Understanding Cardinality Estimation using Entropy Maximization

Christopher Ré
University of Wisconsin-Madison
chrisre@cs.wisc.edu

Dan Suciu
University of Washington, Seattle
suciu@cs.washington.edu

ABSTRACT

Cardinality estimation is the problem of estimating the number of tuples returned by a query; it is a fundamentally important task in data management, used in query optimization, progress estimation, and resource provisioning. We study cardinality estimation in a principled framework: given a set of statistical assertions about the number of tuples returned by a fixed set of queries, predict the number of tuples returned by a new query. We model this problem using the probability space, over possible worlds, that satisfies all provided statistical assertions and maximizes entropy. We call this the Entropy Maximization model for statistics (MaxEnt). In this paper we develop the mathematical techniques needed to use the MaxEnt model for predicting the cardinality of conjunctive queries.

Categories and Subject Descriptors

H.2.4 [**Systems**]: Relational Databases

General Terms

Theory

Keywords

Cardinality Estimation, Database Theory, Maximum Entropy, Distinct Value Estimation

1. INTRODUCTION

Cardinality estimation is the process of estimating the number of tuples returned by a query. In relational database query optimization, cardinality estimates are key statistics used by the optimizer to choose an (expected) lowest cost plan. As a result of the importance of the problem, there are many sources of statistical information available to the engine, e.g., *query feedback records* [6,31] and *distinct value counts* [3], and many models to capture some portion of the available statistical information, e.g., *histograms* [17,23], *samples* [12], and *sketches* [2,26]; but on any given cardinality estimation task, each method may return a different (and so, conflicting) estimate. Consider the following cardinality estimation task:

> *"Suppose one is given a binary relation $R(A,B)$ along with estimates for the number of distinct values in R.A, R.B, and for the number of tuples in R. Given a query q, how many tuples should one expect to be returned by q?"*

Each of the preceding methods is able to answer the above question with varying degrees of accuracy; nevertheless, the optimizer still needs to make a single estimate, and so, the task of the optimizer is then to choose a single (best) estimate. Although the preceding methods are able to produce an estimate, none is able to say that it is the best estimate (even for our simple motivating example above). In this paper, our goal is to understand the question raised by this observation: *Given some set of statistical information, what is the best cardinality estimate that one can make?* Building on the principle of *entropy maximization*, we are able to answer this question in special cases (including the above example). Our hope is that the techniques that we use to solve these special cases will provide a starting point for a comprehensive theory of cardinality estimation.

Conceptually, our approach to cardinality estimation has two phases: we first build a consistent probabilistic model that incorporates all available statistical information, and then we use this probabilistic model to estimate the cardinality of a query q. The standard model used in cardinality estimation is the frequency model [30]. For example, this model can express that the frequency of the value a_1 in $R.A$ is f_1, and the frequency of another value a_2 in $R.A$ is f_2. The frequency model is a probability space over a set of possible tuples. For example, histograms are based on the frequency model. This model, however, cannot express cardinality statistics, such as $\sharp R.A = 2000$ (the number of distinct values in A is 2000). To capture these, we use a model where the probability space is over the set of possible instances of R, also called *possible worlds*. To make our discussion precise, we consider a language that allows us to make *statistical assertions* which are pairs (v, d) where v is a view (first order query) and $d > 0$ is a real number. An assertion is written $\sharp v = d$, and its informal meaning is that *"the estimated number of distinct tuples returned by v is d"*. A *statistical program*, $\Sigma = (\bar{v}, \bar{d})$, is a set of statistical assertions, possibly with some constraints. In our language, our motivating question is modeled as a simple statistical program: $\sharp R = d_R$, $\sharp R.A = d_A$, and $\sharp R.B = d_B$. A statistical program defines the statistical information available to the cardinality estimator when it makes its prediction. We give a semantics to this program following prior work [16, 19, 30]: our chief desideratum is that our semantic for statistical programs should take into consideration all of the provided statistical information and nothing else. This is the essence of our study: we want to understand what we can conclude from a given set of statistical information without making ad hoc assumptions. Although the preceding desideratum may seem vague and non-technical, as we explain in §2, mathematically this can be made precise using the *entropy maximization principle*. In prior

PODS'10, June 6–11, 2010, Indianapolis, Indiana, USA.

work [16], we showed that this principle allows us to give a semantics to any consistent set of statistical estimates.[1]

Operationally, given a statistical program Σ, the entropy maximization principle tells us that we are not looking for an arbitrary probability distribution function, but one with a prescribed form. For an arbitrary discrete probability distribution over M possible worlds one needs to specify $M-1$ numbers; in the case of a binary relation $R(A,B)$ over a domain of size N, there are $M = 2^{N^2}$ possible worlds. In contrast, a maximum entropy distribution (MaxEnt) over a program Σ containing t statistical assertions is completely specified by a tuple of t parameters, denoted $\bar{\alpha}$. In our motivating question, for example, the maximum entropy distribution is completely determined by three parameters: one for each statistical assertion in Σ. This raises two immediate technical challenges for cardinality estimation: Given a statistical program Σ, how do we compute the parameters $\bar{\alpha}$? We call this the *model computation problem*. Then, given the parameters $\bar{\alpha}$ and a query q, how does one estimate the number of tuples returned by q? We call this the *prediction problem*. In this work, we completely solve this problem for many special cases, including binary relations where q is a full query (i.e., a conjunctive query without projection).

Our first technical result is an explicit, closed-form formula for the expected size of a conjunctive query without projection for a large class of programs called *hierarchical normal form programs* (HNF programs). The formula expresses the expected size of the query in terms of moments of the underlying MaxEnt distribution: the number of moments and their degree depends on the query, and the size of the formula for a query q is $O(|q|)$. As a corollary, we give a formula for computing the expected size of any conjunctive query (with projection) that uses a number of moments that depends on the size of the domain. Next, we show how to extend these results to more statistical programs. For that, we introduce a general technique called *normalization* that transforms arbitrary statistical programs into normal form programs. A large class of statistical programs are normalized into HNF programs, where we can use our estimation techniques. We solve our motivating question with an application of this technique: to make predictions in this model we normalize it first into an HNF program, then express the expected size of any projection-free query in terms of moments. By combining these two techniques, we solve size estimation for projection-free queries on a large class of models.

To support prediction, we need to compute both the parameters of the MaxEnt distribution and the moments of the MaxEnt distribution efficiently. The first problem is model computation: given the observed statistics, compute the parameters of the MaxEnt distribution that corresponds to those statistics. This is, in general, a very difficult problem and is intimately related to the problem of learning in *statistical relational models* [32]. We show that for *chain programs* the parameters can be computed exactly, for *hypergraph programs* and *binary relational programs* the parameters can be computed asymptotically (as the domain size N grows to infinity), and for general *relational programs* the parameters can be computed numerically. For the last two methods we have observed empirically that the approximations error is quite low even for relatively small values of N (say 300), which makes these approximations useful in practice (especially as input to a numeric solving method). The second problem is: once we have the parameters of the model, compute any given moment. Once the parameters are known, any moment can be computed in time $N^{O(t)}$, where t is the number of parameters of the model, but in some applications this may be too costly. We give explicit closed formulas for approximating the moments, allowing them to be computed in $O(t)$ time.[2] Thus, combining with our previous solution for prediction, we can estimate the expected output size of a projection-free conjunctive query q in time $O(|q|)$.

Our main tool in deriving asymptotic approximation results is a novel approximation technique, called a *peak approximation* that approximates the MaxEnt distribution with a convex sum of simpler distributions. In some cases, the peak approximation is very strong: all finite moments of the MaxEnt distribution are closely approximated by the peak approximation. A classical result in probability theory states that, if two finite, discrete distributions agree on all finite moments then they are the same distribution [29, pg. 35]. And so, if our approximation were not asymptotic then the peak approximation would not be an approximation – it would be the actual MaxEnt distribution.

Outline In §2, we discuss the basics of the MaxEnt model and explain our first technical contribution, *normalization*. In §3, we address *prediction* by showing how to estimate the size of a full query in terms of the moments of an MaxEnt model. Then, we discuss the *model computation* problem and solve several special cases using a novel technique, the *peak approximation*. In addition, we provide source code for Sage programs[3] that demonstrate both the rapid convergence of our asymptotic claims and a proof of concept that our techniques can be implemented efficiently. We discuss related work (§5) and finally conclude (§6).

2. THE MAXENT MODEL FOR STATISTICAL PROGRAMS

We introduce basic notations then review the MaxEnt. CQ denotes the class of conjunctive queries over a relational schema R_1, …, R_m. A *full conjunctive query* is a conjunctive query that contains no variables. A *projection query* is a query that contains a single subgoal without repeated variables. For example, $q(x) :- R(x,y)$ is a projection query, while $q(x) :- R(x,x)$ is not. We also denote projection queries using a named perspective [1], e.g., $R_i(A_1,\ldots,A_t)$ then $R_i.A_1A_2$ denotes the projection of R_i onto the attributes A_1A_2. To specify statistics for range values, as in a histogram, one needs arithmetic predicates such as $x < y$. To simplify presentation, our queries do not contain arithmetic predicates. In Appendix A.6, we extend our results to handle arithmetic predicates.

Let Γ be a set of *full inclusion constraints*, i.e., statements of the form $\forall\bar{x}.R_i(\bar{x}) \Rightarrow R_j(\bar{x})$, R_i and R_j are relation names, and $R_i(\bar{x})$ contains all variables in $\bar{x}$; equivalently, $R_i.X \subseteq R_j$, where X is a set of attributes of R_i.

2.1 Background: The MaxEnt Model

For a fixed, finite domain D and constraints Γ we denote $\mathsf{I}(\Gamma)$ the set of all instances over D that satisfy Γ; the set of all instances over D is $\mathsf{I}(\emptyset)$, which we abbreviate I. A probability distribution on $\mathsf{I}(\Gamma)$ is a set of numbers $\bar{p} = (p_I)_{I\in\mathsf{I}(\Gamma)}$ in $[0,1]$ that sum up to 1. We use the notations p_I and $\mathbf{P}[I]$ interchangeably in this paper.

A *statistical program* is a triple $\Sigma = (\Gamma, \bar{v}, \bar{d})$, where Γ is a set of constraints, $\bar{v} = (v_1,\ldots,v_s)$ and each v_i is a projection query, and $(d_1,\ldots,d_s)$ are positive real numbers. A pair (v_i,d_i) is a *statistical assertion* that we write informally as $\#v_i = d_i$; in the simplest case it can just assert the cardinality of a relation, $\#R_i = d_i$. A probability distribution on $\mathsf{I}(\Gamma)$ *satisfies* a statistical program Σ if $\mathbf{E}_{\bar{p}}[|v_i|] = d_i$,

[1] Intuitively, a program is consistent if there is at least one probability distribution that satisfies it (see §2 for more detail).

[2] We assume here the *unit cost model* [22, pg. 40], i.e., arithmetic operations are constant cost.

[3] Sage is a popular open-source mathematical framework [27].

