

ECEN 662 REPORT

BELIEF PROPAGATION

Digveer De, Richard Marvelloss, Virisha Timmaraju

Abstract-Message passing can be considered as a constraint Satisfaction Problem, and It appears in a large number of problems. In any constraint satisfaction problem, we are interested in assigning a certain set of variables a value, while satisfying a set of constraints on them. The belief propagation /message passing algorithm calculates exact solution marginals when the factor graph is a tree. In every iteration, messages are passed between variables and checks, till we converge to the exact solution marginals. In this work, we shall discuss message passing and its applications, followed by an application in the form of a Sudoku puzzle solver.

I. INTRODUCTION

Graph theory models large collections of random variables with complex interdependencies, in a simplified manner. It strikes a balance between the biggest problems posed by mathematics, uncertainty and complexity. Probabilistic graphical models are graphs in which the nodes represent random variables and the edges between nodes represent their conditional probability density functions. There are two kinds of graphical models:

- Directed Models
- Undirected Models

Directed and undirected models make different assertions of conditional independence. So, the problems solved by the directed ones cannot be solved by the undirected model and vice versa. They are extensively used for the following purposes;

- Image Recognition: Deep Belief Networks
- Pagerank: Random Walk based inference learning
- Illness Diagnosis: Bayesian Trees
- Text Analysis: Latent Dirichlet Analysis

A. *Directed Models*

They represent ‘causal’ relationships between nodes with a Directed Acyclic Graph (DAG). For every graphical model, we represent its parameters, the conditional probability distribution at every node, in the case of directed models. If the random variables are discrete, we represent a conditional probability table.[1]

In the example depicted in figure-1, we see that the conditional probability table is mentioned for every node in the graph. If I walk out of my house and find that the grass is wet, I would evaluate whether it is because of the rain or sprinkler, but would not consider its conditional probability when it is cloudy, since the parent nodes for wet grass are only sprinkler and rain. In this way if we had n binary nodes, the full joint would require $O(2^n)$ space to represent, but the factored form would require $O(n \cdot 2^k)$ space to represent, where k is the maximum fan-in of a node. This clearly indicates the need for a probabilistic graphical model.

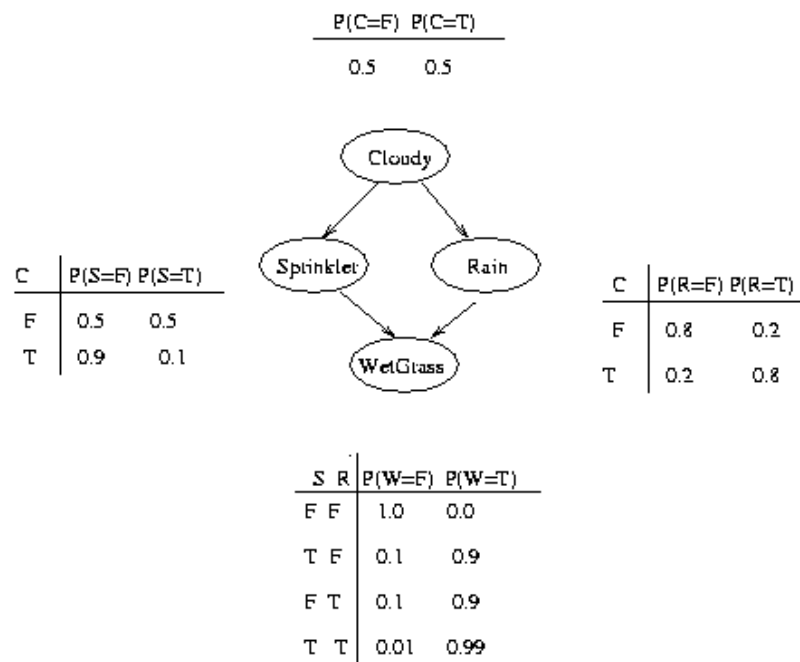


Figure 1 An example for Directed Model (Bayesian Network)

