Survey Propagation: Iterative Solutions to Constraint Satisfaction Problems

Constantine Caramanis*

September 28, 2003

Abstract

Iterative algorithms, such as the well known Belief Propagation algorithm, have had much success in solving problems in statistical inference and coding and information theory. Survey Propagation attempts to apply iterative message passing algorithms to solve difficult combinatorial problems, in particular constraint satisfaction problems such as k-SAT and coloring problems. Intuition from statistical physics, involving evidence of phase transitions and clustering phenomena in the solution space, motivate some key modifications to well-known message passing algorithms, to yield effective tools for large instances of constraint satisfaction problems. The main algorithm, Survey Propagation, is motivated and developed, and then realized as a Belief Propagation algorithm, with an addition of a joker state.

1 Introduction

At the core of the difficult optimization and combinatorial problems, and in particular of NP-hard problems, is a non-local quality which stymies methods like local searches and descents, which have such success in convex problems. One of the most well known NP-complete problems, one might say the archetypal NP-complete problem, is 3-SAT. The problem here is, given a boolean formula in N variables, that is a conjunction of M clauses, each of which is a disjunction of three literals, to find an assignment of the variables in $\{T, F\}$ that satisfies all the clauses. To make things concrete, consider the following two examples:

- $(x_1 \vee x_4 \vee \tilde{x}_5) \wedge (\tilde{x}_2 \vee \tilde{x}_3 \vee \tilde{x}_4) \wedge (\tilde{x}_1 \vee \tilde{x}_4 \vee x_3) \wedge (\tilde{x}_3 \vee \tilde{x}_4 \vee \tilde{x}_5) \wedge (\tilde{x}_1 \vee x_4 \vee x_2) \wedge (\tilde{x}_1 \vee \tilde{x}_2 \vee x_3)$
- $(x_1 \vee \tilde{x}_2 \vee x_3) \wedge (\tilde{x}_3 \vee \tilde{x}_4 \vee x_5) \wedge (x_5 \vee \tilde{x}_6 \vee \tilde{x}_7) \wedge (x_7 \vee x_8 \vee x_9) \wedge (\tilde{x}_9 \vee x_{10} \vee x_{11}) \wedge (\tilde{x}_{11} \vee \tilde{x}_{12} \vee \tilde{x}_{13})$.

The reader might enjoy trying to find a few satisfying assignments for the above two problems. While neither is particularly difficult, the second, even with more variables, is without doubt the easier of the two to solve. While these are quite small examples, they nevertheless underscore some important themes. The second example above is particularly easy because variables appear, and then do not reappear later on. When assigning values, the problem kindly allows us to focus our attention locally – we are not compelled to check the fifth or sixth clause, when we are trying to satisfy the first or second. In addition, because of the plethora of variables, there are many readily available satisfying assignments, many of which are only single bit-flips away from each other. The first example, on the other hand, does not enjoy this acyclic property, and when assigning the variables in the first two clauses, we need to make sure we do not rule out the satisfiability of, say, the fifth or sixth clauses. In addition, the variables are more tightly related. For example, if we select $x_1 = T = x_2$, then we must set $x_3 = T$, and consequently $x_4 = F$. Had we chosen $x_2 = F$, we would have been forced to assign $x_4 = F$.

*cmcaram@mit.edu

There are two main points illustated here. First, the cyclic nature of the first example (for instance, x_1 and x_5 are linked in the first clause, x_5 and x_4 in the fourth, x_4 and x_2 in the fifth, and x_2 and x_1 in the sixth) makes finding a solution more difficult. Second, the many variables in the second instance above, actually make the problem easier, since different satisfying assignments are "closer" together, in the sense of hamming distance.

