On the Complexity of the Marginal Satisfiability Problem

Surya Bhupatiraju

November 18th, 2012

Abstract. The marginal satisfiability problem (MSP) asks: Given desired marginal distributions D_S for every subset S of c variable indices from $\{1, \ldots, n\}$, does there exist a distribution D over n-tuples of values in $\{1, \ldots, m\}$ with those S-marginals D_S? Previous authors have studied MSP in fixed dimensions, and have classified the complexity up to certain upper bounds. However, when using general dimensions, it is known that the size of distributions grows exponentially, making brute force algorithms impractical. This presents an incentive to study more general, tractable variants, which in turn may shed light on the original problem's structure.

Thus, our work seeks to explore MSP and its variants for arbitrary dimension, and pinpoint its complexity more precisely. We solve MSP for $n = 2$ and completely characterize the complexity of three closely related variants of MSP. In particular, we detail novel greedy and stochastic algorithms that handle exponentially-sized data structures in polynomial time, as well as generate accurate representative samples of these structures in polynomial time. These algorithms are also unique in that they represent possible protocols in data compression for communication purposes. Finally, we posit conjectures related to more generalized MSP variants, as well as the original MSP.

1 Introduction

The transportation problem (TP) in linear programming is a classic question of feasibility and optimization, and studying it has been the basis of several subfields of mathematics and optimization theory. In one formulation, it asks about the optimized cost of transporting goods from i factories to j depositories, where a cost function gives the cost of transporting from each factory to each depository. The total space of possibilities is a $i \times j$ matrix whose solution space can be represented as a type of polytope known as a transportation polytope. However, when the transport costs are all equal, every valid solution is equally good, and one can concentrate solely on feasibility. In this case, a particular transportation problem is feasible exactly if we can fill in an $n \times n$ grid with prescribed row and column sums. Using this correspondence, we can study this more general question of feasibility and equate TP to the following, similarly practical problem.

In statistics and data security, when entire distributions are either too large to convey or contain confidential information, abridged forms of data are often given to summarize sets of data and to protect sensitive data. One way to abridge data is to provide *marginal distributions*, or distributions over a subset of categories. A pertinent example is the Census records of the U.S. Census Bureau. These tables of data are unwieldy in size and carry private information about individuals, and the U.S. Census Bureau faces the task of conveying this information to citizens and government agencies without the data being overwhelming or compromising privacy. One way to compress the data is to take sums across different characteristsics, and reveal only these sums to clients. This way, individual information is impossible to ascertain, and the size of the data is significantly reduced, alleviating bandwidth issues.

However, in some cases, it may be desirable to construct the distribution itself given the marginal sums, while in other cases, it may suffice simply to check that such a distribution exists. For example, the client receiving the marginalized data may wish to check the veracity of the data and construct an example population in a timely manner. Given only these marginal sets of data, how hard is it to check that a distribution exists that satisfies each of the constraints, and how hard is it to calculate such a distribution? Marginal problems such as these arise in many natural, interesting questions and have foundational applications in statistical inquiry, data security, mathematical

programming and transportation theory.

Consider tableaus of fixed dimensions, where we are given constraints on the sum of subsets of the elements in the tableau. For example, these constraints might bound the sum of two-dimensional slices in a three-dimensional tableau. A collection of constraints constitutes a *constraint problem*. If there exists an assignment of values to each entry in the tableau such that all of the constraints are satisfied, then the constraint problem is said to be *satisfiable*. The guiding question is: When all or some of these constraints are provided to a client, how hard is it for the client to determine if the constraint problem is satisfiable?

We study this problem using computational complexity theory − a subfield of theoretical computer science and mathematics whose goal is twofold: to classify computational problems into different complexity classes based on their difficulty and to equate or compare these classes. *Complexity classes* are sets of problems bounded by some resource constraints, where the resource may be time or space. Understanding the complexity of this problem will not only help to determine the degree of tractability of the problem, but also reveal the practicality of solving other similar problems. The relevant classes under study in this paper are P, the class of problems that can be efficiently solved and NP, the class of problems whose solutions can be efficiently checked.

Loera et al. [3] studied this problem when restricted to three dimensions, and showed that when all of the column, row, and depth sums are provided, the problem is generally computationally infeasible, or NP-complete, even when the length of the tableau is held small. In addition, Loera et al. found that counting the number of valid tables and finding the existence of unique tables are both difficult as well. Liu [5] showed that the generalized problem is in the computational class NP-hard, and that a quantum variant of the problem is QMA-complete, the quantum version of NP. Gusfield [1] studied two-dimensional tableaus when only some of the entries are provided, and studied these tableaus under the theme of statistical data security. He found that data protection in these tableaus is generally in NP.

The **marginal satisfiability problem** (MSP) asks whether a polynomial-size collection of constraints given as input, each over at most a constant number c of indices, is satisfiable. We ask whether the local constraints, in the sense of each being checkable by looking through at most

c indices at a time, admit a global distribution. Formally, MSP asks: Given desired marginal distributions D_S for every subset S of at most c variable indices from $\{1, \ldots, n\}$, does there exist a nonnegative distribution, or an assignment of values D to n-tuples of values in $\{1, \ldots, m\}$ with those S-marginals D_S?

