Polynomial Approximation Algorithms for Belief Matrix Maintenance in Identity Management

Hamsa Balakrishnan, Inseok Hwang, Claire J. Tomlin
Dept. of Aeronautics and Astronautics, Stanford University, CA 94305
hamsa,ishwang,tomlin@stanford.edu

***Abstract*— Updating probabilistic belief matrices as new observations arrive, in the presence of noise, is a critical part of many algorithms for target tracking in sensor networks. These updates have to be carried out while preserving sum constraints, arising for example, from probabilities. This paper addresses the problem of updating belief matrices to satisfy sum constraints using scaling algorithms. We show that the convergence behavior of the Sinkhorn scaling process, previously used for scaling belief matrices, can vary dramatically depending on whether the prior unscaled matrix is *exactly scalable* or only *almost scalable*. We give an efficient polynomial-time algorithm based on the maximum-flow algorithm that determines whether a given matrix is exactly scalable, thus determining the convergence properties of the Sinkhorn scaling process. We prove that the Sinkhorn scaling process *always* provides a solution to the problem of minimizing the Kullback-Leibler distance of the physically feasible scaled matrix from the prior constraint-violating matrix, even when the matrices are not exactly scalable. We pose the problem as a linearly constrained convex optimization problem, and solve it using an interior-point method. We prove that even in cases when the matrices are not exactly scalable, the problem can be solved to $\epsilon-$optimality in strongly polynomial time, improving the best known bound for the problem of scaling arbitrary nonnegative rectangular matrices to prescribed row and column sums.**

I. Introduction

Identity management refers to the probabilistic management of the identities of multiple interacting objects. This has become an important problem in control with the advent of large scale networks of systems, such as sensor networks. Our motivation for this work stems from distributed identity management algorithms in air traffic control and sensor networks [12].

The identity or belief matrix was proposed in [18] as a possible method of integrating information available in the system with external information which might be available sporadically. The belief matrix is a matrix B, in which elements b_{ij} represent the probability of object j having identity i. Updating belief matrices as new information is obtained is a crucial part of many algorithms in identity management. These require methods for constraining matrices to prescribed row and column sums. While the belief matrix of the entire system is doubly-stochastic (*i.e.*, the row sums and column sums are 1), in distributed identity management, in which a particular sensor might only detect a subset of the objects in the system, the belief matrix might be constrained to some prespecified (but not doubly-stochastic) row and column sums. This paper addresses the problem of updating belief matrices by scaling in the face of uncertainty in the system and the observations.

This work was supported by ONR MURI Agreement No. N00014-02-1-0720. H. Balakrishnan is supported by a Stanford Graduate Fellowship.

For example, consider the case of the belief matrix for a system with three objects (labelled 1, 2 and 3). Suppose that, at some instant, we are unsure about their identities (tagged X, Y and Z) completely, and our belief matrix is a 3×3 matrix with every element equal to 1/3. Let us suppose the we receive additional information that object 3 is definitely Z. Then our prior, but constraint violating matrix,

$$\begin{bmatrix} \frac{1}{3} & \frac{1}{3} & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 \\ \frac{1}{3} & \frac{1}{3} & 1 \end{bmatrix} \text{ would logically scale to } \begin{bmatrix} .5 & .5 & 0 \\ .5 & .5 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Although the solution is simple in this case, it is not clear how one would scale an arbitrary rectangular matrix to prescribed row and column sums. A natural way is to simply normalize alternately the rows and columns until the constraints are met. This method of scaling by repeated normalization is called the Sinkhorn scaling process [20]. However, it is not obvious that such a process would always converge; and if it does, that it would converge in a reasonable amount of time. It is also not clear what the quality of the resulting solution is, whether the process always maintains the quality of its solution, or whether there might be faster methods of arriving at the same solution. These are issues that we will address in this paper.

Sinkhorn iterations were first proposed as a method of scaling matrices to make them doubly-stochastic. This method was shown to converge for different classes of matrices, in [21], [20] and [17]. Its properties were analyzed, for the special case of matrices known as *exactly scalable* matrices, in [3], [17] and [8]. This technique was analyzed further and applied to the problem of identity management for Air Traffic Control in [10], [11].

The convergence behavior of the Sinkhorn scaling process for a nonnegative matrix depends greatly on the sparsity structure of the matrix, and can fall into one of two regimes. Most studies, such as those mentioned above, have been restricted to only one of the two regimes, namely, the class of exactly scalable matrices. Belief matrices in distributed sensor networks are sparse matrices, since most interactions between objects are local. In addition, the type of local information that is most desirable from a practical

point of view is identity-type information, *i.e.*, information that determines the identity of one of the objects with certainty. This is also the type of local information that is most likely to make the prior matrix not exactly scalable. In this paper, we consider the general problem of scaling arbitrary rectangular nonnegative matrices to prespecified row and column sums.

The main contributions of this paper are as follows: We prove that the Sinkhorn scaling process *always* converges to the sum-constrained matrix that minimizes the Kullback-Leibler distance from the unscaled prior matrix, even when the matrices are not exactly scalable (Section II). This property of the solution justifies the use of Sinkhorn scaling in belief matrix updates. Because the convergence behavior of the Sinkhorn scaling process can vary widely depending on whether the matrix is exactly scalable or not, we give an efficient polynomial time algorithm that determines whether exact scalability is achievable, for an arbitrary nonnegative rectangular matrix with prescribed row and column sums (Section III). The key contribution of this paper is to show that the Sinkhorn scaling process may be posed as a linearly constrained convex optimization problem. We show that, even when the matrix is not exactly scalable, an interior-point method (Section IV) is a strongly polynomial approximation algorithm which attains $\epsilon-$optimality with complexity $\mathcal{O}(n^6 \log(n/\epsilon))$ for an $n \times n$ matrix (Section V).

Our approach to the problem is different from the only other strongly polynomial approximation scheme for matrix balancing [13], which proposes a modified Sinkhorn algorithm; in Section V we also compare the complexity of the two schemes and show that for the class of square matrices, the algorithm based on the barrier method has lower complexity. In Section VI, we present some examples.

