Multiple Constraint Satisfaction by Belief Propagation: An Example Using Sudoku

Todd K. Moon and Jacob H. Gunther
Utah State University

Abstract— **The popular Sudoku puzzle bears structural resemblance to the problem of decoding linear error correction codes: solution is over a discrete set, and several constraints apply. We express the constraint satisfaction using a Tanner graph. The belief propagation algorithm is applied to this graph. Unlike conventional computer-based solvers, which rely on humanly specified tricks for solution, belief propagation is generally applicable, and requires no human insight to solve a problem. The presence of short cycles in the graph creates biases so that not every puzzle is solved by this method. However, all puzzles are at least partly solved by this method. The Sudoku application thus demonstrates the potential effectiveness of BP algorithms on a general class of constraint satisfaction problems.**

I. Introduction

The belief propagation (BP) paradigm (also known as message passing (MP)) realizes Bayesian inference on graphs without cycles [1], [2], [3], and performs nearly Bayesian belief propagation on graphs with cycles [4], [5]. BP algorithms can be used to describe a variety of algorithms, including fast Hadamard transforms, the Kalman filter, fast Fourier transforms, MAP decoding algorithms (including decoding algorithms for turbo codes), and the Viterbi algorithm. Most relevant to the current consideration, BP algorithms are also the means by which low density parity check (LDPC) codes are decoded [6]. In LDPC decoding, information about received bits that is implied collectively by the set of parity constraints is combined together in a (nearly) Bayesian way with information from the received data to provide information about the bits that were originally transmitted. In this paper, we demonstrate how the BP paradigm — borrowed closely from LDPC decoding — can be applied to a problem with multiple local combinatorial constraints, namely, the popular Sudoku puzzle. While there are other methods of solving Sudoku puzzles, the BP method is very general and does not require any human insight or tricks, nor does it require building solution trees. It is thus potentially applicable to a broad variety of problems as a general tool.

Easy Sudoku puzzles can be solved by simple elimination. Difficult Sudoku puzzles are actually NP-complete [7]. These puzzles are thus well-scoped examples worthy of studying the general class of NP-complete problems. The error correction decoding problem is also NP-complete [8], but very effective sub-optimal decoding algorithms exist. Even for codes with not particular structure, message passing algorithms can be somewhat effective, but not perfect [9]. In this paper, we present progress toward a solution, but do not provide a solution that works in all cases.

Interestingly, the algorithm works from general rules of inference, not from particular special cases and tricks, which are probably employed in the Sudoku puzzle posers and solvers which are widespread. It is thus applicable to a wide variety of constraint satisfaction problems.

In applying BP methods, the problem is mapped to a graph and messages representing Bayes probabilities are passed among the nodes of the graph. For the Sudoku puzzle, evidence about the contents of cells of the Sudoku puzzle is learned in a soft (probabilistic) way from the degree to which the cell contents satisfy the constraints of the puzzle. This application thus demonstrates application of BP to problems with multiple local constraints.

BP is Bayesian optimal for graphs without cycles, but suffers from biases for graphs with cycles. The graph associated with Sudoku, like the graphs for LDPC codes, does have cycles. Every node in the Sudoku graph lies on two cycles of length four. The biases introduced by these short cycles cause failure of the BP method for more difficult puzzles.

Unlike the LDPC case, in which the combinatorial complexity of the marginalizations can be efficiently treated using a lattice structure, the marginalization associated with Sudoku still retains combinatorial complexity. However, it is localized only to individual constraints, so that a search over the global set of constraints is avoided.

Despite these shortcomings, the application reveals the possibility of applying BP techniques to multiconstraint problems, at least to eliminate many of the possibilities, perhaps leaving the problem sufficiently small that a global search may be possible.

1-4244-0166-6/06/$20.00 (c)2006IEEE

II. Puzzle description and notation

In a 9×9 Sudoku puzzle, the problem is to place integers from 1 to 9 in a 9×9 array in such a way that each integer appears once on each row, once in each column, and once in each of the nine 3×3 blocks partitioning the array. The puzzle is seeded with some cells filled in. The difficulty of the puzzle generally increases as the number of seed values decreases. An example Sudoku puzzle is shown here.