It is known that at three dimensions MSP is already NP-complete, due to the practicality of this problem, it is still useful to find practical and related problems that can be solved; this is the main motivation for studying variants, which in turn may shed light on the structure of the original problem. We mainly ask about the computational complexity of these problems, which can be either bounded above by an algorithm solving the problem using certain resources, or below via a reduction or proof of completeness. We also ask about efficient algorithms for giving an explicit satisfying distribution when n is small. Note that for any fixed c, such a collection of constraints requires specifying a number of values that is polynomial in m and n, and thus only takes polynomial space to specify to fixed precision. On the other hand, a candidate satisfying the distribution requires specifying m^n values, making brute force impractical.

Therefore, MSP is not immediately known to be in NP for general dimensions. Past work has identified complexity results for fixed dimensions, yet our work is unique in that it focuses on finding complexity results and *algorithms* for solving MSP instances of arbitrary dimensions. In general, satisfiability problems are usually in NP, yet, in this case, MSP does not seem to lie in NP because of its exponential nature and the inability of clients to even look at all of the data due to pure bandwidth issues. In light of this, we have devised novel algorithms that are able to handle exponentially sized data structures in polynomial time.

We study three main variants of MSP, characterize their solutions with algorithms, and pinpoint their complexity. In Section 5, we first study a variant of MSP where negative values are allowed, as might appear when values represent economic losses and profits, and detail an efficient greedy algorithm that can not only determine if an n-dimensional instance is satisfiable, but if so, can also provide an explicit, sparse satisfying assignment in polynomial time. In addition, even if some entries are filled in arbitrarily, the algorithm is flexible enough to accomodate for these. The same algorithm extends to the modulo-n case of MSP when all the values are integral and marginals are

evaluated modulo n. This demonstrates that both of the variants above are in P.

Lastly in Section 6, we study an approximate version of MSP, where the client is only interested in seeing if the constraint problem can be satisfied up to an error ϵ, which is a realistic assumption since often constraints are given to certain tolerances. We develop a randomized algorithm to take specific samples of an exactly satisfying assignment, and prove that we can reduce the structure to a polynomially-sized output that can be used to show approximate satisfiability. Thus, we prove that the approximate variant of MSP is in promise-NP. This randomized algorithm is particularly useful, and reflects a protocol that may be used in data compression for communcation. For example, the Census can create a representative subpopulation by taking a sample with this algorithm, and constructively prove that the marginals it is providing are reasonable. Finally in Section 7, we posit conjectures regarding the complexity of the generalized MSP and other, more general variants.

2 Preliminaries

To specify the distributions that constitute marginal distributions, we represent each distribution in a tableau. This way, understanding and visualizing marginal sums becomes more intuitive.

Definition 2.1. A set of values $\{1, \ldots, m\}$ is defined as the **value set** $[m]$.

Definition 2.2. For an n-dimensional tableau, there are n **variables**, denoted by $X_1, X_2, \ldots X_n$, each of which take a **value** in the value set $[m]$, where m is the size of each dimension. For each X_i, the ith **index** takes values from the value set $[n]$. The set $U = [n]$ is the complete set of indices.

Definition 2.3. A **tableau** T is the set of all locations, or $[m]^U = \underbrace{\{1, \ldots, m\} \times \ldots \times \{1, \ldots, m\}}_{n \text{ times}}$, where a **location** is an individual cell in a tableau, or an element from $[m]^U$.

Example 2.4. A $5 \times 5 \times 5$ tableau has $n = 3$ variables, $m = 5$ values, $m^n = 125$ entries. One particular entry may have coordinates (1, 4, 3), where the second index is 4.

Definition 2.5. An **assignment** is a choice of real values for every location. In other words, it is a map $f : [m]^U \to \mathbb{R}$. An **entry** is a location with an assigned value.

Although with the original MSP we require a nonnegative assignment, we give a definition without nonnegativity so we can readily discuss variants. We are primarily interested in subsets of indices where some variables are fixed and some are unspecified, leading to the following definitions.

Definition 2.6. A **partially defined location** (PDL) is a specification of a value for each index in a subset $S \subseteq U$.

We can write a PDL as a tuple (S, U, p) where p is a function that maps indices to values and S is a subset of indices. We can also put the specified values in the specified locations and asterisks in the unspecified locations.

Definition 2.7. The **slice** corresponding to a PDL is the subset of locations in the tableau whose specified coordinates match. Formally, we define a slice as

$$\text{slice}(S, U, p) = \{\vec{X} \in [m]^U |\ X_i = p(i) \text{ for all } i \in S\},$$

where p is a mapping from indices to values: $p : S \to [m]$.

Example 2.8. The locations in the slice$(\{1, 2, 3\}, [6], (X_1 = 1, X_2 = 5, X_3 = 3))$ are those that match the PDL: $(1, 5, 3, *, *, *)$.