II. Sinkhorn Scaling

The Sinkhorn scaling procedure was proposed in [19] as a method for scaling positive matrices to doubly stochastic matrices. Since then, there have been several extensions to treat the case of nonnegative matrices [21], to scaling positive rectangular matrices to prespecified row and column sums [20], and to scaling nonnegative rectangular matrices to prespecified row and column sums [17].

Unless otherwise specified, throughout this paper, the prior sum-constraint violating matrix is denoted by A and the sum-constrained belief matrix is denoted by B. The prespecified row and column sums to be achieved are denoted by r and c respectively. Since the sum of all the elements in the matrix is both the sum of the row sums and the sum of the column sums, $\sum_i r_i = \sum_j c_j$.

We first formalize a few definitions. Let A be an $m \times n$ matrix, and $r \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$ be the prescribed row and column sums. A zero minor $Z \times L$ of A is a matrix such that for $Z \subseteq \{1, \cdots, m\}$ and $L \subseteq \{1, \cdots, n\}$, the matrix $A_{ZL} = 0$. Then, following the definitions in [13], we define the concepts of *exact scalability* and *almost scalability*.

Definition 1: [13] A nonnegative matrix A is exactly scalable to row and column sums r and c if and only if for every zero minor $Z \times L$ of A,

1)
$$\sum_{i \in Z^c} r_i \geq \sum_{j \in L} c_j \Longleftrightarrow \sum_{i \in Z} r_i \leq \sum_{j \in L^c} c_j \tag{1}$$

2) Equality in (1) holds if and only if the $Z^c \times L^c$ minor is all zero as well.

A matrix A is *almost scalable* to r and c if (1) holds.

Almost scalability is a weaker condition than *exact scalability*. We sometimes refer to matrices that are almost but not exactly scalable as *only almost scalable*.

A. Sinkhorn Scaling Algorithm

Algorithm 1: (Sinkhorn Scaling Algorithm):
Given an nonnegative, $m \times n$ matrix A, and specified vectors of the row sums ($r \in \mathbb{R}^m$) and column sums ($c \in \mathbb{R}^n$), we iterate the following until convergence, with initial value $a_{ij}^{(0)} = a_{ij}$,$k = 1$:

1) Multiply every element $a_{ij}^{(k-1)}$ by the ratio of the desired row sum r_i to the actual row sum $\sum_{j=1}^n a_{ij}^{(k-1)}$

$$a_{ij}^{(k-)} = \frac{r_i a_{ij}^{(k-1)}}{\sum_{j=1}^n a_{ij}^{(k-1)}} \tag{2}$$

2) Multiply every element of the matrix from (2) by the ratio of the desired column sum c_j to the actual column sum $\sum_{i=1}^m a_{ij}^{(k-)}$

$$a_{ij}^{(k)} = \frac{c_j a_{ij}^{(k-)}}{\sum_{i=1}^m a_{ij}^{(k-)}} \tag{3}$$

It can be shown that under the condition that the matrix A is almost scalable, the Sinkhorn scaling process will converge to a unique matrix B that satisfies the row and column sum constraints. The following theorem is a unified statement of the convergence of the Sinkhorn scaling process, from various previous results in literature.

Theorem 1: ([21], [20], [17], [13]): Consider $A \in \mathbb{R}^{m \times n}$, a nonnegative matrix, and desired row sums $r \in \mathbb{R}^m$ and column sums $c \in \mathbb{R}^n$. Then there exists a unique matrix $B \in \mathbb{R}^{m \times n}$ which satisfies these prescribed row and column sums, where $B = D_1 A D_2$ for $D_1 \in \mathbb{R}^{m \times m}$ and $D_2 \in \mathbb{R}^{n \times n}$, D_1 and D_2 both diagonal, positive definite matrices, *if and only if* A is exactly scalable. Furthermore, if the above is true, the Sinkhorn scaling of A will converge to such a matrix B. If A is only almost scalable but not exactly scalable, the Sinkhorn scaling would converge to a unique limit of the form $\lim_{k \to \infty} D_1^{(k)} A D_2^{(k)}$ which satisfies the row and column constraints. However, the individual matrix sequences, $D_1^{(k)}$ and $D_1^{(k)}$ would not converge. ∎

B. Exact scalability vs. Almost scalability

We briefly address the practical implications of exact vs. almost scalability to the Sinkhorn scaling process. It can be shown that while for exactly scalable matrices, $b_{ij} = 0 \Leftrightarrow a_{ij} = 0$, for almost scalable matrices it is only true that

$a_{ij} = 0 \Rightarrow b_{ij} = 0$. This implies that a matrix is almost but not exactly scalable, if and only if there exists at least one element $a_{ij} > 0$ which has to be scaled to zero ($b_{ij} = 0$). Since the Sinkhorn scaling process tries to achieve this by multiplying repeatedly by a sequence of positive numbers, this clearly cannot be done in a finite number of steps. In practice, it could take a very long time to reach a desired accuracy ($b_{ij} < \epsilon$). In Section III we formulate an efficient polynomial time algorithm that determines whether a matrix is exactly or only almost scalable, which in turn determines the convergence behavior of the Sinkhorn scaling process.