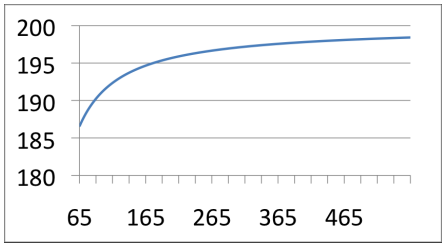

Figure 1: A graph that plots the domain size (on x-axis) versus $\mathbf{E}[\#R.AC]$ (y-axis) for the program $R(A,B,C)$: $\#R = 200$, $\#R.A = 20$, $\#R.B = 30$, $\#R.C = 40$.

for all $i = 1, \ldots, s$. Here $E_{\bar{p}}[|v_i|]$ denotes the expected value of the size of the view v_i, i.e., $\sum_{I \in \mathsf{I}(\Gamma)} |v_i(I)| p_I$. We will also allow the domain size N to grow to infinity. For fixed values $\bar{d}$ we say that a sequence of probability distributions $(\bar{p}^{(N)})_{N>0}$ *satisfies* $\Sigma = (\bar{v}, \bar{d})$ *asymptotically* if $\lim_{N\to\infty} \mathbf{E}_{\bar{p}^{(N)}}[|v_i|] = d_i$, for $i = 1, \ldots, s$.

Given a program Σ, we want to determine the most "natural" probability distribution $\bar{p}$ that satisfies Σ and use it to estimate query cardinalities. In general, there may not exist any probability distribution that satisfies Σ; in this case, we say that Σ is unsatisfiable. We say that a program $\Sigma = (\bar{v}, \bar{d})$ is *satisfiable* if there exists a distribution $\bar{p}$ such that for all i, $\mathbf{E}_{\bar{p}}[|v_i|] = d_i$ and *unsatisfiable* otherwise.[4] On the other hand, there may exist many solutions. To choose a canonical one, we apply the principle of Maximum Entropy (MaxEnt).

Definition 2.1. *A probability distribution $\bar{p} = (p_I)_{I \in \mathsf{I}(\Gamma)}$ is a* MaxEnt *distribution associated to Σ if the following two conditions hold: (1) $\bar{p}$ satisfies Σ, and (2) it has the maximum entropy among all distributions that satisfy Σ, where the entropy of $\bar{p}$ is $H(\bar{p}) = -\sum_{I \in \mathsf{I}(\Gamma)} p_I \log p_I$.*

We refer to a MaxEnt distribution as *the* MaxEnt model, since, as we later show, it is uniquely defined. For a simple illustration, consider the following program on the relation $R(A,B,C)$: $\#R = 200$, $\#R.A = 20$, $\#R.B = 30$, $\#R.C = 40$. Thus, we know the cardinality of R and the number of distinct values of each of the attributes A, B, C. We want to estimate $\#R.AB$, i.e., the number of distinct values of pairs AB. Clearly this number can be anywhere between 30 and 200, but currently there does not exists a principled approach for query optimizers to estimate the number of distinct pairs AB from the other four statistics. The MaxEnt model gives such a principled approach. According to this model, R is a random instance over a large domain D of size N, according to a probability distribution described by the probabilities p_I, for $I \subseteq D^3$. The distribution p_I is defined precisely: it satisfies the four statistical assertions above, and is such that the entropy is maximized. Therefore, the estimate we seek also has a well-defined semantics, as $\mathbf{E}_{\bar{p}}[\#R.AB] = \sum_{I \subseteq D^3} p_I |I.AB|$. This estimate will certainly be between 30 and 200; it will depend on N, which is an undesirable property, but a sensible thing to do is to let N grow to infinity, and compute the limit of $\mathbf{E}_{\bar{p}}[\#R.AB]$. In Figure 1, we plot $\mathbf{E}_{\bar{p}}[\#R.AB]$ as a function of the domain size (N). Interestingly, it very quickly goes to 200, even for small values of N. Thus, the MaxEnt model offers a principled and uniform approach to query size estimation.

To describe the general form of a MaxEnt distribution, we need some definitions. Fix a program $\Sigma = (\Gamma, \bar{v}, \bar{d})$, and so a set of constraints Γ and views $\bar{v} = (v_1, \ldots, v_s)$.

[4] Using a compactness argument, we show in Appendix A.2 that if a program is satisfiable, there is at least one distribution that maximizes entropy.

Definition 2.2. *The* partition function *for $\Sigma = (\Gamma, \bar{v}, \bar{d})$ is the following polynomial T with s variables $\bar{x} = (x_1, \ldots, x_s)$:*

$$T^{\Sigma}(\bar{x}) \quad = \quad \sum_{I \in \mathsf{I}(\Gamma)} x_1^{|v_1(I)|} \cdots x_s^{|v_s(I)|}$$

Let $\bar{\alpha} = (\alpha_1, \ldots, \alpha_s)$ be s positive real numbers. The probability distribution associated to $(\Sigma, \bar{\alpha})$ *is:*

$$p_I \quad = \quad \omega \alpha_1^{|v_1(I)|} \cdots \alpha_s^{|v_s(I)|} \qquad (1)$$

where $\omega = 1/T^{\Sigma}(\bar{\alpha})$.

We write T instead of T^{Σ} when $\Gamma, \bar{v}$ are clear from the context (notice that T does not depend on $\bar{d}$). The partition function can be written more compactly as:

$$T(\bar{x}) \quad = \quad \sum_{k_1, \ldots, k_s} C_{\Gamma}(N, k_1, \ldots, k_s) x_1^{k_1} \cdots x_s^{k_s}$$

where $C_{\Gamma}(N, k_1, \ldots, k_s)$ denotes the number of instances I over a domain of size N that satisfy Γ and for which $|v_i(I)| = k_i$, for all $i = 1, \ldots, s$.

The following is a key characterization of MaxEnt distributions.

Theorem 2.3. *[15, page 355] Let $\Sigma = (\bar{v}, \bar{d})$ be a statistical program. For any probability distribution $\bar{p}$ that satisfies the statistics Σ the following holds: $\bar{p}$ is a* MaxEnt *distribution iff there exists parameters $\bar{\alpha}$ s.t. $\bar{p}$ is given by the Equation (1) (equivalently: $\bar{p}$ is associated to $(\Sigma, \bar{\alpha})$).*

We refer to Jaynes [15, page 355] for a full proof; the "only if" part of the proof is both simple and enlightening, and we include it in Appendix A.1 for completeness. To justify the statement "the MaxEnt model", we need some notation: we say that a tuple of m views $\bar{v}$ is *affinely dependent* over a set of instances $\mathsf{I}(\Gamma)$ if there exist $m+1$ real numbers $\bar{c}, d$, not all zero, such that:

$$\forall I \in \mathsf{I}(\Gamma). \quad \sum_{j=1,\ldots,s} |v_j(I)| c_j = d$$

We say $\bar{v}$ is affinely independent over $\mathsf{I}(\Gamma)$ if no such $\bar{c}, d$ exist. We now justify the term "the MaxEnt Model":

Theorem 2.4. *Let $\Sigma = (\Gamma, \bar{v}, \bar{d})$ be a satisfiable statistical program where $\bar{v}$ is affinely independent over $\mathsf{I}(\Gamma)$, then there is a unique tuple of parameters $\bar{\alpha}$ that satisfies Σ and maximizes entropy.*

For completeness we include a full proof in Appendix A.2. From now on, for any program $\Sigma = (\Gamma, \bar{v}, \bar{d})$ that we consider, we assume that $\bar{v}$ is affinely independent over $\mathsf{I}(\Gamma)$. We verify this assumption for the programs that we consider in Appendix A.3. We illustrate with examples:

Example 2.5 The Binomial-Model Consider a relation $R(A,B)$ and the statistical assertion $\#R = d$. The partition function is the binomial, $T(x) = \sum_{k=0,N^2} \binom{N^2}{k} x^k = (1+x)^{N^2}$, and the MaxEnt model turns out to be the probability model that randomly inserts each tuple in R independently, with probability $p = d/N^2$. We need to check that this is a MaxEnt distribution: given an instance I of size k, $\mathbf{P}[I] = p^k(1-p)^{N^2-k}$, which we rewrite as $\mathbf{P}[I] = \omega\alpha^k$. Here $\alpha = p/(1-p)$ is the *odds* of a tuple, and $\omega = (1-p)^{N^2} = \mathbf{P}[I = \emptyset]$. This is indeed a MaxEnt distribution by Theorem 2.3. Asymptotic query evaluation on a generalization of this distribution to multiple tables was studied in Dalvi *et al.* [8]. □

In this example, α is the odds of a particular tuple. In general, the MaxEnt parameters may not have a simple probabilistic interpretation.

We define a normal form for statistical program.

Definition 2.6. Σ *is in* normal form *(NF) if all statistical assertions are on base tables; otherwise, it is in* non-normal form *(NNF).*

For illustration, consider the relation $R(A_1, A_2)$. The program $\#R = 20$, $\#R.A_1 = 10$, and $\#R.A_2 = 5$ where $\Gamma = \emptyset$ is in NNF. Consider three relation names $S(A_1, A_2)$, $R_1(A_1)$, $R_2(A_2)$. The program with constraints $S.A_i \subseteq R_i$ for $i = 1, 2$ and statistical assertions $\#S = 20$, $\#R_1 = 10$, $\#R_2 = 5$ is in NF.

We will show that any statistical program can be translated into a statistical program in normal form, but first we illustrate some important statistical programs.

2.2 Important Programs

We describe two classes of programs that are central to this paper: *relational programs* and *hypergraph programs*.

2.2.1 Relational Statistical Programs

Definition 2.7. *Fix a single relation name $R(A_1, \ldots, A_m)$. A* relational program *is a program $\Sigma = (\bar{v}, \bar{d})$ where every statistical assertion is of the form $\#R.X = d$ for $X \subseteq \{A_1, \ldots, A_m\}$.*

There are no constraints in a relational program. Relational programs are in NNF.

A relational program is called *hierarchical* if for any two sets of attributes X, Y occurring in statistical assertions, the following condition holds:

$$X \cap Y = \emptyset \text{ or } X \subseteq Y \text{ or } Y \subseteq X$$

A relational program is called *simple* it consists of $m+1$ assertions: $\#R.A_i = d_i$ for $i = 1, \ldots, m$, and[5] $\#R = d_R$. Clearly, a simple program is also hierarchical. We always order the parameters and assume w.l.o.g. $d_1 \leq d_2 \leq \ldots d_m \leq d_R$. Our motivating example in the introduction is a simple relational program of arity 2.

We give now the partition function for a simple relational program. Consider m sets, $A_1, \ldots, A_m$, such that $|A_i| = k_i$ for $i = 1, \ldots, m$. Denote by $r(\bar{k}, l) = r(k_1, \ldots, k_m, l)$ the number of relations $R \subseteq A_1 \times \cdots \times A_m$ such that $|R| = l$ and $|R.A_i| = k_i$ for $i = 1, \ldots, k$.

Proposition 2.8. *The partition function for a simple relational program Σ_R of arity m is:*

$$T^{\Sigma_R}(\bar{\alpha}, \gamma) = \sum_{\bar{k}, l} \binom{N}{\bar{k}} \alpha^{\bar{k}} r(\bar{k}, l) \gamma^l$$

Here, $\binom{N}{\bar{k}}\alpha^{\bar{k}}$ is a short hand for $\prod_{i=1,\ldots,m} \binom{N}{k_i}\alpha_i^{k_i}$. Note that the binomial coefficient ensures that T has only finitely many non-zero terms (finite support).