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

(1)

We refer to the elements of the matrix as "cells," and denote the contents of cell n by $S_n \in \{1, 2, \ldots, 9\}$ for $n = 1, 2, \ldots, 81$. Cells are numbered in row-scan order. The constraints of the puzzle can be described as follows. Each constraint involves nine cells. A constraint is satisfied if all nine cells associated with it — all nine arguments to the constraint function — are distinct. A constraint function $C_m : \{1, \ldots, 9\}^9 \to \{0, 1\}$ is defined as

$$C_m(s_1, s_2, \ldots, s_9) = \begin{cases} 1 & s_1, s_2, \ldots, s_9 \text{ are all distinct} \\ 0 & \text{otherwise.} \end{cases}$$

Let C_1 through C_9 denote the constraints associated with the rows of the puzzle, C_{10} through C_{18} the constraints associated with the columns, and C_{19} through C_{27} the constraints associated with the 3×3 blocks. Generally we denote cell indices with n or n' and denote constraint indices with m or m'.

We can associate a bipartite graph with the Sudoku puzzle, consisting of a set containing 81 nodes corresponding to the cells S_n of the puzzle, and another set containing 27 nodes corresponding to the constraints C_m of the puzzle. The graph has an edge between cell node S_n and a constraint node C_m if and only if S_n is involved in constraint C_m. Figure 1 diagrams the graph. In the error correction coding literature, such a graph is referred to as a Tanner graph [10]. (In the error correction coding application, the constraint functions are parity check functions.) The Tanner graph for the Sudoku puzzle reveals that structurally, the LDPC decoding problem is similar to solving the Sudoku puzzle.

We denote the set of indices of the cells (that is, the n values) that participate in constraint C_m by $\mathcal{N}_m$, and

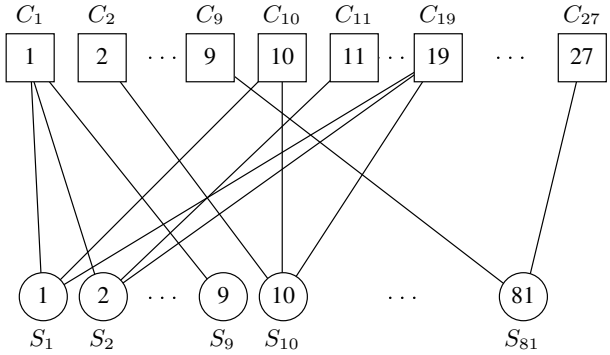


Fig. 1. Tanner graph associated with Sudoku puzzle

the set of indices of the constraints (the m values) that associate with cell S_n by $\mathcal{M}_n$. For example,

$$\mathcal{N}_1 = \{1, 2, 3, 4, 5, 6, 7, 8, 9\},$$

$$\mathcal{N}_{10} = \{1, 10, 19, 28, 37, 46, 55, 44, 73\},$$

$$\mathcal{N}_{19} = \{1, 2, 3, 10, 11, 12, 19, 20, 21\},$$

$$\mathcal{M}_1 = \{1, 10, 19\} \qquad \mathcal{M}_2 = \{1, 11, 19\}.$$

We use a double subscript notation to indicate that elements are removed from these sets. For example, $\mathcal{N}_{m,n} = \mathcal{N}_m \backslash n$ denotes the cells involved in constraint m, except for cell n. For example,

$$\mathcal{N}_{10,19} = \{1, 10, 28, 37, 46, 55, 44, 73\}.$$

III. Belief Propagation Formulation

In the belief propagation algorithm, the nodes in the Tanner graph send messages to each other, representing local information about the nodes. For the Sudoku puzzle, a constraint node sends a message about the probability that the constraint is satisfied, which it computes using information from the cell nodes about the probabilities of the cell contents. A cell node, on the other hand, sends a message about the probabilities of the various cell contents, given information about the constraints associated with that cell. For a graph with cycles, nodes in the graph send information to each other until the messages converge, or until all constraints are satisfied, or until some maximum number of iterations is reached.

We model the contents of the cells probabilistically. Let

$$\mathbf{p}_n = \begin{bmatrix} P(S_n = 1) & P(S_n = 2) & \cdots & P(S_n = 9) \end{bmatrix}$$

be the probability vector associated with cell S_n. Cells which are specified initially place all their probability mass on the specified value, while unspecified cells have probability uniformly distributed over possible outcome values. (The possible outcome values are obtained by

eliminating values from consideration which would violate the three constraints associated with that cell. This is not strictly necessary; initial probabilities could be uniformly distributed over all nine possibilities. However, eliminating some contents based on constraints reduces the number of iterations of the algorithm.)

