Sinkhorn Solves Sudoku

Todd K. Moon, *Senior Member, IEEE*, Jacob H. Gunther, *Member, IEEE*, and Joseph J. Kupin

***Abstract*—The Sudoku puzzle is a discrete constraint satisfaction problem, as is the error correction decoding problem. We propose here an algorithm for solution to the Sinkhorn puzzle based on Sinkhorn balancing. Sinkhorn balancing is an algorithm for projecting a matrix onto the space of doubly stochastic matrices. The Sinkhorn balancing solver is capable of solving all but the most difficult puzzles. A proof of convergence is presented, with some information theoretic connections. A random generalization of the Sudoku puzzle is presented, for which the Sinkhorn-based solver is also very effective.**

***Index Terms*—Belief propagation (BP), constraint satisfaction, low-density parity-check (LDPC) decoding, Sinkhorn, Sudoku.**

I. Introduction

THE Sudoku puzzle is a discrete constraint satisfaction problem, where the constraints address uniqueness of subsets of the puzzle, based on some observed "clues." Error correction decoding is also a discrete constraint satisfaction problem, where the constraints address parity of subsets of the observed values. The low-density parity-check (LDPC) belief propagation (BP) decoding algorithm can be adapted to solve Sudoku, since the puzzle has a Tanner graph representation. On the other hand, a different solver for Sudoku may suggest an alternative decoding algorithm, resulting in an interesting interplay between Sudoku and error correction coding.

A Sudoku puzzle may be solved by logical elimination, as exhaustively described in [1], where nearly a dozen distinct rules are described. By contrast, our method employs a probabilistic representation of the puzzle. The probabilistic representation provides a "suspension of belief" that allows searching of potential solutions.

The connection between decoding and Sudoku has been previously noted. The paper [2] gives an explicit formulation of the BP algorithm for solving Sudoku. BP appears to work only for easier puzzles, with the probable cause being the "loopy" nature of the Tanner graph associated with the puzzle (all cells are in cycles of length four). The presentation [3] also discusses BP. In this paper, we present a solution algorithm based on Sinkhorn balancing, which has both lower computational complexity (per iteration) than BP and does not apparently suffer from cycles in the graph. Sinkhorn balancing [4] is a means of obtaining a unique doubly stochastic matrix from a (nearly) arbitrary matrix. Extensions to produce matrices with arbitrary row and column sums appear in [5]. Sinkhorn balancing (sometimes called Sinkhorn scaling) has been widely studied, and makes its appearance in a variety of applications. (See, for example [6].) The Sinkhorn balancing approach to solution is successful at solving all but the most difficult Sudoku puzzles. Sinkhorn balancing furthermore generalizes well to situations in which clues are presented as random elements in a set.

As there are other methods of solving Sudoku puzzles, the method presented here needs some justification. Our exploration was motivated by a desire to develop decoding algorithms for linear codes having many cycles in their Tanner graphs. While the BP algorithm fares poorly for such codes (and Sudoku puzzles), the Sinkhorn balancing approach appears to be much more robust. This success may in the future lead to insights on how to apply Sinkhorn-like techniques to the decoding problem.

Manuscript received December 13, 2006; revised May 08, 2007. Current version published March 18, 2009.

T. K. Moon and J. H. Gunther are with the Electrical and Computer Engineering Department, Utah State University, Logan, UT 84322 USA (e-mail: Todd.Moon@usu.edu; jake@ece.usu.edu).

J. J. Kupin is with the Center for Communications Research, Princeton, NJ 08544 USA (e-mail: joseph.kupin@verizon.net).

Communicated by G. Seroussi, Associate Editor for Coding Theory.

Digital Object Identifier 10.1109/TIT.2009.2013004

II. Puzzle Description and Representation

A Sudoku puzzle is an $N \times N$ grid of cells partitioned into N smaller blocks of N elements each. The puzzle problem is to fill in the cells so that the digits $1, \ldots, N$ appear uniquely in each row and column of the grid and in each block, starting from some initial set of filled-in cells called "clues." The uniqueness requirement imposes $3N$ constraints on the puzzle. The following is an example of a 9×9 puzzle:

	10	11	12	13	14	15	16	17	18
1				4		9	6	7	
2					7	6	9		
3									3
4						1	7	4	
5	6	4						1	8
6		2	1	6					
7	1								
8			4	3	2				
9		6	2	9		4			

(1)

We denote the contents of cell n by $S_n \in \{1, 2, \ldots, N\}$ for $n = 1, 2, \ldots, N^2$, with cells numbered in row-scan order. The row constraints are indexed by the numbers $1, \ldots, N$ down the side of the puzzle [as indicated in (1)]; the column constraints are indexed by the numbers $N+1, \ldots, 2N$ across the top of the puzzle. [The box constraints are not shown in (1).] Constraint m of the puzzle is satisfied if all N cells associated with it are distinct.