C. Kullback-Leibler distance as cost

Given a matrix which represents our *a priori* belief (A), but violates physical constraints such as prespecified row and column sums, we would like to compute the sum-constrained (physically feasible) matrix B that represents the closest distribution to the (infeasible) given distribution. In determining a suitable measure for this "distance", we need to bear the following in mind: if the given distribution A satisfies the constraints (row and column sums equal to the prescribed values), then the scaled distribution $B = A$; if there is no *a priori* distribution, no bias is introduced in B; and finally, B uses all the information available from A, but scrupulous care is taken not to make any assumptions not presented by A. Bearing all this in mind, a suitable measure is the Kullback-Leibler measure (also known as the KL-distance or the cross-entropy [5]) given by:

$$I(B : A) = \sum_{j=1}^{n} \sum_{i=1}^{m} b_{ij} \log \frac{b_{ij}}{a_{ij}} \tag{4}$$

This is sometimes also called the directed divergence (since it measures the divergence of distribution B from distribution A), and is denoted by $D(B \parallel A)$.
Our problem therefore reduces to

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{m} \sum_{j=1}^{n} b_{ij} \log \frac{b_{ij}}{a_{ij}} \\ \text{subject to} & \sum_{j=1}^{n} b_{ij} = r_i \ \ \forall\ i = 1, \cdots, m \\ & \sum_{i=1}^{m} b_{ij} = c_j \ \forall\ j = 1, \cdots, n \\ & b_{ij} \geq 0 \ \forall\ i = 1, \cdots, m;\ j = 1, \cdots, n \end{array} \tag{5}$$

where $r \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$ are the prescribed row and column sums, the constraints on the matrix B. We use the following convention throughout this paper: $0 \log 0 = 0$, $0 \log \frac{0}{0} = 0$, and $a \log \frac{a}{0} = \infty$, if $a > 0$.

D. Sinkhorn scaling and the Kullback-Leibler distance

In this section, we prove that the Sinkhorn scaling process always minimizes the KL-distance, irrespective of whether the matrix is exactly scalable or only almost scalable. Consider problem (5). We compute the Lagrangian dual of this problem. The Lagrangian is given by

$$\begin{aligned} L(B, \lambda, \mu) &= \sum_{i=1}^{m} \sum_{j=1}^{n} b_{ij} \log \frac{b_{ij}}{a_{ij}} + \sum_i \lambda_i (r_i - \sum_j b_{ij}) \\ &\quad + \sum_j \mu_j (c_j - \sum_i b_{ij}) \end{aligned} \tag{6}$$

where $\lambda_i,\ \mu_j \in \mathbb{R}$ are the Lagrange multipliers. The Lagrangian dual of this problem is

$$\begin{aligned} g(\lambda, \mu) &= \inf_B L(B, \lambda, \mu) \\ &= \sum_{i=1}^{m} \lambda_i r_i + \sum_{j=1}^{n} \mu_j c_j - \sum_{i=1}^{m} \sum_{j=1}^{n} \frac{1}{e} e^{\lambda_i} a_{ij} e^{\mu_j} \end{aligned} \tag{7}$$

Consider the derivatives of L with respect to b_{ij}

$$\frac{\partial L}{\partial b_{ij}} = \log \frac{b_{ij}}{a_{ij}} + 1 - \lambda_i - \mu_j \tag{8}$$

Setting the derivatives to zero, for optimality, we get

$$\arg \inf_B L(B, \lambda, \mu) = b_{ij} = \frac{1}{e} e^{\lambda_i} a_{ij} e^{\mu_j} \tag{9}$$

We know that for an exactly scalable matrix, the Sinkhorn process converges to a solution $B = D_1 A D_2$, or in other words, $b_{ij} = d_{1_i} a_{ij} d_{2_j}$ where $D_1 = \text{diag}(d_{1_1}, \cdots, d_{1_m})$ and $D_2 = \text{diag}(d_{2_1}, \cdots, d_{2_n})$. Therefore,

$$b_{ij} = d_{1_i} a_{ij} d_{2_j} \text{where } d_{1_i} = e^{\lambda_i - 1},\ d_{2_j} = e^{\mu_j} \tag{10}$$

satisfies the condition (9) for optimality. We also notice that B satisfies the nonnegativity constraint. The second derivative of L is

$$\frac{\partial^2 L}{\partial b_{ij}^2} = \frac{1}{b_{ij}} > 0 \tag{11}$$

which implies that L is indeed minimized. B is therefore a scaled matrix with row sums given by the vector r and column sums by the vector c, which can be expressed as $D_1 A D_2$, where D_1 and D_2 are diagonal, and A is exactly scalable. Thus, from Theorem 1, the Sinkhorn iterations of A will converge to B.
Suppose A is almost scalable, but not exactly scalable. Then, as before, the Lagrangian is given by (6) and the dual by (7). The dual problem is

$$\text{minimize} \sum_{i=1}^{m} \lambda_i + \sum_{j=1}^{n} \mu_j - \frac{1}{e} \sum_{i=1}^{m} \sum_{j=1}^{n} e^{\lambda_i} a_{ij} e^{\mu_j} \tag{12}$$

Taking derivatives, we get

$$\frac{\partial g}{\partial \lambda_i} = r_i - \frac{1}{e} \sum_{j=1}^{n} e^{\lambda_i} a_{ij} e^{\mu_j} \tag{13}$$

$$\frac{\partial g}{\partial \mu_j} = c_j - \frac{1}{e} \sum_{i=1}^{m} e^{\lambda_i} a_{ij} e^{\mu_j} \tag{14}$$

Therefore, for optimality, we require

$$e^{\lambda_i} \sum_{j=1}^{n} a_{ij} e^{\mu_j} = e r_i \ \text{ and } \ e^{\mu_j} \sum_{i=1}^{m} a_{ij} e^{\lambda_i} = e c_j \tag{15}$$

Since A is almost scalable, we know from Theorem 1 that the Sinkhorn iterations converge to a solution of the form $\lim_{k \to \infty} D_1^{(k)} A D_2^{(k)}$, which satisfies the row and column sum constraints. Let us therefore consider the limit of the Sinkhorn iterations,

$$b_{ij} = \lim_{k \to \infty} d_{1_i}^{(k)} a_{ij} d_{2_j}^{(k)}$$

Using this in (9), we find that

$$b_{ij} = \frac{1}{e} e^{\lambda_i} a_{ij} e^{\mu_j} = \lim_{k\to\infty} d_{1_i}^{(k)} a_{ij} d_{2_j}^{(k)} \tag{16}$$

satisfies the optimality conditions, (15). Therefore, the limit of the sequence of matrices generated by the Sinkhorn process minimizes the KL-distance from the *a priori* distribution.