For conciseness, when U is clear, a slice(S, U, p) can be categorized as a disjoint S-slice, and further categorized as an $|S|$-slice when we are only interested in the number of variables being held constant. It is important to note that with each abbreviation, information is lost.

We look at a specific subset of indices and ignore the rest to compute marginal sums. Therefore, we define the following marginal map:

Definition 2.9. Let the S-marginal $W = S\text{-}\mathbf{marg}(V)$ of an assignment $V\colon\ [m]^U \to \mathbb{R}$ be the assignment $W : [m]^S \to \mathbb{R}$, given by

$$W(p) = \sum_{a \in \text{slice}(S,U,p)} V(a).$$

We can abbreviate this marginal map as Π_{US} because we produce a S-slice from a U-slice. $\Pi_{XY} \circ \Pi_{YZ}$ denotes two consecutive applications of the marg function.

Example 2.10. For $U = [5]$ and $S = \{2, 5\}$, we compute the S-marginal $W(*, k_2, *, *, k_5)$ of an assignment V as

$$\sum V(*, k_2, *, *, k_5) = \sum_{z \in (\text{slice}(\{2,5\},[5],(X_2=k_2, X_5=k_5))} V(z).$$

We show that this definition is equivalent to the standard definition of marginalizing in distributions in Appendix A.

3 Two-Dimensional MSP

The smallest case to consider is the two-dimensional case.

Theorem 3.1. *A two-dimensional instance of MSP (two variables) is satisfiable if and only if the row and column constraints sum to the same value.*

Proof. Adding the entries in either order (row first, then column, or column first, then row) produces the same sum. To show the only-if-direction, we provide two constructions to give a satisfying assignment for any two-dimensional MSP instance with equal row and column sums. We first define notation used in the following constructions. Let entries in the tableau be denoted as $V_{i,j}$, where i is the column number and j the row number. Similarly, let S_i^{row} and S_j^{col} represent the ith and jth column and row sums as given by constraints respectively. Let $S_{\text{total}} = \sum_i S_i^{\text{row}} = \sum_j S_j^{\text{col}}$ be the sum of all the entries.

Construction 3.2. (Fractional Construction) Given all of the 1-marginals, each entry is assigned the value $V_{i,j} = \frac{S_i^{\text{row}} \cdot S_j^{\text{col}}}{S_{\text{total}}}$.

It is clear from this construction that the constraints are satisfied:

$$\sum_i V_{i,j} = \sum_i \frac{S_i^{\text{row}} \cdot S_j^{\text{col}}}{S_{\text{total}}} = \frac{S_{\text{total}} \cdot S_j^{\text{col}}}{S_{\text{total}}} = S_j^{\text{col}} \text{ and } \sum_j V_{i,j} = \sum_j \frac{S_i^{\text{row}} \cdot S_j^{\text{col}}}{S_{\text{total}}} = \frac{S_i^{\text{row}} \cdot S_{\text{total}}}{S_{\text{total}}} = S_j^{\text{row}},$$

Construction 3.3. (Sorted Construction) This construction relies on the fact that the constraints are ordered, and that there exists a minimal constraint. Given constraints S_i^{row} and S_j^{col}, we first sort the constraints in ascending order, and we assign the first entry $V_{1,1} = \min\{S_1^{\text{row}}, S_1^{\text{col}}\}$, and

subtract this value from each of the constraints. One of the constraints will therefore be reduced to 0, and in general, when we reach the $V_{i,j}$ entry, we assign it $V_{i,j} = \min\{S_i^{\text{row}}, S_j^{\text{row}}\}$. From here there are two possibilities: if $S_i^{\text{row}} = 0$, then we assign $V_{i,j+1}$. Otherwise, we assign $V_{i+1,j}$, and we continue until we reach $V_{n,n}$.

This construction not only admits a sparse solution where $n^2 - 2n - 1$ of the entries are 0, but this also guarantees anatural number solution when the marginal values are natural numbers.

□

Construction 3.3 implies Corollary 3.4:

Corollary 3.4. *For any two-dimensional MSP instance with natural number constraints, if there exists a satisfying assignment, then there also exists a natural number solution.*

The more general statement of this corollary, whether or not given any n-dimensional satisfiable tableau with natural number constraints has an natural number solution, is still open. It may be the case that when only natural number constraints are specified, the constraints can only be satisfied by a rational solution.

4 Local Consistency

For small dimensions, proving that a constraint problem is satisfiable is easy − check a satisfying assignment. Thus we can conclude that MSP is in NP for small dimensions. However, verifying that a constraint problem is not satisfiable is harder − one cannot show with a single contradicting assignment that the tableau is not satisfiable. Some constraints are so directly contradictory that it is easy to prove that they do not allow a satisfying assignment. Thus, we introduce the notion of *local consistency*. In particular, for a constraint problem to be satisfiable, it must at least be satisfiable at local levels.

Definition 4.1. Consider the marginal constraints on two slices, an S-slice and a T-slice, where S and T are sets of indices. These two constraints are **locally consistent** if $(S \cap T)$-marg$(D_S) = (S \cap T)$-marg(D_T).

