcal{H}} M^\lambda(\sigma')$ and $|\mathcal{H}| \le K$, where $K := |\text{supp}(f)|$; Example II-C.1 illustrates such a scenario. Thus, a sufficient condition for f to be the unique sparsest solution of (1) is that not only is $\{M^\lambda(\sigma)\}_{\sigma \in \text{supp}(f)}$ linearly independent, but $\{M^\lambda(\sigma), M^\lambda(\sigma')\}_{\sigma \in \text{supp}(f), \sigma' \in \mathcal{H}}$ is linearly independent for all $\mathcal{H} \subset S_n$ such that $|\mathcal{H}| \le K$; in other words, not only we want $M^\lambda(\sigma)$ for $\sigma \in \text{supp}(f)$ to be linearly independent, but we want them to be linearly independent even after the addition of at most K permutations to the support of f. Note that this condition is similar to the Restricted Isometry Property (RIP) introduced in [10], which roughly translates to the property that ℓ_0 optimization recovers x of sparsity K from $y = Ax$ provided every subset of $2K$ columns of A is linearly independent. Motivated by this, we impose the following conditions on f.

Condition (Sufficiency Conditions): Let f satisfy the following:

- *Unique Witness*: for any $\sigma \in \text{supp}(f)$, there exists $1 \le i_\sigma, j_\sigma \le D_\lambda$ such that $M^\lambda_{i_\sigma j_\sigma}(\sigma) = 1$, but $M^\lambda_{i_\sigma j_\sigma}(\sigma') = 0$, for all $\sigma' (\neq \sigma) \in \text{supp}(f)$.

[2]Throughout, we will assume that the random selection is done *with* replacement.

- *Linear Independence*: for any collection of integers $c_1, \ldots, c_K$ taking values in $\{-K, \ldots, K\}$, $\sum_{k=1}^{K} c_k p_k \neq 0$, unless $c_1 = \ldots = c_K = 0$.

The above discussion motivates the "unique witness" condition; indeed, $M^{\lambda}(\sigma)$ for σ satisfying the "unique witness" condition are linearly independent because every permutation has a unique witness and no nonzero linear combination of $M^{\lambda}(\sigma)$ can yield zero. On the other hand, as shown in the proof of Theorem III.1, the *linear independence* condition is required for the uniqueness of the sparsest solution.

Now we state a formal result that establishes Condition 1 as sufficient for recovery of f as the unique solution of ℓ_0 optimization problem. Further, it allows for a simple, iterative recovery algorithm. Thus, Theorem III.1 provides answers to questions *One* and *Two* of Section II-C.

Theorem III.1: Under Condition 1, the function f is the unique solution to the ℓ_0 optimization problem (1). Further, a simple, iterative algorithm called the sparsest-fit algorithm, described in Section IV, recovers f.

Linear Programs Don't Work: Theorem III.1 states that under Condition 1, the ℓ_0 optimization recovers f and the sparsest-fit algorithm is a simple iterative algorithm to recover it. In the context of compressive sensing literature (cf. [11], [13], [14], [21]), it has been shown that convex relaxation of ℓ_0 optimization, such as the Linear Programing relaxation, have the same solution in similar scenarios. Therefore, it is natural to wonder whether such a relaxation would work in our case. To this end, consider the Linear Programing relaxation of (1) stated as the following ℓ_1 minimization problem:

$$\begin{aligned} &\text{minimize } \|g\|_1 \qquad \text{over} \qquad g : S_n \to \mathbb{R}_+ \\ &\text{subject to } \hat{g}(\lambda) = \hat{f}(\lambda). \end{aligned} \tag{2}$$

Example II.C.1 provides a scenario where ℓ_1 relaxation fails in recovery. In fact, we can prove a stronger result. The following result establishes that—with a high probability—a function generated randomly as per Definition II.2 cannot be recovered by solving the linear program (2) because there exists a function g such that $\hat{g}(\lambda) = \hat{f}(\lambda)$ and $\|g\|_1 = \|f\|_1$.

Theorem III.2: Consider a function f randomly generated as per Definition II.2 with sparsity $K \geq 2$. Then, as longs as λ is not the partition $(1, 1, \ldots, 1)$ (n times), with probability $1 - o(1)$, there exists a function g distinct from f such that $\hat{g}(\lambda) = \hat{f}(\lambda)$ and $\|g\|_1 = \|f\|_1$.

Answer Three: Next, we turn to the third question. Specifically, we study the conditions for high probability recoverability of a random function f in terms of its sparsity. That is, we wish to identify the high probability recoverability threshold $K(\lambda)$. In what follows, we spell out the result starting with few specific cases so as to better explain the dependency of $K(\lambda)$ on D_λ.

Case 1: $\lambda = (n-1, 1)$. Here $D_\lambda = n$ and $\hat{f}(\lambda)$ provides the *first order* marginal information. As stated next, for this case the achievable recoverability threshold $K(\lambda)$ scales[3] as $n \log n$.

[3]Throughout this paper, by $\log$ we mean the natural logarithm, i.e., $\log_e$, unless otherwise stated.

Theorem III.3: A randomly generated f as per Definition II.2 can be recovered by the sparsest-fit algorithm with probability $1 - o(1)$ as long as $K \leq (1-\varepsilon) n \log n$ for any fixed $\varepsilon > 0$.

Case 2: $\lambda = (n-m, m)$ with $1 < m = O(1)$. Here $D_\lambda = \Theta(n^m)$ and $\hat{f}(\lambda)$ provides the *mth order* marginal information. As stated next, for this case we find that $K(\lambda)$ scales at least as $n^m \log n$.

Theorem III.4: A randomly generated f as per Definition II.2 can be recovered from $\hat{f}(\lambda)$ by the sparsest-fit algorithm for $\lambda = (n-m, m), m = O(1)$, with probability $1 - o(1)$ as long as $K \leq \frac{(1-\varepsilon)}{m!} n^m \log n$ for any fixed $\varepsilon > 0$.

In general, for any λ with $\lambda_1 = n - m$ and $m = O(1)$, arguments of Theorem III.4 can be adapted to show that $K(\lambda)$ scales as $n^m \log n$. Theorems III.3 and III.4 suggest that the recoverability threshold scales $D_\lambda \log D_\lambda$ for $\lambda = (\lambda_1, \ldots, \lambda_r)$ with $\lambda_1 = n - m$ for $m = O(1)$. Next, we consider the case of more general λ.

Case 3: $\lambda = (\lambda_1, \ldots, \lambda_r)$ with $\lambda_1 = n - O\left(n^{\frac{2}{9}-\delta}\right)$ for any $\delta > 0$. As stated next, for this case, the recoverability threshold $K(\lambda)$ scales at least as $D_\lambda \log \log D_\lambda$.

Theorem III.5: A randomly generated f as per Definition II.2 can be recovered from $\hat{f}(\lambda)$ by the sparsest-fit algorithm for $\lambda = (\lambda_1, \ldots, \lambda_r)$ with $\lambda_1 = n - n^{\frac{2}{9}-\delta}$ for any $\delta > 0$, with probability $1 - o(1)$ as long as $K \leq (1-\varepsilon) D_\lambda \log \log D_\lambda$ for any fixed $\varepsilon > 0$.

Case 4: Any $\lambda = (\lambda_1, \ldots, \lambda_r)$. The results stated thus far suggest that the threshold is essentially D_λ, ignoring the logarithm term. For general λ, we establish a bound on $K(\lambda)$ as stated in Theorem III.6 below. Before stating the result, we introduce some notation. For given λ, define $\alpha = (\alpha_1, \ldots, \alpha_r)$ with $\alpha_i = \lambda_i / n, 1 \leq i \leq r$. Let

$$H(\alpha) = -\sum_{i=1}^{r} \alpha_i \log \alpha_i, \quad \text{and} \quad H'(\alpha) = -\sum_{i=2}^{r} \alpha_i \log \alpha_i.$$

Theorem III.6: Given $\lambda = (\lambda_1, \ldots, \lambda_r)$, a randomly generated f as per Definition II.2 can be recovered from $\hat{f}(\lambda)$ by the sparsest-fit algorithm with probability $1 - o(1)$ as long as

$$K \leq C\, D_\lambda^{\gamma(\alpha)} \tag{3}$$

where

$$\gamma(\alpha) = \frac{M}{M+1}\left[1 - C' \frac{H(\alpha) - H'(\alpha)}{H(\alpha)}\right] \quad \text{with } M = \left\lfloor \frac{1}{1-\alpha_1} \right\rfloor$$

and $0 < C, C' < \infty$ are constants.

At a first glance, the above result seems very different from the crisp formulas of Theorems III.3–III.5. Therefore, let us consider a few special cases. First, observe that as $\alpha_1 \uparrow 1, M/(M+1) \to 1$. Further, as stated in Lemma III.1, $H'(\alpha)/H(\alpha) \to 1$. Thus, we find that the bound on sparsity essentially scales as D_λ. Note that the cases 1, 2 and 3 fall squarely under this scenario since $\alpha_1 = \lambda_1/n = 1 - o(1)$. Thus, this general result contains the results of Theorems III.3–III.5 (ignoring the logarithm terms).

Next, consider the other extreme of $\alpha_1 \downarrow 0$. Then, $M \to 1$ and again by Lemma III.1, $H'(\alpha)/H(\alpha) \to 1$. Therefore, the bound on sparsity scales as $\sqrt{D_\lambda}$. This ought to be the case because for $\lambda = (1, \ldots, 1)$ we have $\alpha_1 = 1/n \to 1, D_\lambda = n!$, and unique witness property holds only up to $o(\sqrt{D_\lambda}) = o(\sqrt{n!})$ due to the standard Birthday paradox.

In summary, Theorem III.6 appears reasonably tight for the general form of partial information λ. We now state the Lemma III.1 used above (proof in Appendix A).

Lemma III.1: Consider any $\alpha = (\alpha_1, \ldots, \alpha_r)$ with $1 \geq \alpha_1 \geq \cdots \geq \alpha_r \geq 0$ and $\sum_{i=1}^r \alpha_r = 1$. Then

$$\lim_{\alpha_1 \uparrow 1} \frac{H'(\alpha)}{H(\alpha)} = 1$$
$$\lim_{\alpha_1 \downarrow 0} \frac{H'(\alpha)}{H(\alpha)} = 1.$$

Answer Four: Finally, we wish to understand the fundamental limitation on the ability to recover f from $\hat{f}(\lambda)$ by any algorithm. To obtain a meaningful bound (cf. Example II-C.1), we shall examine this question under an appropriate information theoretic setup.

To this end, as in random model $R(K, \mathscr{C})$, consider a function f generated with given K and λ. For technical reasons (or limitations), we will assume that the values p_is are chosen from a discrete set. Specifically, let each p_i be chosen from integers $\{1, \ldots, T\}$ instead of compact set $\mathscr{C}$. We will denote this random model by $R(K, T)$.

Consider any algorithm that attempts to recover f from $\hat{f}(\lambda)$ under $R(K, T)$. Let h be the estimation of the algorithm. Define probability of error of the algorithm as

$$p_{\text{err}} = \Pr(h \neq f).$$

We state the following result.

Theorem III.7: With respect to random model $R(K, T)$, the probability of error is uniformly bounded away from 0 for all n large enough and any λ, if

$$K \geq \frac{3D_\lambda^2}{n \log n} \left[\log \left(\frac{D_\lambda^2}{n \log n} \vee T \right) \right] \tag{4}$$

where for any two numbers x and y, $x \vee y$ denotes $\max\{x, y\}$.

IV. Sparsest-Fit Algorithm

As mentioned above, finding the sparsest distribution that is consistent with the given partial information is in general a computationally hard problem. In this section, we propose an efficient algorithm to fit the sparsest distribution to the given partial information $\hat{f}(\lambda)$, for any partition λ of n. The algorithm we propose determines the sparsest distribution *exactly* as long as the underlying distribution belongs to the general family of distributions that satisfy the 'unique witness' and 'linear independence' conditions; we call this the 'sparsest-fit' algorithm. In this case, it follows from Theorem III.1 that the 'sparsest-fit' algorithm indeed recovers the underlying distribution $f(\cdot)$ exactly from partial information $\hat{f}(\lambda)$. When the conditions are not satisfied, the algorithm produces a certificate to that effect and aborts.