From the above, we arrive at the following theorem:

Theorem 2: Given $A \in \mathbb{R}^{m\times n}$, the optimal solution to (5), $B \in \mathbb{R}^{m\times n}$, is *always* the solution to the Sinkhorn iteration process.

Proof: Theorem 1 states if the matrix A is at least almost scalable, then the Sinkhorn process will converge; the form of the solution is either $B = D_1 A D_2$ or $B = \lim_{k\to\infty} D_1^{(k)} A D_2^{(k)}$, depending on whether the matrix is exactly or only almost scalable. However, we have shown that in either case ((10) for exact scalability and (16) for only almost scalability) the Sinkhorn scaling process converges to the minimum KL-distance matrix that satisfies the row/column constraints. ∎

This shows that from the information-theoretic perspective, the Sinkhorn scaling process gives us the best solution to the problem of incorporating local information into belief matrices.

E. Sinkhorn scaling and KL distance: some intuition

That the Sinkhorn iterations minimize the Kullback-Leibler distance from the *a priori* distribution agrees with intuition. Let us consider the argument:

The logarithm is a concave function, and the function $f(t) = t\log t$ is strictly convex. We can use this property of the logarithm to prove the *log sum inequality*, as in [5]. For the sake of brevity, we only reproduce the relevant theorem here.

Theorem 3: ([5], Log sum inequality): For nonnegative numbers, $a_1, a_2, \cdots, a_n$ and $b_1, b_2, \cdots, b_n$,

$$\sum_{i=1}^{n} b_i \log \frac{b_i}{a_i} \leq \left(\sum_{i=1}^{n} b_i\right) \log \frac{\sum_{i=1}^{n} b_i}{\sum_{i=1}^{n} a_i} \tag{17}$$

with equality if and only if $\frac{b_i}{a_i}$ is a constant. ∎

In the case of an $m \times n$ matrix, we can treat every row (or column) as a set of nonnegative numbers. The log sum inequality implies that for every row, the set of possible new rows that minimize the KL-distance are the ones in which the elements of the new row are obtained by scaling all the elements of the old row by the same amount. But this is exactly what the Sinkhorn iteration does at every iteration - it scales the entire row by the same amount, and the scaling factor is chosen in a way that satisfies the row sum constraint. It then repeats this for the column distributions. As long as this process of scaling rows and columns alternately converges (as it does, by Theorem 1), the matrix it converges to will minimize the KL-distance.

F. Complexity and convergence of the Sinkhorn scaling algorithm

The Sinkhorn iterations are a natural way of scaling a matrix to achieve prescribed row and column sums. The complexity of each iteration is very small, and for an $m \times n$ matrix, simply involves dividing mn numbers by their row sums or column sums. While Sinkhorn and others ([20], [21], [14], [17]) proved that the iterative procedure converges for appropriate matrices, they did not study the rate of convergence. Franklin and Lorenz [7] showed that each iteration of Sinkhorn scaling for an exactly scalable matrix is a contraction map in the Hilbert projective metric, and they concluded that the number of iterations is bounded by $\mathcal{O}\big(L(A)\cdot 1/\epsilon\big)$, where $L(A)$ is the binary input length (the log of the ratio of the largest to smallest non-zero elements of A) and ϵ is the desired accuracy in some metric of interest. Thus the Sinkhorn iteration process is an approximation scheme, but is not polynomial in $\log(1/\epsilon)$, even for positive, exactly scalable matrices. For an only almost scalable matrix, there are no known bounds on the rate of convergence of the Sinkhorn process.

III. Feasibility of the problem

Let us consider the optimization problem (5). Then, for the Sinkhorn iterations to converge, at the very least, the problem must be feasible. We first note that the feasibility check can be carried out in polynomial time by identifying an equivalent problem [13]. The feasibility test is equivalent to a check for almost scalability. We also formulate an approximation that checks for the infeasibility of exact scalability.

A. Feasibility of scaling algorithm

The feasibility of (5) is equivalent to the maximum-flow problem on a graph with $m+n+2$ nodes. Consider the graph in Fig. 1. The flow on source-adjacent and sink-adjacent arcs is denoted by f_i and g_j respectively. The flow on the arc (i,j), $i = 1,\cdots m$, $j = 1,\cdots n$ is denoted by b_{ij}.

Proposition 1: If $\sum_i r_i = \sum_j c_j = K$, there exists a feasible matrix scaling if and only if the maximum source-to-sink flow equals K.

Proof: Suppose the maximum flow equals K. Then, we have a flow in the network that saturates the source- and sink-adjacent arcs, does not violate flow conservation, and does not violate capacity restrictions ($b_{ij} = 0\ \forall\{(i,j)|\ a_{ij} = 0\}$). Therefore,

$$\sum_j b_{ij} = f_i = r_i, \quad \forall\ i \in \{1,\cdots,m\}$$
$$\sum_i b_{ij} = g_j = c_j, \quad \forall\ j \in \{1,\cdots,n\}$$

which is the definition of a feasible point for the optimization problem, whose elements are given by b_{ij}.

Suppose the value of the maximum flow is less than K. Then, the value of every feasible flow in the network is

also less than K. Given such a flow, there exists at least one unsaturated source-adjacent arc, *i.e.*,

$$\exists\ i \in \{1, \cdots, m\} \text{ such that } \sum_j b_{ij} < r_i, \qquad (18)$$

which violates the row sum constraint. Therefore, every feasible flow in the network violates at least one of the row sum constraints, which implies that there is no feasible matrix solution to the optimization, and hence no feasible solution to the Sinkhorn scaling process. ∎

The maximum flow problem in bipartite networks can be solved in $\mathcal{O}(pq \log(q^2/p))$, where p is the number of non-zero elements in $A = (a_{ij})$, and q is $\min\{n, m\}$ ([2], [9]).