Bayesian networks are most commonly find uses in solving probabilistic inference problems. In our example concerning the wet grass, we could use the bayesian

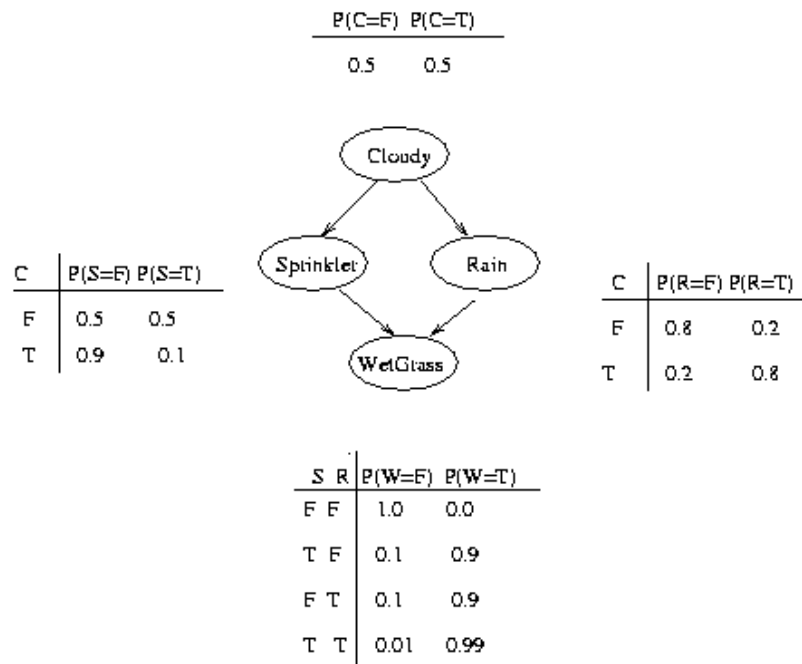
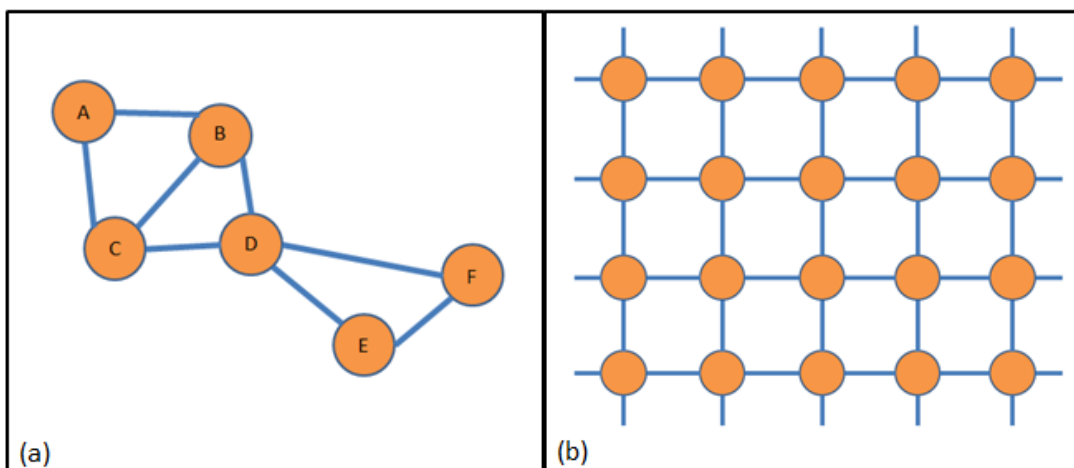


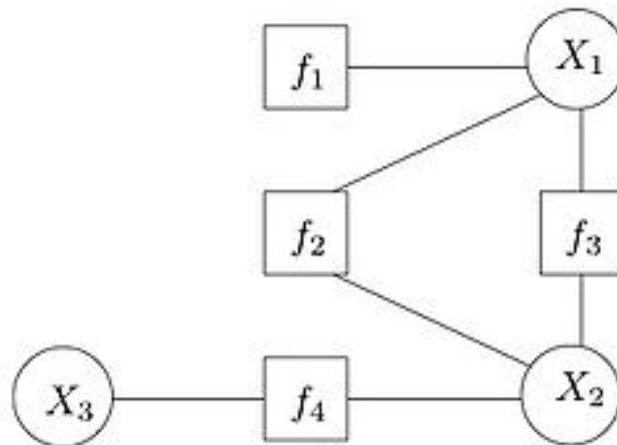
Figure 2 Bayesian Network for Weather Determination

network to find the posterior probabilities of the cause for wet grass being the sprinkler and then the rain.