For example, for the puzzle in (1),

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

$$\mathbf{p}_1 = \frac{1}{4}\begin{bmatrix}0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0\end{bmatrix}$$

$$\mathbf{p}_2 = \frac{1}{4}\begin{bmatrix}1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0\end{bmatrix}, \qquad \text{etc.}$$

where $\mathbf{e}_k$ is a vector of length 9 with a single 1 at position k and zeros in other positions. We also use $\mathbf{1}$ as a vector of all ones.

Let $\mathcal{C}_m$ denote the event that the mth constraint is satisfied; that is, the event that all of the cells in that constraint are distinct.

Belief propagation operates by sending probabilistic messages between adjacent nodes in the graph. The message that constraint node C_m sends to cell S_n is

$$r_{mn}(x) = P(C_m \text{ is satisfied} | S_n = x) = P(\mathcal{C}_m | S_n = x)$$

that is, the probability that constraint C_m is satisfied when the cell S_n contains x (see [11] for related discussion for LDPC decoding). The "message" from C_m to S_n is actually a probability vector, e.g.,

$$\mathbf{r}_{mn} = \begin{bmatrix} r_{mn}(1), r_{mn}(2), \ldots, r_{mn}(9) \end{bmatrix}.$$

The message that cell node S_n sends to constraint node C_m is

$$\begin{aligned} q_{mn}(x) &= P(S_n = x | \text{all the constraints except } C_m \\ &\qquad \text{involving } S_n \text{ are satisfied}) \\ &= P(S_n = x | \{\mathcal{C}_{m'}, m' \in \mathcal{M}_{nm}\}), \end{aligned}$$

that is, the probability that $S_n = x$ given that all of the constraints connected to S_n are satisfied, except the constraint to which the message is being sent.

The decision values are based upon the message that cell node S_n obtains from *all* of the constraints,

$$\begin{aligned} q_n(x) &= P(S_n = x | \text{all constraints involving } S_n \\ &\qquad \text{are satisfied}) \\ &= P(S_n = x | \{\mathcal{C}_{m'}, m' \in \mathcal{M}_n\}). \end{aligned}$$

If there were no cycles in the graph, belief propagation theory asserts that, after a sufficiently large number of message passing steps, $q_n(x)$ would be the Bayesian posterior probability, incorporating information both from the prior probabilities and the evidence provided by the constraints [1], [2]. If there are cycles in the graph, then evidence recirculates around the graph, leading to potentially biased results. However, experience has shown that the results are usually still useful [4].

The belief propagation rules are derived under certain assumptions of statistical independence. Strictly speaking, cycles in the graph lead to violation of these assumptions. However, the assumptions are approximately true, and lead to tractable, and useful, results. It now remains to develop expressions for these probabilities.

A. Constraint to Cell Message $r_{mn}(x)$

We can write

$$\begin{aligned} r_{mn}(x) &= P(\mathcal{C}_m | S_n = x) \\ &= \sum_{\{x_{n'}, n' \in \mathcal{N}_{m,n}\}} P(\mathcal{C}_m, \{S_{n'} = x_{n'}\} | S_n = x) \\ &= \sum_{\{x_{n'}, n' \in \mathcal{N}_{m,n}\}} \Big(P(\mathcal{C}_m | S_n = x, \{S_{n'}, n' \in \mathcal{N}_{m,n}\}) \\ &\qquad \times P(\{S_{n'}, n' \in \mathcal{N}_{m,n}\} | S_n = x) \Big). \end{aligned}$$

We now invoke an assumption that the cells in the set $\{S_{n'}, n' \in \mathcal{N}_{m,n}\}$ are independent. This is clearly not true, since cells associated with a constraint must have distinct contents; if $S_1 = 1$, it cannot be the case the $S_2 = 1$ also. However, following the spirit of the LDPC decoder we use that assumption. We thus have

$$\begin{aligned} r_{mn}(x) = \sum_{\{x_{n'}, n' \in \mathcal{N}_{m,n}\}} & \Big(P(\mathcal{C}_m | S_n = x, \{S_{n'}, n' \in \mathcal{N}_{m,n}\}) \\ & \times \prod_{l \in \mathcal{N}_{m,n}} P(S_l = x_l | S_n = x) \Big). \end{aligned}$$