Using the degree D_λ representation of the permutations, the algorithm processes the elements of the partial information matrix $\hat{f}(\lambda)$ sequentially and incrementally builds the permutations in the support. We describe the sparsest-fit algorithm as a general procedure to recover a set of non-negative values given sums of these values over a collection of subsets, which for brevity we call subset sums. In this sense, it can be thought of as a linear equation solver customized for a special class of systems of linear equations.

Next we describe the algorithm in detail and prove the relavant theorems.

A. Sparsest-Fit Algorithm

We now describe the sparsest-fit algorithm that was also referred to in Theorems III.1, III.3–III.6 to recover function f from $\hat{f}(\lambda)$ under Condition 1.

Setup: The formal description of the algorithm is given in Fig. 1. The algorithm is described there as a generic procedure to recover a set of non-negative values given a collection of their subset sums. As explained in Fig. 1, the inputs to the algorithm are L positive numbers $q_1, \ldots, q_L$ sorted in ascending order $q_1 \leq q_2 \leq \cdots \leq q_L$. As stated in assumptions C1–C3 in Fig. 1, the algorithm assumes that the L numbers are different subset sums of K distinct positive numbers $p_1, \ldots, p_K$ i.e., $q_\ell = \sum_{T_\ell} p_k$ for some $T_\ell \subset \{1, 2, \ldots, K\}$, and the values and subsets satisfy the conditions: for each $1 \leq k \leq K, p_k = q_\ell$ for some $1 \leq \ell \leq L$ and $\sum_T p_k \neq \sum_{T'} p_k$ for $T \neq T'$. Given this setup, the sparsest-fit algorithm recovers the values p_k and subset membership sets $A_k := \{\ell : k \in T_\ell\}$ for $1 \leq k \leq K$ using q_ℓ, but without any knowledge of K or subsets $T_\ell, 1 \leq \ell \leq L$.

Before we describe the algorithm, note that in order to use the sparsest-fit algorithm to recover $f(\cdot)$ we give the nonzero elements of the partial information matrix $\hat{f}(\lambda)$ as inputs q_ℓ. In this case, L equals the number of nonzero entries of $\hat{f}(\lambda), p_k = f(\sigma_k)$, and the sets A_k correspond to $M^\lambda(\sigma_k)$. Here, assumption C1 of the algorithm is trivially satisfied. As we argue in Section V, assumptions C2, C3 are implied by the 'unique witness' and 'linear independence' conditions.

Description: The formal description is given below in the Fig. 1. The algorithm processes elements $q_1, q_2, \ldots, q_L$ sequentially and builds membership sets incrementally. It maintains the number of nonempty membership sets at the end of each iteration ℓ as $k(\ell)$. Partial membership sets are maintained as sets A_k, which at the end of iteration ℓ equals $\{1 \leq \ell' \leq \ell : k \in T_{\ell'} \text{ for some } \ell' \leq \ell\}$. The values found are maintained as $p_1, p_2, \ldots, p_{k(\ell)}$. The value of $k(0)$ is initialized to zero and the sets A_k are initialized to be empty.

In each iteration ℓ, the algorithm checks if the value q_ℓ can be written as a subset sum of values $p_1, p_2, \ldots, p_{k(\ell-1)}$ for some subset T. If q_ℓ can be expressed as $\sum_{k \in T} p_k$ for some $T \subset \{1, 2, \ldots, k(\ell-1)\}$, then the algorithm adds ℓ to sets A_k for $k \in T$ and updates $k(\ell)$ as $k(\ell) = k(\ell-1)$ before ending the iteration. In case there exists no such subset T, the algorithm updates $k(\ell)$ as $k(\ell-1)+1$, makes the set $A_{k(\ell)}$ nonempty

Input: Positive values $\{q_1, q_2, \ldots, q_L\}$ sorted in ascending order i.e., $q_1 \leq q_2 \leq \ldots \leq q_L$.

Assumptions: $\exists$ positive values $\{p_1, p_2, \ldots, p_K\}$ such that:

C1. For each $1 \leq \ell \leq L$, $q_\ell = \sum_{k \in T_\ell} p_k$, for some $T_\ell \subseteq \{1, 2, \ldots, K\}$

C2. For each $1 \leq k \leq K$, there exists a q_ℓ such that $q_\ell = p_K$.

C3. $\sum_{k \in T} p_k \neq \sum_{k' \in T'} p_{k'}$, for all $T, T' \subseteq \{1, 2, \ldots, K\}$ and $T \cap T' = \emptyset$.

Output: $\{p_1, p_2, \ldots, p_K\}$, $\forall\ 1 \leq k \leq K$ set A_k s.t.

$$A_k = \{\ell : q_\ell = \sum_{j \in T} p_j \text{ and index } k \text{ belongs to set } T\}.$$

Algorithm:

initialization: $p_0 = 0, k(0) = 0$, $A_k = \emptyset$ for all possible k.
for $\ell = 1$ to L
  if $q_\ell = \sum_{k \in T} p_k$ for some $T \subseteq \{0, 1, \ldots, k(\ell - 1)\}$
    $k(\ell) = k(\ell - 1)$
    $A_k = A_k \cup \{\ell\} \quad \forall \quad k \in T$
  else
    $k(\ell) = k(\ell - 1) + 1$
    $p_{k(\ell)} = q_\ell$
    $A_{k(\ell)} = A_{k(\ell)} \cup \{\ell\}$
  end if
end for
Output $K = k(L)$ and $(p_k, A_k), 1 \leq k \leq K$.

Fig. 1. Sparsest-fit algorithm.

by adding ℓ to it, and sets $p_{k(\ell)}$ to q_ℓ. At the end the algorithm outputs (p_k, A_k) for $1 \leq k \leq k(L)$.

We now argue that under assumptions C1–C3 stated in Fig. 1, the algorithm finds (p_k, A_k) for $1 \leq k \leq K$ accurately. Note that by Assumption C2, there exists at least one q_ℓ such that it is equal to p_k, for each $1 \leq k \leq K$. Assumption C3 guarantees that the condition in the **if** statement is not satisfied whenever $q_\ell = p_{k(\ell)}$. Therefore, the algorithm correctly assigns values to each of the p_ks. Note that the condition in the **if** statement being true implies that q_ℓ is a subset sum of some subset $T \subset \{p_1, p_2, \ldots, p_{k(\ell-1)}\}$. Assumption C3 ensures that if such a combination exists then it is unique. Thus, when the condition is satisfied, index ℓ belongs only to the sets A_k such that $k \in T$. When the condition in the **if** statement is false, then from Assumptions C2 and C3 it follows that ℓ is contained only in $A_{k(\ell)}$. From this discussion we conclude that the sparsest-fit algorithm correctly assigns all the indices to each of the A_ks. Thus, the algorithm recovers p_k, A_k for $1 \leq k \leq K$ under Assumptions C1, C2 and C3. We summarize it in the following Lemma.

Lemma IV.1: The sparsest-fit algorithm recovers p_k, A_k for $1 \leq k \leq K$ under Assumptions C1, C2 and C3.

Complexity of the Algorithm: Initially, we sort at most D_λ^2 elements. This has a complexity of $O(D_\lambda^2 \log D_\lambda)$. Further, note that the **for** loop in the algorithm iterates for at most D_λ^2 times. In each iteration, we are solving a subset-sum problem. Since there are at most K elements, the worst-case complexity of subset-sum in each iteration is $O(2^K)$. Thus, the worst-case complexity of the algorithm is $O(D_\lambda^2 \log D_\lambda + D_\lambda^2 2^K)$. However, using the standard balls and bins argument, we can prove that for $K = O(D_\lambda \log D_\lambda)$, with a high probability, there are at most $O(\log D_\lambda)$ elements in each subset-sum problem. Thus, the complexity would then be $O(\exp(\log^2 D_\lambda))$ with a high probability.

V. Proof of Theorem III.1

The proof of Theorem III.1 requires us to establish two claims: under Condition 1, (i) the sparsest-fit algorithm finds f and (ii) the ℓ_0 optimization (1) has f as it's unique solution. We establish these two claims in that order.

The Sparsest-Fit Algorithm Works: As noted in Section IV, the sparsest-fit algorithm can be used to recover f from $\hat{f}(\lambda)$. As per Lemma IV.1, the correctness of the sparsest-fit algorithm follows under Assumptions C1, C2 and C3. The Assumption C1 is trivially satisfied in the context of recovering f from $\hat{f}(\lambda)$ as discussed in Section IV. Next, we show that Condition 1 implies C2 and C3. Note that the *unique witness* of Condition 1 implies C2 while C3 is a direct implication of *linear independence* of Condition 1. Therefore, we have established that the sparsest-fit algorithm recovers f from $\hat{f}(\lambda)$ under Condition 1.

Unique Solution of ℓ_0 Optimization: To arrive at a contradiction, assume that there exists a function $g: S_n \to \mathbb{R}_+$ such that $\hat{g}(\lambda) = \hat{f}(\lambda)$ and $L \triangleq ||g||_{\ell_0} \leq ||f||_{\ell_0} = K$. Let

$$\text{supp}(f) = \{\sigma_k \in S_n : 1 \leq k \leq K\},\ f(\sigma_k) = p_k, 1 \leq k \leq K$$
$$\text{supp}(g) = \{\rho_\ell \in S_n : 1 \leq \ell \leq L\}, g(\rho_\ell) = q_\ell, 1 \leq \ell \leq L.$$

By hypothesis of Theorem III.1, f satisfies Condition 1. Therefore, entries of matrix $\hat{f}(\lambda)$ contains the values $p_1, p_2, \ldots, p_K$. Also, by our assumption $\hat{f}(\lambda) = \hat{g}(\lambda)$. Now, by definition, each entry of the matrix $\hat{g}(\lambda)$ is a summation of a subset of L numbers, $q_\ell, 1 \leq \ell \leq L$. Therefore, it follows that for each $k, 1 \leq k \leq K$, we have

$$p_k = \sum_{j \in T_k} q_j, \qquad \text{for some} \quad T_k \subseteq \{1, 2, \ldots, L\}.$$

Equivalently

$$p = Aq \tag{5}$$

where $p = [p_k]_{1 \leq k \leq K}, q = [q_\ell]_{1 \leq \ell \leq L}$ $A \in \{0, 1\}^{K \times L}$.

Now consider the matrix $\hat{f}(\lambda)$. As noted before, each of its entries is a summation of a subset of numbers $p_k, 1 \leq k \leq K$. Further, each $p_k, 1 \leq k \leq K$ contributes to exactly D_λ distinct entries of $\hat{f}(\lambda)$. Therefore, it follows that the summation of all entries of $\hat{f}(\lambda)$ is $D_\lambda(p_1 + \cdots + p_K)$. That is

$$\sum_{ij} \hat{f}(\lambda)_{ij} = D_\lambda \left(\sum_{k=1}^{K} p_k \right).$$

Similarly

$$\sum_{ij} \hat{g}(\lambda)_{ij} = D_\lambda \left(\sum_{\ell=1}^{L} q_\ell \right).$$

But $\hat{f}(\lambda) = \hat{g}(\lambda)$. Therefore

$$p \cdot \mathbf{1} = q \cdot \mathbf{1} \tag{6}$$

where $\mathbf{1}$ is vector of all 1s of appropriate dimension (we have abused the notation $\mathbf{1}$ here): in LHS, it is of dimension K, in RHS it is of dimension L. Also, from (5) we have

$$\begin{aligned} p \cdot \mathbf{1} &= Aq \cdot \mathbf{1} \\ &= \sum_{\ell=1}^{L} c_\ell q_\ell, \end{aligned} \tag{7}$$

for some $c_j \in \mathbb{Z}_+$. From (6) and (7), it follows that

$$\sum_j q_j = \sum_j c_j q_j. \tag{8}$$

Now, there are two options: (1) either all the c_ℓs are >0, or (2) some of them are equal to zero. In the case (1), when $c_\ell > 0$ for all $1 \leq \ell \leq L$, it follows that $c_\ell = 1$ for each $1 \leq \ell \leq L$; or else, RHS of (8) will be strictly larger than LHS since $q_\ell > 0$ for all $1 \leq \ell \leq L$ by definition. Therefore, the matrix A in (5) must contain exactly one nonzero entry, i.e., 1, in each column. Since $p_k > 0$ for all $1 \leq k \leq K$, it follows that there must be at least K nonzero entries in A. Finally, since $L \leq K$, it follows that we must have $L = K$. In summary, it must be that A is a $K \times K$ matrix with each row and column having exactly one 1, and rest of the entries 0. That is, A is a permutation matrix. That is, $p_k, 1 \leq k \leq K$ is permutation of $q_1, \ldots, q_L$ with $L = K$. By relabeling the q_ℓs, if required, without loss of generality, we assume that $p_k = q_k$, for $1 \leq k \leq K$. Since $\hat{g}(\lambda) = \hat{f}(\lambda)$ and $p_k = q_k$ for $1 \leq k \leq K$, it follows that g also satisfies Condition 1. Therefore, the sparsest-fit algorithm accurately recovers g from $\hat{g}(\lambda)$. Since the input to the algorithm is only $\hat{g}(\lambda)$ and $\hat{g}(\lambda) = \hat{f}(\lambda)$, it follows that $g = f$ and we have reached contradiction to our assumption that f is not the unique solution of optimization problem (1).