No matter by what sequence we take marginals, we get the same result if we end with the same set of variables, resulting in Theorem 4.2:

Theorem 4.2. *Let S_0 and S_1 be sets of indices, and let $S_2 = S_0 \cap S_1$. The composition of two marginal maps, or two applications of the marg function, $U_{ST} \circ U_{US}$, is well defined.*

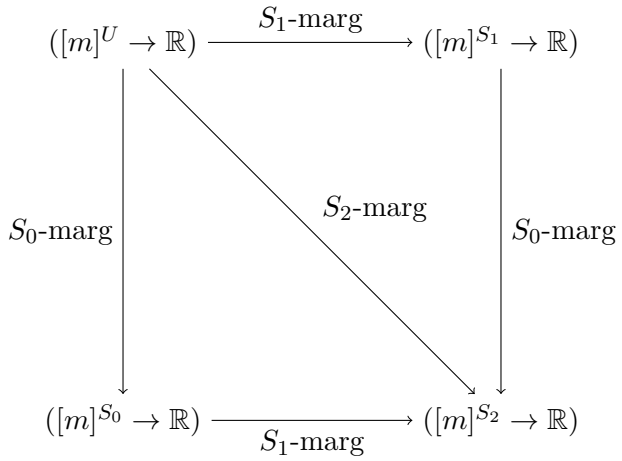

Figure 1: Commutativity of Marginal Maps

We illustrate Theorem 4.2 with the commutative diagram in Figure 1.

Proof. We write out each individual projection, and consider their composition. Straightforward evaluation of the marg functions shows that the composition evaluates to the correct expression. □

Theorem 4.3. *If a constraint problem is satisfiable, then each pair of marginal constraints is locally consistent.*

Proof. Assume that two constraints, S_1 and S_2, are not locally consistent. This implies that each of them induces different marginal constraints on their intersection $S_1 \cap S_2$. However, any assignment has a single $(S_1 \cap S_2)$-marg(D), contradicting the original assumption. □

We can use this terminology to restate the two-dimensional result of MSP:

Theorem 4.4. *For two dimensions, local consistency implies global satisfiability.*

The proof is the same as the proof of Theorem 3.1. However, we find that local consistency is no longer a sufficient condition to guarantee a satisfying assignment to any constraint problem beyond two dimensions.

Theorem 4.5. *For dimension $n > 2$, local consistency does not imply global satisfiability.*

Proof. To disprove the theorem, we construct the following counterexample. Suppose we have these marginal constraints on a $2 \times 2 \times 2$ tableau, where $V_{X,Y,Z}$ represents coordinates in the tableau:

Marginal Constraints

$$V_{0,0,0} + V_{0,0,1} = 0 \quad V_{0,0,0} + V_{1,0,0} = 0.5 \quad V_{1,1,0} + V_{1,1,1} = 0 \quad V_{0,1,1} + V_{1,1,1} = 0.5$$
$$V_{0,1,0} + V_{0,1,1} = 0.5 \quad V_{0,1,0} + V_{1,1,0} = 0 \quad V_{0,0,0} + V_{0,1,0} = 0.5 \quad V_{1,0,0} + V_{1,1,0} = 0$$
$$V_{1,0,0} + V_{1,0,1} = 0.5 \quad V_{0,0,1} + V_{1,0,1} = 0 \quad V_{0,0,1} + V_{0,1,1} = 0 \quad V_{1,0,1} + V_{1,1,1} = 0.5$$

Straightforward computation shows that each pair of constraints is locally consistent, and it is easy to see that the 0-constraint equations force every value in the table to be 0, resulting in a contradiction. □

5 Weak Satisfiability is in P

For general dimensions, we first consider a weaker variant of MSP where the nonnegativity constraint is removed.

Definition 5.1. A constraint problem is **weakly satisfiable** if there exists an assignment, not necessarily nonnegative, with the desired marginals. The *weak marginal satisfiability problem* (WMSP) asks whether a marginal constraints problem is weakly satisfiable.

Definition 5.2. A constraint problem is **weakly sparsely satisfiable** if there exists an assignment, not necessarily nonnegative, with the desired marginals and polynomially many nonzero entries.

The relaxation of nonnegativity changes the problem to a more linear algebra-esque problem, in the sense that the problem is reduced to solving large systems of equations. In general, because there are many more variables than equations at higher dimensions, the problem is determinable.

At the same time, solving exponentially many constraint equations is still outside computational range, and we want an algorithm for a polynomial-time constructible solution. To show that such a construction is computationally feasible, we require two things: that the construction satisfies sparsity, and for the prover to only verify polynomially many constraints, in which case we can solely utilize local consistency.

Note that for c-marginals, there are a total of $\binom{n}{c}m^c$ constraints. Thus, when c is allowed to vary, the number of constraints grows exponentially in m and n. However, when the type of marginals is fixed, or when c is held constant, the number of constraints is polynomial in m and n. We describe the following polynomial-time constructible algorithm to choose a full assignment for any n-dimensional tableau in Theorem 5.3. One important aspect of the algorithm is that even when initial values are filled in arbitrarily, it is still possible to complete the assignment and choose a weak satisfying assignment.