The function $r(\bar{k}, l)$ is difficult to compute. One can show, using the inclusion/exclusion principle, that, for $m = 2$:

$$r(k_1, k_2, l) = \sum_{\substack{j_1 = 0, k_1 \\ j_2 = 0, k_2}} (-1)^{j_1 + j_2} \binom{k_1}{j_1} \binom{k_2}{j_2} \binom{(k_1 - j_1)(k_2 - j_2)}{l}$$

This generalizes to arbitrary m. To the best of our knowledge, there is no simple closed form for r: we will circumvent computing r using normalization.

[5] $\#R$ is equivalent to $\#R.A_1A_2 \ldots A_m$.

2.2.2 Hypergraph Statistical Programs

Definition 2.9. *Fix a set of relation names $R_1, R_2, \ldots, R_m$. A hypergraph program consists of Σ, Γ, where Σ has one statistical assertion $\#R_i = d_i$ for every relation name R_i, and Γ consists of inclusion constraints of the form $R_i.X \subseteq R_j$, where X is a subset of the attributes of R_i.*

A hypergraph program is in NF. If there are no constraints, then a hypergraph program consists of m independent Binomial models. The addition of constraints changes the model considerably.

We consider two important special cases of hypergraph programs in this paper. The first is a chain program. Fix m relation names: $R_1(A_1, \ldots, A_m)$, $R_2(A_2, \ldots, A_m)$, …, $R_m(A_m)$. A *chain program* of size m, Σ_{Cm}, is a hypergraph program where the set of constraints are: $R_{i-1}.A_iA_{i+1} \ldots A_m \subseteq R_i$, for $i = 2, \ldots, m$. For example, Σ_{C2} is the following program on $R_1(A_2, A_1)$, and $R_2(A_2)$: $\#R_1 = d_1$, $\#R_2 = d_2$, and $R_1.A_2 \subseteq R_2$.

Proposition 2.10. *(Chain Partition Function) Let Σ_{Cm} be a chain program of size $m \geq 1$. Denote the parameters of Σ_{Cm} as $\alpha_1, \ldots, \alpha_m$. Then its partition function satisfies the recursion:*

$$\begin{aligned} T^{\Sigma_{C1}}(\alpha_1) &= (1 + \alpha_1)^N \\ T^{\Sigma_{Cj+1}}(\alpha_1, \ldots, \alpha_{j+1}) &= \left(1 + \alpha_{j+1} T^{\Sigma_{Cj}}(\alpha_1, \ldots, \alpha_j)\right)^N \end{aligned}$$

for $j = 1, 2, \ldots, m-1$.

The partition function $T^{\Sigma_{Cm}}$ is sometimes referred to as a *cascading binomial* [8].

Example 2.11 For Σ_{C2}, the partition function on a domain of size N is:

$$T^{\Sigma_{C2}}(\bar{\alpha}) = (1 + \alpha_2(1 + \alpha_1)^N)^N$$

Given $\bar{d} = (d_1, d_2)$, we need to find the parameters α_1, α_2 for which the probability distribution defined by $T^{\Sigma_{C2}}$ has $E[|R_1|] = d_1$ and $E[|R_2|] = d_2$. We show in Appendix A.4. that the solutions are $\alpha_1 = \frac{d_1}{d_2 N - d_1}$ and $\alpha_2 = \frac{d_2}{N - d_2}(1 + \alpha_1)^{-N}$.

The second special case is the following. A *simple hypergraph program* of size m is a hypergraph program over $S(A_1, \ldots, A_m)$, $R_1(A_1), \ldots, R_m(A_m)$, where the constraints are $S.A_i \subseteq R_i$ for $i = 1, \ldots, m$. We denote by Σ_{Hm} a simple hypergraph program of size m, and will refer to it, with some abuse, as a hypergraph program. Its partition function is:

Proposition 2.12 (Hypergraph Partition Function). *Given a hypergraph program Σ_{Hm} let $\bar{\alpha}$ be a tuple of m parameters (one for each R_i) and γ be the parameter associated with the assertion on S. Then, the partition function is given by:*

$$T^{\Sigma_{Hm}}(\bar{\alpha}, \gamma) = \sum_{\bar{k}} t(\bar{\alpha}, \gamma; \bar{k}) \text{ where } t(\bar{\alpha}, \gamma; \bar{k}) = \binom{N}{\bar{k}} \bar{\alpha}^{\bar{k}} (1 + \gamma)^{\prod_i k_i}$$

We call $t(\bar{\alpha}; \bar{k})$ a term function.

Here $\binom{N}{\bar{k}}$ denotes $\prod_i \binom{N}{k_i}$, and $\bar{\alpha}^{\bar{k}}$ denotes $\prod_i \alpha_i^{k_i}$. Note that the term function is simpler than that in Prop. 2.8.

This partition function corresponds to a simple random process: select random values for R_i from the domain using a Binomial distribution, then we select a (random) subset of edges (hyperedges) from their cross product using another Binomial distribution.

Example 2.13 The hypergraph program Σ_{H2} is over three relations, $S(A_1, A_2)$, $R_1(A_1)$, and $R(A_2)$, two constraints $S.A_1 \subseteq R_1$, $S.A_2 \subseteq R_2$, and three statistical assertions: $\#R_1 = d_1$, $\#R_2 = d_2$, $\#S = d_S$. Denoting α_1, α_2, and γ the parameters of the MaxEnt model, we have:

$$T^{\Sigma_{H2}}(\alpha_1, \alpha_2, \gamma) = \sum_{k_1,k_2} \binom{N}{k_1}\binom{N}{k_2}\alpha_1^{k_1}\alpha_2^{k_2}(1+\gamma)^{k_1k_2}$$

This expression is much simpler than that in Prop. 2.8, but it still does not have a closed form. To compute moments of this distribution (needed for expected values) one needs sums of N^2 terms. The difficulty comes from $(1+\gamma)^{k_1k_2}$: when $k_1k_2\gamma = o(1)$, this term is $O(1)$ and the partition function behaves like a product of two Binomials, but when $k_1k_2\gamma = \Omega(1)$ it behaves differently.

In the full paper, we generalize hypergraphs to define hierarchical normal form programs; these programs play the role of hypergraphs for (non-simple) hierarchical relational programs.

2.3 Normalization

We give here a general, and non-obvious procedure for converting any NNF statistical program Σ into an NF program, with additional inclusion constraints; in fact, this theorem is the reason why we consider inclusion constraints as part of our statistical programs.

Theorem 2.14 below shows one step of the normalization process: how to replace a statistical assertion on a projection with a statistical assertion on a base table, plus one additional inclusion constraint. Repeating this process normalizes Σ.

We describe the notation in the theorem. Recall that $\bar{R} = (R_1, \ldots, R_m)$. Let $\bar{v}$ be a set of s projection views, and assume that v_s is not a base relation. Thus, the statistic $\#v_s = d_s$ is in NNF. Let Q be a new relational symbol of the same arity as v_s, and set $\bar{R}' = \bar{R} \cup \{Q\}$, $\Gamma' = \Gamma \cup \{v_s \subseteq Q\}$. Replace the statistical assertion $\#v_s = d_s$ with $\#Q = d'_s$ (where the number d'_s is computed as described below). Denote $a = \text{arity}(Q)$. Denote $\bar{w}$ the set of views obtained from $\bar{v}$ by replacing v_s with Q.

Let's examine the MaxEnt distributions for $(\Gamma, \bar{v})$ and for $(\Gamma', \bar{w})$. Both have the same number of parameters (s). The former has m relations as outcomes: $R_1, \ldots, R_m$; the latter has $m+1$ outcomes $R_1, \ldots, R_m, Q$. Consider a MaxEnt distribution for the latter, and examine what happens if we compute the marginals over $R_1, \ldots, R_m$: it turns out that the marginal is another MaxEnt distribution. More precisely:

Theorem 2.14 (Normalization). *Consider a* MaxEnt *distribution for $\bar{w}$, with parameters $\beta_1, \ldots, \beta_s$ and outcomes $R_1, \ldots, R_m, Q$. Then the marginal distribution over $R_1, \ldots, R_m$ is a* MaxEnt *distribution, with parameters given by $\alpha_i = \beta_i$ for $i = 1, \ldots, s-1$, and $\alpha_s = \frac{\beta_s}{1+\beta_s}$. In addition, the following relations hold between the partition functions T for $(\Gamma, \bar{v})$ and U for $(\Gamma', \bar{w})$:*

$$T(\bar{\alpha}) = \frac{U(\bar{\beta})}{(1+\beta_s)^{N^a}} \tag{2}$$

Finally, the following relationships holds between the expected sizes of the views in the statistical programs:

$$\begin{aligned} \mathbf{E}_T[|v_s|] &= N^a\alpha_s + (1-\alpha_s)\mathbf{E}_U[|Q|] \\ \mathbf{E}_T[|v_i|] &= \mathbf{E}_U[|w_i|] \quad \textit{for } i = 1, \ldots, s-1 \end{aligned} \tag{3}$$

The last equation tells us how to set the expected sized d'_s of Q to obtain the same distributions, namely $d'_s = (d_s - N^a\alpha_s)/(1-\alpha_s)$.

Example 2.15 The A, R-Model (Cascading Binomials) Consider two statistical assertions on $R(A, B)$: $\#R = d_1$ and $\#R.A = d_2$. This is not normalized. We use Theorem 2.14 to normalize it. For that, add a new relation symbol $Q(A)$, the constraint $R.A \subseteq Q$, and make the following two statistical assertions, $|Q| = c$, $|R| = d_1$; the new constant c to be determined shortly. Example 2.11 gives us the solution to the normalized statistic, namely $\beta_1 = d_1/(cN - d_1)$ and $\beta_2 = c/(N-c)(1+\beta_1)^{-N}$. We use these to solve the original, non-normalized model: $\alpha_2 = \beta_2/(1+\beta_2)$, $\alpha_1 = \beta_1$. Next, we use Theorem 2.14 to obtain: $c = N\alpha_2 + (1-\alpha_2)d_2$. When $N \to \infty$ this equation becomes $c = ce^{-d_2/c} + d_2$, which yields a unique c for any (d_1, d_2). See Appendix A.5 for an explicit computation of c in terms of d_1, d_2.

Example 2.16 To appreciate the power of normalization, we will illustrate on the NNF program on $R(A, B)$: $\#R.A = d_1$, $\#R.B = d_1$, and $\#R = d$. Let $\alpha_1, \alpha_2, \gamma$ be the associated parameters of MaxEnt. Its partition function $T(\alpha_1, \alpha_2, \gamma)$ is a complicated expression given by Prop.2.8. The NF Program has three relations $R_1(A_1)$, $R_2(A_2)$ and $R(A_1, A_2)$, statistics $\#R_1 = c_1, \#R_2 = c_2, \#R = c$, and constraints $R.A_1 \subseteq R_1$, $R.A_2 \subseteq R_2$. Its partition is $U(\beta_1, \beta_2, \gamma) = \sum_{k_1,k_2}\binom{N}{k_1}\binom{N}{k_2}\beta_1^{k_1}\beta_2^{k_2}(1+\gamma)^{k_1k_2}$ (see Example 2.13). After applying the normalization theorem twice, we obtain the following identity:

$$T(\alpha_1, \alpha_1, \gamma) \quad = \quad (1+\beta_1)^{-N}(1+\beta_2)^{-N}U(\beta_1, \beta_2, \gamma)$$

where $\alpha_i = \beta_i/(1+\beta_i)$ for $i = 1, 2$. Moreover, $d_i = N\alpha_i + (1-\alpha_i)c_i$ for $i = 1, 2$ and $d = c$. This translation allows us to do predictions for the NNF program by reduction to the (more manageable) NF hypergraph program. This justifies the normalization theorem, and our interest in hypergraph programs.