Figure-2a represents a cyclic relationship between the nodes. Node A is dependent on Node B, which is dependent on C, which is again dependent on node A. Figure-2b represents a grid like graphical model, with symmetric dependencies between nodes, as in the case of pixels in an image, where every pixel is symmetrically dependent on all its neighbors.



The parameters in an undirected graphical model are represented by factors, in turn leading to a factor graph. In a factor graph, the factors are nodes, representing the interaction between one or more of the variables, connected to them with edges. An example factor graph is depicted in figure-3, where the factor f_1 acts only on X_1 , f_2 , f_3 act on both X_1 and X_2 , f_4 acts on X_2 and X_3 . So the joint distribution function can be simply represented as a product of four terms



$f_1(X_1), f_2(X_1, X_2), f_3(X_1, X_2), f_4(X_2, X_3)$, greatly easing the complexity of computations. Here, if we merge f_2 and f_3 into a single factor, since the factor graph is cyclic, it will become a tree.

$$g(X_1, X_2, X_3) = f_1(X_1) f_2(X_1, X_2) f_3(X_1, X_2) f_4(X_2, X_3)$$

Another useful type of graph is the cluster graph, where each node is a cluster, meaning it is connected with a set of nodes internally. So, in a graph there will be many links within a cluster and fewer links between clusters.

II. CLUSTER GRAPHS

Though both Bayesian or Markov network can be used to perform inference, sometimes the number of random variable required to model a real- world phenomenon can increase to gargantuan proportion. This can be countered by using cluster graphs of which clique tree is special case.

In a cluster graph every node is a collection of nodes called clusters(C_i). Two nodes in a cluster graph are connected if they have a common node that is common among them. The intersection of two cluster $C_i \cap C_j$ is called a sepset (S_{ij}).

For example, given the following Markov network:

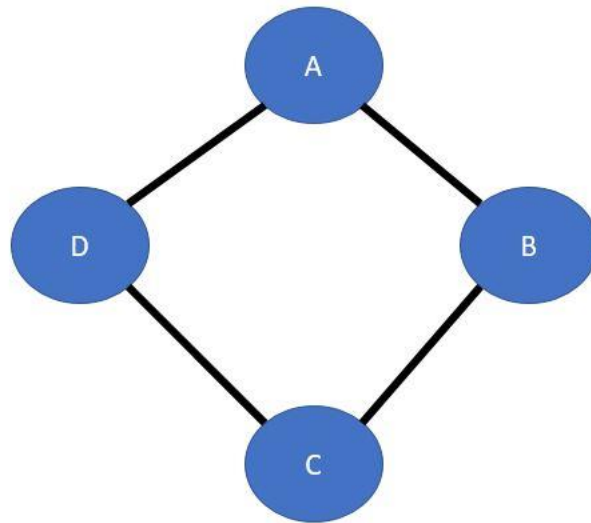


Figure-4: Example of a Markov Network

The equivalent Cluster graph structure will be:

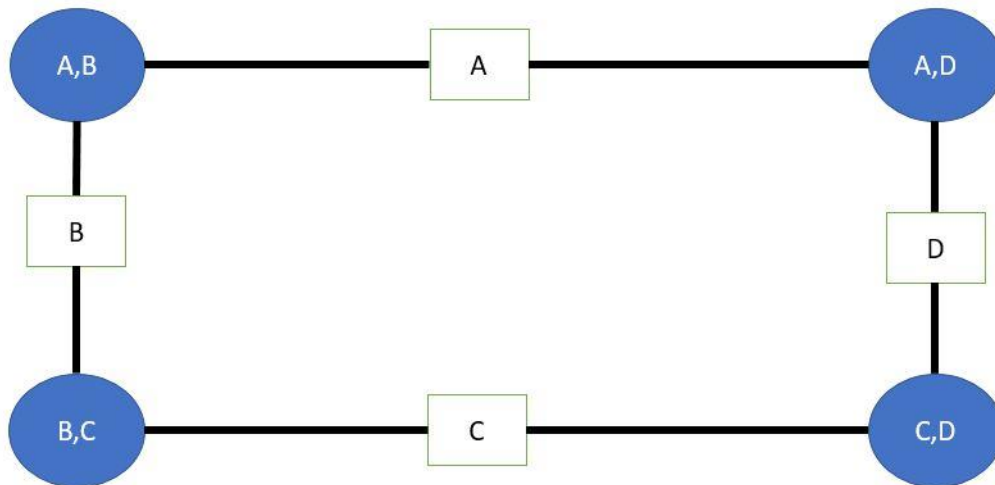


Figure 5: Equivalent cluster graph

The belief propagation algorithm discussed in this report is based on a cluster graph.

III. BELIEF PROPOGATION

Given any Bayesian or Markov network it is converted to the equivalent cluster graph.

Considering the cluster graph in figure-5.

First step is the initialization, this step initializes the beliefs of each node. For example, for figure 5 the initialization is as follows:

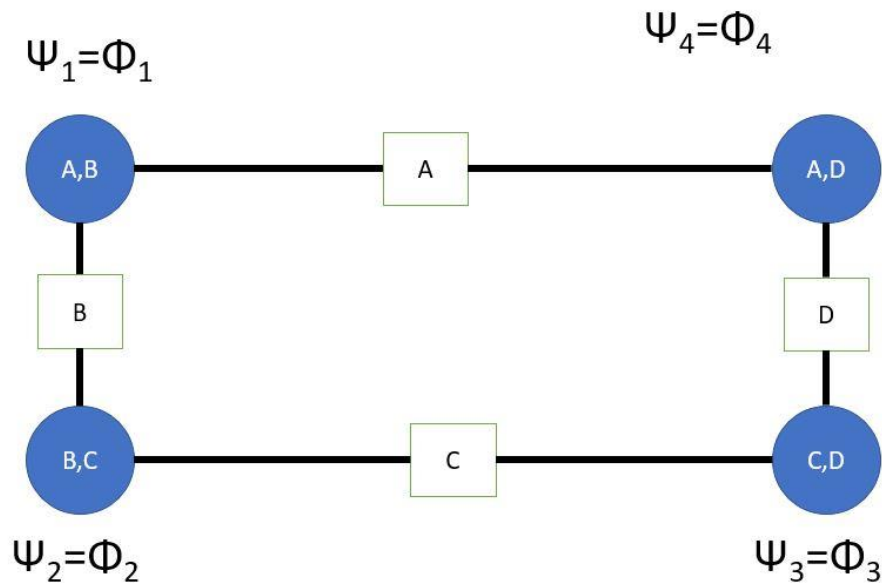



Figure 6: Assignment of initial beliefs

Once the initialization of each node is done then the messages that is to exchange between the nodes are calculated.

Say edge B is picked and the message sent by it from Cluster(B,C) to C(A, ) is set to 1. Once cluster A,B receives the message it computes the message transmitted by it cluster A,D.