We model the contents of the cells probabilistically. Let $\mathbf{p}_n = [P(S_n = 1) \quad P(S_n = 2) \quad \cdots \quad P(S_n = 9)]$ be the probability row vector associated with cell S_n, with individual elements $p_{n,j}$. Cells that are specified initially—the clue cells—place all their probability mass on the specified value. Thus, for the puzzle in (1)

$$\mathbf{p}_4 = \mathbf{e}_4 \quad \mathbf{p}_6 = \mathbf{e}_9 \quad \mathbf{p}_7 = \mathbf{e}_6, \quad \text{etc.}$$

where $\mathbf{e}_k$ is a vector of length N with a single 1 at position k and 0 in other positions (a singleton probability). For the nonclue cells (initially empty), the probabilities are uniformly distributed over the possible outcomes. For example, $\mathbf{p}_1 = \frac{1}{4}[0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0]$.

III. Sinkhorn Balancing Solution

A doubly stochastic matrix is a matrix with nonnegative elements whose rows each sum to 1 and whose columns each sum to 1. Sinkhorn balancing [4] is a method of obtaining a doubly stochastic matrix starting from a (nearly) arbitrary matrix with nonnegative elements. It operates quite simply: each column in succession is normalized so that its sum is 1, then each row in succession is normalized so that its sum is 1. Then repeat until convergence (according to some convergence criterion). We refer to the column-normalizing step as the vertical step and the row-normalizing step as the horizontal step. (Necessary and sufficient conditions for convergence to a unique limit are presented in [7].)

Sinkhorn Balancing (SB)

Input/Initialize: An $N \times N$ matrix Q; max number of iterations M; a tolerance ϵ. Set $k = 0$. Set $Q^{[0]} = Q$.

Repeat:

For $j = 1 : N$ (Vertical: sum and scale columns)

$\chi_j = \sum_l q_{lj}^{[k]}; \quad q_{ij}^{|} = q_{ij}^{[k]}/\chi_j, i = 1, 2, \ldots, N$

End For j

For $i = 1 : N$ (Horizontal: sum and scale rows)

$\rho_i = \sum_l q_{il}^{||}; \quad q_{ij}^{[k+1]} = q_{ij}^{|}/\rho_i, j = 1, 2, \ldots, N$

End For i

If $\|Q^{[k+1]} - Q^{[k]}\| < \epsilon$, Set $Q = Q^{[k+1]}$, Break

$k \leftarrow k + 1$. If $k > M, Q = Q^{[k]}$ Break

end Repeat.

Return Q

Application of Sinkhorn balancing to Sudoku is straightforward. For each constraint m in the puzzle, form an $N \times N$ matrix called the constraint probability matrix $Q_m^{[0]}$ by stacking the probability vectors for the cells associated with that constraint.

Consider now the nature of a constraint probability matrix Q_m if all the cells associated with constraint m were known. Each probability vector would be singleton, putting its mass in distinct columns of Q_m, so that Q_m would be doubly stochastic.

A necessary (but certainly not sufficient) condition, then, for the probabilities associated with cells in this constraint to represent a solution is for the probability constraint matrix to be doubly stochastic. Given the matrix $Q_m^{[0]}$, we produce an associated matrix $Q_m^{[1]}$ that is doubly stochastic by Sinkhorn balancing. This Sinkhorn balanced matrix is, as we prove below, closer in Kullback–Leibler (KL) distance to the constraint probability matrix for a solved puzzle. Sinkhorn balancing is repeated for each probability constraint in turn, then the process is repeated until all constraints are satisfied, or some maximum number of iterations is reached.