Now consider the remaining case (2) and suppose that $c_\ell = 0$ for some ℓ. Then, it follows that some of the columns in the A matrix are zeros. Removing those columns of A we can write

$$p = \tilde{A}\tilde{q}$$

where $\tilde{A}$ is formed from A by removing the zero columns and $\tilde{q}$ is formed from q by removing q_ℓs such that $c_\ell = 0$. Let $\tilde{L}$ be the size of $\tilde{q}$. Since at least one column was removed, $\tilde{L} < L \leq K$. The condition $\tilde{L} < K$ implies that the vector p lies in a lower dimensional space. Further, $\tilde{A}$ is a $0, 1$ valued matrix. Therefore, it follows that p violates the linear independence property of Condition 1 resulting in a contradiction. This completes the proof of Theorem III.1.

VI. Proof of Theorem III.2

We prove this theorem by showing that when two permutations, say σ_1, σ_2, are chosen uniformly at random, with a high probability, the sum of their representation matrices $M^\lambda(\sigma_1) + M^\lambda(\sigma_2)$ can be decomposed in at least two ways. For that, note that a permutation can be represented using cycle notation, e.g., for $n = 4$, the permutation $1 \mapsto 2, 2 \mapsto 1, 3 \mapsto 4, 4 \mapsto 3$ can be represented as a composition of two cycles $(12)(34)$. We call two cycles *distinct* if they have no elements in common, e.g., the cycles (12) and (34) are distinct. Given two permutations σ_1 and σ_2, let $\sigma_{1,2} = \sigma_1\sigma_2$ be their composition.

Now consider two permutations σ_1 and σ_2 such that they have distinct cycles. For example, $\sigma_1 = (1, 2)$ and $\sigma_2 = (3, 4)$ are permutations with distinct cycles. Then $\sigma_{1,2} = \sigma_1\sigma_2 = (12)(34)$. We first prove the theorem for $\lambda = (n-1, 1)$ and then extend it to a general λ; thus, we fix the partition $\lambda = (n-1, 1)$. Then, we have

$$M^\lambda(\sigma_1) + M^\lambda(\sigma_2) = M^\lambda(\sigma_{1,2}) + M^\lambda(\mathrm{id}) \tag{9}$$

where σ_1 and σ_2 have distinct cycles and id is the identity permutation. Now, assuming that $p_1 \leq p_2$, consider the following:

$$\begin{aligned} &p_1 M^\lambda(\sigma_1) + p_2 M^\lambda(\sigma_2) \\ =&p_1 M^\lambda(\sigma_{1,2}) + p_1 M^\lambda(\mathrm{id}) + (p_2 - p_1) M^\lambda(\sigma_2). \end{aligned}$$

Thus, given $\hat{f}(\lambda) = p_1 M^\lambda(\sigma_1) + p_2 M^\lambda(\sigma_2)$, it can be decomposed in two distinct ways with both having the same ℓ_1 norm. Of course, the same analysis can be carried out when f has a sparsity K. Thus, we conclude that whenever f has two permutations with distinct cycles in its support, the ℓ_1 minimization solution is not unique. Therefore, to establish claim of Theorem III.2, it is sufficient to prove that when we choose two permutations uniformly at random, they have distinct cycles with a high probability.

To this end, let $\mathscr{E}$ denote the event that two permutations chosen uniformly at random have distinct cycles. Since permutations are chosen uniformly at random, $\Pr(\mathscr{E})$ can be computed by fixing one of the permutations to be id. Then, $\Pr(\mathscr{E})$ is the probability that a permutation chosen at random has more than one cycle.

Let us evaluate $\Pr(\mathscr{E}^c)$. For that, consider a permutation having exactly one cycle with the cycle containing l elements. The number of such permutations will be $\binom{n}{l}(l-1)!$. This is because we can choose the l elements that form the cycle in $\binom{n}{l}$ ways and the l numbers can be arranged in the cycle in $(l-1)!$ ways. Therefore

$$\Pr(\mathscr{E}^c) = \frac{1}{n!}\sum_{l=1}^{n}\binom{n}{l}(l-1)! = \sum_{r=1}^{n}\frac{1}{l(n-l)!}. \tag{10}$$

Now, without loss of generality let's assume that n is even. Then

$$\sum_{l=1}^{n/2}\frac{1}{l(n-l)!} \leq \sum_{l=1}^{n/2}\frac{1}{(\frac{n}{2})!} = \frac{1}{(\frac{n}{2}-1)!}. \tag{11}$$

The other half of the sum becomes

$$\sum_{l=n/2}^{n}\frac{1}{l(n-l)!} \leq \sum_{k=0}^{n/2}\frac{1}{\frac{n}{2}k!} \leq \frac{2}{n}\sum_{k=0}^{\infty}\frac{1}{k!} \leq \frac{O(1)}{n}. \tag{12}$$

Putting everything together, we have

$$\begin{aligned} \Pr(\mathscr{E}) \geq 1 - \Pr(\mathscr{E}^c) &\geq 1 - \left(\frac{1}{(\frac{n}{2}-1)!} + \frac{O(1)}{n}\right) \\ &\to 1 \text{ as } n \to \infty. \end{aligned}$$

Thus, Theorem III.2 is true for $\lambda = (n-1, 1)$.

In order to extend the proof to a general λ, we observe that the standard cycle notation for a permutation we discussed above can be extended to λ partitions for a general λ. Specifically, for any given λ, observe that a permutation can be imagined as a perfect matching in a $D_\lambda \times D_\lambda$ bipartite graph, which we call the λ-bipartite graph and denote it by $G^\lambda = (V_1^\lambda \times V_2^\lambda, E^\lambda)$; here V_1^λ and V_2^λ respectively denote the left and right vertex sets with $|V_1^\lambda| = |V_2^\lambda| = D_\lambda$ with a node for every λ partition of n. Let $t_1, t_2, \ldots, t_{D_\lambda}$ denote the D_λ λ-partitions of n; then, the nodes in V_1^λ and V_2^λ can be labeled by $t_1, t_2, \ldots, t_{D_\lambda}$. Since every perfect matching in a bipartite graph can be decomposed into its corresponding distinct cycles (the cycles can be obtained by superposing the bipartite graph corresponding to identity permutation with the λ-bipartite graph of the permutation), every permutation can be written as a combination of distinct cycles in its λ-bipartite graph. The special case of this for $\lambda = (n-1, 1)$ is the standard cycle notation we discussed above; for brevity, we call the λ-bipartite graph for $\lambda = (n-1, 1)$ the standard bipartite graph.

In order to prove the theorem for a general λ, it follows from an argument as above that it is sufficient to prove that a randomly chosen permutation contains at least two distinct cycles in its λ-bipartite graph with a high probability. We prove this by that a permutation with at least two distinct cycles in its standard bipartite graph has at least two distinct cycles in its λ-bipartite graph for any general λ. The theorem then follows from the result we established above that a randomly chosen permutation has at least two distinct cycles in its standard bipartite graph with a high probability.

To that end, consider a permutation, σ, with at least two distinct cycles in the standard bipartite graph. Let $A := (a_1, a_2, \ldots, a_{\ell_1})$ and $B := (b_1, b_2, \ldots, b_{\ell_2})$ denote the first two cycles in the standard bipartite graph; clearly, $\ell_1 \ell_2 \geq 2$ and at least one of ℓ_1, ℓ_2 is $\leq n/2$. Without loss of generality we assume that $\ell_2 \leq n/2$. Let $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_r)$. Since $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_r$, we have $\lambda_r \leq n/2$. First, we consider the case when $\lambda_r < n/2$. Now consider the λ-partition, t_1, of n constructed as follows: a_1 placed in the rth partition, a_2 in the first partition, all the elements of the second cycle $b_1, b_2, \ldots, b_{\ell_2}$ arbitrarily in the first $r-1$ partitions and the rest placed arbitrarily. Note that such a construction is possible by the assumption on λ_r. Let t_1' denote $\sigma(t_1)$; then, $t_1' \neq t_1$ because t_1 does not contain a_2 in the rth partition while t_1' contains $\sigma(a_1) = a_2$ in the rth partition. Thus, the partition t_1 belongs to a cycle that has a length of at least 2 partitions. Thus, we have found one cycle, which we denote by C_1. Now consider a second partition t_2 constructed as follows: b_1 placed in the rth partition, b_2 in the first and the rest placed arbitrarily. Again, note that $\sigma(t_2) \neq t_2$. Thus, t_2 belongs to a cycle of length at least 2, which we denote by C_2. Now we have found two cycles C_1, C_2, and we are left with proving that they are distinct. In order to establish the cycles are distinct, note that none of the partitions in cycle C_1 can be t_2. This is true because, by construction, t_2 contains b_1 in the rth partition while none of the partitions in C_1 can contain any elements from the cycle B in the rth partition. This finishes the proof for all λ such that $\lambda_r < n/2$.

We now consider the case when $\lambda_r = n/2$. Since $\lambda_1 \geq \lambda_r$, it follows that $r = 2$ and $\lambda = (n/2, n/2)$. For $\ell_2 < n/2$, it is still feasible to construct t_1 and t_2, and the theorem follows from the arguments above. Now we consider the case when $\ell_1 = \ell_2 = n/2$; let $\ell := \ell_1 = \ell_2$. Note that now it is infeasible to construct t_1 as described above. Therefore, we consider $t_1 = \{a_1, b_2, \ldots, b_\ell\}\{b_1, a_2, \ldots, a_\ell\}$ and $t_2 = \{b_1, a_2, \ldots, a_\ell\}\{a_1, b_2, \ldots, b_\ell\}$. Clearly, $t_1 \neq t_2$, $\sigma(t_1) \neq t_1$ and $\sigma(t_2) \neq t_2$. Thus, t_1 and t_2 belong to two cycles, C_1 and C_2, each with length at least 2. It is easy to see that these cycles are also distinct because every λ-partition in the cycle C_1 will have only one element from cycle A in the first partition and, hence, C_1 cannot contain the λ-partition t_2. This completes the proof of the theorem.

VII. Proof of Theorem III.3: $\lambda = (n-1, 1)$

Our interest is in recovering a random function f from partial information $\hat{f}(\lambda)$. To this end, let

$$K = \|f\|_0, \quad \text{supp}(f) = \{\sigma_k \in S_n : 1 \leq k \leq K\} \quad \text{and} \quad f(\sigma_k) = p_k, 1 \leq k \leq K.$$

Here σ_k and p_k are randomly chosen as per the random model $R(K, \mathscr{C})$ described in Section II. For $\lambda = (n-1, 1)$, $D_\lambda = n$; then $\hat{f}(\lambda)$ is an $n \times n$ matrix with its (i,j)th entry being

$$\hat{f}(\lambda)_{ij} = \sum_{k:\sigma_k(j)=i} p_k, \qquad \text{for} \qquad 1 \leq i, j \leq n.$$

To establish Theorem III.3, we prove that as long as $K \leq C_1 n \log n$ with $C_1 = 1 - \varepsilon$, f can be recovered by the sparsest-fit algorithm with probability $1 - o(1)$ for any fixed $\varepsilon > 0$. Specifically, we show that for $K \leq C_1 n \log n$, Condition 1 is satisfied with probability $1 - o(1)$, which in turn implies that the sparsest-fit algorithm recovers f as per Theorem III.1. Note that the "linear independence" property of Condition 1 is satisfied with probability 1 under $R(K, \mathscr{C})$ as p_k are chosen from a distribution with continuous support. Therefore, we are left with establishing "unique witness" property.