Theorem 5.3. *For a subset P of the tableau T defined as*

$$P = \{(x_1, \ldots, x_n) \mid \textit{fewer than } n - c \textit{ of } x_i \textit{ are } 1\},$$

for every partial assignment $f : P \to \mathbb{R}$, and for a choice of all the c-marginal constraints that are locally consistent, there exists a unique, full assignment $g : T \to \mathbb{R}$ that satisfies these constraints and is consistent with the partial assignment.

Proof. We complete the partial assignment recursively. At the kth step of the recursion, where k ranges from 0 to c, we define the partial assignment $g_k : E_k \to \mathbb{R}$ whose domain is the subset

$$E_k = \{(x_1, \ldots, x_n) \mid \text{exactly } n - c + k \text{ of } x_i \text{ are } 1\}.$$

Our proof strategy is to show that once we have chosen $g_0, \ldots, g_{k-1}$, we can uniquely choose g_k. The partial assignments $f, g_0, g_1, \ldots, g_c$ have disjoint respective domains, $P, E_0, E_1, \ldots, E_c$, whose union is the entire tableau T. By assigning each individual domain, we will construct a full assignment $g = f \cup g_0 \cup g_1 \cup \cdots \cup g_c$ that satisfies all of the constraints. First, for the purpose of the proof, we define a *forcing slice*: a slice S is *forcing* for a location e if e is the last unassigned location in the

slice. The proofs for the following lemmata are in Appendix B.

Lemma 5.4. *A slice S is forcing for a location e in S if and only if the PDL for S only leaves the 1-valued indices of e unspecified.*

By Lemma 5.4, we show that every location has a slice that forces it:

Lemma 5.5. *For each k, and each location e in E_k, there exists a forcing c-slice S which contains e and no other location from $E_k \cup \cdots \cup E_c$.*

Suppose we have already assigned $g_0, \ldots, g_{k-1}$. For each location e in E_k, choose S according to Lemma 5.5, so that e is the only unassigned location in S. To satisfy the marginal constraint on the slice S, the sum of the value assigned by g_k to e and the remaining entries in S which are assigned via $g_0, \ldots, g_{k-1}$ must equal the constraint value. Thus, e is uniquely chosen.

Using Lemma 5.5, the algorithm for assigning each entry is as follows. Of the $n - c + k$ 1's for an entry e in E_k, we pick $n - c$ of them to be unspecified. Assign the entry e the value forced by its forcing slice and continue for all the entries in E_k. We assign recursively up to E_c from the partial assignment until the entire tableau has a full assignment, in which case the assignment is complete and unique. Every constraint is satisfied because by Lemma 5.4, it has a location it forces that is not in P. Thus, when that location is assigned, that constraint is satisfied.

We now show that for each element e in E_k, the assignment of e to a particular value does not contradict any marginal sums on slices that contain e. This shows that the final assignment given by g satisfies all the constraints.

Lemma 5.6. *Any two forcing slices that force a location e are contained in a slice that forces e.*

We now use Lemma 5.6 to show that local consistency forces each pair of constraints to induce the same constraint on their variable intersection. One example of a slice that contains two slices, a S-slice and a T-slice, is the $(S \cap T)$-slice produced by the variable intersection of S and T. Local consistency and Lemma 5.6 now dictate that the $(S \cap T)$-slice must force the same value as the individual S-slice and T-slice. Therefore, we can be sure that no marginal sum is contradicted by the assignment of e. This completes the proof of Theorem 5.3.

□

Indeed, for fixed c, there are only polynomially many entries to fill in after the initial partial assignment:

$$\begin{aligned}|T-p| &= \binom{n}{0}m^0 + \binom{n}{1}m^1 + \ldots + \binom{n}{c}m^c \\ &\leq c\binom{n}{c}m^c \\ &= \mathcal{O}\left(c\binom{n}{c}m^c\right).\end{aligned}$$

Furthermore, Theorem 5.3 not only provides an explicit solution, but also proves that local consistency is a sufficient condition for weak sparse satisfiability, leading to Corollary 5.7:

Corollary 5.7. *Any WMSP instance that is locally consistent is weakly sparsely satisfiable.*

Therefore, for each WMSP instance, the verifier only has to check that polynomially-many constraints are locally consistent. Moreover, instead of assigning arbitrary values to the initial assignment P in Theorem 5.3, we can assign every location in the initial partial assignment a 0, leaving only polynomially many nonzero entries and guaranteeing a sparse solution. This way, any prover only has to convey polynomially many bits to the verifier. Corollary 5.7 directly implies Corollary 5.8 in that any weak, sparse solution is constructible from a weak solution.

Corollary 5.8. *Any marginal constraints problem that is weakly satisfiable is sparsely weakly satisfiable.*

Additionally, because the only arithmetic operation we use is addition and subtraction, the closure property implies Corollary 5.9 as well.