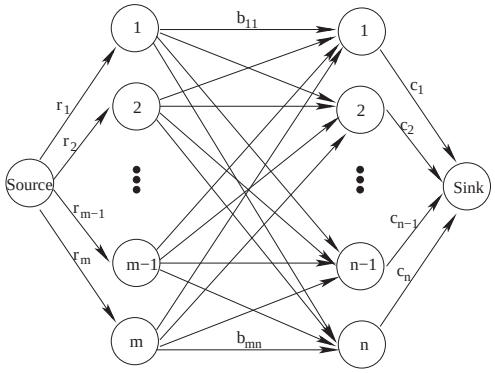

Fig. 1. Equivalence of feasibility problem to maximum-flow problem

B. Infeasibility of an exactly scaled solution

We consider the case in which we might expect the existence of an only almost scaled but not an exactly scaled solution, *i.e.*, a solution B such that $b_{ij} = 0$ even though $a_{ij} \neq 0$. As we might expect, this solution, although feasible, can be theoretically reached by the Sinkhorn scaling process only after an infinite number of iterations. We formulate the infeasibility of an exactly scaled solution as the following equivalent network flow problem.

We are interested in checking whether there is some element b_{ij} such that $a_{ij} \neq 0$ whose value is exactly zero in every feasible matrix scaling (assuming one exists). This is equivalent to asking if there is a feasible scaling such that $b_{ij} > 0\ \forall\{(i,j)|a_{ij} \neq 0\}$. While it is not possible to answer this question exactly, it is possible to check (in polynomial time) if there exists a feasible scaling such that $b_{ij} \geq \epsilon\ \forall\{(i,j)|a_{ij} \neq 0\}$, for arbitrarily small values of ϵ. We work on the same graph as before (Figure 1), but impose a lower bound of ϵ on the flow on arcs $\{(i,j)|a_{ij} \neq 0, i \in \{1, \cdots m\}, j \in \{1, \cdots, n\}\}$. By the same argument as before, there exists a feasible matrix scaling B such that $b_{ij} \geq \epsilon \forall\{(i,j)|a_{ij} \neq 0\}$ if and only if the maximum flow equals K. The problem of finding the maximum flow in a network with arc lower bounds is as hard as solving two maximum flow problems [1] (Section 6.7, *Flows with lower bounds*). Both maximum flow problems are solved on bipartite graphs (one on the original graph, and one on a transformed graph); therefore, the complexity of finding an ϵ-accurate solution to the question of the infeasibility of an exactly scaled solution is also $\mathcal{O}(pq \log(q^2/p))$, where, as before, p is the number of non-zero elements in $A = (a_{ij})$, and q is $\min\{n, m\}$ [2]. We note that for this infeasibility check, which is $\epsilon-$approximate, the run time is independent of ϵ. We also note that it is not possible to identify the exact element in B that needs to be zero – we can only prove that such an element necessarily exists.

IV. Scaling algorithm based on an Interior Point Method

A. Interior-point or barrier method

Let us denote the $mn-$dimensional vector of the elements of the matrix B by x, *i.e.*, $x = [b_{11}, b_{21}, \cdots, b_{m1}, b_{12}, b_{22}, \cdots, b_{mn}]^{\mathrm{T}}$. Similarly, let $y = [a_{11}, a_{21}, \cdots, a_{m1}, a_{12}, a_{22}, \cdots, a_{mn}]^{\mathrm{T}}$. Then we can reformulate (5) as

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{mn} x_i \log \frac{x_i}{y_i} \\ \text{subject to} & -x_i \leq 0,\ i = 1 \cdots mn \\ & Cx = d \end{array} \qquad (19)$$

where $Cx = d$ is the linear equality constraint derived from the row and column sum constraints. We note that the elements of C are zeros and ones.

In the following analysis, we draw heavily from concepts in convex optimization, and refer the reader to [4] for further details. We consider optimization programs of the form

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0,\ i = 1 \cdots M \\ & C_0 x = d_0 \end{array} \qquad (20)$$

where $f_0, \cdots, f_M : \mathbb{R}^n \to \mathbb{R}$ are convex, and $C_0 \in \mathbb{R}^{p \times n}$ is full rank with $p > n$.

The optimization program (19), which is a special case of (20), is a linear inequality constrained problem, with linear equality constraints and a convex cost. The barrier method solves this problem by solving a sequence of equality constrained problems, using Newton's method. It is also called the interior-point method, since all the iterates of x are strictly feasible, *i.e.*, they lie in the relative *interior* of the feasible set.

B. Initial point and sublevel set assumptions

We have already seen how the maximum-flow formulation can be used to compute a feasible point $x^{(0)}$ in polynomial time.

Interior point methods make two assumptions about the problem. First, they assume that the sublevel set

$$\{x|f_0(x) \leq f_0(x^{(0)}), f_i(x) \leq 0, i = 1 \cdots M, C_0 x = d_0\}$$

is bounded. Let us consider this condition for the program (19). For any feasible starting point $x^{(0)}$,

$$\begin{aligned} \{x|f_0(x) \leq f_0(x^{(0)}), x_i \geq 0, i = 1 \cdots mn, Cx = d\} \\ \subseteq \{x|x_i \geq 0, i = 1 \cdots mn, Cx = d\} \end{aligned}$$

But in (19), the linear equality constraint corresponds to constraints on the row and column sums, and therefore includes the constraint $\sum_{i=1}^{mn} x_i = \sum_{k=1}^{m} r_k$. This implies that

$$\begin{aligned} x^{\mathrm{T}}x &= \textstyle\sum_i x_i^2 = (\sum_i x_i)^2 - 2\sum_{i,j} x_i x_j \\ &\leq (\textstyle\sum_i x_i)^2 = (\sum_k r_k)^2 \end{aligned} \tag{21}$$

which is bounded. The second assumption is that every strictly feasible point is in the domain of the objective, which is also satisfied by the nonnegative constraints with the KL-distance as the objective function.