TABLE I
Comparison of SSS and BP Solutions

	A	B	C	D
No. of Puzzles	10	10	25	23
No. SSS solved	10	10	25	23
Av. No. SSS Iterations	35.5	56.9	169.4	177.4
No. BP solved	5	3	3	4
Av. No. BP Iterations	5.8	7	6.3	7.5

Sinkhorn Sudoku Solution (SSS)

Initialization: Set initial probability vectors $\mathbf{p}_1, \ldots, \mathbf{p}_{N^2}$ according to initial clues and uniformly in cells with no clues. Set $k = 0$

Repeat:

For each constraint $m \in \{1, 2, \ldots, 3N\}$:

Form the probability constraint matrix $Q_m^{[k]}$

Sinkhorn balance: $Q_m^{[k+1]} = \mathrm{SB}(Q_m^{[k]})$.

Extract the probabilities $\mathbf{p}_n$ from $Q_m^{[k+1]}$.

End for m

Determine most probable contents S_n from $\mathbf{p}_n$:

$$S_n = \arg\max_j p_{n,j}$$

If all constraints are satisfied, Break with success.

Increment iteration count: $k \leftarrow k + 1$.

If too many iterations, Break with failure

End Repeat

A. Some Experimental Sinkhorn Results

The SSS algorithm was applied to puzzles from [8], ten puzzles each from the rated difficulty levels "Light&Easy" (A); "Moderate" (B); and 25 of the "Demanding" (C); and 23 "Beware! Very Challenging" (D) puzzles. SSS found correct solutions in all cases. Performance comparisons with BP solution are shown in Table I.

Fig. 1 shows the number of unsatisfied constraints as a function of the number of iterations for these puzzles. There tend to be long stretches in which there is no change in the number of constraints satisfied. Also, there is no monotonic improvement: occasionally on the way to the solution the number of unsatisfied constraints will actually *increase*.

The SSS algorithm was also applied to all 250 puzzles in [9], which contains puzzles ranging in size from 6×6 through 16×16. All of these puzzles were also solved.

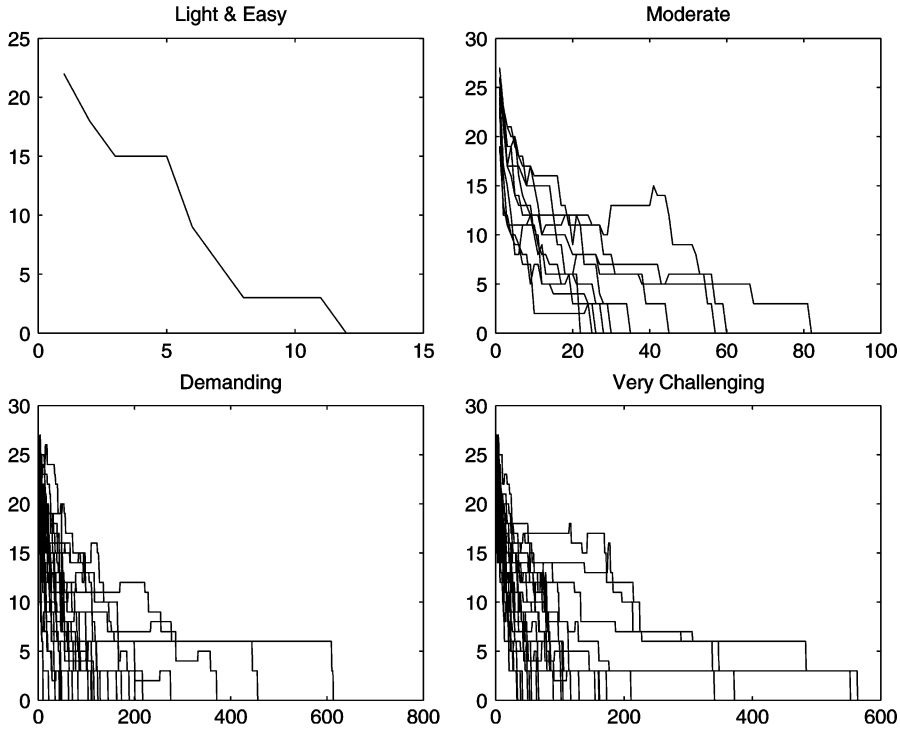

Fig. 1. Convergence of the SSS: horizontal axes show number of iterations; vertical axes show number of unsatisfied constraints.