To this end, let $4\delta = \varepsilon$ so that $C_1 \leq 1 - 4\delta$. Let $\mathscr{E}_k$ be the event that σ_k satisfies the unique witness property, $1 \leq k \leq K$. Under $R(K, \mathscr{C})$, since K permutations are chosen from S_n independently and uniformly at random, it follows that $\Pr(\mathscr{E}_k)$ is the same for all k. Therefore, by union bound, it is sufficient to establish that $K \Pr(\mathscr{E}_1^c) = o(1)$. Since we are interested in $K = O(n \log n)$, it is sufficient to establish $\Pr(\mathscr{E}_1^c) = O(1/n^2)$. Finally, once again due the symmetry, it is sufficient to evaluate $\Pr(\mathscr{E}_1)$ assuming $\sigma_1 = \text{id}$, i.e., $\sigma_1(i) = i$ for all $1 \leq i \leq n$. Define

$$\mathscr{F}_j = \{\sigma_k(j) \neq j, \quad \text{for} \quad 2 \leq k \leq K\}, \text{ for } 1 \leq j \leq n.$$

It then follows that

$$\Pr(\mathscr{E}_1) = \Pr\left(\cup_{j=1}^{n} \mathscr{F}_j\right).$$

Therefore, for any $L \leq n$, we have

$$\begin{aligned}\Pr(\mathscr{E}_1^c) &= \Pr\left(\cap_{j=1}^{n} \mathscr{F}_j^c\right) \\ &\leq \Pr\left(\cap_{j=1}^{L} \mathscr{F}_j^c\right) \\ &= \Pr\left(\mathscr{F}_1^c\right)\left[\prod_{j=2}^{L} \Pr\left(\mathscr{F}_j^c \middle| \cap_{\ell=1}^{j-1} \mathscr{F}_\ell^c\right)\right]. \quad (13)\end{aligned}$$

Next we show that for the selection of $L = n^{1-\delta}$, the RHS of (13) is bounded above by $\exp(-n^{\delta}) = O(1/n^2)$. That will complete the proof of achievability.

For that, we start by bounding $\Pr(\mathscr{F}_1^c)$

$$\begin{aligned}\Pr\left(\mathscr{F}_1^c\right) &= 1 - \Pr(\mathscr{F}_1) \\ &= 1 - \left(1 - \frac{1}{n}\right)^{K-1}. \quad (14)\end{aligned}$$

The last equality follows because all permutations are chosen uniformly at random. For $j \geq 2$, we now evaluate $\Pr\left(\mathscr{F}_j^c \middle| \cap_{\ell=1}^{j-1} \mathscr{F}_\ell^c\right)$. Given $\cap_{\ell=1}^{j-1} \mathscr{F}_\ell^c$, for any $k, 2 \leq k \leq K$, $\sigma_k(j)$ will take a value from $n-j+1$ values, possibly including j, uniformly at random. Thus, we obtain the following bound:

$$\Pr\left(\mathscr{F}_j^c \middle| \cap_{\ell=1}^{j-1} \mathscr{F}_\ell^c\right) \leq 1 - \left(1 - \frac{1}{n-j+1}\right)^{K-1}. \quad (15)$$

From (13)–(15), we obtain that

$$\begin{aligned}\Pr(\mathscr{E}_1^c) &\leq \prod_{j=1}^{L}\left(1 - \left(1 - \frac{1}{n-j+1}\right)^{K-1}\right) \\ &\leq \left[1 - \left(1 - \frac{1}{n-L}\right)^{K}\right]^{L} \\ &\leq \left[1 - \left(1 - \frac{1}{n-L}\right)^{C_1 n \log n}\right]^{L} \quad (16)\end{aligned}$$

where we have used $K \leq C_1 n \log n$ in the last inequality. Since $L = n^{1-\delta}, n - L = n(1 - o(1))$. Using the standard fact $1 - x = e^{-x}(1 + O(x^2))$ for small $x \in [0,1)$, we have

$$\left(1 - \frac{1}{n-L}\right) = \exp\left(-\frac{1}{n-L}\right)\left(1 + O\left(\frac{1}{n^2}\right)\right). \quad (17)$$

Finally, observe that

$$\left(1 + O\left(\frac{1}{n^2}\right)\right)^{C_1 n \log n} = \Theta(1).$$

Therefore, from (16) and (17), it follows that

$$\begin{aligned}\Pr(\mathscr{E}_1^c) &\leq \left[1 - \Theta\left(\exp\left(-\frac{C_1 \log n}{1 - n^{-\delta}}\right)\right)\right]^{L} \\ &\leq \left[1 - \Theta\left(\exp(-(C_1 + \delta)\log n)\right)\right]^{L} \\ &= \left[1 - \Theta\left(\frac{1}{n^{C_1+\delta}}\right)\right]^{L} \\ &\leq \exp\left(-\Theta\left(\frac{L}{n^{C_1+\delta}}\right)\right) \\ &= \exp(-\Omega(n^{2\delta})) \quad (18)\end{aligned}$$

where we have used the fact that $1 - x \leq e^{-x}$ for $x \in [0,1]$ and $L = n^{1-\delta}, C_1 \leq 1 - 4\delta$. From (18), it follows that $\Pr(\mathscr{E}_1) = O(1/n^2)$. This completes the proof of achievability.

VIII. Proof of Theorem III.4 : $\lambda = (n-m, m)$

Our interest is in recovering the random function f from partial information $\hat{f}(\lambda)$. As in proof of Theorem III.3, we use the notation

$$K = \|f\|_0, \quad \text{supp}(f) = \{\sigma_k \in S_n : 1 \leq k \leq K\} \quad \text{and} \quad f(\sigma_k) = p_k, 1 \leq k \leq K.$$

Here σ_k and p_k are randomly chosen as per the random model $R(K, \mathscr{C})$ described in Section II. For $\lambda = (n-m, m)$, $D_\lambda = \frac{n!}{(n-m)!m!} \sim n^m$ and $\hat{f}(\lambda)$ is an $D_\lambda \times D_\lambda$ matrix.

To establish Theorem III.4, we shall prove that as long as $K \leq C_1 n^m \log n$ with $0 < C_1 < \frac{1}{m!}$ a constant, f can be recovered by the sparsest-fit algorithm with probability $1 - o(1)$. We shall do so by verifying that the Condition 1 holds with probability $1 - o(1)$, so that the sparsest-fit algorithm will recover f as per Theorem III.1. As noted earlier, the "linear independence" of Condition 1 is satisfied with probability 1 under $R(K, \mathscr{C})$. Therefore, we are left with establishing the "unique witness" property.

To this end, for the purpose of bounding, without loss of generality, let us assume that $K = \frac{(1-2\delta)}{m!} n^m \log n$ for some $\delta > 0$. Set $L = n^{1-\delta}$. Following arguments similar to those in the proof of Theorem III.3, it will be sufficient to establish that $\Pr(\mathscr{E}_1^c) = O(1/n^{2m})$; where $\mathscr{E}_1$ is the event that permutation $\sigma_1 = \text{id}$ satisfies the unique witness property.

To this end, recall that $\hat{f}(\lambda)$ is a $D_\lambda \times D_\lambda$ matrix. Each row (and column) of this matrix corresponds to a distinct λ partition of $n : t_i, 1 \leq i \leq D_\lambda$. Without loss of generality, let us order the D_λ λ partitions of n so that the ith partition, t_i, is defined as follows: $t_1 = \{1, \ldots, n-m\}\{n-m+1, \ldots, n\}$, and for $2 \leq i \leq L$

$$t_i = \{1, \ldots, n - im, n - (i-1)m + 1, \ldots, n\} \{n - im + 1, \ldots, n - (i-1)m\}.$$

Note that since $\sigma_1 = \text{id}$, we have $\sigma_1(t_i) = t_i$ for all $1 \leq i \leq D_\lambda$. Define

$$\mathscr{F}_j = \{\sigma_k(t_j) \neq t_j, \quad \text{for} \quad 2 \leq k \leq K\}, \text{ for } 1 \leq j \leq D_\lambda.$$

Then it follows that

$$\Pr(\mathscr{E}_1) = \Pr\left(\cup_{j=1}^{D_\lambda} \mathscr{F}_j\right).$$

Therefore

$$\begin{aligned}\Pr(\mathscr{E}_1^c) &= \Pr\left(\cap_{j=1}^{D_\lambda} \mathscr{F}_j^c\right)\\ &\le \Pr\left(\cap_{j=1}^{L} \mathscr{F}_j^c\right)\\ &= \Pr\left(\mathscr{F}_1^c\right)\left[\prod_{j=2}^{L} \Pr\left(\mathscr{F}_j^c \middle| \cap_{\ell=1}^{j-1} \mathscr{F}_\ell^c\right)\right]. \quad (19)\end{aligned}$$

First, we bound $\Pr(\mathscr{F}_1^c)$. Each permutation $\sigma_k, k \neq 1$, maps $t_1 = \{1,\dots,n-m\}\{n-m+1,\dots,n\}$ to $\{\sigma_k(1),\dots\sigma_k(n-m)\}\{\sigma_k(n-m+1),\dots,\sigma_k(n)\}$. Therefore, $\sigma_k(t_1) = t_1$ iff σ_k maps set of elements $\{n-m+1,\dots,n\}$ to the same set of elements. Therefore

$$\begin{aligned}\Pr(\sigma_k(t_1) = t_1) &= \frac{1}{\binom{n}{m}}\\ &= \frac{m!}{\prod_{\ell=0}^{m-1}(n-\ell)}.\\ &\le \frac{m!}{(n-Lm)^m}. \quad (20)\end{aligned}$$

Therefore, it follows that

$$\begin{aligned}\Pr\left(\mathscr{F}_1^c\right) &= 1 - \Pr(\mathscr{F}_1)\\ &= 1 - \Pr(\sigma_k(t_1) \neq t_1, 2 \le k \le K)\\ &= 1 - \prod_{k=2}^{K}(1 - \Pr(\sigma_k(t_1) = t_1))\\ &\le 1 - \left(1 - \frac{m!}{(n-Lm)^m}\right)^K. \quad (21)\end{aligned}$$

Next we evaluate $\Pr\left(\mathscr{F}_j^c \middle| \cap_{\ell=1}^{j-1} \mathscr{F}_\ell^c\right)$ for $2 \le j \le L$. Given $\cap_{\ell=1}^{j-1} \mathscr{F}_\ell^c$, we have (at least partial) information about the action of $\sigma_k, 2 \le k \le K$ over elements $\{n-(j-1)m+1,\dots,n\}$. Conditional on this, we are interested in the action of σ_k on t_j, i.e., $\{n-jm+1,\dots,n-jm+m\}$. Specifically, we want to (upper) bound the probability that these elements are mapped to themselves. Given $\cap_{\ell=1}^{j-1} \mathscr{F}_\ell^c$, each σ_k will map $\{n-jm+1,\dots,n-jm+m\}$ to one of the $\binom{n-(j-1)m}{m}$ possibilities with equal probability. Further, $\{n-jm+1,\dots,n-jm+m\}$ is not a possibility. Therefore, for the purpose of upper bound, we obtain that

$$\begin{aligned}\Pr\left(\mathscr{F}_j^c \middle| \cap_{\ell=1}^{j-1} \mathscr{F}_\ell^c\right) &\le 1 - \left(1 - \frac{1}{\binom{n-(j-1)m}{m}}\right)^{K-1}\\ &\le 1 - \left(1 - \frac{m!}{(n-Lm)^m}\right)^K. \quad (22)\end{aligned}$$

From (19)–(22), we obtain that

$$\Pr(\mathscr{E}_1^c) \le \left[1 - \left(1 - \frac{m!}{(n-Lm)^m}\right)^K\right]^L. \quad (23)$$

Now $Lm = o(n)$ and hence, $n - Lm = n(1-o(1))$. Using $1 - x = e^{-x}(1+O(x^2))$ for small $x \in [0,1)$, we have

$$\begin{aligned}&\left(1 - \frac{m!}{(n-Lm)^m}\right)\\ &= \exp\left(-\frac{m!}{(n-Lm)^m}\right)\left(1 + O\left(\frac{1}{n^{2m}}\right)\right). \quad (24)\end{aligned}$$

Finally, observe that since $K = O(n^m \log n)$

$$\left(1 + O\left(\frac{1}{n^{2m}}\right)\right)^K = \Theta(1).$$

Thus, from (23) and (24), it follows that

$$\begin{aligned}\Pr(\mathscr{E}_1^c) &\le \left[1 - \Theta\left(\exp\left(-\frac{Km!}{n^m(1-Lm/n)^m}\right)\right)\right]^L\\ &\le \left[1 - \Theta\left(\exp\left(-\frac{(1-2\delta)\log n}{(1-n^{-\delta}m)^m}\right)\right)\right]^L\\ &\le [1 - \Theta(\exp(-(1-3\delta/2)\log n))]^L\\ &\le \left[1 - \Theta\left(\frac{1}{n^{1-3\delta/2}}\right)\right]^L\\ &\le \exp(-\Omega(Ln^{-1+3\delta/2}))\\ &\le \exp(-\Omega(n^{\delta/2}))\\ &= O\left(\frac{1}{n^{2m}}\right). \quad (25)\end{aligned}$$

In above, we have used the fact that $1 - x \le e^{-x}$ for $x \in [0,1]$ and choice of $L = n^{1-\delta}$. This completes the proof of Theorem III.4.