Indeed, constraint satisfaction problems, such as the satisfiability problem above, or others, such as graph coloring problems, or decoding corrupted codewords, illustrate a *fundamental global* nature, which is intimately related with their reluctance to yield to efficient solution methods. These computational problems are also closely related to the problem in statistical physics of computing ground state configurations of spin-glasses subjected to magnetic fields. For more on this see [**?**] and the references therein. Of particular interest is the behavior of randomly generated instances of 3-SAT, where the number of variables N goes to infinity, while the number of clauses, M, is kept in constant proportion to the variables: $\alpha = M/N$. The behavior is analyzed in the asymptotic regime, while the parameter α is varied. It has long been observed that for certain ranges of values of α, problems may be easy, while in other ranges of α, problems may be difficult to solve, and in fact in different regimes, solutions may exhibit rather diverse characteristics. Some recent progress in statistical physics has shed light on phase transition phenomena in the solution space of some of these problems. Intuition from these results has been used to develop iterative algorithms akin to the well-known message passing algorithm (and its variants) belief propagation (BP) from statistical inference, to solve random instances of certain constraint satisfaction problems. Survey Propagation (SP) is an iterative, message passing algorithm that is at its core, a belief propagation algorithm.

In this paper we describe survey propagation, and develop it in the context of the belief propagation algorithm. In Section 2 below we describe the physical phenomenon of clustering. This is a phase transition phenomenon, which at the same time illustrates the difficulties that local search methods face, and also motivates possible ways to improve and work around their inherent problems. In short, this section motivates the survey propagation algorithm. Section 3 is the bulk of this paper. First, in 3.2 we develop the belief propagation message passing algorithm in the context of constraint satisfaction problems, specifically the 3-SAT problem. Then in Section 3.3 we show how one can generalize the belief propagation algorithm in order to obtain the survey propagation algorithm. Then in Section 3.4 we rederive the survey propagation algorithm. First, we develop a warning propagation algorithm, then from this we obtain the survey propagation algorithm. Finally we show that this is no more than belief propagation, with the addition of an extra state. In Section 4 we bring up some of the issues that remain unclear, to the general community, but in particular to the author, and thus conclude.

2 Some Physical Intuition

In this section we describe the physical situation in the solution space. This gives an intuitive idea first of why general local algorithms run into trouble, and also motivates the Survey Propagation algorithm.

Recall that in all this discussion, we are assuming that the number of variables, N, is very large, while the number of constraints, or clauses in the case of 3-SAT, M, is always determined by the parameter α, where

$$\alpha = M/N.$$

To generate our random instances, the variables that appear in each clause are chosen from a uniform distribution, and each variable appears negated or unnegated with equal probability.

We are interested in understanding the asymptotic behavior when α is kept fixed, and N goes to infinity. The actual experiments have been run with $N \sim 10^7$ variables.

As α increases, we observe a phase transition around a critical value $\alpha = \alpha_c$, where α_c has been observed to be close to 4.27. For $\alpha < \alpha_c$ most problems are satisfiable and are generally underconstrained. Indeed in this region, with probability one we generate an instance that is satisfiable.

This is called the SAT region. In addition, when α is far from the critical value α_c, it is generally easy to find a satisfying assigment. When alpha increases and $\alpha > \alpha_c$, we are in the UNSAT region, where with probability one random instances have no satisfying assignments. The picture we have described is given in figure 1. Consider the second example given above, where we had six clauses

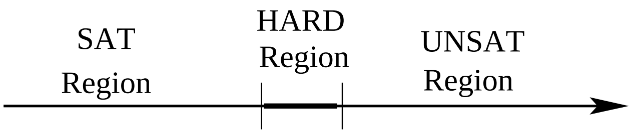

Figure 1: A picture of the phase transitions for α.

and 13 variables. One of its characteristics is the plethora of satisfying assignments. In addition, given any two satisfying assignments, it is possible to move from one to the other, in a sequence of steps where each is a satisfying assignment, and very few "bits" need to be flipped in any given step. This turns out to be in general true.

Indeed, experiments have been done to determine what happens in the solution space, as α approaches the critical value α_c. For small α, as mentioned, the generated instances are under-constrained and problems are satisfiable. Moreover, in the solution space, the satisfiable solutions form one big cluster, where one can move from one solution to another along a path consisting of satisfiable assignments, and where each move requires only few "bit flips". This structure of the solution space provides fertile ground for local search algorithms that can focus on different clauses sequentially, and thus obtain a satisfying assignment.