However, the algorithm does not work on all puzzles; the following puzzle stalls with seven constraints still unsatisfied:

		3			9		8	1
			2				6	
5				1		7		
8	9							
		5	6		1	2		
							3	7
		9		2				8
	7				4			
2	5		8			6		

(2)

B. Proof of Convergence

Let $Q_m^{[k]}$ denote the constraint probability matrix for constraint m at the kth iteration of the SSS algorithm, with elements $q_{m,ij}^{[k]}$, and let $\mathcal{Q}^{[k]}$ denote the set of constraint probability matrices for all constraints

$$\mathcal{Q}^{[k]} = \left\{ Q_1^{[k]}, Q_2^{[k]}, \ldots, Q_{3N}^{[k]} \right\}.$$

Let P_m denote the constraint probability matrix containing singleton probabilities, with elements $p_{m,ij} \in \{0,1\}$. If P_m represents the solution, then P_m is a permutation matrix. Let $\mathcal{P}$ denote the set of P_m matrices for all constraints for a solution, $\mathcal{P} = \{P_1, P_2, \ldots, P_{3N}\}$. Let $D(\mathcal{P}\|\mathcal{Q}^{[k]})$ denote the total KL distance between the sets $\mathcal{P}$ and $\mathcal{Q}^{[k]}$

$$D\left(\mathcal{P}||\mathcal{Q}^{[k]}\right) = \sum_{m=1}^{3N} D\left(P_m||Q_m^{[k]}\right) \tag{3}$$

where

$$D(P_m||Q_m^{[k]}) = \sum_{i,j=1}^{N} p_{m,ij} \log \frac{p_{m,ij}}{q_{m,ij}^{[k]}}$$

is the KL distance between the argument matrices.

Theorem 1: Let $\mathcal{P}$ denote the set of probability constraint matrices for the solution of a puzzle with a unique solution. Then, $D(\mathcal{P}||\mathcal{Q}^{[0]}) < \infty$ and

$$D(\mathcal{P}\left\|\mathcal{Q}^{[k+1]}\right.) \leq D(\mathcal{P}\left\|\mathcal{Q}^{[k]}\right.). \tag{4}$$

That is, every iteration gets closer to (or more precisely, no further from) the solution.

Furthermore, if $\bar{\mathcal{P}}$ is the solution to another puzzle, then $D(\bar{\mathcal{P}}||\mathcal{Q}^{[k]}) = \infty$ for all k.

Proof: For notational brevity, use Q to denote $Q_m^{[k]}$ and P to denote P_m. First, observe that $D(\mathcal{P}\|\mathcal{Q}^{[0]}) < \infty$. The locations where there might be trouble are those terms in $\sum_{i,j} p_{ij} \log p_{ij}/q_{ij}$ where $p_{ij} = 1$ and $q_{ij} = 0$. However, the only place where $q_{ij} = 0$ are those that are already known to be 0 by the presence of clue cells, and hence for which $p_{ij} = 0$ also.

We will show that every Sinkhorn step on every constraint probability matrix is nonincreasing. Prior to Sinkhorn balancing, the matrix Q is row stochastic, but not necessarily column stochastic.

For a matrix Q, the vertical step produces a matrix $Q^{|}$ with elements $q_{ij}^{|} = \frac{q_{ij}}{\sum_l q_{lj}} \triangleq \frac{q_{ij}}{\chi_j}$, where χ_j is the sum of column j. Then

$$\begin{aligned} D(P||Q^{|}) &= \sum_{i,j} p_{ij} \log \frac{p_{ij}}{q_{ij}/\chi_j} \\ &= D(P||Q) + \sum_{i,j} p_{ij} \log \chi_j. \end{aligned} \tag{5}$$