IX. Proof of Theorem III.5: $\lambda_1 = n - n^{\frac{2}{9}-\delta}, \delta > 0$

So far we have obtained the sharp result that the sparsest-fit algorithm recovers f up to sparsity essentially $\frac{1}{m!}n^m \log n$ for λ with $\lambda_1 = n - m$ where $m = O(1)$. Now we investigate this further when m scales with n, i.e., $m = \omega(1)$. Let $\lambda_1 = n - \mu$ with $\mu \le n^{\frac{2}{9}-\delta}$ for some $\delta > 0$. For such $\lambda = (\lambda_1,\dots,\lambda_r)$

$$\begin{aligned}D_\lambda &= \frac{n!}{\prod_{i=1}^{r}\lambda_i!}\\ &\le \frac{n!}{\lambda_1!}\\ &\le n^{n-\lambda_1} = n^\mu. \quad (26)\end{aligned}$$

Our interest is in the case when $K \le (1-\varepsilon)D_\lambda \log\log D_\lambda$ for any $\varepsilon > 0$. For this, the structure of arguments will be similar to those used in Theorems III.3 and III.4. Specifically, it will be sufficient to establish that $\Pr(\mathscr{E}_1^c) = O(1/D_\lambda^2)$, where $\mathscr{E}_1$ is the event that permutation $\sigma_1 = \mathrm{id}$ satisfies the unique witness property.

To this end, we order the rows (and corresponding columns) of the $D_\lambda \times D_\lambda$ matrix $\hat{f}(\lambda)$ in a specific manner. Specifically, we are interested in the $L = 3n^{\frac{4}{9}-2\delta}\log^3 n$ rows that we call $t_\ell, 1 \le \ell \le L$ and they are as follows: the first row, t_1 corresponds to a partition where elements $\{1,\dots,\lambda_1\}$ belong to the first partition and $\{\lambda_1+1,\dots,n\}$ are partitioned into remaining $r-1$ parts of size $\lambda_2,\dots,\lambda_r$ in that order.

The partition t_2 corresponds to the one in which the first part contains the λ_1 elements $\{1,\ldots,n-2\mu,n-\mu+1,\ldots,n\}$, while the other $r-1$ parts contain $\{n-2\mu+1,\ldots,n-\mu\}$ in that order. More generally, for $3 \le \ell \le L, t_\ell$ contains $\{1,\ldots,n-\ell\mu,n-(\ell-1)\mu+1,\ldots,n\}$ in the first partition and remaining elements $\{n-\ell\mu+1,\ldots,n-(\ell-1)\mu\}$ in the rest of the $r-1$ parts in that order. By our choice of $L, L\mu = o(n)$ and, hence, the above is well defined. Next, we bound $\Pr(\mathscr{E}_1^c)$ using these L rows.

Now $\sigma_1 = \mathrm{id}$ and hence, $\sigma_1(t_i) = t_i$ for all $1 \le i \le D_\lambda$. Define

$$\mathscr{F}_j = \{\sigma_k(t_j) \neq t_j, \quad \text{for} \quad 2 \le k \le K\}, \text{ for } 1 \le j \le D_\lambda.$$

Then it follows that

$$\Pr(\mathscr{E}_1) = \Pr\left(\cup_{j=1}^{D_\lambda} \mathscr{F}_j\right).$$

Therefore

$$\begin{aligned}\Pr(\mathscr{E}_1^c) &= \Pr\left(\cap_{j=1}^{D_\lambda} \mathscr{F}_j^c\right) \\ &\le \Pr\left(\cap_{j=1}^{L} \mathscr{F}_j^c\right) \\ &= \Pr\left(\mathscr{F}_1^c\right)\left[\prod_{j=2}^{L} \Pr\left(\mathscr{F}_j^c \,\middle|\, \cap_{\ell=1}^{j-1} \mathscr{F}_\ell^c\right)\right]. \quad (27)\end{aligned}$$

First, we bound $\Pr(\mathscr{F}_1^c)$. Each permutation $\sigma_k, 1 \le k \le K$ maps t_1 to one of the D_λ possible other λ partitions with equal probability. Therefore, it follows that

$$\Pr(\sigma_k(t_1) = t_1) = \frac{1}{D_\lambda}. \quad (28)$$

Thus

$$\begin{aligned}\Pr\left(\mathscr{F}_1^c\right) &= 1 - \Pr(\mathscr{F}_1) \\ &= 1 - \Pr\left(\sigma_k(t_1) \neq t_1, 2 \le k \le K\right) \\ &= 1 - \prod_{k=2}^{K} \left(1 - \Pr\left(\sigma_k(t_1) = t_1\right)\right) \\ &= 1 - \left(1 - \frac{1}{D_\lambda}\right)^K. \quad (29)\end{aligned}$$

Next we evaluate $\Pr\left(\mathscr{F}_j^c \,\middle|\, \cap_{\ell=1}^{j-1} \mathscr{F}_\ell^c\right)$ for $2 \le j \le L$. Given $\cap_{\ell=1}^{j-1} \mathscr{F}_\ell^c$, we have (at least partial) information about the action of $\sigma_k, 2 \le k \le K$ over elements $\{n-(j-1)\mu+1,\ldots,n\}$. Conditional on this, we are interested in the action of σ_k on t_j. Given the partial information, each of the σ_k will map t_j to one of at least $D_{\lambda(j)}$ different options with equal probability for $\lambda(j) = (\lambda_1 - (j-1)\mu, \lambda_2, \ldots, \lambda_r)$—this is because the elements $1,\ldots,\lambda_1-(j-1)\mu$ in the first part and all elements in the remaining $r-1$ parts are mapped completely randomly conditional on $\cap_{\ell=1}^{j-1} \mathscr{F}_\ell^c$. Therefore, it follows that

$$\Pr\left(\mathscr{F}_j^c \,\middle|\, \cap_{\ell=1}^{j-1} \mathscr{F}_\ell^c\right) \le 1 - \left(1 - \frac{1}{D_{\lambda(j)}}\right)^K. \quad (30)$$

From (27)–(30), we obtain that

$$\begin{aligned}\Pr(\mathscr{E}_1^c) &\le \prod_{j=1}^{L}\left[1 - \left(1 - \frac{1}{D_{\lambda(j)}}\right)^K\right] \\ &\le \left[1 - \left(1 - \frac{1}{D_{\lambda(L)}}\right)^K\right]^L. \quad (31)\end{aligned}$$

In the above, we have used the fact that

$$D_\lambda = D_{\lambda(1)} \ge \ldots \ge D_{\lambda(L)}.$$

Consider

$$\begin{aligned}\frac{D_{\lambda(j)}}{D_{\lambda(j+1)}} &= \frac{(n-(j-1)\mu)!(\lambda_1 - j\mu)!}{(n-j\mu)!(\lambda_1-(j-1)\mu)!} \\ &= \prod_{\ell=0}^{\mu-1} \frac{(n-(j-1)\mu-\ell)}{(\lambda_1-(j-1)\mu-\ell)} \\ &= \left(\frac{n}{\lambda_1}\right)^\mu \prod_{\ell=0}^{\mu-1} \frac{1 - \frac{(j-1)\mu-\ell}{n}}{1 - \frac{(j-1)\mu-\ell}{\lambda_1}}. \quad (32)\end{aligned}$$

Therefore, it follows that

$$\frac{D_{\lambda(1)}}{D_{\lambda(L)}} = \left(\frac{n}{\lambda_1}\right)^{(L-1)\mu} \prod_{\ell=0}^{(L-1)\mu} \frac{1 - \frac{\ell}{n}}{1 - \frac{\ell}{\lambda_1}}. \quad (33)$$

Using $1 + x \le e^x$ for any $x \in (-1,1), 1 - x \ge e^{-2x}$ for $x \in (0,1/2)$ and $L\mu = o(n)$, we have that for any $\ell, 0 \le \ell \le (L-1)\mu$

$$\begin{aligned}\frac{1 - \frac{\ell}{n}}{1 - \frac{\ell}{\lambda_1}} &= \frac{1 - \frac{\ell}{n} + \frac{\ell}{\lambda_1} - \frac{\ell^2}{n\lambda_1}}{1 - \frac{\ell^2}{\lambda_1^2}} \\ &\le \exp\left(-\frac{\ell^2 - \ell\mu}{n\lambda_1} + \frac{2\ell^2}{\lambda_1^2}\right) \\ &\le \exp\left(\frac{\ell\mu}{n\lambda_1} + \frac{2\ell^2}{\lambda_1^2}\right). \quad (34)\end{aligned}$$

Therefore, we obtain

$$\frac{D_{\lambda(1)}}{D_{\lambda(L)}} \le \left(\frac{n}{\lambda_1}\right)^{L\mu} \exp\left(\Theta\left(\frac{L^2\mu^3}{n\lambda_1} + \frac{2L^3\mu^3}{\lambda_1^2}\right)\right). \quad (35)$$

Now

$$\begin{aligned}\left(\frac{n}{\lambda_1}\right)^{L\mu} &= \left(1 + \frac{\mu}{\lambda_1}\right)^{L\mu} \\ &\le \exp\left(\frac{L\mu^2}{\lambda_1}\right). \quad (36)\end{aligned}$$

It can be checked that for given choice of L, μ, we have $L\mu^2 = o(\lambda_1), L^3\mu^3 = o(\lambda_1^2)$ and $L^2\mu^3 = o(n\lambda_1)$. Therefore, in summary we have that

$$\frac{D_{\lambda(1)}}{D_{\lambda(L)}} = 1 + o(1). \tag{37}$$

Using similar approximations to evaluate the bound on RHS of (31) along with (26) yields

$$\begin{aligned}
\Pr(\mathscr{E}_1^c) &\le \exp\left(-L\exp\left(-\frac{K}{D_{\lambda(L)}}\right)\right) \\
&= \exp(-L\exp(-(1-\varepsilon)\log\log D_\lambda(1+o(1)))) \\
&\le \exp(-L\exp(-\log\log D_\lambda)) \\
&= \exp\left(-\frac{L}{\log D_\lambda}\right) \\
&= \exp\left(-\frac{3n^{\frac{4}{9}-2\delta}\log^3 n}{\log D_\lambda}\right) \\
&\le \exp(-2\log D_\lambda) \\
&= \frac{1}{D_\lambda^2}.
\end{aligned} \tag{38}$$

This completes the proof of Theorem III.5.