As α approaches the critical value α_c, the cluster of solutions breaks up into many small clusters that are far from each other in distance. In each cluster, some of the variables are frozen to some value, while the others are free to vary. This property will prove crucial later, when in trying to locate a satisfying assignment of the variables, we will essentially be trying to find the values of the frozen variables (the "address") of one of the clusters of satisfying assignments.

Further complicating matters, in addition to the multiple separated clusters, there are exponentially more clusters of partial solutions. These partial solutions are pitfalls for local search algorithms, since many clauses are satisfied, but there is no satisfying assignment within a distance that can be traversed by any local algorithm. In other words, as we have when we are in a local minimum, in order to escape, we may need to deteriorate our solution. In this case, this would mean that any path leading to a satisfying assignment would have to travel through assignments with many more unsatisfied clauses than we might have in one of the clusters of partial solutions. In figure 2 we have an "artist's" depiction of this picture.

3 Iterative Methods

Iterative message passing methods have had much success in statistical inference, and information theory and specifically efficient decoding of parity check codes. Message passing algorithms work on problems where many variables (or agents, or atoms, etc...) interact because of local constraints. For these classes of problems it is convenient to use the formalism of graphical models in order to conveniently represent these local interactions. In a graphical model, the nodes represent the variables (agents, atoms, etc.) and the edges are drawn such that nodes A are independent of nodes B when conditioned on nodes C if and only if any path from nodes A to nodes B must pass through one of the nodes of C. This is illustrated in figure 3. Message passing algorithms, then, pass messages along edges of the graph in the graphical model. Intuitively speaking, node v collects information from its neighbors, and then broadcasts to its neighbors based on this information. This process is iterated until the messages that are sent and received converge to some steady values. One of the most important ideas is that if the graph is a tree, or singly connected, then the messages that node v receives are independent of each other. If the graph has cycles, on the other hand, then what node v receives from nodes w_1, w_2 ar some iteration may be correlated because of what w_1 may have sent

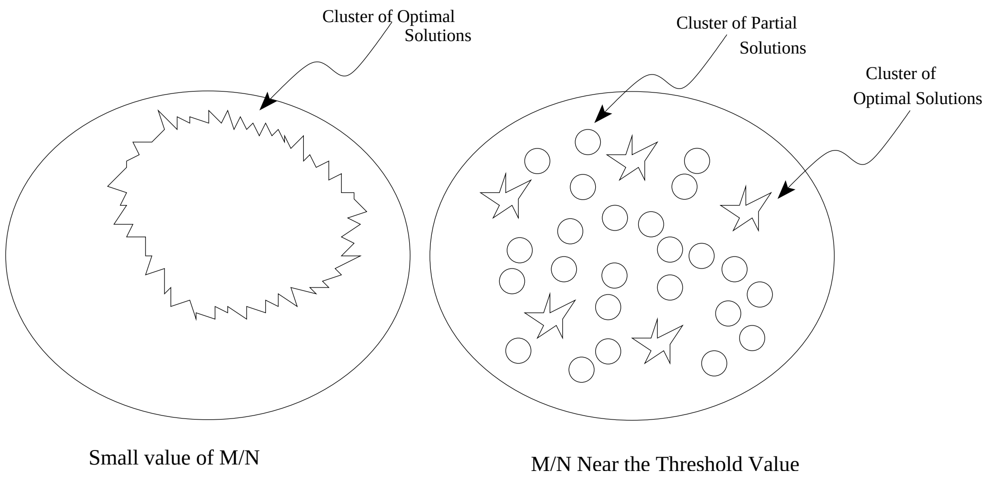

Figure 2: A picture of the phase transitions for α much smaller than α_c, and then for α in the hard region, close to α_c. We see the two salient features are first the breaking up of the single big cluster, and second, the presence of the many clusters of partial solution.