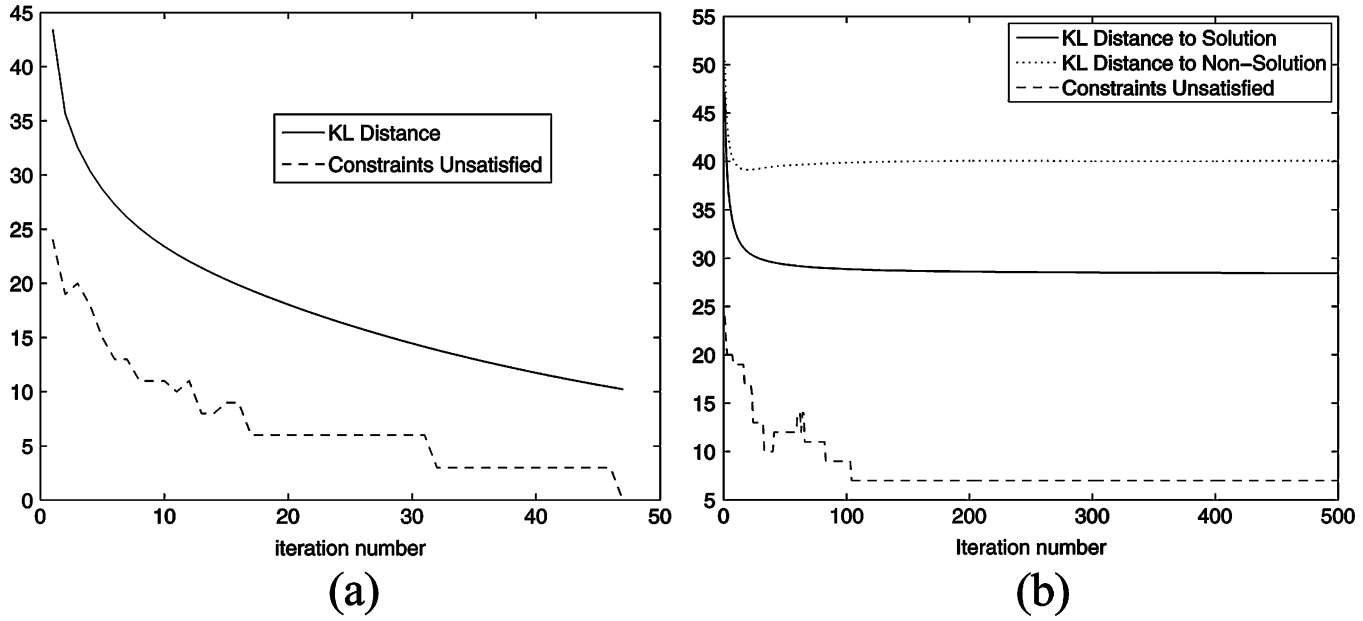

Fig. 2. Convergence results for a solved an unsolved puzzle. (a) Constraints unsatisfied and $D(\mathcal{P}\|\mathcal{Q}^{[k]})$ for a puzzle. (b) Convergence results for the puzzle in (2).

Using the inequality $\log x \leq x - 1$ and recalling that for a permutation matrix $\sum_{i,j} p_{ij} = N$ and that $\sum_j \chi_j = N$, the second term can be written as

$$\begin{aligned}\sum_{i,j} p_{ij} \log \chi_j &\leq \sum_{i,j} p_{ij}(\chi_j - 1) \\ &= \sum_{i,j} p_{ij}\chi_j - N \\ &= \sum_j \chi_j - N = N - N = 0 \end{aligned} \tag{6}$$

so that $D(P||Q^{|}) \leq D(P||Q)$. Equality holds in (6) if and only if $\chi_j = 1$ for all j; that is, if the Sinkhorn balancing has already converged, so that Q is doubly stochastic.

Similarly, letting $Q^{-} = Q_m^{[k+1]}$ denote the results of the row step, we obtain $D(P||Q^{-}) \leq D(P||Q^{|})$. That is, for every m, $D(P_m||Q_m^{[k+1]}) \leq D(P||Q^{|}) \leq D(P_m||Q_m^{[k]})$.

If $\bar{\mathcal{P}}$ is the solution to another puzzle, let n be a cell number that has a clue specified by $\bar{\mathcal{P}}$ that differs from the solution given by $\mathcal{P}$. There must be at least one such cell, or $\bar{\mathcal{P}}$ would be consistent with all clues and hence, by the uniqueness assumption, would be a solution to the initial puzzle. Let Q_m be an initial probability constraint matrix having $\mathbf{p}_n$ as a row, and $\bar{P}_m$ the corresponding constraint matrix $\in \bar{\mathcal{P}}$. Then, there is an element of that row such that $\bar{p}_{ij} = 1$ but $q_{ij} = 0$, which causes $D(\bar{P}_m||Q_m) = \infty$, hence $D(\bar{\mathcal{P}}||\mathcal{Q}^{[0]}) = \infty$, and so $D(\bar{\mathcal{P}}||\mathcal{Q}^{[k]}) = \infty$ for every $k \geq 0$. □

Fig. 2(a) shows the number of constraints unsatisfied and $D(\mathcal{P}||\mathcal{Q}^{[k]})$ as a function of the iteration number for the puzzle in (1). While the number of constraints unsatisfied does not decrease monotonically, $D(\mathcal{P}||\mathcal{Q}^{[k]})$ does.