X. Proof of Theorem III.6: General λ

We shall establish the bound on sparsity up to which recovery of f is possible from $\hat{f}(\lambda)$ using the sparsest-fit algorithm for general λ. Let $\lambda = (\lambda_1, \ldots, \lambda_r), r \ge 2$ with $\lambda_1 \ge \cdots \ge \lambda_r \ge 1$. As before, let

$$K = \|f\|_0, \quad \mathrm{supp}(f) = \{\sigma_k \in S_n : 1 \le k \le K\} \quad \text{and} \quad f(\sigma_k) = p_k, 1 \le k \le K.$$

Here σ_k and p_k are randomly chosen as per the random model $R(K, \mathscr{C})$ described in Section II. And, we are given partial information $\hat{f}(\lambda)$ which is $D_\lambda \times D_\lambda$ matrix with

$$D_\lambda = \frac{n!}{\prod_{i=1}^r \lambda_i!}.$$

Finally, recall definition $\alpha = (\alpha_i)_{1 \le i \le r}$ with $\alpha_i = \lambda_i/n, 1 \le i \le r$

$$H(\alpha) = -\sum_{i=1}^r \alpha_i \log \alpha_i, \quad \text{and} \quad H'(\alpha) = -\sum_{i=2}^r \alpha_i \log \alpha_i.$$

As usual, to establish that the sparsest-fit algorithm recovers f from $\hat{f}(\lambda)$, we will need to establish "unique witness" property as "linear independence" is satisfied due to choice of p_ks as per random model $R(K, \mathscr{C})$.

For the ease of exposition, we will need an additional notation of λ-bipartite graph: it is a complete bipartite graph $G^\lambda = (V_1^\lambda \times V_2^\lambda, E^\lambda)$ with vertices V_1^λ, V_2^λ having a node each for a distinct λ partition of n and thus $|V_1^\lambda| = |V_2^\lambda| = D_\lambda$. Action of a permutation $\sigma \in S_n$, represented by a $0/1$ valued $D_\lambda \times D_\lambda$ matrix, is equivalent to a perfect matching in G^λ. In this notation, a permutation σ has "unique witness" with respect to a collection of permutations, if and only if there is an edge in the matching corresponding to σ that is not present in any other permutation's matching.

Let $\mathscr{E}_L$ denote the event that $L \ge 2$ permutations chosen uniformly at random satisfy the "unique witness" property. To establish Theorem III.6, we wish to show that $\Pr(\mathscr{E}_K^c) = o(1)$ as long as $K \le K_1^*(\lambda)$ where $K_1^*(\lambda)$ is defined as per (3). To do so, we shall study $\Pr(\mathscr{E}_{L+1}^c \,|\, \mathscr{E}_L)$ for $L \ge 1$. Now consider the bipartite graph, G_L^λ, which is subgraph of G^λ, formed by the superimposition of the perfect matchings corresponding to the L random permutations, $\sigma_i, 1 \le i \le L$. Now, the probability of $\mathscr{E}_{L+1}^c$ given that $\mathscr{E}_L$ has happened is equal to the probability that a new permutation, generated uniformly at random, has its perfect matching so that all its edges end up overlapping with those of G_L^λ. Therefore, in order to evaluate this probability we count the number such permutations.

For the ease of exposition, we will first count the number of such permutations for the cases when $\lambda = (n-1, 1)$ followed by $\lambda = (n-2, 2)$. Later, we shall extend the analysis to a general λ. As mentioned before, for $\lambda = (n-1, 1)$, the corresponding G^λ is a complete graph with n nodes on left and right. With a bit of abuse of notation, the left and right vertices be labeled $1, 2, \ldots, n$. Now each permutation, say $\sigma \in S_n$, corresponds to a perfect matching in G^λ with an edge from left i to right j if and only if $\sigma(i) = j$. Now, consider G_L^λ, the superimposition of all the perfect matching of the given L permutations. We want to count (or obtain an upper bound on) the number of permutations that will have corresponding perfect matching so that all of its edges overlap with edges of G_L^λ. Now each permutation maps a vertex on left to a vertex on right. In the graph G_L^λ, each vertex i on the left has degree of at most L. Therefore, if we wish a choose a permutation so that all of its perfect matching's edges overlap with those of G_λ^L, it has at most L choices for each vertex on left. There are n vertices in total on left. Therefore, total number of choices are bounded above by L^n. From this, we conclude that for $\lambda = (n-1, 1)$

$$\Pr(\mathscr{E}_{L+1}^c | \mathscr{E}_L) \le \frac{L^n}{n!}.$$

In a similar manner, when $\lambda = (n-2, 2)$, the complete bipartite graph G^λ has $D_\lambda = \binom{n}{2}$ nodes on the left and right; each permutation corresponds to a perfect matching in this graph. We label each vertex, on left and right, in G^λ by unordered pairs $\{i, j\}$, for $1 \le i < j \le n$. Again, we wish to bound given $\Pr(\mathscr{E}_{L+1}^c | \mathscr{E}_L)$. For this, let G_L^λ, a subgraph of G^λ, be obtained by the union of edges that belong to the perfect matchings of given L permutations. We would like to count the number possible permutations that will have corresponding matching with edges overlapping with those of G_L^λ. For this, we consider the $\lfloor n/2 \rfloor$ pairs $\{1, 2\}, \{3, 4\}, \ldots, \{2\lfloor n/2 \rfloor - 1, 2\lfloor n/2 \rfloor\}$. Now if n is even then they end up covering all n elements. If not, we consider the last, nth element, $\{n\}$ as an additional set.

Now using a similar argument as before, we conclude that there are at most $L^{\lfloor n/2 \rfloor}$ ways of mapping each of these $\lfloor n/2 \rfloor$ pairs such that all of these edges overlap with the edges of G_L^λ. Note that this mapping fixes what each of these $\lceil n/2 \rceil$ unordered

pairs get mapped to. Given this mapping, there are 2! ways of fixing the order in each unordered pair. For example, if an unordered pair $\{i,j\}$ maps to unordered pair $\{k,l\}$ there there are $2!=2$ options: : $i\mapsto k, j\mapsto l$ or $i\mapsto l, j\mapsto k$. Thus, once we fix the mapping of each of the $\lceil n/2\rceil$ disjoint unordered pairs, there can be at most $(2!)^{\lceil n/2\rceil}$ permutations with the given mapping of unordered pairs. Finally, note that once the mapping of these $\lfloor n/2\rfloor$ pairs is decided, if n is even that there is no element that is left to be mapped. For n odd, since mapping of the $n-1$ elements is decided, so is that of $\{n\}$. Therefore, in summary in both even n or odd n case, there are at most $L^{\lfloor n/2\rfloor}(2!)^{\lceil n/2\rceil}$ permutations that have all of the edge of corresponding perfect matching in G^λ overlapping with the edges of G_L^λ. Therefore

$$\Pr\left(\mathscr{E}_{L+1}^c \middle| \mathscr{E}_L\right) \le \frac{L^{\lfloor n/2\rfloor}(2!)^{\lfloor n/2\rfloor}}{n!}.$$

Now consider the case of general $\lambda=(\lambda_1,\lambda_2,\ldots,\lambda_r)$. Let $M=\lfloor n/(n-\lambda_1)\rfloor$ and $N=n-M(n-\lambda_1)$. Clearly, $0\le N<n-\lambda_1$. Now we partition the set $\{1,2,\ldots,n\}$ into $M+1$ partitions covering all elements: $\{1,\ldots,n-\lambda_1\},\ldots,\{(n-\lambda_1)(M-1)+1,\ldots,(n-\lambda_1)M\}$ and $\{(n-\lambda_1)M+1,\ldots,n\}$. As before, for the purpose of upper bounding the number of permutations that have corresponding perfect matchings in G^λ overlapping with edges of G_L^λ, each of the first M partitions can be mapped in L different ways; in total at most L^M ways. For each of these mappings, we have options at the most

$$(\lambda_2!\lambda_3!\ldots\lambda_r!)^M.$$

Given the mapping of the first M partitions, the mapping of the N elements of the $M+1$st partition is determined (without ordering). Therefore, the additional choice is at most $N!$. In summary, the total number of permutations can be at most

$$L^M\left(\prod_{i=2}^r\lambda_i!\right)^M N!.$$

Using this bound, we obtain

$$\Pr\left(\mathscr{E}_{L+1}^c \middle| \mathscr{E}_L\right) \le \frac{1}{n!}L^M\left(\prod_{i=2}^r\lambda_i!\right)^M N!. \tag{39}$$

Let

$$x_L \triangleq \frac{1}{n!}L^M\left(\prod_{i=2}^r\lambda_i!\right)^M N!.$$

Note that $\mathscr{E}_{k+1}\subset\mathscr{E}_k$ for $k\ge 1$. Therefore, it follows that

$$\begin{aligned}\Pr(\mathscr{E}_K) &= \Pr(\mathscr{E}_K\cap\mathscr{E}_{K-1})\\ &= \Pr(\mathscr{E}_K\,|\,\mathscr{E}_{K-1})\Pr(\mathscr{E}_{K-1}).\end{aligned} \tag{40}$$

Recursive application of argument behind (40) and fact that $\Pr(\mathscr{E}_1)=1$, we have

$$\begin{aligned}\Pr(\mathscr{E}_K) &= \Pr(\mathscr{E}_1)\prod_{L=1}^{K-1}\Pr(\mathscr{E}_{L+1}\,|\,\mathscr{E}_L)\\ &= \prod_{L=1}^{K-1}\left(1-\Pr\left(\mathscr{E}_{L+1}^c\middle|\,\mathscr{E}_L\right)\right)\\ &= \prod_{L=1}^{K-1}(1-x_L)\\ &\ge 1-\left(\sum_{L=1}^{K-1}x_L\right).\end{aligned} \tag{41}$$

Using (39), it follows that $x_{k+1}\ge x_k$ for $k\ge 1$. Therefore

$$\begin{aligned}\sum_{L=2}^K x_L &\le Kx_K\\ &\le \frac{1}{n!}K^{M+1}\left(\prod_{i=2}^r\lambda_i!\right)^M N!\\ &= \frac{1}{n!}K^{M+1}\left(\frac{n!}{\lambda_1!D_\lambda}\right)^M N!\\ &= \frac{K^{M+1}}{D_\lambda^M}\left(\frac{n!}{\lambda_1!}\right)^M\frac{N!}{n!}\\ &= \frac{K^{M+1}}{D_\lambda^M}\left(\frac{n!}{\lambda_1!(n-\lambda_1)!}\right)^M\frac{N!((n-\lambda_1)!)^M}{n!}.\end{aligned} \tag{42}$$

Since $n=N+M(n-\lambda_1)$, we have a binomial and a multinomial coefficient in RHS of (42). We simplify this expression by obtaining an approximation for a multinomial coefficient through Stirling's approximation. For that, first consider a general multinomial coefficient $m!/(k_1!k_2!\ldots k_l!)$ with $m=\sum_i k_i$. Then, using the Stirling's approximation $\log n!=n\log n-n+0.5\log n+O(1)$, for any n, we obtain

$$\begin{aligned}&\log\left(\frac{m!}{k_1!k_2!\ldots k_l!}\right)\\ =&m\log m-m+0.5\log m+O(1)-\\ &\sum_{i=1}^l(k_i\log k_i-k_i+0.5\log k_i+O(1))\\ =&m\sum_{i=1}^l\frac{k_i}{m}\log\frac{m}{k_i}+0.5\log\frac{m}{k_1k_2\ldots k_l}-O(l).\end{aligned}$$

Thus, we can write

$$\begin{aligned}&M\log\frac{n!}{\lambda_1!(n-\lambda_1)!}\\ =&Mn\alpha_1\log\frac{1}{\alpha_1}+Mn(1-\alpha_1)\log\frac{1}{1-\alpha_1}\\ &+0.5\log\frac{1}{n^M\alpha_1^M(1-\alpha_1)^M}-O(M)\end{aligned} \tag{43}$$

where $\alpha_1 = \lambda_1/n$. Similarly, we can write

$$\begin{aligned}&\log \frac{n!}{N!((n-\lambda_1)!)^M}\\ =&n\delta \log \frac{1}{\delta} + Mn(1-\alpha_1)\log\frac{1}{1-\alpha_1}\\ &+ 0.5\log\frac{1}{n^M\delta(1-\alpha_1)^M} - O(M)\end{aligned} \tag{44}$$

where $\delta = N/n$. It now follows from (43) and (44) that

$$\begin{aligned}&M\log\frac{n!}{\lambda_1!(n-\lambda_1)!} - \log\frac{n!}{N!((n-\lambda_1)!)^M}\\ =&-Mn\alpha_1\log\alpha_1 + \delta n\log\delta\\ &+0.5\log\frac{\delta}{\alpha_1^M} + O(M)\end{aligned} \tag{45}$$