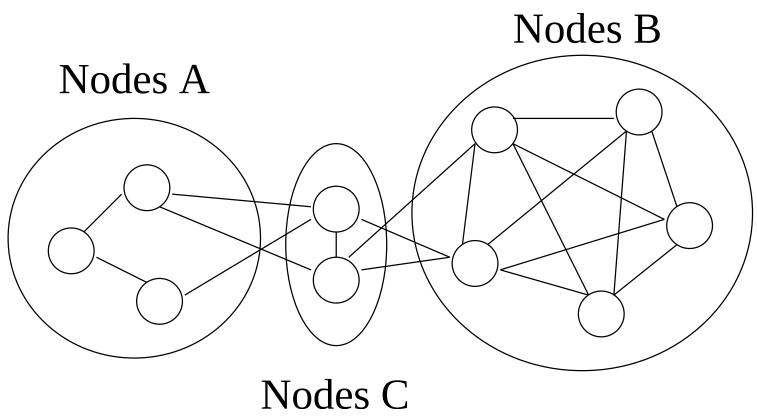

Figure 3: Nodes A and B are independent when conditioned on C.

to w_2 along some path, in previous iterations. It turns out that for this reason, graphs with cycles often resist solution by message passing algorithms, causing message passing algorithms either to fail to converge, or to converge to the wrong answer. On graphs with tree-structure, however, message passing algorithms converge quickly, and are (provably) exact.

The difficulty of solving problems on cyclic graphs is merely another expression of the fact that problems that are somehow inherently global, are difficult to solve. In order for dependencies in the received messages to be properly untangled, the node must have observed the message passing more globally than merely observing its nearest neighbors.

Recently, message passing algorithms have been generalized to work on hypergraphs, passing messages from one clique to another, along hyperedges. For more on this, see [8].

Also, there are many equivalent ways to formulate the graphical model formalism. For more on this, see [1].

3.1 Graphical Model of 3-SAT

In this section we discuss the graphical model formulation of the 3-SAT problem. We use the so-called factor graph representation. We construct our graph as follows: For every variable, we have a variable node, and for every clause we have a function node. We connect variable node i to function node j if and only if variable i appears in clause j. Also we need to keep track of whether the variables appear negated or unnegated. We do this by using two types of edges. We label J_a^i the edge from clause a to node i. If variable i appears unnegated, then $J_a^i = -1$, otherwise we set $J_a^i = +1$. We let $V(i)$ denote the function nodes attached to i by an edge, denoting by $V_-(i)$ and $V_+(i)$ the function nodes where i appears negated and unnegated respectively. Similarly we let $V(a) = V_-(a) \cup V_+(a)$ denote the variable nodes that appear in clause a. Also, in order to unify notation, Braunstein et al. introduce the sets $V_a^s(j)$ and $V_a^u(j)$. Suppose j appears negated in clause a. Then if it appears unnegated in clause b, say, then we can say that clause b is encouraging j to not satisfy clause a. It is these types of clauses that make up $V_a^u(j)$. Meanwhile, if variable j appears negated in clause c, then we can say that clause c is encouraging j to satisfy clause a, and hence $c \in V_a^s(j)$. As an example, figure 4 below shows the two factor graphs for our two examples

- $(x_1 \vee x_4 \vee \tilde{x}_5) \wedge (\tilde{x}_2 \vee \tilde{x}_3 \vee \tilde{x}_4) \wedge (\tilde{x}_1 \vee \tilde{x}_4 \vee x_3) \wedge (\tilde{x}_3 \vee \tilde{x}_4 \vee \tilde{x}_5) \wedge (\tilde{x}_1 \vee x_4 \vee x_2) \wedge (\tilde{x}_1 \vee \tilde{x}_2 \vee x_3)$
- $(x_1 \vee \tilde{x}_2 \vee x_3) \wedge (\tilde{x}_3 \vee \tilde{x}_4 \vee x_5) \wedge (x_5 \vee \tilde{x}_6 \vee \tilde{x}_7) \wedge (x_7 \vee x_8 \vee x_9) \wedge (\tilde{x}_9 \vee x_{10} \vee x_{11}) \wedge (\tilde{x}_{11} \vee \tilde{x}_{12} \vee \tilde{x}_{13})$.