There is also a sort of quasi-converse to Theorem 1, which roughly states the following: eventually the distance to a nonsolution may grow larger. A *nonsolution* is an assignment to cells that does not satisfy all of the constraints. That is, in some constraint set, one or more of the digits is repeated, so that another digit is left out.

If $\tilde{\mathcal{P}}$ represents a nonsolution (that is, one or more of the $\tilde{P}_m \in \tilde{\mathcal{P}}$ are not permutation matrices), then the inequality (4) may not hold: It could be the case that $D(\tilde{\mathcal{P}}||\mathcal{Q}^{[k+1]}) > D(\tilde{\mathcal{P}}||\mathcal{Q}^{[k]})$. To examine this, let $\tilde{\mathcal{P}}$ be a nonsolution, so that one or more of the $\tilde{P}_m \in \tilde{\mathcal{P}}$ is not a permutation matrix. For such a matrix $\tilde{P}_m$, let i_1 and i_2 represent cells that contain duplicate values for some value j_1 (that is, $\tilde{p}_{i_1 j_1} = 1$ and $p_{i_2 j_1} = 1$), and let j_2 be a value that is not in the constraint set, so that $\tilde{p}_{ij_2} = 0$ for all $i \in \{1, \ldots, N\}$. As in (5), we have

$$D(\tilde{P}||Q^{|}) = D(\tilde{P}||Q) + \sum_{i,j} \tilde{p}_{ij} \log \chi_j.$$

Consider the term $\sum_{i,j} \tilde{p}_{ij} \log \chi_j$, using the same inequality as before

$$\sum_{i,j} \tilde{p}_{ij} \log \chi_j \leq \sum_{i,j} \tilde{p}_{ij}\chi_j - \sum_{i,j} \tilde{p}_{ij}.$$

If $\chi_{j_1} > \chi_{j_2}$, then it is the case that $\sum_{i,j} \tilde{p}_{ij}\chi_j - \sum_{i,j} \tilde{p}_{ij} > 0$. There is thus "room" provided by this inequality, but no guarantee, for $D(\tilde{P}||Q^{|}) = D(\tilde{P}||Q)+$ some positive quantity, or $D(\tilde{P}||Q^{|}) > D(\tilde{P}||Q)$. Whether this happens depends on if χ_{j_1} is sufficiently greater than χ_{j_2}.

Since the row sums of $\tilde{P}_m$ are all still 1, it is still the case that $D(\tilde{P}_m||Q^{-}) \leq D(\tilde{P}_m||Q^{|})$. However, if $D(\tilde{P}_m||Q^{-})$ is very nearly equal to $D(\tilde{P}_m||Q^{|})$, then we will still have $D(\tilde{P}_m||Q^{-}) > D(\tilde{P}_m||Q)$.

Furthermore, if the other terms in the sum (3) are sufficiently small (e.g., the algorithm is converging for those constraints so that $D(P_{m'}||Q^{[k+1]}) \approx D(P'_m||Q^{[k]})$, then their contribution to the sum may be negligible.

This explanation is full of caveats and conditions. What happens in practice is suggested by this description. For the

first few iterations, we may see $D(\tilde{\mathcal{P}}||Q^{[k+1]}) < D(\tilde{\mathcal{P}}||Q^{[k]})$. However, after the values have started to converge, so that $D(P_m||Q^{[k+1]}) \approx D(P_m||Q^{[k]})$ for consistent constraints m, most of the terms in the sum in (3) become negligible. However, for a nonsolution matrix, the conditions allowing for $D(\tilde{P}_m||Q^{[k+1]}) > D(\tilde{P}_m||Q^{[k]}))$ occur, to the extent that the overall distance $D(\tilde{\mathcal{P}}||Q^{[k+1]}) > D(\tilde{\mathcal{P}}||Q^{[k]})$. To illustrate this, Fig. 2(b) shows the KL distance as a function of iteration to the true solution to the puzzle in (2) and the KL distance to a nonsolution. The distance to the nonsolution begins decreasing, but quickly fails to do as well as the solution to the true solution. After about 20 iterations, the distance begins to increase.