C. The barrier or path following method

We use a logarithmic barrier, defined by

$$\phi(x) = -\sum_{i=1}^{M} \log(-f_i(x)) = -\sum_{i=1}^{mn} \log(x_i) \tag{22}$$

ϕ is a barrier function; it is analytic, convex, its domain is the set of strictly feasible points, and it tends to infinity as the value of x approaches the boundary of **dom** ϕ. We solve the minimization problem

$$\begin{aligned} &\text{minimize} && tf_0(x) + \phi(x) \\ &\text{subject to} && C_0 x = d_0. \end{aligned} \tag{23}$$

The central path is the set of points $x^*(t),\ t > 0$ such that

$$x^*(t) = \arg\min_{C_0 x = d_0} (tf_0(x) + \phi(x)). \tag{24}$$

The barrier method can be written as follows [4]:

Algorithm 2: (Interior-point or Barrier method):
given strictly feasible x, $t := t^{(0)}$, $\mu > 1$, $\epsilon > 0$
repeat

1) *Centering step*: Compute $x^*(t)$ by minimizing $tf_0(x) + \phi(x)$ subject to $C_0 x = d_0$, starting at x.
2) *Update*: $x := x^*(t)$.
3) *Stopping criterion*: If $M/t < \epsilon$, quit.
4) *Increase* t: $t := \mu t$.

Step 1 is known as the *centering step* or *outer iteration*. In each outer iteration, we carry out the centering using Newton's method. We refer to the steps executed during centering as *Newton steps* or *inner iterations*, and the points x^* produced by the centering step are said to lie on the central path.

We give a brief outline Newton's method with equality constraints for the sake of completeness. This is executed during every centering step.

Algorithm 3: (Newton's method):
given starting point $x \in$ **dom**f with $C_0 x = d_0$, tolerance $\epsilon_{nt} > 0$
repeat

1) Compute Newton step Δx_{nt} and decrement $\lambda(x)$ such that

$$\begin{bmatrix} \nabla^2 f(x) & C_0^{\mathrm{T}} \\ C_0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x_{nt} \\ \nu \end{bmatrix} = \begin{bmatrix} -\nabla f(x) \\ 0 \end{bmatrix}$$

$$\lambda(x) = (\nabla f(x)^{\mathrm{T}} \nabla^2 f(x)^{-1} \nabla f(x))^{1/2} \tag{25}$$

where ν is the associated optimal dual variable for the quadratic problem.
2) *Stopping criterion*: quit if $\lambda^2/2 \leq \epsilon_{nt}$.
3) *Line search*: Choose step size τ using backtracking line search. The backtracking line search procedure can be briefly described as follows:
$\tau := 1$; given $0 < \alpha < 0.5$; $0 < \beta < 1$
while $\big[f(x + \tau\delta x_{nt}) > f(x) + \alpha\tau\nabla f(x)^{\mathrm{T}}\delta x_{nt}\big]$
$\tau := \beta\tau$
end
4) *Update*: $x := x + \tau\Delta x_{nt}$.

D. Self-concordant functions

A class of functions for which it is possible to analyze the convergence and complexity of Newton's method, and therefore the barrier method, is the class of self-concordant functions ([4], [16]).

Definition 2: Self-concordant functions [4]: A convex function $f : \mathbb{R} \to \mathbb{R}$ is self-concordant if

$$|f'''(x)| \leq 2f''(x)^{3/2} \tag{26}$$

for all $x \in \mathbf{dom} f$. A function $f : \mathbb{R}^n \to \mathbb{R}$, $n > 1$, is said to be self-concordant if it is self-concordant along every line in its domain, *i.e.*, if the function $\tilde{f}(t) = f(x + tv)$ is a self-concordant function of t for all $x \in \mathbf{dom} f$ and for all v.

Self-concordance is preserved by scaling by a factor greater than one; it is also preserved by addition. Using these properties, it is possible to show that the objective function in the barrier method,

$$t\sum_{i=1}^{mn} x_i \log\frac{x_i}{y_i} - \sum_{i=1}^{mn} \log(x_i) \text{ is self-concordant} \tag{27}$$

for all x and t. For strictly convex self-concordant functions like our objective function, we can obtain bounds on the sub-optimality of a point x in terms of the norm of the gradient of x [4]. Such bounds are unaffected by affine changes in coordinates.

V. Complexity Analysis for the Interior-point Method

In this section, we compute bounds on the complexity of the proposed interior-point method, using a logarithmic barrier and the Newton method for centering.

A. Number of outer iterations for $\epsilon-$optimality

As we have already mentioned, points on the central path are minimizers of $tf_0(x) + \phi(x)$ subject to $C_0 x = d_0$. In our particular instance, the objective function is $\sum_{i=1}^{mn} x_i \log\frac{x_i}{y_i}$, and $\phi(x) = -\sum_{i=1}^{mn} log(x_i)$. Using duality gap arguments, as shown in [4], we can find a bound on the number of centering steps needed. We can show that the minimum number of centering steps needed (l) to achieve $\epsilon-$optimality, satisfies

$$l = \frac{1}{\log\mu}\log\frac{mn}{\epsilon t^{(0)}} = \mathcal{O}\big(\log\Big(\frac{mn}{\epsilon}\Big)\big) \tag{28}$$

B. Number of Newton iterations per centering step

To bound the number of Newton iterations per centering step, we use the fact that the objective function in problem (19) is self-concordant (27). We refer the reader to [4] for details on how the bounds are derived. It is important, however, to note that convexity and self-concordance make it possible to derive bounds on the suboptimality of any feasible point in terms of the Newton decrement. Such bounds are independent of affine coordinate changes. Self-concordant functions form a small class of functions for which complexity analysis is possible for a barrier method; fortunately the KL-distance combined with a logarithmic barrier falls into this class of functions, enabling us to obtain polynomial bounds on the way the complexity scales with the size of the problem. The properties of self-concordant functions allow us to obtain an upper bound on the number of Newton iterations per centering step as:

$$\begin{aligned} N_{nt} &\leq \frac{mn(\mu - 1 - \log\mu)}{\gamma} + \log_2\left(\frac{1}{6}\log_2(\frac{4}{\epsilon_{nt}})\right) \\ &= \mathcal{O}(mn), \end{aligned} \tag{29}$$

since we are interested only in how the complexity scales with problem size. γ is a constant that depends on the line search parameters, α and β.