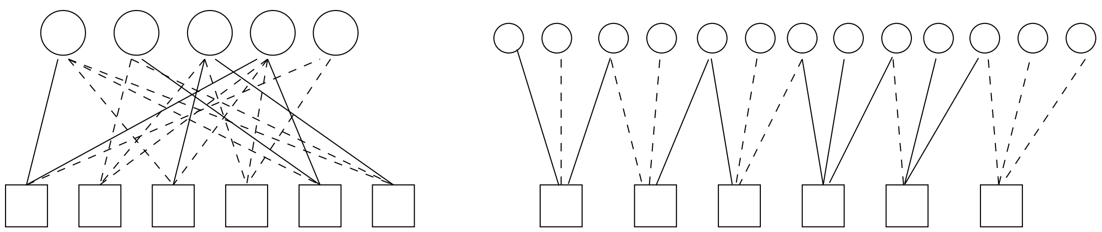

Figure 4: Here we show the factor graph representation of our two examples. Dashed lines indicate that the variable appears negated in the particular clause. The graphical model itself shows that our second example, even though it has more than twice the variables, it has a much more simple, and more local nature. In the factor graph representation, this local property is expressed in its lack of cycles.

3.2 Belief Propagation

Belief Propagation (BP) is one of the most well known variants of message passing algorithms. In this algorithm, the variable nodes pass as messages the probability that they will assume state $\{0, 1\}$, (where we take 1 to be T and 0 to be F) and the function nodes broadcast the probability that they will be satisfied if their neighboring variable nodes take certain values. Belief Propagation is an inherently local algorithm. This is reflected in the manner in which the nodes update their messages after each iteration. Since each node has only knowledge of the messages it received and nothing more, it assumes that each of these messages is statistically independent of the others. Note that on a tree structured graph, this indeed would be the case.

The BP algorithm has been studied because of its successful application to statistical inference and coding. In [7], Yedidia et al. show that the inference problem can be cast as a minimization of a free energy. They go on to show that the BP algorithm solves exactly an approximation called the Bethe free energy. The Bethe approximation is exact on graphs that are tree-structured. In his thesis, Wainwright shows that BP can be interpreted as enforcing local single node and pairwise marginal constraints on the distributions of neighboring nodes, and that it then tries in effect to patch up these locally consistent distributions to form globally consistent distributions. For more on this, see [9].

3.2.1 The BP Details

In this section we develop the belief propagation equations for the factor graph representation of a 3-SAT problem. Recall that the main idea is that variable nodes pass on the probabilities that they will assume each possible value, and function nodes pass the probability that they will be satisfied should variable node i take value x_i. The messages are updated at each iteration under the assumption that the incoming messages are all statistically independent. The messages are:

(i) $\mu_{a\to i}(x_i)$ is the probability that clause a is satisfied if variable i is set to $x_i \in \{0,1\}$.

(ii) $\mu_{i\to a}(x_i)$ is the probability that variable i takes value x_i when clause a is ignored.

Function node a receives messages from its neighbors. Then to neighboring node i, it sends a message that it has updated using all its *new* information, that is, information that it received from nodes other than i. Similarly, variable node i sends neighboring function node a a message that it has updated based only on information received from neighoring function nodes other than a. We update these messages recursively using the following rules:

$$\begin{aligned}
\mu_{i\to a}(x_i) &= C_{i\to a}\cdot \prod_{b\in V(i)\setminus a} \mu_{b\to i}(x_i) \\
&= \text{The probability that the clauses other than } a \text{ will be satisfied} \\
&\quad\ \text{if variable } i \text{ takes value } x_i.\ C_{i\to a} \text{ is a normalization factor.} \\
\mu_{a\to i}(x_i) &= \sum_{\{x_j : j\neq i\}} f_a(X) \prod_{j\in V(a)\setminus i} \mu_{j\to a}(x_j) \\
&= \text{The sums of the probabilities of all satisfying assignments for clause} \\
&\quad\ a\text{, given that variable } i = x_i\text{, and also assuming the messages from} \\
&\quad\ \text{variables } j\in V(a)\setminus i \text{ are independent.}
\end{aligned}$$

In the last expression, the function $f_a(X)$ is an indicator function of the assignments that satisfy clause a, when $i = x_i$.