Since $\delta < 1, \delta n\log\delta \le 0$ and $\log(\delta/\alpha_1^M) \le -M\log\alpha_1$. Thus, we can write

$$\begin{aligned}&M\log\frac{n!}{\lambda_1!(n-\lambda_1)!} - \log\frac{n!}{N!((n-\lambda_1)!)^M}\\ \le&Mn\alpha_1\log(1/\alpha_1) + O(M\log(1/\alpha_1))\\ =&O(Mn\alpha_1\log(1/\alpha_1)).\end{aligned} \tag{46}$$

It now follows from (42), (45), and (46) that

$$\begin{aligned}&\log\left(\sum_{L=2}^{K} x_L\right)\\ \le&(M+1)\log K - M\log D_\lambda + O(Mn\alpha_1\log(1/\alpha_1)).\end{aligned} \tag{47}$$

Therefore, for $\Pr(\mathscr{E}_K) = 1 - o(1)$, a sufficient condition is

$$\begin{aligned}&\log K + \frac{c\log n}{M+1}\\ \le&\frac{M}{M+1}\log D_\lambda - \frac{M}{M+1}O(n\alpha_1\log(1/\alpha_1))\end{aligned} \tag{48}$$

for some $c > 0$. We now claim that $\log n = O(Mn\alpha_1\log(1/\alpha_1))$. The claim is clearly true for $\alpha_1 \to \theta$ for some $0 < \theta < 1$. Now suppose $\alpha_1 \to 1$. Then, $M \ge 1/(1-\alpha_1) - 1 = \alpha_1/(1-\alpha_1) = x$, say. This implies that $M\alpha_1\log(1/\alpha_1) \ge \alpha_1 x\log(1+1/x) \to 1$ as $\alpha_1 \to 1$. Thus, $Mn\alpha_1\log(1/\alpha_1) = n(1+o(1))$ for $\alpha_1 \to 1$ as $n \to \infty$. Hence, the claim is true for $\alpha_1 \to 1$ as $n \to \infty$. Finally, consider $\alpha_1 \to 0$ as $n \to \infty$. Note that the function $h(x) = x\log(1/x)$ is increasing on $(0,\epsilon)$ for some $0 < \epsilon < 1$. Thus, for n large enough, $n\alpha_1\log(1/\alpha_1) \ge \log n$ since $\alpha_1 \ge 1/n$. Since $M \ge 1$, it now follows that $Mn\alpha_1\log(1/\alpha_1) \ge \log n$ for n large enough and $\alpha_1 \to 0$. This establishes the claim.

Since $\log n = O(Mn\alpha_1\log(1/\alpha_1))$, it now follows that (48) is implied by

$$\begin{aligned}\log K &\le \frac{M}{M+1}\log D_\lambda - \frac{M}{M+1}O(n\alpha_1\log(1/\alpha_1))\\ &= \frac{M}{M+1}\log D_\lambda\left[1 - \frac{O(n\alpha_1\log(1/\alpha_1))}{\log D_\lambda}\right].\end{aligned} \tag{49}$$

Now consider $D_\lambda = n!/(\lambda_1!\lambda_2!\ldots\lambda_r!)$. Then, we claim that for large n

$$\log D_\lambda \ge 0.5nH(\alpha). \tag{50}$$

In order to see why the claim is true, note that Stirling's approximation suggests,

$$\begin{aligned}\log n! &= n\log n - n + 0.5\log n + O(1)\\ \log\lambda_i! &= \lambda_i\log\lambda_i - \lambda_i + 0.5\log\lambda_i + O(1).\end{aligned}$$

Therefore

$$\log D_\lambda \ge nH(\alpha) + 0.5\log(n/\lambda_1) - \sum_{i=2}^{r} 0.5(O(1) + \log\lambda_i).$$

Now consider

$$\begin{aligned}&\lambda_i\log(n/\lambda_i) - \log\lambda_i - O(1)\\ =&\left(\lambda_i - \frac{\log\lambda_i}{\log(n/\lambda_i)}\right)\log(n/\lambda_i) - O(1)\end{aligned} \tag{51}$$

Since $\lambda_i \le n/2$ for $i \ge 2, \log(n/\lambda_i) \ge \log 2$. Thus, the first term in the RHS of (51) is non-negative for any $\lambda_i \ge 1$. In addition, for every λ_i, either $\lambda_i - \log\lambda_i \to \infty$ or $\log(n/\lambda_i) \to \infty$ as $n \to \infty$. Therefore, the term on the RHS of (51) is asymptotically non-negative. Hence

$$\log D_\lambda \ge 0.5nH(\alpha). \tag{52}$$

Thus, it now follows from (50) that (49) is implied by

$$\log K \le \frac{M}{M+1}\log D_\lambda\left[1 - \frac{O(\alpha_1\log(1/\alpha_1))}{H(\alpha)}\right].$$

That is, we have "unique witness" property satisfied as long as

$$K = O\left(D_\lambda^{\gamma(\alpha)}\right) \tag{53}$$

where

$$\gamma(\alpha) = \frac{M}{M+1}\left[1 - C'\frac{H(\alpha) - H'(\alpha)}{H(\alpha)}\right] \tag{54}$$

and C' is some constant. This completes the proof of Theorem III.6.

XI. Proof of Theorem III.7: Limitation on Recovery

In order to make a statement about the inability of *any* algorithm to recover f using $\hat{f}(\lambda)$, we rely on the formalism of classical information theory. In particular, we establish a bound on the sparsity of f beyond which recovery is not asymptotically reliable (precise definition of asymptotic reliability is provided later).

A. Information Theory Preliminaries

Here we recall some necessary Information Theory preliminaries. Further details can be found in the book by Cover and Thomas [26].

Consider a discrete random variable X that is uniformly distributed over a finite set $\mathscr{X}$. Let X be *transmitted* over a *noisy* channel to a receiver; suppose the receiver receives a random variable Y, which takes values in a finite set $\mathscr{Y}$. Essentially, such "transmission over noisy channel" setup describes any two random variables X, Y defined through a joint probability distribution over a common probability space.

Now let $\hat{X} = g(Y)$ be an estimation of the transmitted information that the receiver produces based on the observation Y using some function $g : \mathscr{Y} \to \mathscr{X}$. Define probability of error as $p_{\text{err}} = \Pr(X \neq \hat{X})$. Since X is uniformly distributed over $\mathscr{X}$, it follows that

$$p_{\text{err}} = \frac{1}{|\mathscr{X}|} \sum_{x \in \mathscr{X}} \Pr(g(Y) \neq x | x). \tag{55}$$

Recovery of X is called asymptotically reliable if $p_{\text{err}} \to 0$ as $|\mathscr{X}| \to \infty$. Therefore, in order to show that recovery is not asymptotically reliable, it is sufficient to prove that p_{err} is bounded away from 0 as $|\mathscr{X}| \to \infty$. In order to obtain a lower bound on p_{err}, we use Fano's inequality

$$H(X \,|\, \hat{X}) \leq 1 + p_{\text{err}} \log |\mathscr{X}|. \tag{56}$$

Using (56), we can write

$$\begin{aligned} H(X) &= I(X; \hat{X}) + H(X|\hat{X}) \\ &\leq I(X; \hat{X}) + p_{\text{err}} \log |\mathscr{X}| + 1 \\ &\overset{(a)}{\leq} I(X; Y) + p_{\text{err}} \log |\mathscr{X}| + 1 \\ &= H(Y) - H(Y \,|\, X) + p_{\text{err}} \log |\mathscr{X}| + 1 \\ &\leq H(Y) + p_{\text{err}} \log |\mathscr{X}| + 1 \end{aligned} \tag{57}$$

where we used $H(Y \,|\, X) \geq 0$ for a discrete[4] valued random variable. The inequality (a) follows from the data processing inequality: if we have Markov chain $X \to Y \to \hat{X}$, then $I(X; \hat{X}) \leq I(X; Y)$. Since $H(X) = \log |\mathscr{X}|$, from (57) we obtain

$$p_{\text{err}} \geq 1 - \frac{H(Y) + 1}{\log |\mathscr{X}|}. \tag{58}$$

Therefore, to establish that probability of error is bounded away from zero, it is sufficient to show that

$$\frac{H(Y) + 1}{\log |\mathscr{X}|} \leq 1 - \delta \tag{59}$$

for any fixed constant $\delta > 0$.

B. Proof of Theorem III.7

Our goal is to show that when K is large enough (in particular, as claimed in the statement of Theorem III.7), the probability of error of *any* recovery algorithm is uniformly bounded away from 0. For that, we first fix a recovery algorithm, and then utilize the above setup to show that recovery is not asymptotically reliable when K is large. Specifically, we use (59), for which we need to identify random variables X and Y.

To this end, for a given K and T, let f be generated as per the random model $R(K, T)$. Let random variable X represent the support of function f i.e., X takes values in $\mathscr{X} = S_n^K$. Given λ, let $\hat{f}(\lambda)$ be the partial information that the recovery algorithm uses to recover f. Let random variable Y represent $\hat{f}(\lambda)$, the $D_\lambda \times D_\lambda$ matrix. Let $h = h(Y)$ denote the estimate of f, and $g = g(Y) = \text{supp}h$ denote the estimate of the support of f produced by the given recovery algorithm. Then

$$\begin{aligned} \Pr(h \neq f) &\geq \Pr(\text{supp}(h) \neq \text{supp}(f)) \\ &= \Pr(g(Y) \neq X). \end{aligned} \tag{60}$$

Therefore, in order to uniformly lower bound the probability of error of the recovery algorithm, it is sufficient to lower bound its probability of making an error in recovering the support of f. Therefore, we focus on

$$p_{\text{err}} = \Pr(g(Y) \neq X).$$

It follows from the discussion in Section XI-A that in order to show that p_{err} is uniformly bounded away from 0, it is sufficient to show that for some constant $\delta > 0$

$$\frac{H(Y) + 1}{\log |\mathscr{X}|} \leq 1 - \delta. \tag{61}$$

Observe that $|\mathscr{X}| = (n!)^K$. Therefore, using Stirling's approximation, it follows that

$$\log |\mathscr{X}| = (1 + o(1)) K n \log n. \tag{62}$$

Now $Y = \hat{f}(\lambda)$ is a $D_\lambda \times D_\lambda$ matrix. Let $Y = [Y_{ij}]$ with $Y_{ij}, 1 \leq i, j \leq D_\lambda$, taking values in $\{1, \ldots, KT\}$; it is easy to see that $H(Y_{ij}) \leq \log KT$. Therefore, it follows that

$$\begin{aligned} H(Y) &\leq \sum_{i,j=1}^{D_\lambda} H(Y_{ij}) \\ &\leq D_\lambda^2 \log KT = D_\lambda^2 (\log K + \log T). \end{aligned} \tag{63}$$

For small enough constant $\delta > 0$, it is easy to see that the condition of (61) will follow if K satisfies the following two inequalities:

$$\frac{D_\lambda^2 \log K}{Kn \log n} \leq \frac{1}{3}(1 + \delta) \Leftarrow \frac{K}{\log K} \geq \frac{3(1 - \delta/2) D_\lambda^2}{n \log n} \tag{64}$$

$$\frac{D_\lambda^2 \log T}{Kn \log n} \leq \frac{1}{3}(1 + \delta) \Leftarrow K \geq \frac{3(1 - \delta/2) D_\lambda^2 \log T}{n \log n}. \tag{65}$$

In order to obtain a bound on K from (64), consider the following: for large numbers x, y, let $y = (c + \varepsilon) x \log x$, for some constants $c, \varepsilon > 0$. Then, $\log y = \log x + \log \log x + \log(c + \varepsilon)$ which is $(1 + o(1)) \log x$. Therefore

$$\frac{y}{\log y} = \frac{c + \varepsilon}{1 + o(1)} x \geq cx \tag{66}$$

for $x \to \infty$ and constants $c, \varepsilon > 0$. Also, observe that $y / \log y$ is a nondecreasing function; hence, it follows that for $y \geq (c +$

[4]The counterpart of this inequality for a continuous valued random variable is not true. This led us to study the limitation of recovery algorithm over model $R(K, T)$ rather than $R(K, \mathscr{C})$.