Corollary 5.9. *Any natural number-valued constraint problem that is weakly satisfiable is also weakly satisfiable by a natural number-valued assignment.*

From Corollary 5.7, we have the following major result of the complexity of WMSP.

Theorem 5.10. *WMSP is in P.*

The proof is in Appendix C.

5.1 MSP modulo n is in P

We consider another variant of MSP, but one also closely related to WMSP.

Definition 5.11. The *integral marginal satisfiability problem modulo n* (nMSP) asks whether a constraint problem is satisfiable modulo n.

Borrowing the same algorithm from Theorem 5.3, we can solve nMSP similarly, leading to the following major result of nMSP's complexity:

Theorem 5.12. *nMSP is in P.*

Proof. Consider a constraint problem in nMSP. We apply the same construction that we used for WMSP in order to solve this instance. For each assignment of a value in the tableau, we take each value modulo n. Since modular arithmetic respects addition and subtraction, the marginal constraints are still satisfied. Because the original construction takes polynomial time, any nMSP can be solved in polynomial time as well. □

6 Approximate Satisfiability is in promise-NP

We consider another variant of MSP, where instead of relaxing the nonnegativity constraint, we relax the accuracy of the constraints. For each constraint value k, we allow a margin of error, ϵ, and we consider each constraint approximately satisfied if the corresponding sum lies between $k-\epsilon$ and $k+\epsilon$.

Definition 6.1. A constraint problem is **approximately satisfiable** within ϵ if there exists an assignment whose marginal values are all within ϵ of the desired ones. The **approximate satisfiability problem** (AMSP) asks whether a marginal constraints problem is approximately satisfiable for a given ϵ, or ϵ-satisfiable.

Because of the probabilistic proof strategy used to solve AMSP, we introduce the notion of stochastic tableaus.

Definition 6.2. A tableau T is called *stochastic* if all the assigned entries in T are nonnegative and sum to 1. Similarly, a constraint problem is *stochastic* if all of the c-marginal constraint values sum to 1.

If a given assignment is not stochastic, we can scale each entry in the tableau to sum to 1, apply the algorithm, and scale back. Therefore without loss of generality, we assume stochasticity.

Imagine that a prover has a satisfying assignment and wants to tell the verifier that this assignment is satisfiable with error at most ϵ with probability of success $1 - \delta$. However, the prover can only send polynomially many bits N, with respect to ϵ and δ, to show that the problem is approximately satisfiable. We detail Algorithm 6.1 for producing this sparse, approximating assignment and claim that it is sufficient.

Algorithm 6.1. *We are given a free parameter N and a nonnegative satisfying assignment to a constraint problem. We output a stochastic nonnegative sparse assignment with at most N nonzero entries.*

Given: var N, tableau T_{start}

1. Compute sum of all entries in T_{start}. Divide each entry in T_{start} by that sum, establishing stochasticity.
2. Initialize T_{temp} with all entries equal to 0.
3. Prover assigns a weight k to each entry in T_{temp} proportional to its value in the satisfying assignment in T_{start}.
4. For $i := 1$ to N:
   Prover randomly selects an entry with probability according to the weights, and increments the current value of the entry in T_{temp} by one.
5. After the N entries are assigned, divide each entry in T_{temp} by N. T_{temp} is stochastic.

Algorithm 6.1 gives an approximate satisfying tableau with parameters ϵ and δ. With this algorithm, we have the following claim about the size of N needed for the protocol:

Claim 6.3. *There exists a polynomially bounded value $N = p(\delta, \epsilon)$ such that for each AMSP instance, Algorithm 6.1 produces an assignment that satisfies the instance with error at most ϵ with success probability at least $1 - \delta$.*

See Appendix D for the proof sketch. Claim 6.3 therefore implies Corollary 6.4:

Corollary 6.4. *If a constraint problem is satisfiable, then there exists a short proof that it is ϵ-satisfiable.*

By Algorithm 6.1 and Corollary 6.4, we have the following main result of the complexity of AMSP:

Theorem 6.5. *If a constraint problem is satisfiable, then the prover will always be able to find a ϵ-satisfiable assignment. If a constraint problem is not ϵ-satisfiable, then the prover will never find a ϵ-satisfiable assignment. In other words, AMSP is in* promise-NP.

Remark 6.6. Algorithm 6.1 fails to show that a purported ϵ-satisfiable solution is in fact ϵ-satisfiable, because the algorithm relies on the fact that the tableau can be scaled to be stochastic, and rescaled back. It is therefore unclear as to whether or not ϵ-satisfiable constraint problems can be verified without accumulating additional error. To prove this fact and to fill in this gap in the possible inputs, would prove that AMSP is in NP.

7 Future Work

With the complexity of MSP-like problems pinpointed, the heuristics of MSP are much better understood. We therefore conjecture the following result about MSP:

Conjecture 7.1. *MSP is in MA, the probabilistic version of NP.*

We propose the following questions for further inquiry on MSP.

Question 7.2. The **general constraints problem** asks: Instead of $\binom{n}{c}m^c$ c-marginals being specified, some of the constraints may be omitted, and different k-marginals may be specified. Is this general constraints problem harder than MSP?