Next we parametrize $\mu_{i\to a}(x_i)$ by $\gamma_{i\to a}$, the probability that variable i takes state x_i that violates clause a, in a problem where clause a has been removed. Then,

$$\mu_{i\to a}(x_i) = \begin{cases} \gamma_{i\to a}\delta(x_i,0) + (1-\gamma_{i\to a}\delta(x_i,1) & J_i^a = -1, \\ \gamma_{i\to a}\delta(x_i,1) + (1-\gamma_{i\to a}\delta(x_i,0) & J_i^a = 1. \end{cases}$$

Next we write,

$$\begin{aligned}
\delta_{a\to i} &= \prod_{j\in V(a)\setminus i} \gamma_{j\to a} \\
&= \text{probability that all variables } j\neq i \text{ in clause } a \text{ violate it (assuming independence).} \\
P^u_{j\to a} &= \prod_{b\ in V^s_a(j)} (1-\delta_{b\to j}) \\
&= \text{probability that all clauses that would force } j \text{ to satisfy } a \text{ are already satisfied.} \\
P^s_{j\to a} &= \prod_{b\ in V^u_a(j)} (1-\delta_{b\to j}) \\
&= \text{probability that all the clauses that would force } j \text{ to violate } a \text{ are already satisfied.}
\end{aligned}$$

Now the probability of not satisfying clause a is the normalized probability that all the clauses that would force j to satisfy clause a are already satisfied. Thus we have our closed system of equations for belief propagation:

$$
\begin{aligned}
\gamma_{j\to a} &= \frac{P^u_{j\to a}}{P^u_{j\to a}+P^s_{j\to a}}, \\
P^u_{j\to a} &= \prod_{b\ inV^s_a(j)} (1-\delta_{b\to j}) \\
P^s_{j\to a} &= \prod_{b\ inV^u_a(j)} (1-\delta_{b\to j}).
\end{aligned}
$$

3.2.2 The Performance of BP

As we have mentioned, belief propagation is an efficient algorithm that is guaranteed to produce a satisfying assignment when the underlying factor graph is a tree. For large values of N (i.e. when the size of the problem is large) locally the factor graphs look very tree like. In other words, as N becomes large, we have only very long cycles. Therefore BP does a good job of finding local solutions, or partial solutions, that satisfy many of the clauses. Indeed when the ratio $\alpha = M/N$ is not in the hard region (i.e. it is not close to the critical value α_c) then BP typically succeeds in finding satisfiable assignments. Intuitively speaking, this is because the local solutions that BP may be obtaining at different parts of the graph, can be extended to full satisfying solutions, and furthermore, these solutions all belong to the same cluster. However, when α is close to the critical value α_c, and hence we are in the hard SAT region, the solution space breaks up into clusters (figure 2). Then the partial solutions to which the BP local computations converge, may belong to different solution clusters, or even worse, clusters of partial solutions. In these cases, the BP algorithm performs quite poorly. Indeed it is in these cases when BP will not, in general, converge. This physical phenomenon, and (intuitive) explanation of the failure of BP serves as a motivation for the introduction of the survey propagation algorithm.

the outputs here are probabilities that each variable is zero or one. the problem is that the local BP computations, because they are exactly that: local, may be refering to different clusters, and as a result BP may not converge.

3.3 Survey Propagation I

In the last section we outlined a rough intuition of why belief propagation works well in the region where the random SAT instances are underconstrained, yet breaks down in the hard region. This section uses this motivation to develop the Survey Propagation algorithm. We develop only the intuition in this section, and then save the equations and recursive update formulas for the next section, where we present a different development of survey propagation.