As an application of the Normalization theorem we give a non-trivial result both for simple hypergraph, and for simple relational programs. Given a statistical program $\Sigma = (\bar{v}, \bar{d})$, consider the function $F(\bar{\alpha}) = \bar{d}$ in Theorem 2.4: F maps parameters $\bar{\alpha}$ to statistics $\bar{d}$. We say that F is i, j-increasing if $\partial F_i/\partial\alpha_j > 0$. It is well known that, for any MaxEnt distribution, F is i, i-increasing [15, pg. 359], and that this fails in general for $i \neq j$: furthermore, F is i, j-increasing iff it is j, i-increasing.

Theorem 2.17. *For both simple hypergraph programs and simple relational programs, F is i, j-increasing, for all i, j.*

In the full paper, we prove this for hypergraphs directly, by exploiting the special shape of the partition function, then use the normalization theorem to extend it to relational programs.

2.4 Problem Definitions

We study two problems in this paper. One is the *model computation problem*: given a statistical program $\Sigma = (\Gamma, \bar{v}, \bar{d})$, find the parameters $\bar{\alpha}$ for the MaxEnt model such that $\bar{\alpha}$ satisfies Σ. The other is the *prediction* problem, given the parameters of a model and a query $q(\bar{x})$, compute $\mathbf{E}[|q(\bar{x})|]$ in the MaxEnt distribution. We first discuss the prediction problem.

3. PREDICTION

In this section, we describe how to estimate the size of a projection-free conjunctive query q on a hypergraph program. Then using normalization, we show how to estimate the expected size of a query on a relational program. Throughout this section we assume that the parameters of the model are given: we discuss in the next section how to compute these parameters given a statistical program.

3.1 Evaluating Full Queries

Our technique is to rewrite $\mathbf{E}[|q(\bar{x})|]$ in terms of the moments of the MaxEnt distribution. We first reduce computing $\mathbf{E}[|q(\bar{x})|]$ to

computing $\mathbf{P}[q']$ for several Boolean queries q'. Then, we provide an explicit, exact formula for $\mathbf{P}[q']$ in terms of moments of the MaxEnt distribution.

3.1.1 From Cardinalities to Probabilities

We start from the observation:

$$\mathbf{E}[|q(\bar{x})|] = \sum_{\bar{c}\in D^t} \mathbf{P}[q(\bar{x}/\bar{c})]$$

where $q(\bar{x}/\bar{c})$ means substituting x_i with c_i for $i = 1, \ldots, t$, where t is the number of head variables in q. The MaxEnt model is invariant under permutations $f : D \to D$ of the domain: for any instance I, $\mathbf{P}[I] = \mathbf{P}[f(I)]$. Therefore, $\mathbf{P}[q(\bar{x}/\bar{c})]$ is the same for all constants $\bar{c}$ up to a permutation. We exploit this in order to simplify the formula above, as illustrated by this example:

Example 3.1 If $q(x, y, z) = R(x, y), R(y, z), x \neq y, y \neq z, x \neq z$ then:

$$\sum_{c_1,c_2,c_3} \mathbf{P}[q(c_1, c_2, c_3)] \quad = \quad \langle N\rangle_{(3)}\mathbf{P}[q(a_1, a_2, a_3)]$$

where $\langle N\rangle_{(k)} = N(N-1)\cdots(N-k+1)$ is the falling factorial. Here a_1, a_2, a_3 are three fixed (but arbitrary) constants, and $q(a_1, a_2, a_3) = R(a_1, a_2), R(a_2, a_3)$.

In general, let C be the set of all constants appearing in q and in any definition in $\bar{v}$, and let $A = \{a_1, \ldots, a_t\}$ be distinct constants. Consider all substitutions $\theta : \{x_1, \ldots, x_t\} \to A \cup C$: call θ, θ_1 equivalent if there exists a permutation $f : A \to A$ s.t.[6] $\theta_1 = f \circ \theta$. Call θ *canonical* if for any other equivalent substitutions θ_1, $\exists i$ s.t. $\forall j = 1, \ldots, i-1$, $\theta(x_j) = \theta_1(x_j)$, and $\theta(x_i) = a_k$, $\theta_1(x_i) = a_l$ and $k < l$. Let Θ be the set of canonical substitutions.

Proposition 3.2. *With the notations above:*

$$\mathbf{E}[|q(\bar{x})|] \quad = \quad \sum_{\theta\in\Theta} \langle N - |C|\rangle_{(|\theta(\bar{x})\cap A|)}\mathbf{P}[q(\theta(\bar{x}))]$$

The number of terms in the sum is $\leq (|C| + t)^t$; it depends only on the query, not the domain. Thus, the size estimation problem for $q(\bar{x})$ reduces to computing the probability of several Boolean queries. From now on we will consider only Boolean queries in this section.

3.1.2 Query Answering on Simple Programs

A *full query* is a Boolean query without variables; e.g. $q = R(a, b), R(a, d)$. We give here an explicit equation for $\mathbf{P}_\Sigma[q]$, over the MaxEnt distribution given by a program Σ, for the case when Σ is either a simple hypergraph program, or a simple relational program. Note that, in probabilistic databases [9], computing the probability of q for a full query is trivial, because all tuples are assumed to be either independent or factored into independent sets. MaxEnt models, however, are not independent, and cannot be decomposed into simple independent factors. As a result, computing $\mathbf{P}_\Sigma[q]$ is non-trivial. Computing $\mathbf{P}_\Sigma[q]$ intimately relies on the combinatorics of the underlying MaxEnt distribution, and so, we are only able to compute $\mathbf{P}_\Sigma[q]$ directly for hierarchical NF programs.

Simple Hypergraph Programs We start with the case of a simple hypergraph program Σ over $S(A_1, \ldots, A_m)$ and $R_i(A_i)$ for $i = 1, \ldots, m$; recall the constraints $S.A_i \subseteq R_i$, $i = 1, \ldots, m$. Let $q = g_1, g_2, \ldots$ be a full conjunctive query: each g_i is a grounded tuple. Denote:

$$\begin{aligned} q.A_i &= \{a \mid (S(\bar{c}) \in q \text{ and } c_i = a) \text{ or } \exists j.\ R_i(a) = g_j\} \\ u_i &= |q.A_i| \\ u_s &= |\{g \mid g \in q, g = S(\bar{c})\}| \end{aligned}$$

[6] We extend f to $C \cup A \to C \cup A$ by defining it to be the identity on C.

Denote $\langle X\rangle_{(k)} = X(X-1)\cdots(X-k+1)$, the k-falling factorial. Given the probability space $\mathbf{P}_\Sigma$, we write A_i for the random variable $|R_i.A_i|$. Then $\mathbf{E}[\langle A_i\rangle_{(u)}]$ denotes the expected value of the u-falling factorial of A_i; it can be computed directly as $\sum_{\bar{k}}\langle k_i\rangle_{(u)}t(\bar{\alpha}, \gamma, \bar{k})$ in time $O(N^m)$ (see Prop 2.12), and we give more effective methods in the next section.

Theorem 3.3. *Let Σ_{Hm} be a hypergraph program of size m over a domain of size N. Then, following equation holds:*

$$\mathbf{P}_\Sigma[q] = \left(\frac{\gamma}{1+\gamma}\right)^{u_S} \mathbf{E}\left[\prod_{i=1,\ldots,m} \frac{\langle A_i\rangle_{(u_i)}}{\langle N\rangle_{(u_i)}}\right]$$

This theorem allows us to reduce query answering to moment computation. Thus, if we can compute moments of the MaxEnt distribution (and know the parameter γ), we can estimate query cardinalities. We extend this result to *hierarchical NF programs* in the full paper.

Example 3.4 Let $q = S(a, b), S(a, b'), R_1(a'), R_2(b'')$. Then $q.A_1 = \{a, a'\}$ and $q.A_2 = \{b, b', b''\}$, $u_1 = 2, u_2 = 3, u_s = |\{S(a, b), S(a, b')\}| = 2$. We have:

$$\mathbf{P}_\Sigma[q] \quad = \quad \left(\frac{\gamma}{1+\gamma}\right)^2 \frac{\mathbf{E}[A_1(A_1-1)A_2(A_2-1)(A_2-2)]}{N^2(N-1)^2(N-2)}$$

Example 3.5 Given a binary relation $R(A, B)$, the *fanout* X_a of a node a is the number of tuples $(a, b) \in R$. Let X denote the expected fanout over all nodes a. Computing the expected fanout is an important problem in optimization. By linearity of expectation we have $\mathbf{E}[X_a] = (N-1)\mathbf{P}[R(a, b') \mid R(a, b)]$, and Bayes' Rule gives us:

$$\mathbf{E}[X] = (N-1)\frac{\mathbf{P}[R(a,b), R(a,b')]}{\mathbf{P}[R(a,b)]} = \frac{\gamma}{1+\gamma}\frac{\mathbf{E}[A \cdot B \cdot (B-1)]}{\mathbf{E}[A \cdot B]}$$

Theorem 3.3 gives us an identity between $\#S$ and the expectation of the product $\prod_i A_i$. Consider the query $q = S(a, b)$; obviously, $\mathbf{E}[|S|] = N^2\mathbf{P}[q]$ by linearity of expectation. We also have $\mathbf{P}[q] = \frac{\gamma}{1+\gamma}\mathbf{E}[A_1A_2]N^{-2}$ and so $\mathbf{E}[S] = \frac{\gamma}{1+\gamma}\mathbf{E}[A_1A_2]$.

Simple Relational Programs Next, we discuss the case when Σ_R is a simple relational program: $R(A_1, \ldots, A_m)$, with statistics $\#R.A_i = d_i$ for $i = 1, \ldots, m$, $\#R = d$, and no constraints. Let α_i, $i = 1, \ldots, m$ and γ be its parameters. A full query q consists of a set of atoms of the form $R(\bar{c})$. Construct a new hypergraph program Σ_H, by normalizing Σ_R: it has schema $R(A_1, \ldots, A_m)$, $Q_1(A_1)$, $\ldots, Q_m(A_m)$, constraints $R.A_i \subseteq Q_i$, $i = 1, \ldots, m$, and parameters $\beta_i = \alpha_i/(1-\alpha_i)$, $i = 1, \ldots, m$. The MaxEnt distribution given by Σ_H is a probability space with outcomes $R, Q_1, \ldots, Q_m$; from Theorem 2.14 (applied m times) it follows that the marginal distribution of R is precisely the MaxEnt distribution for the Σ_R-program. This discussion implies:

Corollary 3.6. $\mathbf{P}_{\Sigma_R}[q] = \mathbf{P}_{\Sigma_H}[q]$.