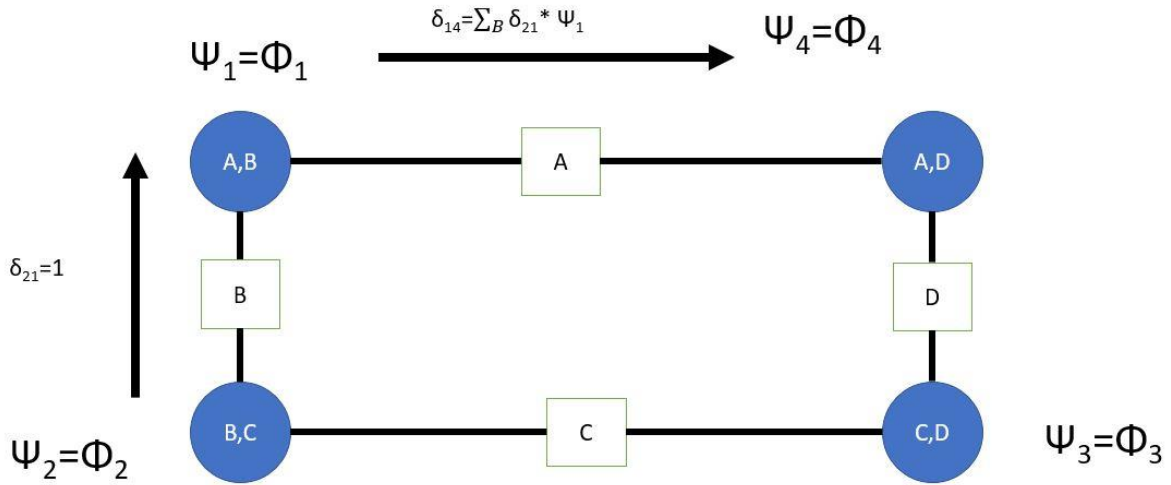


Figure 7: Message passing

Computing the message between each will look like this:

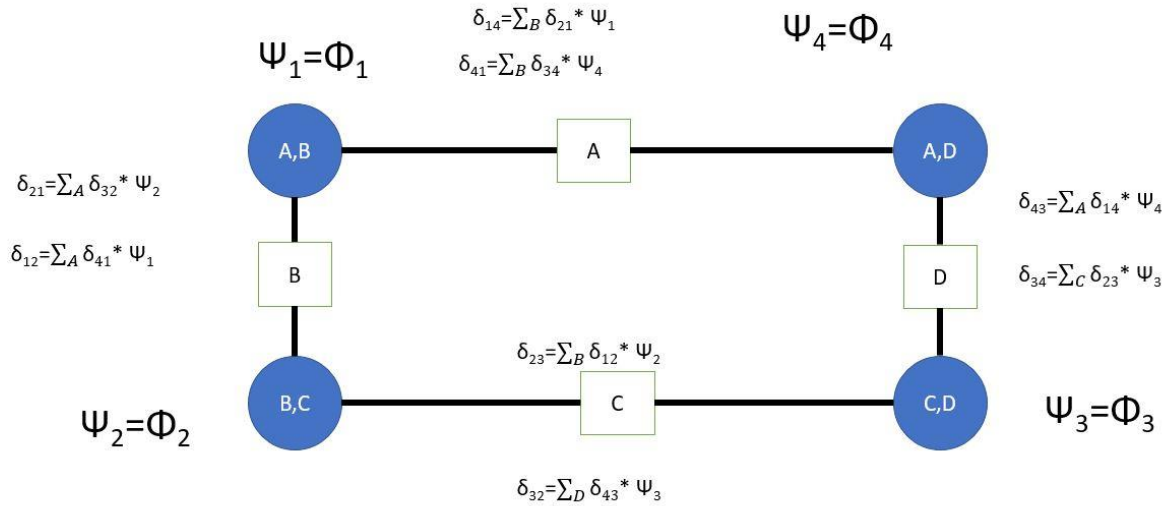


Figure 8: Overall expression for message passing

Once the message is over each edge is calculated the final belief can be computed easily.