We also note that $P(\mathcal{C}_m | S_n = x, \{S_{n'}, n' \in \mathcal{N}_{m,n}\})$ is conditioned upon all of the cells connected to C_m. Constraint is C_m is then either satisfied or not, depending on the values of the arguments. Thus

$$P(\mathcal{C}_m | S_n = x, \{S_{n'}, n' \in \mathcal{N}_{m,n}\}) = \begin{cases} 1 & \text{all } S_n \text{ and } \{S_{n'}, n' \in \mathcal{N}_{m,n}\} \text{ are distinct} \\ 0 & \text{otherwise.} \end{cases}$$

We thus have

$$r_{mn}(x) = \sum_{\substack{\{x_{n'}, n' \in \mathcal{N}_{m,n}\} \\ \{x, x_{n'}\} \text{ all unique}}} \prod_{l \in \mathcal{N}_{m,n}} P(S_l = x_l | S_n = x)$$

To formulate this as a belief propagation step, we invoke the approximation

$$P(S_l = x_l | S_n = x) = q_{ml}(x_l),$$

the probability that cell S_l sends to constraint C_m. We thus obtain

$$r_{mn}(x) = \sum_{\substack{\{x_{n'}, n' \in \mathcal{N}_{m,n}\} \\ \{x, x_{n'}\} \text{ all unique}}} \prod_{l \in \mathcal{N}_{m,n}} q_{ml}(x_l). \quad (2)$$

Unfortunately, the sum is over a combinatorial set. However, when some of the cells in $\mathcal{N}_{m,n}$ are known, it reduces the size of the set. Furthermore, this is only a "local" combinatorial complexity, restricted to the cells involved in a constraint, and not over all the empty cells in the puzzle.

B. Cell to Constraint Message $q_{mn}(x)$

We derive $q_n(x)$; modifications to obtain $q_{mn}(x)$ are straightforward.

$$\begin{aligned} q_n(x) &= P(S_n = x | \{\mathcal{C}_m, m \in \mathcal{M}_n\}) \\ &= \frac{P(S_n = x, \{\mathcal{C}_m, m \in \mathcal{M}_n\})}{P(\{\mathcal{C}_m, m \in \mathcal{M}_n\})} \\ &= \alpha P(\{\mathcal{C}_m, m \in \mathcal{M}_n\} | S_n = x) P(S_n = x), \end{aligned}$$

where α is a normalizing constant. We assume independence again, then recognize $r_{mn}(x)$:

$$\begin{aligned} q_n(x) &= P(S_n = x) \prod_{m \in \mathcal{M}_n} P(\mathcal{C}_m | S_n = x) \\ &= P(S_n = x) \prod_{m \in \mathcal{M}_n} r_{mn}(x). \end{aligned}$$

Similarly,

$$q_{mn}(x) = P(S_n = x) \prod_{m' \in \mathcal{M}_{n,m}} r_{m'n}(x). \qquad (3)$$

In operation, the BP iterates between (2) and (3). However, for a cell whose contents are unambiguously known — such as the cells initially filled in — the cell to constraint message is simply the fixed probability vector.

C. Cycle structure of the graph

The BP method is exact on graphs with no cycles. However, the graph associated with Sudoku has many short cycles in it. In fact, every cell is in four cycles of girth four. There are two of the following form:

cell → row constraint → cell on row → box constraint → cell,

one for each of the two other cells on the row in a box, and

cell → column constraint → cell on column → box constraint → cell,

one for each of the two other cells on the column in a box. For example,

1 → row 1 constraint → 2 → box constraint → 1

1 → row 1 constraint → 3 → box constraint → 1

1 → column 1 constraint → 10 → box constraint → 1

1 → column 1 constraint → 19 → box constraint → 1

These many short cycles will definitely bias the results of the message passing algorithm. What results is that not every puzzle is solvable by this MP technique.

IV. Some Results

In solving the puzzle, several iterations of elimination were computed: the possible contents of each cell were eliminated based on the constraints the cell associated with. This reduced the number of BP iterations, and acts according to how a human would begin solving the puzzle. Following this simple elimination, the BP proceeds. As computation proceeds, as a probability vector emerges that places all of its mass on a single cell, a "hard" decision is declared, establishing the contents of a cell. (Filled cells are important because they reduce the computationally complexity of the sum in (2).)

We consider first the puzzle in (1). The initial elimination phase did not result in any simplifications. This was followed by 8 iterations of message passing. The sequence of steps is as follows.