In other worlds, we can simply compute a query probability or a cardinality estimate in the NNF model Σ_R by simply computing the same query in the NF model Σ_H. When doing so, we must ensure to translate the parameters α_i to β_i correctly, as in Theorem 2.14. Initially, we found the formula for $\mathbf{P}_\Sigma[q]$ where Σ was a simple relational program (NNF program); this formula was a complicated inclusion-exclusion formula, and it was a pleasant surprise that the formula reduced to a closed-form equation via normalization.

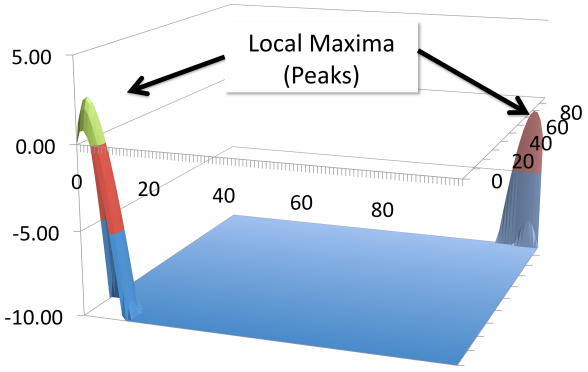

Figure 2: A graph of $\ln t(k,l)$ for the Hypergraph program with $\sharp R.A = 2$ $\sharp R.B = 4$), $\sharp R = 10$ and $N = 99$. For readability, we plot $\ln f(k,l)$ where $f(k,l) = \max\{t(k,l), e^{-10}\}$. Almost all mass comes from the two peaks.

General Conjunctive Queries For a full query q, $\mathbf{P}[q]$ can be computed in terms of one particular moment, of a degree that depends on the query q. For general conjunctive query, one can compute $\mathbf{P}[q]$ in terms of $O(N^v)$ moments, where v is the number of existential variables in the query. We only illustrate here the main idea, by using an example: $q = R(a,x,c)$, where x is an existentially quantified variable. Since $q \equiv \bigvee_{b\in D} R(a,b,c)$, we obtain the following:

$$\begin{aligned}\mathbf{P}[q] &= \sum_{B\subseteq D: B=\{b_1,\ldots,b_k\}} (-1)^{k+1}\mathbf{P}[R(a,b_1,c),\ldots,R(a,b_k,c)] \\ &= \sum_{k\geq 1}\binom{N}{k}\left(\frac{\gamma}{1+\gamma}\right)^k \frac{\mathbf{E}[A\cdot\langle B\rangle_{(k)}C]}{N^2\cdot\langle N\rangle_{(k)}}\end{aligned}$$

Each moment above can be computed in time $O(N^3)$, and there are $O(N)$ moments to compute. In practice, however, one may stop when $k \ll N$. For example, when computing Figure 1, taking $k = 3$, the error ε satisfied $|\varepsilon| \leq 10^{-10}$.

4. MODEL COMPUTATION

We first discuss the peak approximation and then use it to solve the model computation problem for hypergraphs and binary relational programs.

4.1 Peak Approximations

The peak approximation writes a MaxEnt distribution as a convex sum of simpler distributions using two key pieces of intuition: first, in many cases, almost all of the mass in the partition function comes from relatively few terms. Second, around each peak, the function behaves like a simpler function (here, a product of binomials).

To make this intuition more concrete, consider the following hypergraph program: $\sharp R_1.A_1 = 2$, $\sharp R_2.A_2 = 4$ and $\sharp S = 10$ on a domain of size $N = 99$. In Figure 2, we plot $t(k_1,k_2)$ the associated term function: k_1 is on the x axis, and k_2 is on the y axis, and on the z-axis is $\ln t(x,y)$. Most of the mass of $t(k,l)$ is concentrated around $t(2,4)$, i.e., around the expected values given in the program, and some slightly smaller mass is concentrated around $t(99,99)$. The idea of the peak approximation is to locally approximate the term function t in the neighborhood of $(2,4)$ and $(99,99)$ with simpler functions.

The formal setting that we consider in this section is: we are given a hypergraph program Σ_H of size m with relations $R_1\ldots,R_m$ and S, and our goal is to approximate its MaxEnt distribution with a convex sum of products of binomials. We now describe how we approximate the term function of Σ_H (t^{Σ_H}, simply t). Let $\bar{c}$ be a tuple of m constants and denote $P(\bar{c}) = \prod_{i=1,\ldots,m} c_i$. For $i = 1,\ldots,m$, we define a function f_i:

$$f_i(k_i;\alpha_i,\gamma;c_i) = \binom{N}{k_i}\alpha_i^{k_i}(1+\gamma)^{\frac{k_i}{c_i}P(\bar{c})}$$

We think of each c_i as a fixed constant, and so each f_i is a term function for a binomial: to see this, sum over k_i, $\sum_{k_i} f_i(k_i;\alpha_i,\gamma;c_i) = (1+\alpha(1+\gamma)^{P(\bar{c})/c_i})^N$. Then, we define our (local) approximate about $\bar{c}$ using a function $\tilde{f}$ defined as follows:

$$\tilde{f}(\bar{k};\bar{\alpha},\gamma;\bar{c}) = (1+\gamma)^{(1-m)P(\bar{c})}\times\prod_{i=1} f_i(k_i,\alpha_i,\gamma;c_i)$$

It is interesting to compare $\tilde{f}$ with t from Prop. 2.12: we see that the leading $(1+\gamma)^{(1-m)P(\bar{c})}$ term essentially compensates for over counting. In particular, if $\bar{k} = \bar{c}$, then $t(\bar{k};\bar{\alpha}) = \tilde{f}(\bar{k},\bar{\alpha};\bar{c})$, i.e., there is no error in approximating t with $\tilde{f}$ at $\bar{c}$, which provides some intuition as to why $\tilde{f}$ is a good local approximate to t near $\bar{c}$.

To specify the general peak approximation, we choose several different values for $\bar{c}$, say $\bar{c}^{(1)},\bar{c}^{(2)},\ldots$, and then we approximate t around each such $\bar{c}$ as above. Fix a set Peaks $= \left\{\bar{c}^{(1)},\ldots,\bar{c}^{(s)}\right\}$ of s of such tuples (later, we take Peaks to be the local maxima of t). We define the peak approximation for t, denoted $\tilde{t}$, as:

$$\tilde{t}(\bar{k};\bar{\alpha},\gamma) = \sum_{\bar{c}\in\text{Peaks}}\tilde{f}(\bar{k},\bar{\alpha},\gamma;\bar{c})$$

The partition function associated to the peak approximation, $\tilde{T}$ is obtained by summing $\tilde{t}$ over $\bar{k}$:

$$\tilde{T}(\bar{\alpha},\gamma) = \sum_{\bar{c}\in\text{Peaks}}(1+\gamma)^{(1-m)P(\bar{c})}\times\prod_{i=1,\ldots,m}\left(1+\alpha_i(1+\gamma)^{\frac{k_i}{c_i}P(\bar{c})}\right)^N \quad (4)$$

Notice that $\tilde{T}$ has a much simpler form than the original T: it is a mixture of binomial distributions. This simpler form makes it easy to find the local maxima of $\tilde{t}$ analytical, and as we show later, compute all of the moments of $\tilde{T}$ analytically. We call $\tilde{T}$ the peak approximation for T defined by Peaks. Our technique is to replace the complicated MaxEnt distribution T with the simpler partition function $\tilde{T}$. In the next section, we show how to find Peaks and so specify $\tilde{T}$.

4.2 Finding the Peaks

Fix a hypergraph program Σ. We take Peaks to be the set of local maxima for the term function t^Σ. Intuitively, this is where T^Σ's mass is concentrated, so it makes sense to locally approximate t near the peaks. One concern is that the size of Peaks could grow with the domain size, N, which would make our approximation undesirable; below, we show a surprising fact: for hypergraph programs, $|\text{Peaks}| \leq 2$.

Theorem 4.1 (Number of Peaks). *Let t be the term function for any hypergraph program Σ_H. Then, for any fixed $\bar{\alpha}$ such that $\alpha_i > 0$, for $i = 1,\ldots,m$, $t(\alpha,\bar{k})$ has at most 2 local maxima (in $\bar{k}$) and so $|\text{Peaks}(T^\Sigma)| \leq 2$.*

We prove this theorem in several steps: a local maxima of $t(\bar{\alpha};\bar{k})$ function is at critical point; we observe that, by the *mean value theorem* [25, pg. 108], to find such critical points it suffices to find values of $\bar{k}$ such that $t(\bar{k}) = t(\bar{k}+e^{(i)})$ for $i = 1,\ldots,m$ where $e^{(i)}$ is

the unit vector in direction i (also known as a *variational derivative* [15]). This yields a system of equations. We then show that all solutions of this system of equations are the zeros of a single equation in a single variable; then, we show that this function has at most 3 zeros by showing that the third derivative of this function has a constant sign. We conclude that at most 2 solutions can be local maxima. We call $\tilde{T}$ the peak approximation for T where Peaks is the set of local maxima of t. Denote this set Peaks $= \left\{\bar{c}^{(1)}, \bar{c}^{(2)}\right\}$.

We give a sufficient condition under which, informally, the peaks approximation will be a good approximation to the hypergraph partition function. The lemma is unfortunately technical and requires three conditions, which informally say: (1) that the error around each peak is small enough, (2) the peaks are far enough apart, and (3) that the peaks are not in the middle of the space.

Lemma 4.2. *Fix a hypergraph program Σ. Let $N = 1, 2, \ldots,$ and let T_N denote the partition function for Σ on a domain of size. For for each N, let $\tilde{T}_N$ be the peak approximation for T_N and $\bar{c}^{(i,N)}$ for $i = 1, 2$ denote the local maxima of t^N. Assuming that (1) $\ln(1+\gamma)N^{m-2} = o(1)$ and (2) $\min\left|c_i^{(1,N)} - c_j^{(2,N)}\right| \geq N^{-\varepsilon}$ for some $\varepsilon > 0$, and (3) $\exists i$ s.t. $\min\{c_i, N - c_i\} = O(N^{1-\tau})$ for some $\tau > 0$. Then, for any tuple $\bar{s}$ of m positive numbers:*

$$\lim_{N\to\infty} \frac{\mathbf{E}_{T_N}[\prod_{i=1,\ldots,m}\langle A_i\rangle_{(s_i)}]}{\mathbf{E}_{\tilde{T}_N}[\prod_{i=1,\ldots,m}\langle A_i\rangle_{(s_i)}]} = 1$$

We prove this lemma by showing two more general statements: The first informally says that the peaks are a best local, linear approximation (in the exponent), and we use this to write the error in a closed form. The second result is a variation of the standard Chernoff Bound [20], which informally says that binomial distributions are very sharply concentrated. The proof of this sufficient condition then boils down to a calculation that combines these two statements. Next, we use this sufficient condition to verify asymptotic solutions for several statistical programs.

4.3 Model Computation Solutions

We give exact solutions for chain programs, and asymptotic solutions for simple hypergraph programs and simple binary (arity 2) relational programs.

Chain Programs In this section, we abbreviate the chain partition function $T^{C_i}(\alpha_1, \ldots, \alpha_i)$ as $T^{\langle i\rangle}$. We show:

Proposition 4.3. *Given a chain program Σ of size m, then for $j = 1, \ldots, m$*

$$\mathbf{E}[R_j] = \prod_{i=j,\ldots,m} N\frac{\alpha_i T^{\langle i-1\rangle}}{1+\alpha_i T^{\langle i-1\rangle}}$$

Under the convention that $T^{\langle 0\rangle} = 1$.