2.THE ALGORITHM:

2.1 SETTING UP THE INTIAL BELEIFS:

- Given a set of factors(beliefs) to start of with.
- Each factor is assigned to one cluster such that $\text{scope}(\Phi_k)$ is a subset of the cluster.
- Then the initial set of beliefs of the cluster is given by:

$$\Psi_i = \prod_{\Phi_i} \Phi_i \quad \text{where } \Phi_i \text{ are the factors assigned to that cluster.}$$

For example,

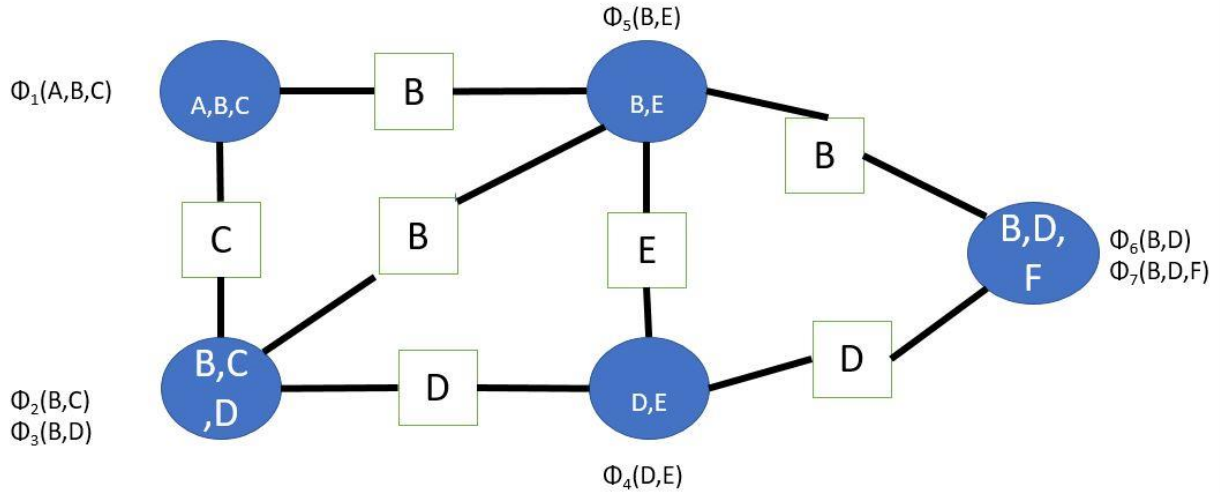


Figure 9: Assignment of factors to each cluster based on scopes

2.2 MESSAGE PASSING:

- Initialize all the messages to 1 to start.
- Selected an edge i,j and pass the message
- The message that is to be exchanged between the clusters are given by:

$$\delta_{i \rightarrow j}(S_{ij}) = \sum_{C_i - S_{ij}} \Psi_i * \prod_{k=N_i - \{j\}} \delta_{k \rightarrow i}$$

where $\delta_{i \rightarrow j}$ representing the message from cluster C_i to C_j , Ψ_i represents the initial beliefs.

- The message passing procedure is repeated until the each value message converges to a constant value or stopped after a fixed number of iterations.

2.2 COMPUTING THE NEW BELIEFS:

- $\beta_i(C_i) = \Psi_i * \prod_{k=N_i} \delta_{k \rightarrow i}$
- Once the beliefs β_i are computed, the individual probabilities can be calculated by summing over the other variable not of interest in the cluster.

Results have shown that running the belief propagation over a cluster graph might not lead to convergence of message signals in all the cases. Rather, it is preferable to use the algorithm over a subset of cluster graph called clique trees. In case of a clique tree the message passing should start from the leaf node and proceed along its parent. In the case of a clique tree the resulting marginal obtained for each random variable is exact unlike in a cluster graph where the marginals obtained are approximate in nature. So, it is preferable to use networks that can be modelled as a clique tree.

IV. USE CASES

A. MOVIE RECOMMENDATION SYSTEM:

- Objective: Based on user preferences, the system can suggest movies to the user.
- Input: Various preferences of the user are collected by the system
- Output: Various features of the movies are extracted and are correlated to derive recommendations. So, probability of a movie being liked by a certain user is evaluated based on the user preferences.

B. Illness Diagnosis:

- Objective: Predict the probability of an illness given the symptoms and other characteristics of a patient, like gender and age.
- Input: The doctor collects basic information regarding the patient and his health conditions.
- Output: The system evaluates all the characteristics and gives probabilities for various diseases.