Theorem 1 falls a little short of proving convergence for every puzzle: if at some iteration of SSS, the Sinkhorn balancing does not change the probability matrices, so that $Q_m^{[k+1]} = Q_m^{[k]}$, or if the change is negligibly small numerically, then there will be no change in the KL distance. In Fig. 2(b), $D(\mathcal{P}||\mathcal{Q}^{[k]})$, the rate of decrease drops nearly to zero. This flattening continues as the number of iterations increases, so that convergence to the final answer never happens: there remain seven constraints unsatisfied. This suggests that there is a "solvability" threshold, analogous to a decoding capacity of an LDPC code under message passing decoding [10], above which puzzles can be correctly solved.

There are some other information theoretic connections between Sinkhorn balancing and theKL distance. At each step in the solution to the puzzle, a solution algorithm should preserve as much information as possible from the original set of clues, or from previously obtained information, while striving to satisfy the uniqueness constraints. In this regard, we pose the following problem: Given a matrix $Q_m^{[k]}$, determine the matrix $Q_m^{[k+1]}$ that is doubly stochastic (to enforce the puzzle constraints), which minimizes $D(Q_m^{[k+1]}||Q_m^{[k]})$. We thus obtain the constrained optimization problem: $\text{minimize}_{\{q_{ij}^{[k+1]}\}} \sum_{i,j} q_{ij}^{[k+1]} \log \frac{q_{ij}^{[k+1]}}{q_{ij}^{[k]}}$, subject to

$$\begin{aligned} \sum_j q_{ij}^{[k+1]} &= 1 \qquad \forall i = 1,2,\ldots,N \\ \sum_i q_{ij}^{[k+1]} &= 1 \qquad \forall j = 1,2,\ldots,N \\ q_{ij}^{[k+1]} &\geq 0 \qquad \forall i,j = 1,2,\ldots,N. \end{aligned} \tag{7}$$

Theorem 2 of [6] has shown that the solution to (7) is equivalent to the result obtained via Sinkhorn balancing. Thus, SSS is, in a sense, an optimal information preserving algorithm.

Another information theoretic connection [6] makes use of the log-sum inequality, which states [11, Th. 2.7.1]: For nonnegative numbers $a_1,\ldots,a_n$ and $b_1,\ldots,b_n$, $\sum_{i=1}^n b_i \log \frac{b_i}{a_i} \geq (\sum_{i=1}^n b_i) \log \frac{\sum_{i=1}^n b_i}{\sum_{i=1}^n a_i}$, with equality if and only if $\frac{b_i}{a_i}$ is a constant. In our case, a row or column of the Q_m matrices forms a nonnegative set of numbers $\{a_i\}$. By the log-sum inequality, of the possible rows (or columns) of a new matrix $\{b_i\}$, the ones that minimize the KL distance are obtained by scaling the elements of the old row (or column) by the same amount, so that b_i/a_i is constant. This is, of course, what Sinkhorn balancing does, scaling each row (or column) by the row (or column) sum. Thus, the matrix that the Sinkhorn process converges to will minimize the KL distance from the original matrix.

IV. Random Sudoku

For the clue cells in a Sudoku puzzle, the representative probability vectors are singleton, so that the row operation in Sinkhorn balancing recovers these singleton probabilities exactly. The information present in the clues is rejuvenated, for every Sinkhorn balance operation. However, since a goal of this exploration is to gain insight into decoding of error correction codes, and since the observed values are subject to uncertainty, a model that introduces some uncertainty into the clues is of interest.