We now give an $O(m)$ time algorithm to solve the model computation problem by observing the following identity:

$$\frac{d_j}{d_{j+1}} = \frac{\mathbf{E}[R_j]}{\mathbf{E}[R_{j+1}]} = N\frac{\alpha_j T^{\langle j-1\rangle}}{1+\alpha_j T^{\langle j-1\rangle}}$$

The recursive procedure starts with $T^{\langle 0\rangle} = 1$ in the base case; recursively, we compute the value $T^{\langle i\rangle}$ and all moments. We observe that this uses no asymptotic approximations. Summarizing, we have shown:

Theorem 4.4. *Given a chain program Σ of arity m the above algorithm solves the model computation problem in time $O(m)$ for any domain size.*

Hypergraph Programs We solve hypergraph programs of any arity. We show:

Theorem 4.5. *Consider a hypergraph programs of arity $m \geq 2$, where (without loss) $0 < d_1 \leq d_2 \leq \cdots \leq d_m < d_R = O(1)$ then the following parameters are an asymptotic solution:*

$$\alpha_i = d_i N^{-1} \text{ and } \gamma = gN^{1-m} + N^{-m}\left(\delta + \ln\frac{d_R}{Ng}\right)$$

where $g = -\sum_{i=1,\ldots,m} \ln\frac{\alpha_i}{1+\alpha_i}$, and we set $\delta = g^2/2 - (d_1 + d_2)$ if $m = 2$ and $\delta = 0$ if $m > 2$.

The strange looking δ term is due to the fact that (1) $w = \Theta(\ln n)$ and (2) $\ln(1+x) = x + \frac{x^2}{2} + \ldots$, and so when $m = 2$ the first term in γ is $\tilde{O}(N^{-1})$ and so when squared interferes with the second term. The technical key to the proof is the following lemma that computes the set Peaks.

Lemma 4.6. *With the parameters and notation of Theorem 4.5,*

$$\text{Peaks} = \left\{\bar{d} + \delta^{(1)}, \bar{c}^{(2)} + \delta^{(2)}\right\}$$

where $\bar{c}^{(2)} = (N - d_2, N - d_1)$ if $m = 2$ and $\bar{c}^{(2)} = (N, \ldots, N)$ otherwise; and $\bar{\delta}^{(i)}$ is a vector such that $\max_j|\delta_j| = O(N^{-1})$ Moreover, define $w_i = \tilde{T}(\bar{\alpha}, \gamma)^{-1}\sum_{\bar{k}} \tilde{f}(\bar{k}, \alpha, \gamma; \bar{c}^{(i)})$ for $i = 1, 2$ then $w_2 = \frac{d_R}{Ng}$ and $w_1 = 1 - w_2$.

Observe that the conditions of Lemma 4.2 are satisfied, so we may use the peaks instead of the MaxEnt to calculate the moments. Then, it is straightforward to calculate the moments and verify the claims of the theorem: $\mathbf{E}[A_i] = d_i \cdot w_1 + N \cdot w_2 \to d_i$ and $\mathbf{E}[R] = 0 \cdot w_1 + N^m\frac{\gamma}{1+\gamma} \cdot w_2 = d_R + o(1)$. Anecdotally, we have implemented this in Sage and verified that it converges for small N (on the order of hundreds) for a broad range of programs.

Binary Relations. Our solution for binary relations combines normalization and the peaks approach, but there is a subtle twist: consider the solutions from Theorem 4.5, we observe that if we set the moments of the hypergraph to any constant, normalization tells us that the moments of $R.A_i$ tend to zero:

$$\mathbf{E}_R[A_i] = (1+\alpha)\mathbf{E}_H[A_i] - N\alpha \approx d_i - d_i \to 0$$

Here $\mathbf{E}_R$ denotes the moment for the relational program and $\mathbf{E}_H$ denotes the hypergraph program. In fact, binary relations require subtle balancing:

Theorem 4.7. *Given $d_A \leq d_B \leq d_R$ for the relational program Σ over $R(A, B)$. Then, the tuple of parameters $(\alpha_1, \alpha_2, \gamma)$ defined as follows is an asymptotic solution for Σ: Let $\frac{\alpha_1}{1+\alpha_1} = aN^{-1}$, $\frac{\alpha_2}{1+\alpha_2} = bg_1^{-1}$ and $\gamma = g_1N^{-1} + g_2N^{-2}$ where*

$$a = (d_A + 1)/(e^b - 1),\ b = d_b/d_a \text{ and } g_1 = -\mathsf{W}_{-1}(-\alpha b)$$

$$g_2 = g_1^2/2 + (1+\beta)\ln(1+\beta)\frac{d_G - d_B}{N\ln g_1}$$

Here, W_{-1} denotes the value of the Lambert W function over the non-principal (but real-valued) branch.[7] Then, $\tilde{\alpha}$ is an asymptotic solution for Σ.

The proof uses normalization to transform the program into a hypergraph program, and then use a peaked approximation (with non-constant moments) instead of the MaxEnt distribution (via Lemma 4.2). For programs with non-binary relations, we are able to solve these programs using numeric techniques.

[7] The Lambert function is defined by $\mathsf{W}(v) = u$ implies that $v = ue^u$. See Corless *et al.* [7].

4.4 Moment Computation to Answer Queries

We give a closed-form solution for moments of the peak approximation:

THEOREM 4.8. *Let $\tilde{T}$ be a peak approximation (Eq. 4) defined by* Peaks *with parameters $\alpha_1, \ldots, \alpha_m, \gamma$. Then, for any $\bar{s} \in \mathbb{N}^m$ the following equation holds:*

$$\mathbf{E}\left[\prod_{i=1,\ldots,m} \langle A_i \rangle_{(s_i)}\right] = \sum_{\bar{c} \in \text{Peaks}} \prod_{i=1,\ldots,m} N \frac{\alpha_i^{s_i}(1+\gamma)^{s_i P(\bar{c})/c_i}}{1+\alpha_i^{s_i}(1+\gamma)^{s_i P(\bar{c})/c_i}} w(\bar{c})$$

where $w(\bar{c}) = \frac{\sum_{\bar{k}} t(\bar{k};\bar{\alpha},\gamma,\bar{c})}{\sum_{\bar{d} \in \text{peaks}} \sum_{\bar{k}} t(\bar{k};\bar{\alpha},\gamma,\bar{d})}$ and N is the size of the domain.

Combining Theorem 4.8 with Theorem 3.3, we can approximate any full query in $O(|q|)$-time using the peak approximation.

5. RELATED WORK

The first body of related work is in cardinality estimation. As noted above, while a variety of synopses structures have been proposed for cardinality estimation [2, 10, 13, 21], they have all focused on various sub-classes of queries and deriving estimates for arbitrary query expressions has involved ad hoc steps such as the independence and containment assumptions which result in large estimation errors [14]). In contrast, we ask the question: *given some statistical information, what is the best estimate that one can make?*

The MAXENT model has been applied in prior work to the problem of cardinality estimation [19, 30]. However, the focus was restricted to queries that consist of conjunctive selection predicates over single tables. In contrast, we explore a full-fledged MAXENT model that can incorporate statistics involving arbitrary first-order expressions. In our previous work [16], we introduced the MAXENT model over possible worlds for computing statistics, and solved it in a very limited setting, when the MAXENT distribution is a random graph. We left open the MAXENT models for cardinality estimation that are not random graphs, such as the models we solve in this paper. In another work [17], we discussed a MAXENT model for set/bag semantics: we did not discuss bag semantics in this paper. Also prior art did not address query estimation. The MAXENT principle also underlies the graphical model approach, notably the model of probabilistic relational model of Getoor *et al.* [11]. Finally, we observe that entropy maximization is a well-established principle in statistics for handling incomplete information [15].

Probabilistic databases [4, 9, 18, 33] focus on efficient query evaluation over a probabilistic database, in which probabilities are specifies with tuples. Our focus is on computing the parameters of a different type of models. The maximum entropy principle underlies graphical models, and so it is interesting future work to explore how the techniques in this paper apply to inference and learning in such approaches, e.g., Sen *et al.* [28] and *Markov Logic Networks* [24].

6. CONCLUSION

In this paper we propose to model database statistics using maximum entropy probability distributions. This model is attractive because any query has a well defined size estimate, all statistics act as a whole, and the model extends smoothly when new statistics are added. As part of our technical development we described three techniques: normalization, query answering via moments, and peak approximations that we believe are of both theoretical and practical interest for solving statistical programs. The next step for our work is to implement a prototype cardinality estimator using the theoretical underpinnings laid out in this paper. We believe that the peak approximation may have broader applications.

Acknowledgments The authors would like to thank the anonymous reviews for their comments that improved the presentation of the paper. We would also like to thank Ben Recht for pointing us to related work in the convex analysis and machine learning literature. This work was partially supported by NSF IIS-0713576.

7. REFERENCES

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison Wesley Publishing Co, 1995.

[2] N. Alon, P. B. Gibbons, Y. Matias, and M. Szegedy. Tracking join and self-join sizes in limited storage. In *PODS*, pages 10–20, 1999.

[3] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. In *STOC*, pages 20–29, 1996.

[4] L. Antova, C. Koch, and D. Olteanu. World-set decompositions: Expressiveness and efficient algorithms. In *ICDT*, pages 194–208, 2007.

[5] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[6] S. Chaudhuri, V. R. Narasayya, and R. Ramamurthy. Diagnosing estimation errors in page counts using execution feedback. In *ICDE*, pages 1013–1022, 2008.

[7] R. M. Corless, D. J. Jeffrey, and D. E. Knuth. A sequence of series for the lambert w function. In *ISSAC*, pages 197–204, 1997.

[8] N. N. Dalvi, G. Miklau, and D. Suciu. Asymptotic conditional probabilities for conjunctive queries. In *ICDT*, pages 289–305, 2005.

[9] N. N. Dalvi and D. Suciu. The dichotomy of conjunctive queries on probabilistic structures. In *PODS*, pages 293–302, 2007.

[10] A. Deligiannakis, M. N. Garofalakis, and N. Roussopoulos. Extended wavelets for multiple measures. *ACM Trans. Database Syst.*, 32(2):10, 2007.

[11] L. Getoor, B. Taskar, and D. Koller. Selectivity estimation using probabilistic models. In *SIGMOD Conference*, pages 461–472, 2001.

[12] P. J. Haas, J. F. Naughton, S. Seshadri, and A. N. Swami. Selectivity and cost estimation for joins based on random sampling. *J. Comput. Syst. Sci.*, 52(3):550–569, 1996.

[13] Y. E. Ioannidis. The history of histograms (abridged). In *VLDB*, pages 19–30, 2003.

[14] Y. E. Ioannidis and S. Christodoulakis. On the propagation of errors in the size of join results. In *SIGMOD Conference*, pages 268–277, 1991.

[15] E. Jaynes. *Probability Theory: The Logic of Science*. Cambridge University Press, Cambridge, UK, 2003.

[16] R. Kaushik, C. Ré, and D. Suciu. General database statistics using entropy maximization. In *DBPL*, pages 84–99, 2009.

[17] R. Kaushik and D. Suciu. Consistent histograms in the presence of distinct value counts. *PVLDB*, 2(1):850–861, 2009.