V. A SUDOKU SOLVER

Sudoku puzzles are a standard puzzle where cells are arranged in nine rows and nine columns. Cells must be filled with numbers one to nine, and at the beginning of the game we are given certain *givens* already filled with numbers. We aim to fill the remaining cells in such a way that each row, column and 3x3 block contains the numbers 1 to 9.

Sudoku gives us a lot of interesting mathematical problems, including, but not limited to, how many possible puzzles exist, how many minimum givens we need to get a unique solution and how we can solve a sudoku.

We look at solving Sudoku's using the sum product algorithm on a graphical interpretation of Sudoku. Other sudoku solvers exist, including a systematic form of trial and error with backtracking, or by mapping it to the satisfiability problem[2].

Belief Propagation

Message passing is used extensively in other different areas, including artificial intelligence [3] and signal processing [4].

In this case, we represent sudoku puzzles as bipartite graphs. A bipartite graph is an example of a graph, where vertices can be divided into two disjoint sets. In this case, we divide the graph into two sets of sets, one containing the variable nodes and the other set containing the check nodes. Thus, in our case, we can have two disjoint sets **S** and **C**, such that **S** contains all the $N \times N$ cells in the Sudoku grid, and **C** contains all the possible constraints, which as $N \times 3$ in number.

Constraints

We represent our constraints by $C_m \in C$, $m \in \{1, 2, 3 \dots 27\}$. Where $C_m = \{0, 1\}$. It would be a indicator, which represents the contents of cell n by $S_n \in \{1, 2, \dots 9\}$, and the numbers from 1 to 9 are candidates for cell $n \in \{1, 2, 3 \dots 81\}$. Each cell is a part of three constraints, as demonstrated in the following figures.

A constraint function is defined as [5], where

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

Associating a bipartite graph with a Sudoku puzzle for a 9×9 sudoku puzzle gives us 27 checks, which are the 27 checks C_m . The figure below diagrams such a graph, which is also called a Tanner graph. Based on structure alone, we will later see that the Sudoku solver is very much similar to the LDPC belief propagation algorithm and graph.

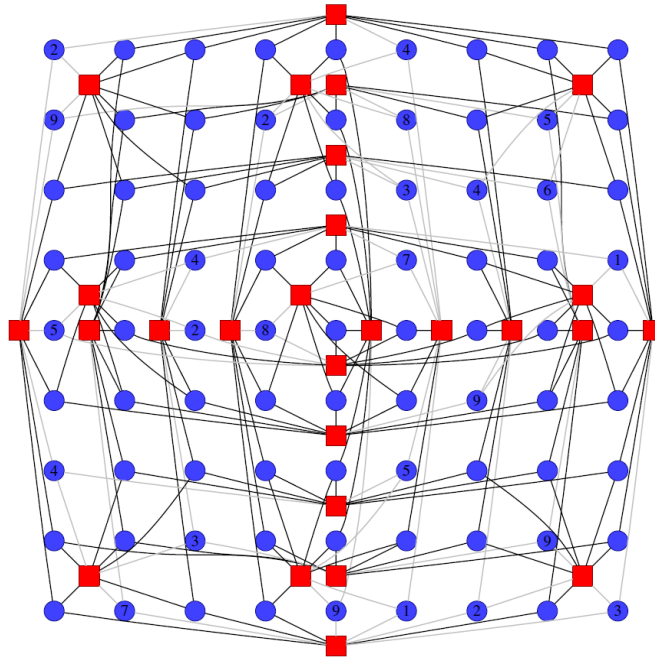


Figure 3 A graph representation of a Sudoku Puzzle. Blue circles are variables nodes or cells. The Red Squares are Checks

Before message passing begins, we can solve the step of elimination. If we were to solve it ourselves, we would fill in the cells which have only one possible value.