$\varepsilon)x\log x, y/\log y \geq cx$ for large x. Now take $x = \frac{D_\lambda^2}{n\log n}, c = 3, \varepsilon = 1$ and $y = K$. Note that $D_\lambda \geq n$ for all λ of interest; therefore, $x \to \infty$ as $n \to \infty$. Hence, (64) is satisfied for the choice of

$$K \geq \frac{4D_\lambda^2}{n\log n}\left(\log\frac{D_\lambda^2}{n\log n}\right). \tag{67}$$

From (61), (64), (65), and (67) it follows that the probability of error of any algorithm is at least $\delta > 0$ for n large enough and any λ if

$$K \geq \frac{4D_\lambda^2}{n\log n}\left[\log\left(\frac{D_\lambda^2}{n\log n} \vee T\right)\right]. \tag{68}$$

This completes the proof of Theorem III.7.

XII. Conclusion

In summary, we considered the problem of *exactly* recovering a non-negative function over the space of permutations from a given partial set of Fourier coefficients. This problem is motivated by the wide ranging applications it has across several disciplines. This problem has been widely studied in the context of discrete-time functions in the recently popular *compressive sensing* literature. However, unlike our setup, where we want to perform exact recovery from a *given set* of Fourier coefficients, the work in the existing literature pertains to the choice of a limited set of Fourier coefficients that can be used to perform exact recovery.

Inspired by the work of Donoho and Stark [1] in the context of discrete-time functions, we focused on the recovery of non-negative functions with a sparse support (support size $\ll$ domain size). Our recovery scheme consisted of finding the function with the sparsest support, consistent with the given information, through ℓ_0 optimization. As we showed through some counterexamples, this procedure, however, does not recover the exact solution in all the cases. Thus, we identified sufficient conditions under which a function can be recovered through ℓ_0 optimization. For each kind of partial information, we then quantified the sufficient conditions in terms of the "complexity" of the functions that can be recovered. Since the sparsity (support size) of a function is a natural measure of its complexity, we quantified the sufficient conditions in terms of the sparsity of the function. In particular, we proposed a natural random generative model for the functions of a given sparsity. Then, we derived bounds on sparsity for which a function generated according to the random model satisfies the sufficient conditions with a high probability as $n \to \infty$. Specifically, we showed that, for partial information corresponding to partition λ, the sparsity bound essentially scales as $D_\lambda^{M/(M+1)}$. For $\lambda_1/n \to 1$, this bound essentially becomes D_λ and for $\lambda_1/n \to 0$, the bound essentially becomes $D_\lambda^{1/2}$.

Even though we found sufficient conditions for the recoverability of functions by finding the sparsest solution, ℓ_0 optimization is in general computationally hard to carry out. This problem is typically overcome by considering its convex relaxation, the ℓ_1 optimization problem. However, we showed that ℓ_1 optimization fails to recover a function generated by the random model with a high probability. Thus, we proposed a novel iterative algorithm to perform ℓ_0 optimization for functions that satisfy the sufficient conditions, and extended it to the general case when the underlying distribution may not satisfy the sufficient conditions and the observations maybe noisy.

We studied the limitation of any recovery algorithm by means of information theoretic tools. While the bounds we obtained are useful in general, due to technical limitations, they do not apply to the random model we considered. Closing this gap and understanding recovery conditions in the presence of noise are natural next steps.

Appendix
Proof of Auxiliary Lemma

Here we present the proof of Lemma III.1. For this, first consider the limit $\alpha_1 \uparrow 1$. Specifically, let $\alpha_1 = 1 - \varepsilon$, for a very small positive ε. Then, $\sum_{i=2}^r \alpha_i = 1 - \alpha_1 = \varepsilon$. By definition, we have $H'(\alpha)/H(\alpha) \leq 1$; therefore, in order to prove that $H'(\alpha)/H(\alpha) \to 1$ as $\alpha_1 \uparrow 1$, it is sufficient to prove that $H'(\alpha)/H(\alpha) \geq 1 - o(1)$ as $\alpha_1 \uparrow 1$. For that, consider

$$\begin{aligned}\frac{H'(\alpha)}{H(\alpha)} &= \frac{H'(\alpha)}{\alpha_1\log(1/\alpha_1) + H'(\alpha)} \\ &= 1 - \frac{\alpha_1\log(1/\alpha_1)}{\alpha_1\log(1/\alpha_1) + H'(\alpha)}.\end{aligned} \tag{69}$$

In order to obtain a lower bound, we minimize $H'(\alpha)/H(\alpha)$ over $\alpha \geq 0$. It follows from (69) that, for a given $\alpha_1 = 1 - \varepsilon, H'(\alpha)/H(\alpha)$ is minimized for the choice of $\alpha_i, i \geq 2$ that minimizes $H'(\alpha)$. Thus, we maximize $\sum_{i=2}^r \alpha_i \log\alpha_i$ subject to $\alpha_i \geq 0$ and $\sum_{i=2}^r \alpha_i = 1 - \alpha_1 = \varepsilon$. Here we are maximizing a convex function over a convex set. Therefore, maximization is achieved on the boundary of the convex set. That is, the maximum is $\varepsilon\log\varepsilon$; consequently, the minimum value of $H'(\alpha) = \varepsilon\log(1/\varepsilon)$. Therefore, it follows that for $\alpha_1 = 1 - \varepsilon$

$$\begin{aligned}1 \geq \frac{H'(\alpha)}{H(\alpha)} &\geq 1 - \frac{-(1-\varepsilon)\log(1-\varepsilon)}{\varepsilon\log(1/\varepsilon) - (1-\varepsilon)\log(1-\varepsilon)} \\ &\approx 1 - \frac{\varepsilon}{\varepsilon\log(1/\varepsilon) + \varepsilon} \\ &\approx 1 - \frac{1}{1 + \log(1/\varepsilon)} \\ &\overset{\varepsilon\to 0}{\to} 1.\end{aligned} \tag{70}$$

To prove a similar claim for $\alpha_1 \downarrow 0$, let $\alpha_1 = \varepsilon$ for a small, positive ε. Then, it follows that $r = \Omega(1/\varepsilon)$ since $\sum_{i=1}^r \alpha_i = 1$ and $\alpha_1 \geq \alpha_i$ for all $i, 2 \leq i \leq r$. Using a convex maximization based argument similar to the one we used above, it can be checked that $H'(\alpha) = \Omega(\log(1/\varepsilon))$. Therefore, it follows that $\alpha_1\log(1/\alpha_1)/H'(\alpha) \to 0$ as $\alpha_1 \downarrow 0$. That is, $H'(\alpha)/H(\alpha) \to 1$ as $\alpha_1 \downarrow 0$. This completes the proof of Lemma III.1.

References

[1] D. Donoho and P. Stark, "Uncertainty principles and signal recovery," *SIAM J. Appl. Math.*, pp. 906–931, 1989.

[2] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar, "Rank aggregation methods for the web," *WWW*, pp. 613–622, 2001.

[3] P. Diaconis, "Group representations in probability and statistics," *IMS Lecture Notes-Monograph Ser.*, vol. 11, 1988.
[4] J. Huang, C. Guestrin, and L. Guibas, "Efficient inference for distributions on permutations," *Adv. Neur. Inf. Process. Syst.*, vol. 20, pp. 697–704, 2008.
[5] R. Kondor, A. Howard, and T. Jebara, "Multi-object tracking with representations of the symmetric group," presented at the 11th Int. Conf. Artificial Intelligence and Statistics, 2007.
[6] P. Dangauthier, R. Herbrich, T. Minka, and T. Graepel, "Trueskill through time: Revisiting the history of chess," *Adv. Neur. Inf. Process. Syst.*, vol. 20, 2007.
[7] K. Kueh, T. Olson, D. Rockmore, and K. Tan, "Nonlinear approximation theory on compact groups," *J. Fourier Anal. Appl.*, vol. 7, no. 3, pp. 257–281, 2001.
[8] C. Shannon, "Communication in the presence of noise," *Proc. IRE*, vol. 37, no. 1, pp. 10–21, 1949.
[9] H. Nyquist, "Certain topics in telegraph transmission theory," *Proc. IEEE*, vol. 90, no. 2, pp. 280–305, Feb. 2002.
[10] E. Candes and T. Tao, "Decoding by linear programming," *IEEE Trans. Inf. Theory*, vol. 51, no. 12, pp. 4203–4215, Dec. 2005.
[11] E. Candes, J. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Commun. Pure Appl. Math.*, vol. 59, no. 8, 2006.
[12] E. Candes and J. Romberg, "Quantitative robust uncertainty principles and optimally sparse decompositions," *Found. Comput. Math.*, vol. 6, no. 2, pp. 227–254, 2006.
[13] E. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Inf. Theory*, vol. 52, no. 2, pp. 489–509, Feb. 2006.
[14] D. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.
[15] R. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory*, vol. 8, no. 1, pp. 21–28, 1962.
[16] M. Sipser and D. A. Spielman, "Expander codes," *IEEE Trans. Inf. Theory*, vol. 42, pp. 1710–1722, 1996.
[17] M. Luby, M. Mitzenmacher, M. Shokrollahi, and D. Spielman, "Improved low-density parity-check codes using irregular graphs," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 585–598, Feb. 2001.
[18] I. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. SIAM*, pp. 300–304, 1960.
[19] J. Tropp, "Just relax: Convex programming methods for identifying sparse signals in noise," *IEEE Trans. Inf. Theory*, vol. 52, no. 3, pp. 1030–1051, Mar. 2006.
[20] J. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Trans. Inf. Theory*, vol. 50, no. 10, pp. 2231–2242, Oct. 2004.
[21] R. Berinde, A. Gilbert, P. Indyk, H. Karloff, and M. Strauss, "Combining geometry and combinatorics: A unified approach to sparse signal recovery," preprint, 2008.
[22] G. Cormode and S. Muthukrishnan, "Combinatorial algorithms for compressed sensing," *Lecture Notes Comput. Sci.*, vol. 4056, p. 280, 2006.
[23] A. C. Gilbert, M. J. Strauss, J. A. Tropp, and R. Vershynin, "One sketch for all: Fast algorithms for compressed sensing," in *Proc. 39th Annu. ACM Symp. Theory of Computing*, New York, 2007, pp. 237–246, ACM.
[24] S. Muthukrishnan, *Data Streams: Algorithms and Applications*, ser. Foundations and Trends in Theoretical Computer Science. Boston, MA: Now Publishers, 2005.
[25] S. Jagabathula and D. Shah, "Inferring rankings under constrained sensing," in *Proc. NIPS*, 2008, pp. 7–1.
[26] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, ser. Wiley Series in Telecommunications and Signal Processing, 2nd ed. Hoboken, NJ: Wiley, Jul. 2006 [Online]. Available: http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20&path=ASIN/0471241954

Srikanth Jagabathula is currently Assistant Professor with the Department of Information, Operations, and Management Sciences at the Leonard N. Stern School of Business, NYU. He is a member of the Operations Management (OM) group. His research focus is on understanding how to handle and extract useful insights from the large quantities of data being generated by businesses. He received the B.Tech degree in Electrical Engineering from IIT Bombay in 2006 with the honor of the President of India Gold Medal. He received the M.S. and Ph.D. degrees from the EECS department at MIT in 2008 and 2011, respectively. He was awarded the "Best Student Paper Award" at NIPS 2008, the Ernst Guillemin award for the best EE SM Thesis, and the first place in the MSOM student paper competition in 2010.

Devavrat Shah is currently a Jamieson career development associate professor with the department of electrical engineering and computer science, MIT. He is a member of the Laboratory of Information and Decision Systems (LIDS) and affiliated with the Operations Research Center (ORC). His research focus is on theory of large complex networks which includes network algorithms, stochastic networks, network information theory and large scale statistical inference. He received his B.Tech. degree in computer science and engg. from IIT-Bombay in 1999 with the honor of the President of India Gold Medal. He received his Ph.D. from the Computer Science department, Stanford University in October 2004. He was a postdoc in the Statistics department at Stanford in 2004–2005. He was co-awarded the best paper awards at the IEEE INFOCOM'04, ACM SIGMETRICS/Performance'06, and best student paper awards at Neural Information Processing Systems'08 and ACM SIGMETRICS/Performance'09. He received 2005 George B. Dantzig best dissertation award from the INFORMS. He received the first ACM SIGMETRICS Rising Star Award 2008 for his work on network scheduling algorithms.