Consider a puzzle in which the clues are provided only up to a possible set. For example, a puzzle might be defined by providing as possible clues all clues on the row, where clues in each box are assumed to be equally likely. Thus, the puzzle in (1) would be presented as a random puzzle as follows:

			4 6 9 7		4 6 9 7	4 6 9 7	4 6 9 7	
				6 7 9	6 7 9	6 7 9		
								3
					1 4 7	1 4 7	1 4 7	
1 4 6 8	1 4 6 8						1 4 1 8	1 4 6 8
	1 2 6	1 2 6	1 2 6					
1								
		2 3 4	2 3 4	2 3 4				
	2 4 6 9	2 4 6 9	2 4 6 9		2 4 6 9			

While this random puzzle might be substantially more difficult for a human puzzle solver, or may involve more branching for a conventional discrete constraint satisfaction solver, the SSS algorithm is able to handle this without trouble in many cases. On 88% of the puzzles tried (all the 9×9 puzzles out of [9] converted to random Sudoku), the solution is found. On puzzles for which the solution is not found, in most cases, the constraints violated uniquely indicate the location of cells in error.

V. Conclusion

Probabilistic representations of the contents of the cells are effective in setting the stage for solution of Sudoku puzzles and their generalization to random puzzles using Sinkhorn balancing. It is interesting that the Sinkhorn balancing technique is more effective than the BP. The fact that there are puzzles not solved by Sinkhorn raises the interesting question of a solvability threshold, which is not pursued here.

The purpose of this investigation was to use these puzzles to gain insight into the error correction decoding process, and in this respect, it holds some interesting promise. The effectiveness of Sinkhorn balancing compared with BP invites exploration of Sinkhorn-like decoding algorithms, which apply to linear block codes, especially codes whose Tanner graphs have a large number of cycles.

REFERENCES

[1] T. Davis, "The mathematics of Sudoku," [Online]. Available: http://www.geometer.org/mathcircles/sudoku.pdf

[2] T. K. Moon and J. H. Gunther, "Multiple constraint satisfaction by belief propagation: An example using Sudoku," in *Proc. SMCals/IEEE Mountain Workshop Adaptive Learn. Syst.*, Jul. 2006, pp. 122–126.

[3] S. Kirkpatrick, "Message-passing and Sudoku," [Online]. Available: http://www.www.cs.huji.ac.il/~kirk/Sudoku.ppt

[4] R. Sinkhorn, "A relationship between arbitrary positive matrices and doubly stochastic matrices," *Ann. Math. Statist.*, vol. 35, pp. 876–879, Jun. 1964.

[5] R. Sinkhorn, "Diagonal equivalence to matrices with prescribed row and column sums," *Amer. Math. Monthly*, vol. 35, pp. 876–879, 1967.

[6] H. Balakrishnan, I. Hwang, and C. J. Tomlin, "Polynomial approximation algorithms for belief matrix maintenance in identity management," in *Proc. 43rd IEEE Conf. Decision Control*, Dec. 14–17, 2004, pp. 4874–4979.

[7] R. Sinkhorn and P. Knopp, "Concerning nonnegative matrices and doubly stochastic matrices," *Pacific J. Mathematics*, vol. 21, no. 2, pp. 343–348, 1967.

[8] W. S. ed. , *Pocket Sudoku*. New York: St. Martins Paperbacks, 2005.

[9] M. Huckvale, *The Big Book of Sudoku # 2*. New York: New Market Press, 2005.

[10] T. J. Richardson and R. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Info. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.

[11] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. New York: Wiley, 2006.

Todd K. Moon (S'86–M'88–SM'00) received the Ph.D. degree in electrical engineering from the University of Utah, Salt Lake City, in 1991.

He has been at Utah State University since 1991 and is currently Professor and Head of the Electrical and Computer Engineering Department. He is the author of *Mathematical Methods and Algorithms for Signal Processing* (Englewood Cliffs, NJ: Prentice-Hall, 2000) and *Error Correction Coding: Mathematical Methods and Algorithm* (New York: Wiley, 2005). His research interests include error correction coding algorithms, multidimensional signal processing, and statistical signal processing.

Jacob H. Gunther (S'91–M'95) received the Ph.D. degree in electrical engineering from Brigham Young University, Provo, UT, in 1998.

He joined the Electrical and Computer Engineering Department, Utah State University, Logan, in 2000, where he is currently an Associate Professor. His research interests span statistical signal processing, communication systems, and radar signal processing.

Joseph J. Kupin received the Ph.D. degree in linguistic from University of Connecticut in 1980.

Then, he joined the Center for Communications Research, Princeton, NJ, as a member of the technical staff, where he develops algorithms and techniques for signal processing and communications.