	C10	C11	C12	C13	C14	C15	C16	C17	C18
C1				7		8			
C2			6					3	1
C3		4			2				
C4		2	4		7				
C5			1		3			8	
C6				6			2	9	
C7				8				7	
C8	8	6					5		
C9			2		6				

Figure 4 A partially filled Sudoku puzzle

Algorithm:

First, form initial beliefs. In Figure 2 , we can perform message passing starting with either constraint to cell messages or cell to constraints. [6]Taking cell S1 as an example, we see a row check would reveal that it can't contain either 7 or 8, a column check would reveal that it can contain an 8, and a 3x3 cell check would confirm that it cannot contain either a four or a 6.

Therefore, assuming our starting cell as S₁, we get

$M1=\{1,10,19\}$

$r_{1,1}=[1/7\ 1/7\ 1/7\ 1/7\ 1/7\ 1/7\ 0\ 0\ 1/7]$

$r_{10,1}=[1/8\ 1/8\ 1/8\ 1/8\ 1/8\ 1/8\ 1/8\ 0\ 1/8]$

$r_{19,1}=[1/7\ 1/7\ 1/7\ 0\ 1/7\ 0\ 1/7\ 1/7\ 1/7]$, based on what the check nodes know about the contents of the cells.

And we can calculate $q_{1,1}$ as $[1/6\ 1/6\ 1/6\ 0\ 1/6\ 0\ 1/6\ 0\ 1/6]$.

Continuing on, we can get a cyclic structure of the graph, which goes like either of :

1. cell \rightarrow row constraint \rightarrow cell on row \rightarrow box constraint \rightarrow cell
2. cell \rightarrow column constraint \rightarrow cell on column \rightarrow box constraint \rightarrow cell,

Termination

As we can see in the figure below, the message passing solver is not able to solve all possible sudokus. This is because:

- All Sudoku's are not implicitly solvable. A lower bound of 17 cells are needed to have a unique solution. [7]
- Certain combinations have stopping sets or sets which are indeterminate.
- However, if we allow for restarts, most Sudoku's are solvable.

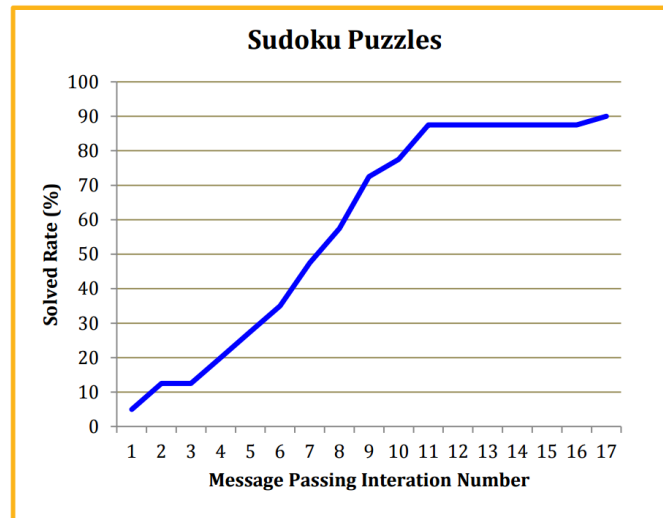


Figure 5 Experimental results for the Message Passing Solver

V. REFERENCES

- [1] S. Aksoy, "Probabilistic Graphical Models Part III: Example Applications."
- [2] I. Lynce and J. Ouaknine, "Sudoku as a SAT Problem," *Symp. A Q. J. Mod. Foreign Lit.*, pp. 1–9, 2006.

- [3] K. B. Korb and A. E. Nicholson, *Bayesian artificial intelligence*. CRC press, 2010.
- [4] S. J. Johnson, “Introducing low-density parity-check codes,” *Univ. Newcastle, Aust.*, 2006.
- [5] T. K. Moon and J. H. Gunther, “Multiple constraint satisfaction by belief propagation: An example using Sudoku,” in *2006 IEEE Mountain Workshop on Adaptive and Learning Systems, SMCals 2006*, 2006, pp. 122–126.
- [6] F. Wu, “Using Probabilistic Graphical Models to Solve NP-complete Puzzle Problems,” 2015.
- [7] A. Forrow and J. R. Schmitt, “Approaching the minimum number of clues Sudoku problem via the polynomial method.” submitted, 2013.