[18] C. Koch and D. Olteanu. Conditioning probabilistic databases. *PVLDB*, 1(1):313–325, 2008.

[19] V. Markl, N. Megiddo, M. Kutsch, T. M. Tran, P. J. Haas, and U. Srivastava. Consistently estimating the selectivity of conjuncts of predicates. In *VLDB*, pages 373–384, 2005.

[20] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, New York, NY, USA, 2005.

[21] F. Olken. *Random Sampling from Databases*. PhD thesis, University of California at Berkeley, 1993.
[22] C. Papadimitriou. *Computational Complexity*. Addison Wesley Publishing Company, 1994.
[23] V. Poosala and Y. E. Ioannidis. Selectivity estimation without the attribute value independence assumption. In *VLDB*, pages 486–495, 1997.
[24] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.
[25] W. Rudin. *Principles of Mathematical Analysis, Third Edition*. McGraw-Hill Science/Engineering/Math, 3rd edition, January 1976.
[26] F. Rusu and A. Dobra. Sketches for size of join estimation. *ACM Trans. Database Syst.*, 33(3), 2008.
[27] Sage. Open-source mathematics software. `http://sagemath.org`, 2009.
[28] P. Sen and A. Deshpande. Representing and querying correlated tuples in probabilistic databases. In *ICDE*, pages 596–605, 2007.
[29] J. Shao. *Mathematical Statistics*. Springer, 2nd edition, 2003.
[30] U. Srivastava, P. J. Haas, V. Markl, M. Kutsch, and T. M. Tran. Isomer: Consistent histogram construction using query feedback. In *ICDE*, page 39, 2006.
[31] M. Stillger, G. M. Lohman, V. Markl, and M. Kandil. Leo - db2's learning optimizer. In *VLDB*, pages 19–28, 2001.
[32] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.
[33] J. Widom. Trio: A system for integrated management of data, accuracy, and lineage. In *CIDR*, pages 262–276, 2005.

APPENDIX

A. CALCULATION HELPERS

We observe that moments can be written as appropriate derivative operators on a partition function:

Proposition A.1. *Let T^Σ be a partition function for $\Sigma = (\bar{v}, \bar{d})$ with parameters α_v for $v \in \bar{v}$, then we have:*

$$T^\Sigma(\bar{\alpha}) \times \mathbf{E}[|v|^k] = \left(\alpha_v \frac{\partial}{\partial \alpha_v}\right)^k T^\Sigma(\bar{\alpha})$$

where $(\alpha_v \frac{\partial}{\partial \alpha_v})^k$ denotes applying the operator $\alpha_v \frac{\partial}{\partial \alpha_v}$ k times, and

$$T^\Sigma(\bar{\alpha}) \times \mathbf{E}[\langle |v| \rangle_{(k)}] = \alpha_v^k \frac{\partial^k}{\partial^k \alpha_v} T^\Sigma(\bar{\alpha})$$

The proof is straightforward: apply the operators directly to the partition function, T^Σ, in compact form and use linearity of the derivative operator. Since MaxEnt distributions are polynomials, computing derivatives is straightforward (but possibly expensive).

A.1 Proof of Theorem 2.3

The "only if" direction is very simple to derive by using the Lagrange multipliers for solving:

$$F_0 = \sum_{I \in \mathsf{I}} p_I - 1 = 0 \tag{5}$$

$$\forall i = 1, \ldots, s: \qquad F_i = \sum_{I \in \mathsf{I}} |v_i(I)| p_I - d_i = 0 \tag{6}$$

$$H = \text{maximum, where} \qquad H = \sum_{I \in \mathsf{I}} p_I \log p_I \tag{7}$$

According to that method, one has to introduce $s + 1$ additional unknowns, $\lambda, \lambda_1, \ldots, \lambda_s$: an MaxEnt distribution is a solution to a system of $|\mathsf{I}| + s + 1$ equations consisting of Eq. (5), (6), and the following $|\mathsf{I}|$ equations:

$$\forall I \in \mathsf{I}: \qquad \frac{\partial (H - \sum_{i=0,s} \lambda_i G_i)}{\partial p_I} = \log p_I - (\lambda_0 + \sum_{i=1,s} \lambda_i |v_i(I)|) = 0$$

This implies $p_I = \mathtt{exp}(\lambda_0 + \sum_{i=1,s} \lambda_i |v_i(I)|)$, and the claim follows by denoting $\omega = \mathtt{exp}(\lambda_0)$, and $\alpha_i = \mathtt{exp}(\lambda_i)$, $i = 1, \ldots, s$.

A.2 Proof of Theorem 2.4

In this section, we reprove some foklore statements; for a variant of these results see Wainwright and Jordan [32, §3.2].

Fix a domain size N. Given a program $\Sigma = (\Gamma, \bar{v}, \bar{d})$ over a space of instances $\mathsf{I}(\Gamma)$, let $P(\Gamma)$ denote all probability distributions over $\mathsf{I}(\Gamma)$. The set $P(\Gamma)$ is a closed, bounded subset of $\mathbb{R}^{|\mathsf{I}(\Gamma)|}$, thus it is compact. Moreover, $P(\Gamma)$ is convex.

We say that Σ is *satisfiable* if there exists $\bar{p} \in P(\Gamma)$ such that $F(\bar{p}) = \bar{d}$. A hypergraph program $\Sigma_H = (\bar{v}, \bar{d})$ is consistent over a domain of size N if $\bar{d}$ is in the convex hull of the following vectors: $(\bar{c}, z)$ where $z = \prod_{i=1,m} c_i$ where $\bar{c} \in \{0, \ldots, N\}^m$.

Let H denote the entropy, i.e., $H(\bar{p}) = -\sum_{I \in \text{Inst}} \bar{p}_I \log \bar{p}_I$. H is a continuous, real-valued function. Moreover $-H(\bar{p})$ is a convex function since its Hessian is only non-zero on the diagonal, $\frac{\partial}{\partial p_I}^2 - H(\bar{p}) = p_I^{-1}$ and all other (mixed) second derivatives are 0. This shows that $-H$ it is positive definite on the interior of $P(\Gamma)$, which is equivalent to convexity [5, pg. 65].

Given a set of views $\bar{v}$ define $E : P \to \mathbb{R}^t$ by $E(\bar{p}) = \bar{c}$ where

$$\bar{c}_j = \sum_{I \in \text{Inst}} \bar{p}_I \left| v_j(I) \right|$$

Proposition A.2. *The set $E^{-1}(\bar{d})$ is compact.*

Proof. We observe that E is continuous. Hence, $E^{-1}(\bar{d})$ is a closed set. Since $P(\Gamma)$ is compact, this means that $E^{-1}(\bar{d})$ is a closed subset of a compact set, and so compact. □

Thus, the entropy H takes a maximum value on the set. Formally,

$$\sup_{\bar{p} \in E^{-1}(d)} H(\bar{p}) = H(\bar{q})$$

for some $\bar{q} \in E^{-1}(\bar{d})$, which proves that there is at least one maximum entropy probability distribution.

A.3 Uniqueness

Proposition A.3. *Given a satisfiable statistical program Σ, then there is a unique probability distribution that satisfies Σ.*

Proof. Consider the negative entropy function $-H(\bar{p})$. By compactness and continuity of $-H$, $-H(\bar{p})$ attains a minimum value on $P(\Gamma)$ provided $P(\Gamma)$ is not empty (which since Σ is satisfiable it is not). By convexity of $P(\Gamma)$ and $-H(\bar{p})$, there is a single point that obtains a minimum value. Thus, there is a unique minimal value of the negative entropy, and hence a single distribution with maximum entropy. □

Given a set of $|\bar{v}|$ parameters, $\bar{\alpha}$, let P be the function that maps $\bar{\alpha}$ to a probability distributions p_α over $\mathsf{I}(\Gamma)$ defined by

$$p_\alpha(I) = \frac{1}{Z} \prod_{i=1,m} \alpha_i^{|v_i(I)|} \text{ where } Z = \sum_{J \in \mathsf{I}(\Gamma)} \prod_{i=1,m} \alpha_i^{|v_i(J)|}$$

We now give a sufficient condition for P to be injective. We say that a set of views $\bar{v}$ where $|\bar{v}| = m$ is *affinely dependent* over $\mathsf{I}(\Gamma)$ if there exist real numbers $\bar{c}$ and a value d such that (1) c_i are not all zero and (2) the following holds:

$$\forall I \in \mathsf{I}(\Gamma). \quad \sum_{j=1,m} |v_j(I)|c_j = d$$

If no such $(\bar{c}, d)$ exists, we say that the views are *affinely independent*.

Proposition A.4. *Fix a set* $\mathsf{I}(\Gamma)$. *If* $\bar{v}$ *is affinely independent over* $\mathsf{I}(\Gamma)$ *then, P mapping* α *to* p_α *is injective.*

Proof. Suppose not, then there exists $\bar{\alpha}, \bar{\beta}$ such that $P(\bar{\alpha}) = P(\bar{\beta})$. This implies that for each I, $\log p_\alpha(I) - \log p_\beta(I) = 0$ so that:

$$\log(Z) - \log(Z') = \sum_{j=1,m} |v_j(I)|(\log \alpha_j - \log \beta_j)$$

But then, define $c_j = \log \alpha_j - \log \beta_j$ and $d = \log(Z) - \log(Z')$, then $(\bar{c}, d)$ is a tuple of constants violating the affine independence condition, a contradiction. □

Now we are ready to show:

Theorem A.5. *If* $\Sigma = (\Gamma, \bar{v}, \bar{d})$ *and* $\bar{v}$ *is affinely independent over* $I(\Gamma)$ *and* Σ *is satisfiable then there is a unique solution* $\bar{\alpha}$ *that maximizes entropy.*

Proof. Suppose not, then there are two solutions and both are of the form $P(\bar{\alpha})$ and $P(\bar{\beta})$, but this means that $P(\bar{\alpha}) = P(\bar{\beta})$ by Prop A.3. On the other hand, since $\bar{v}$ is affinely independent (by assumption) we have that P is injective (Prop A.4), and so $\bar{\alpha} = \bar{\beta}$, a contradiction. □

Remark A.1. *The reverse direction of Prop. A.4 holds.*

Chains, Hypergraphs, and Relations are Affinely Independent

Proposition A.6. *A set of vectors is* $\{\boldsymbol{x}^{(i)}\}_{i=1,\ldots,m}$ *is affinely independent over* $\mathbb{R}^N$ *if and only if* $\{\boldsymbol{y}^{(j)}\}_{j=1,\ldots,m}$ *where* $y^{(j)} = (x^{(j)}, 1)$ *is linearly independent over* $\mathbb{R}^{N+1}$.

Fix a tuple of views $\bar{v}$. Denote by $\tau_{\bar{v}} : I \to \mathbb{N}^{m+1}$ as $\tau(I) = \bar{t}$ where $t_i = |v_i(I)|$ for $i = 1, m$ and $\tau_{m+1} = 1$. We denote the unit vector in direction i as $e^{(i)}$.

Proposition A.7. *A chain program* Σ *of size* $m \geq 2$ *is affinely independent for domain sizes* $N \geq 1$.