Question 7.3. The **integral conjecture** asks: Does a satisfiable constraint problem with integer constraints always have an integral solution? We showed that this is true in two dimensions and for WMSP.

Understanding the difficulty of these problems represents an advance in a fundamental question in statistical studies and data security; applying the results from this paper to future studies will prove fruitful, and using the structure revealed by the variants will be pivotal in solving the general problem. The marginal satisfiability problem is an essential problem in data security, and the results addressed in this paper reflect protocols that are able to handle exponentially-sized data efficiently and generate representative, sparse samplings of large sets of data; these kinds of algorithms show special promise, particularly in the context of data compression.

Additionally, randomized and probabilistic algorithms have become an increasingly attractive technique to solve difficult problems. Several essential NP-complete problems that were deemed computationally infeasible now have clever randomized algorithms that rely on probabilistic protocols to achieve a high level of optimization. This research plays on the strengths of randomized algorithms, again in the context of exponentially-sized data strucures, and presents algorithms can take representative samples of these large data sets to find approximating solutions with polynomially many samples.

8 Acknowledgements

I would like to thank Alex Arkhipov for his guidance and assistance as my mentor, Dr. John Rickert, Sitan Chen, Matthew Babbitt, Kevin Garbe, for their editorial assistance on this paper, and Tanya Khovanova and Pavel Etingof for their supervision and support. I would also like to thank the Massachusetts Institute of Technology, the Research Science Institute, and the Center for Excellence in Education for helping and sponsoring this project.

A Consistency of S-marg with Marginalization in Distributions

A marginal distribution of a subset of a collection of random variables is the probability distribution of the variables contained in the subset. In our definition of marginal, it too is similarly a distribution across subsets of indices.

Given two random variables X and Y whose distribution is known, the marginal distribution of X is the probability distribution of X over information about Y. For random variables, the marginal probability mass function can be written as the following:

$$\Pr(X = x) = \sum_y \Pr(X = x, Y = y) = \sum_y \Pr(X = x | Y = y) \Pr(Y = y).$$

Our definition of marginal is similarly stated:

Definition A.1. The **slice** corresponding to a PDL is the subset of locations in the tableau whose specified coordinates match. Formally, we define a slice as

$$\text{slice}(S, U, p) = \{\vec{X} \in [m]^U |\ X_i = p(i) \text{ for all } i \in S\},$$

where p is a mapping from indices to values: $p : S \to [m]$.

Thus, we rewrite

$$\Pr(X_1 = x) = \sum_y \Pr(X_1 = x, X_2 = y) = \sum_{z \in \text{slice}(\{1\},\{1,2\},(X_1 = x))} = \sum V(x, \cdot).$$

This clearly generalizes to any number of random variables.

To show that the two definitions are equivalent, we show the reverse direction as well. Namely, for any marginal instance, we convert it to an instance of a marginal constraint. Consider the general marginal form, $\sum_{z \in \text{slice}(S,U,p)} V(z)$ where we set $S = \{X_{x_1}, X_{x_2}, \ldots, X_{x_m}\}$, $U = \{X_{x_1}, \ldots, X_{x_n}\}$, so

that $S \subseteq U$. We also have $p = \{X_{x_1} = y_1, \ldots, X_{x_m} = y_m\}$, where $y_i \in [m]$. We can rewrite this as

$$\begin{aligned}\sum_{z \in \text{slice}(S,U,p)} V(z) &= \sum V(X_{x_1}, \ldots, X_{x_m}, \underbrace{\cdot, \ldots, \cdot}_{n-m \ x_i's}) \\ &= \Pr(X_{x_1} = y_1, \ldots, X_{x_m} = y_m) \\ &= \sum_{y_{m+1}, \ldots, y_n} \Pr(X_{x_1} = y_1, \ldots, X_{x_m} = y_m, X_{x_{m+1}} = y_{m+1}, \ldots X_{x_n} = y_n).\end{aligned}$$

Thus, the two definitions are equivalent.

B Proofs of WMSP Lemmata

We restate the definition of a forcing slice for convenience:

Definition B.1. A slice S is **forcing** for a location e in E_k if S contains no other element from $E_k \cup \cdots \cup E_c$.

Lemma B.2 (Lemma 4.4)**.** *A slice S is forcing for a location e it contains if and only if the PDL for S leaves unspecified only indices of e that are 1.*

Proof. Because e lies in E_k, $n - c + k$ of its coordinates are 1. Since each location in the slice S is specified by a choice of values for the unspecified coordinates, the only way to return to having at least $n - c$ 1's is to switch all the $n - c$ indices back to 1's, which gives e back. Thus, e is the only location from $E_k \cup \cdots \cup E_c$ in this slice S, and therefore S is forcing. □

Lemma B.3 (Lemma 4.5)**.** *For each k, and each location e in E_k, there exists a forcing c-slice S which contains e and no other location from $E_k \cup \cdots \cup E_c$.*