C. Complexity of Newton iteration

In each Newton iteration, the chief complexity arises from the elimination of (25) to compute the Newton step. In solving

$$\begin{bmatrix} \nabla^2 f(x) & C^{\mathrm{T}} \\ C & 0 \end{bmatrix} \begin{bmatrix} \Delta x_{nt} \\ \nu \end{bmatrix} = \begin{bmatrix} -\nabla f(x) \\ 0 \end{bmatrix}$$
$$\lambda(x) = (\nabla f(x)^{\mathrm{T}} \nabla^2 f(x)^{-1} \nabla f(x))^{1/2} \tag{30}$$

we use block elimination, and the fact the $\nabla^2 f(x)$ is diagonal for $f = \sum_{i=1}^{mn} x_i \log\frac{x_i}{y_i} - \sum_{i=1}^{mn}\log x_i$. Since the linear equality constraints are on the row and column sums, C has mn columns and the number of rows is $m + n -$ (number of redundant constraints), so that C is full rank. It is easy to show that the complexity of this block elimination [4] is

$$\mathcal{O}\left(2mn(m+n)^2 + \frac{2}{3}(m+n)^3\right) \tag{31}$$

D. Total complexity of scaling using the barrier method

We do not go into the detail of the various parameters associated with the interior-point method and the Newton iterations, such as μ, $t^{(0)}$, α and β. [4] contains a detailed discussion on the trade-offs involved in the choice of these parameters. It is also possible to carry out a more careful analysis [16] and find better (less conservative) bounds on the complexity. However, we are interested primarily in the complexity with respect to the size of the problem and the (very) conservative bounds we have derived are sufficient for this purpose.

Theorem 4: The complexity of scaling an $m \times n$ matrix to specified row and column sums using the proposed interior-point method with a logarithmic barrier and the KL-distance as the objective function is

$$\mathcal{O}\left(m^2 n^2 (m+n)^2 \log(\frac{mn}{\epsilon})\right) \tag{32}$$

In particular, if $n \geq m$, we can equivalently bound the complexity by $\mathcal{O}\left(n^6 \log(\frac{n}{\epsilon})\right)$.

Proof: The interior-point algorithm sequentially performs outer iterations, and a centering step in every outer iteration, which in turn involves a number of the Newton steps. Therefore an upper bound on the total complexity is obtained by combining (28), (29) and (31). ∎

E. Discussion on the relative computational efficiency of various algorithms for scaling

We have already seen that while the Sinkhorn process is an approximate algorithm, it is not polynomial in $\log(1/\epsilon)$. For reasonably small and exactly scalable matrices, the Sinkhorn process is a very attractive option because of its ease of computation and reasonable computational times. However, for larger or only almost scalable matrices (a property that we have shown can be checked in polynomial time), we need to use more general and efficient polynomial approximation schemes. For the problem of scaling square matrices to prescribed row and column sums, [13] developed an iterative process that is a modification of the Sinkhorn scaling process, and which has complexity $\mathcal{O}\left(n^7 \log(1/\epsilon)\right)$ for an $n \times n$ matrix. This is the first (and to our knowledge only) existing strongly polynomial-time algorithm for general matrix scaling. In this paper, we approach the problem in an optimization framework and develop an algorithm that scales nonnegative rectangular matrices to prescribed row and column sums, with a complexity of $\mathcal{O}\left(n^6 \log(n/\epsilon)\right)$ for square matrices.

VI. Examples

We compare the performance of the Sinkhorn scaling process and the barrier method through a few examples. Let us first consider a very basic example, demonstrative of the kind of scenarios we are likely to encounter during tracking and identity management in a small sensor network. Suppose the system has 4 objects (1,2,3 and 4) which are initially given the identities W, X, Y and Z. During the process of tracking multiple maneuvering objects, when the objects come close to each other, it becomes almost impossible to maintain the distinct identities of the objects. Let us consider the case in which after repeated interaction between the objects, the belief matrix is confused. Suppose, at this instant, one of the sensors notices a physical attribute of Object 4 which distinguishes it as Z for certain. Then, our belief matrix before the observation and the prior (unscaled) distribution after the observation are given by

$$\begin{bmatrix} 0.1 & 0.1 & 0.3 & 0.5 \\ 0.2 & 0.4 & 0.3 & 0.1 \\ 0.4 & 0.2 & 0.1 & 0.3 \\ 0.3 & 0.2 & 0.3 & 0.2 \end{bmatrix} \text{ and } \begin{bmatrix} 0.1 & 0.1 & 0.3 & 0 \\ 0.2 & 0.4 & 0.3 & 0 \\ 0.4 & 0.2 & 0.1 & 0 \\ 0.3 & 0.2 & 0.3 & 1 \end{bmatrix}.$$

The maximum-flow formulation of Section III tells us that the prior matrix is almost but not exactly scalable to a doubly-stochastic matrix. We choose an ϵ of 10^{-8}. A MATLAB implementation of the Sinkhorn scaling process takes 1.718 seconds (and 7105 iterations) to converge to a solution, while an implementation of the interior-point method in AMPL [6] using the MINOS [15] solver for the centering takes 0.0156 seconds to reach the same solution.