Proof. Let $I_k = \{R_1(\bar{a}), \ldots, R_i(\bar{a})\}$ so that $\tau(I_k) = \boldsymbol{x}^{(k)}$ where $\boldsymbol{x}^{(k)}_j = 1$ if $j = \{1, \ldots, k\} \cup \{m+1\}$ and $\boldsymbol{x}^{(k)}_j = 0$ otherwise. The set $\{x^{(k)}\}_{k=0,m}$ is a set of $m+1$ linearly independent vectors. □

Proposition A.8. *A hypergraph program of size* $m - 1$ *where* $m \geq 2$ *is affinely independent for for any* $\mathsf{I}(\Gamma)$ *where the domain size is* $N \geq 1$.

Proof. Let $I_i = \{R_i(a)\}$ then $\tau(I_i) = e^{(i)} + e^{(m+2)}$ and $I_{m+1} = \{R_i(a), S(\bar{a})\}$ then $\tau(I_i) = \mathbf{1}$ which is linearly independent. Moreover, $\tau(\emptyset) = e^{(m+1)}$. It is straightforward that this is a linearly independent set. □

Proposition A.9. *A relational program of size* $m - 1$ *where* $m \geq 2$ *is affinely independent over domains of size* $N \geq 2$.

Proof. The vectors are $x^{(i)} = \mathbf{1} + e^{(i)} + e^{(m+1)}$ for $i = 1, m-1$ (a world with two tuples that differ on one attribute) and $x^{(m)} = \mathbf{1}$ (a world with one tuple) and $x^{(m+1)} = e^{(m+1)}$ (the empty world). □

A.4 Calculations for Example 2.11

Recall the example: Continuing with Σ_{C2}, the partition function on a domain of size N is then:

$$T^{\Sigma_{C2}}(\bar{\alpha}) = (1 + \alpha_2(1 + \alpha_1)^N)^N$$

Given $\bar{d} = (d_1, d_2)$, we observe that by setting $\bar{\alpha}$ as follows is a solution to Σ_{C2}: set $\alpha_1 = \frac{d_1}{d_2 N - d_1}$ and $\alpha_2 = \frac{d_2}{N - d_2}(1 + \alpha_1)^{-N}$.

Now, we observe $z = \frac{x}{1+x} \implies \frac{z}{1-z} = x$ so that:

$$\mathbf{E}[A_2] = \frac{T}{\alpha_2}\frac{\partial}{\partial \alpha_2} T^{\Sigma_{C2}} = N\frac{\alpha_2(1+\alpha_1)^N}{1 + \alpha_2(1+\alpha_1)^N} = d_2$$

$$\mathbf{E}[A_1] = \mathbf{E}[A_2]N\frac{\alpha_1}{1+\alpha_1} = d_1$$

A.5 Calculations for Example 2.15

We have $\beta_1 = \frac{d_1}{cN - d_1}$ and $\beta_2 = \frac{c}{N-c}(1+\beta_1)^{-N}$ which implies that $\alpha_1 = \beta_1$ and $\alpha_2 = \frac{c}{(N-c)(1+\beta_1)^N + c}$.

Now, we solve the following equation for large N:

$$c = N\alpha_2 + (1 - \alpha_2)d_2$$

Now,

$$\lim_{N\to\infty} N\alpha_2 = \lim_{N\to\infty} c(1+\beta_1)^{-N}\frac{1}{1 + N^{-1}c(1 - (1+\beta)^{-N})} \to ce^{-d_1/c}$$

Thus, we are left with:

$$c = ce^{-d_1/c} + d_2$$

Let $v = 1/c$ which leaves $e^{-d_1 v} = 1 - d_2 v$ now we apply the substitution $t = d_1 v + \frac{d_1}{d_2}$ so that $v = d_1^{-1}(t - \frac{d_1}{d_2})$ and

$$\begin{aligned} e^{-(t - \frac{d_1}{d_2})} &= \frac{d_2}{d_1}t \\ te^t &= \frac{d_1}{d_2}e^{-\frac{d_1}{d_2}} \\ t &= \mathsf{W}\left(\frac{d_1}{d_2}e^{-\frac{d_1}{d_2}}\right) \\ v &= \frac{\mathsf{W}\left(\frac{d_1}{d_2}e^{-\frac{d_1}{d_2}}\right)}{d_1} - \frac{1}{d_2} \end{aligned}$$

Notice W is a function for positive reals, and $\mathsf{W}(xe^{-x}) = x$ occurs only at $x = 0$, thus $v > 0$ for all $d_1, d_2 > 0$. This implies that $1/v = c$ is a well-defined.

A.6 Extension: Bucketization

An arithmetic predicate, or range predicate, has the form $x \ \mathsf{op} \ c$, where $\mathsf{op} \in \{<, \leq, >, \geq\}$ and c is a constant; we denote by $\mathsf{P}^{\leq}$ the set of project queries with range predicates. We introduce range predicates like $x < c$, both in the constraints and in the statistical assertions. To extend the asymptotic analysis, we assume that all constants are expressed as fractions of the domain size N, e.g., in Ex. A.10 we have $v_1(x, y) :- R(x, y), x < 0.25N$.

Example A.10 Overlapping Ranges Consider two views[8]:

$$v_1(x, y) :- R(x, y), x < .60N \text{ and } v_2(x, y) :- R(x, y), .25N \leq x$$

and the statistical program $\#v_1 = d_1, \#v_2 = d_2$. Assuming $N = 100$, the views partition the domain into three buckets, $D_1 = [1, 24]$,

[8]We represent range predicates as fractions of N so we can allow N to go to infinity.

$D_2 = [25, 59]$, $D_3 = [60, 100]$, of sizes N_1, N_2, N_3. Here we want to say that we observe d_1 tuples in $D_1 \cup D_2$ and d_2 tuples in $D_2 \cup D_3$. The MaxEnt model gives us a precise distribution that represents only these observations and nothing more. The partition function is $(1 + x_1)^{N_1}(1 + x_1x_2)^{N_2}(1 + x_2)^{N_3}$, and the MaxEnt distribution has the form $\mathbf{P}[I] = \omega\alpha_1^{k_1}\alpha_2^{k_2}$, where $k_1 = |I \cap (D_1 \cup D_2)|$ and $k_2 = |I \cap (D_2 \cup D_3)|$.

Suppose we assert the number of tuples in each bucket, say $d_1 = 550$, $d_2 = 126$, $d_3 = 772$, then we can compute the MaxEnt distribution by finding the right parameters $\alpha_1, \alpha_2, \alpha_3$; one can check that these values are $\alpha_i = d_i/(N_iN - d_i)$, for $i = 1, 3$. Note that the statistics Σ resemble superficially a histogram with three buckets D_1, D_2, D_3: both the histogram and Σ make statements about the number of tuples in the three buckets. But histograms do not define a probability distribution, and therefore questions like "what is the estimated size of the query $q(x, z) :- R(x, y), R(z, y)$?" has no meaning over histograms. Instead, it has a well defined meaning for the MaxEnt distribution associated to Σ.

Let $\bar{R} = R_1, \ldots, R_m$ be a relational schema, and consider a statistical program Σ, Γ with range queries, over the schema $\bar{R}$. We translate it into a *bucketized* statistical program Σ^0, Γ^0, over a new schema $\bar{R}^0$, as follows. First, use all the constants that occur in the constraints or in the statistical assertions to partition the domain into b buckets, $D = D_1 \cup D_2 \cup \ldots \cup D_b$. Then define as follows:

- For each relation name R_j of arity a define b^a new relation symbols, $R_j^{i_1 \cdots i_a} = R_j^{\bar{i}}$, where $i_1, \ldots, i_a \in [b]$; then $\bar{R}^0$ is the schema consisting of all relation names $R_j^{i_1 \cdots i_a}$.
- For each conjunctive query q with range predicates, denote $\texttt{buckets}(q) = \{q^{\bar{i}} \mid \bar{i} \in [b]^{|Vars(q)|}\}$ the set of queries obtained by associating each variable in q to a unique bucket, and annotating the relations accordingly. Each query in $\texttt{buckets}(q)$ is a conjunctive query over the schema $\bar{R}^0$, without range predicates, and q is logically equivalent to their union.
- Let $BV = \bigcup\{\texttt{buckets}(v) \mid (v, d) \in \Sigma\}$ (we include in BV queries up to logical equivalence), and let c_u denote a constant for each $u \in BV$, s.t. for each statistical assertion $\#v = d$ in Σ the following holds

$$\sum_{u \in \texttt{buckets}(v)} c_u = d \tag{8}$$

  Denote Σ^0 the set of statistical assertions $\#u = c_u$, $u \in BV$.
- For each inclusion constraint $w \Rightarrow R$ in Γ, create $b^{|Vars(w)|}$ new inclusion constraints, of the form $w^{\bar{j}} \Rightarrow R^{\bar{i}}$; call Γ^0 the set of new inclusion constraints.

Then the following holds:

Proposition A.11. *Let Σ^0, Γ^0 be the bucketized program for Σ, Γ. Let $\bar{\beta} = (\beta_k)$ be the* MaxEnt *model of the bucketized program. Consider some parameters $\bar{\alpha} = (\alpha_j)$. Suppose that for every statistical assertion $\#v_j = d_j$ in Σ condition (8) holds, and the following condition holds for every query $u_k \in BV$:*

$$\beta_k = \prod_{j: u_k \in \texttt{buckets}(v_j)} \alpha_j \tag{9}$$

Then $\bar{\alpha}$ is a solution to the MaxEnt *model for Σ, Γ.*

This gives us a general procedure for solving the MaxEnt model for programs with range predicates: introduce new unknowns $c_j^{\bar{i}}$ and add Equations (8) and (9), then solve the MaxEnt model for the bucketized program under these new constraints.

Example A.12 Recall Example A.10. we are given two statistics $\#\sigma_{A \le 0.60N}(R) = d_1$, and $\#\sigma_{A \ge 0.25N}(R) = d_2$. The domain D is partitioned into three domains, $D_1 = [1, 0.25N)$, $D_2 = [0.25N, 0.60N)$, and $D_3 = [0.60N, N]$, and we denote N_1, N_2, N_3 their sizes. The bucketization procedure is this. Define a new schema R^1, R^2, R^3, with the statistics $\#R^1 = c^1$, $\#R^2 = c^2$, $\#R^3 = c^3$, then solve it, subject to the Equations (9):

$$\begin{aligned} \beta_1 &= \alpha_1 \\ \beta_2 &= \alpha_1\alpha_2 \\ \beta_3 &= \alpha_2 \end{aligned}$$

We can solve for R^1, R^2, R^3, since each R^i is given by a binomial distribution with tuple probability $\beta_i/(1 + \beta_i) = c^i/N_i$. Now use Equations (8), $c^1 + c^2 = d_1$ and $c^2 + c^3 = d_2$ to obtain:

$$\begin{aligned} N_1\frac{\alpha_1}{1 + \alpha_1} + N_2\frac{\alpha_1\alpha_2}{1 + \alpha_1\alpha_2} &= d_1 \\ N_3\frac{\alpha_2}{1 + \alpha_2} + N_2\frac{\alpha_1\alpha_2}{1 + \alpha_1\alpha_2} &= d_2 \end{aligned}$$

Solving this gives us the MaxEnt model. Consistent histograms [30] had a similar goal of using MaxEnt to capture statistics on overlapping intervals, but use a different, simpler probabilistic model based on frequencies.