Proof. Because e lies in E_k, $n - c + k$ of its coordinates are 1's. Let S be a c-slice defined by the partial assignment where we leave $n - c$ of the $n - c + k$ indices that are 1's unspecified. By Lemma 4.4, however, e is the only location from $E_k \cup \cdots \cup E_c$ in this slice S. □

Lemma B.4 (Lemma 4.6)**.** *Any two forcing slices, slice(S, U, p) and slice(T, U, q), that force a location e are contained in a slice that also forces e.*

Proof. Given two slices that share a location, one slice that contains both is their bounding slice: $\text{union}(\text{slice}(S, U, p), \text{slice}(T, U, q)) = \text{slice}(S \cap T, U, p \cap q)$. The notation $S \cap T$ denotes the intersection of sets of indices, and the domain of $p \cap q$ is the restriction of p and q to the intersection of their domains, on which they have to be equal because they both share a location. Both $\text{slice}(S, U, p)$ and $\text{slice}(T, U, q)$ are contained in $\text{slice}(S \cap T, U, p \cap q)$.

Note that the unspecified indices of $\text{slice}(S \cap T, U, p \cap q)$ are the result of the unioning over the unspecified indices of two component slices. If both $\text{slice}(S, U, p)$ and $\text{slice}(T, U, q)$ are forcing for e, then by Lemma 5.4, they only leave the 1-valued coordinates of e unspecified. However, the bounding slice also leaves these coordinates unspecified, and it is thus also forcing for e.

Therefore, by Lemma 5.4, because every other element is from $P \cup E_0 \cup \cdots \cup E_{k-1}$, and because more of the 1's are left unspecified, $\text{slice}(S \cap T, U, p \cap q)$ is also forcing.

□

C Proof of WMSP Complexity

Theorem C.1. *WMSP is in P.*

Proof. For fixed c, Corollary 5.7 states that local consistency is a sufficient condition for weak satisfiability. In this case, the verifier has to check every pair of constraints to prove global consistency. There are

$$\binom{\binom{n}{c}m^c}{2} = \frac{\left(\binom{n}{c}m^c\right)\left(\binom{n}{c}m^c - 1\right)}{2}$$

pairs of constraints, and because only polynomially-valued expressions are being multiplied, and because the constraints themselves are polynomially-sized, the total number of constraints that needs to be checked is still polynomial.

Independently from checking local consistency, WMSP is also in P because the verifier can simply attempt to construct his own sparse solution to a constraint problem using the algorithm, and check that it works in polynomial time. Therefore, WMSP is in P. □

D Proof of Polynomial Bound on $p(\delta, \epsilon)$

Claim D.1. *There exists a polynomially bounded value $N = p(\delta, \epsilon)$ such that for each AMSP instance, Algorithm 6.1 produces an assignment that satisfies the instance with error at most ϵ with success probability at least $1 - \delta$.*

Proof. We consider the application of this inequality to an arbitrary marginal with constraint value p. The probability of an iteration of the loop in Algorithm 6.1 selecting a location in the corresponding slice is equal to its marginal value. Each incrementation produces a Bernoulli random variable, and after N trials, the sum of these random variables is a binomial distribution with probability p. After averaging, we find the actual marginal value to be $X = \frac{\text{binom}(N,p)}{N}$. In this case, the problem is analogous to finding how many flips are needed to calculate the fairness of a biased coin with error ϵ and confidence δ. In other words, we want an inequality of the form

$$\Pr[|X - p| \geq \epsilon] \leq \delta.$$

We match the previous inequality with Chebyshev's Inequality, which states that if X is a random variable with finite expected value p and finite non-zero variance σ^2, then for any real number $k \geq 0$,

$$\Pr[|X - p| \geq k\sigma] \leq \frac{1}{k^2}.$$

In doing so, we find that $N = \dfrac{p(1-p)}{\epsilon^2 \delta}$ suffices for a single marginal.

We have shown that the probability of one marginal failing is small; so by a standard union bound argument, it suffices to use

$$N = \binom{n}{c} m^c \cdot \tfrac{p(1-p)}{\epsilon^2 \delta}$$

to have a probability of δ of any constraint failing more than ϵ. Note that N is polynomial in the inverse of the square of the error and the confidence, as desired.

Although not required, using a stronger binomial-specific inequality known as Hoeffding's Inequality can show that a smaller number of trials, namely $N = \binom{n}{c} m^c \cdot \frac{1}{2\epsilon^2} \ln \frac{2}{\delta}$, suffices. □

References

[1] D. Gusfield. A Graph Theoretic Approach to Statistical Data Security. *SIAM J. Comput.*, 17:552-571, 1988.

[2] J.D. Loera. Transportation Polytopes: A Twenty Year Update. *U of California, Davis*, 2008.

[3] J.D. Loera and S. Onn. The Complexity of Three-Way Statistical Tables. *SIAM J. Comput.*, 2004.

[4] J. Sobel. Linear Programming Notes VIII: The Transportation Problem. *U of California, San Diego*, 2002.

[5] Y.-K. Liu. Consistency of Local Density Matrices is QMA-Complete. *Springer Berlin / Heidelberg*, 4110:438-449, 2006.