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

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

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

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

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

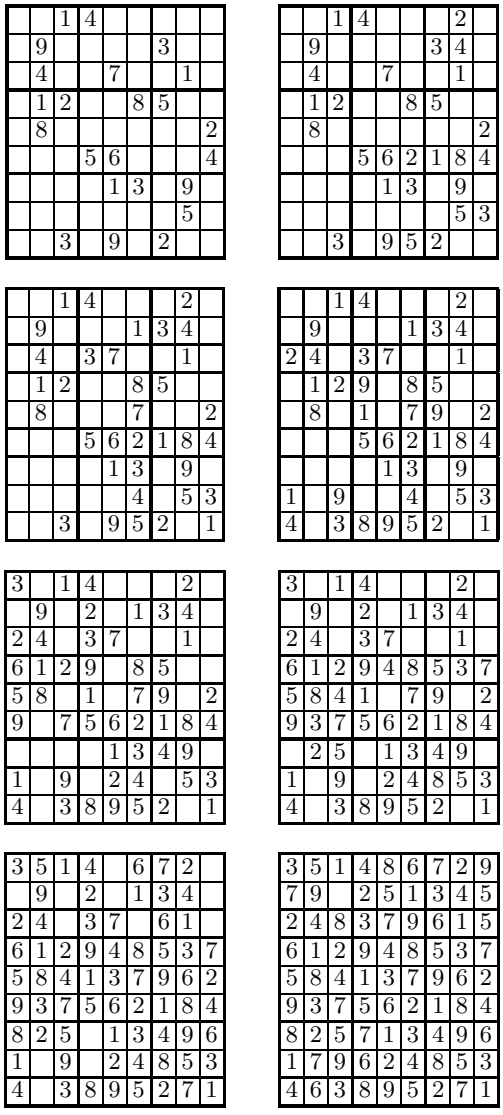
2			4	3	9	6	7	1
4	1	3		7	6	9		2
9		6	1		2	4		3
3			2		1	7	4	6
6	4		5	9	3	2	1	8
8	2	1	6	4		3	9	5
1	3	9		6			2	4
		4	3	2		1	6	9
	6	2	9	1	4		3	7

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

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

After the last step (eight iterations) the puzzle is correctly completed.

As a second example, we present a puzzle whose solution is not completely determined. After easy elimination, we obtain the following.



Note that incorrect decisions have been made: the second row has two 5s on it, and the last column has two 5s on it. Ultimately, this is due to biases introduced due to cycles.

V. Conclusions

The MP paradigm is straightforward to apply to some problems with multiple constraints, with solutions obtained over discrete sets. The computational complexity is localized to each constraint.

Cycles lead to failures in some cases, due to biases in the MP process. Addressing these biases is the topic for future investigations.

References

[1] S. M. Aji and R. J. McEliece, "The Generalized Distributive Law," *IEEE Trans. Info. Theory*, vol. 46, pp. 325–343, Mar. 2000.

[2] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor Graphs and the Sum-Product Algorithm," *IEEE Trans. Info. Theory*, vol. 47, pp. 498–519, Feb. 2001.

[3] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*. San Mateo, CA: Morgan Kaufmann, 1988.

[4] Y. Weiss, "Correctness of Local Probability Propagation in Graphical Models with Loops," *Neural Computation*, vol. 12, pp. 1–41, 2000.

[5] Y. Weiss and W. T. Freeman, "Correctness of Belief Propagation in Gaussian Graphical Models of Arbitrary Topology," *Neural Computation*, vol. 13, pp. 2173–2200, 2001.

[6] R. G. Gallager, "Low-Density Parity-Check Codes," *IRE Trans. on Info. Theory*, vol. IT-8, pp. 21–28, Jan. 1962.

[7] L. Aaronson, "Sudoku science," *IEEE Spectrum*, Feb. 2006.

[8] E. R. Berlekamp, R. J. McEliece, and H. C. van Tilborg, "On the Inherent Intractability of Certain Coding Problems," *IEEE Trans. Info. Theory*, vol. 24, pp. 384–386, May 1978.

[9] T. Moon, "On general linear block code decoding using the sum-product iterative decoder," *IEEE Comm. Lett.*, 2005.

[10] R. Tanner, "A Recursive Approach To Low Complexity Codes," *IEEE Trans. Info. Theory*, vol. 27, pp. 533–547, Sept. 1981.

[11] T. K. Moon, *Error Correction Coding: Mathematical Methods and Algorithms*. Wiley, 2005.