Finally, we present a 100 trial Monte Carlo simulation over a range of matrix sizes, for two different cases - Sinkhorn scaling for only almost scalable matrices, and the interior-point method, whose performance is independent of scalability. The random matrices for the Sinkhorn scaling were generated such that the elements of the prior matrix that had to be scaled to zero were no more than 0.1 in magnitude, *i.e.*, they only violated the exact scalability condition weakly. The matrices for the interior-point method were a random combination of exact and only almost scalable matrices. The average computational times are plotted in Figure 2. While the Sinkhorn scaling process would perform very well for exactly scalable matrices, there is a dramatic deterioration in its performance when the prior matrix is only almost scalable, *even* if the elements that need to be scaled to zero are small in magnitude. On the other hand, the interior-point algorithm scales much better and is independent of the scalability of the prior matrix, as long as it is at least almost scalable.

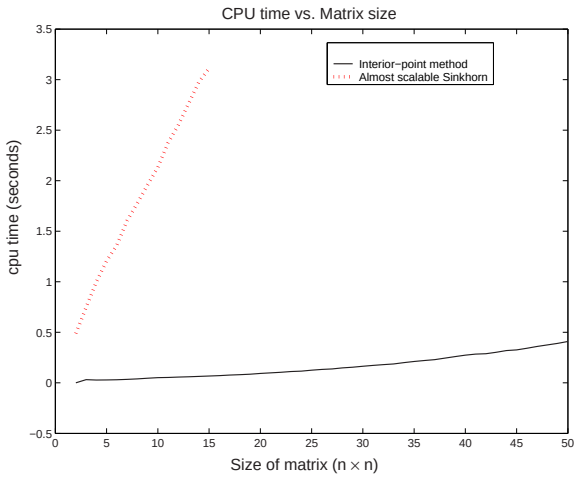

Fig. 2. Computational time comparisons- Sinkhorn and Barrier methods

VII. Conclusions

The main aim of this paper was to develop efficient algorithms for belief matrix maintenance, for the purpose of identity management in large, possibly distributed, systems with multiple objects. We identified the chief problems as being (1) the efficient scaling of large rectangular nonnegative matrices to prescribed row and column sums, and (2) the efficient diagnosis of the behavior of the easy-to-implement Sinkhorn iterations. We began with an analysis of the properties of the solution of the Sinkhorn process for the case when the matrix is only almost scalable, and showed that the process always minimized the Kullback-Leibler distance, even if it was slow to converge. We formulated a maximum-flow with lower bounds algorithm to efficiently predict the behavior of the Sinkhorn process, and to generate a feasible point. We then formulated an equivalent convex optimization problem, and showed that the interior-point method was strongly polynomial in complexity. We demonstrated through simulations that the proposed algorithm is not sensitive to the sparsity structure of the matrix, and performs better than the Sinkhorn algorithm.

References

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows: Theory, Algorithms and Applications*. Prentice-Hall, 1993.

[2] R.K. Ahuja, J. B. Orlin, and C. Stein. Improved algorithms for bipartite network flow. *SIAM Journal on Computing*, 23(5):906–933, 1994.

[3] M. Bacharach. *Biproportional Matrices and Input-Output Change*. Number 16 in University of Cambridge, Department of Applied Economics Monographs. Cambridge University Press, 1970.

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

[5] T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley and Sons, 1991.

[6] R. Fourer, D.M. Gay, and B.W. Kernighan. *AMPL: A Modelling Language for Mathematical Programming*. Brooks/Cole-Thomson Learning, 2003.

[7] J. Franklin and J. Lorenz. On the scaling of multidimensional matrices. *Linear Algebra and its Applications*, 114/115:717–735, 1989.

[8] A. Golan, G. Judge, and S. Robinson. Recovering information from incomplete or partial multisectoral economic data. *Review of Economics and Statistics*, 76(3):541–549, August 1994.

[9] A. V. Goldberg and R. E. Tarjan. A new approach to the maximum-flow problem. *Journal of the Association of Computational Machinery*, 35:921–940, 1988.

[10] I. Hwang, H. Balakrishnan, K. Roy, J. Shin, L. Guibas, and C. Tomlin. Multiple-target Tracking and Identity Management algorithm for Air Traffic Control. In *Proceedings of IEEE Sensors (The Second IEEE International Conference on Sensors)*, Toronto, Canada, October 2003.

[11] I. Hwang, H. Balakrishnan, K. Roy, and C.J. Tomlin. Multiple-target Tracking and Identity Management in clutter with application to aircraft tracking. In *Proceedings of the American Control Conference, 2004*. To appear.

[12] I. Hwang, K. Roy, H. Balakrishnan, and C.J. Tomlin. Distributed multiple-target tracking and identity management in sensor networks. In *IEEE Conference on Decision and Control, 2004*. Submitted.

[13] N. Linial, A. Samorodnitsky, and A. Wigderson. A deterministic strongly polynomial algorithm for matrix scaling and approximate permanents. *Combinatorica*, 20:531–544, 2000.

[14] A.W. Marshall and I. Olkin. Scaling of matrices to achieve specified row and column sums. *Numerische Mathematik*, 12:83–90, 1968.

[15] B.A. Murtagh and M.A. Saunders. Minos 5.5 user's guide. Technical Report SOL 83-20R, Department of Operations Research, Stanford University, 1998.

[16] Y. Nesterov and A. Nemirovskii. *Interior-point Polynomial Methods in Convex Programming*. Society for Industrial and Applied Mathematics, 1994.

[17] U. Rothblum and H. Schneider. Scaling of matrices which have prespecified row and column sums via optimization. *Linear Algebra and its Applications*, 114/115:737–764, 1989.

[18] J. Shin, L.J. Guibas, and F. Zhao. A distributed algorithm for managing multi-target identities in wireless ad-hoc sensor networks. In F. Zhao and L. Guibas, editors, *Information Processing in Sensor Networks*, Lecture Notes in Computer Science 2654, pages 223–238, Palo Alto, CA, April 2003.

[19] R. Sinkhorn. A relationship between arbitrary positive matrices and stochastic matrices. *Annals of Mathematical Statistics*, 35:876–879, 1964.

[20] R. Sinkhorn. Diagonal equivalence to matrices with prescribed row and column sums. *American Mathematical Monthly*, 74:402–405, 1967.

[21] R. Sinkhorn and P. Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967.