We saw above that BP essentially passes probabilities that variables will take the values 0 or 1. The problem is that these calculations may refer to different clusters in the solution space, and thus this may keep the BP algorithm from converging. Recall that in the multiple clusters of satisfying solutions that appear near the hard SAT region, some variables are frozen to particular values, while others are unfrozen and can vary. It is these frozen values that correspond to the "address" of the satisfying cluster. The main idea behind survey propagation is to try to identify the value of the frozen values corresponding to one of the satisfying clusters. As we identify frozen variables, the problem is decimated, and the algorithm rerun on the smaller problem until the problem is small enough to be solved exactly with some local search method. Indeed, as we fix values of the frozen variables, we revert to the picture which algorithms like BP can handle effectively: a single cluster of solutions, where satsifying assignments are close together.

Instead of passing probabilities that variables are either 1 or 0, and trying to converge to a delta function, we wish to consider, over all the satisfiable clusters, the distribution of the probabilities that the BP algorithm passes. In other words, we would like to consider passing more complicated messages: we wish to keep track of probabilities of probabilities – entire *surveys* of the BP outputs over all satisfiable clusters.

But this is actually more information than we need. The main objective is to identify frozen variables, and thus to single out one of the satisfying clusters in the solutions space. Therefore, rather than keep the survey of the entire probability distribution, we keep only a coarse version. We can quantize the possible messages passed by the variables in belief propagation, to one of three possibilities: either the variable is set to value 0 with probability 1, or it is set to value 1 with probability 1, or it is set to either zero or one with some nonzero probability.

The decimation process proceeds according to the output of the survey propagation algorithm. We freeze the node (or nodes) that has the highest probability of being assigned a 1 or 0 by the belief propagation algorithm.

We observe that this final algorithm can easily be recast as a belief propagation algorithm. The main difference now is that our variables do not just take two states, 0 or 1. Instead, since we are looking at a distribution of quantized distributions, our variables can take values: "frozen to 1" or "frozen to 0" or "unfrozen". Let us identify the delta function at 1 with the value 1, and similarly for 0. Next, we assigne a new "joker state" which we call $*$ to any nontrivial mixture of the delta functions. Then survey propagation passes messages which are probabilities that the variables take values in the set $\{0, 1, *\}$. In other words, we can realize survey propagation, as belief propagation on a problem with an augmented state space.

3.4 Survey Propagation II

In this section we give another derivation of survey propagation, and, as we did with belief propagation above, we explicitly develop the recursive update equations. In the previous section, we considered belief propagation, where the messages are probability distributions. Then from here we considered passing messages that are surveys of these distributions, and we then coarsened the surveys. In this section, rather than consider belief propagation, and then survey propagation as a coarsened generalization of belief propagation, we can do the coarsening first, and then consider belief propagation. We consider first a message passing algorithm called Warning Propagation (WP). In this algorithm, variable node pass messages which indicate which value they want to take, and function nodes pass messages which say which values the variable nodes should *not* take. Then we consider passing surveys of these coarse messages. The result is the same, and we retrieve the survey propagation equations.

3.4.1 Warning Propagation

In warning propagation, variable nodes indicate to the clauses which variables they would like to take, and the clauses in return tell the variable nodes which values they can not take. The messages passed are:

(i) $u_{a\to i}$: Clause a passes a 1 to node i if none of its other neighbors satisfy the clause.

(ii) $h_{j\to a}$: Variable node j tells node a what value it will take in order to satisfy the greatest number of its neighboring clauses.

The updating rules are given by

$$\begin{aligned} h_{j\to a} &= \left(\sum_{b\in V_+(j)\setminus a} u_{b\to j}\right) - \left(\sum_{b\in V_-(j)\setminus a} u_{b\to j}\right) \\ u_{a\to i} &= \prod_{j\in V(a)\setminus i} \theta(h_{j\to a} J_j^a), \end{aligned}$$

where θ is the indicator function of the strictly positive numbers.

We note that warning propagation may be considered as a coarse version of belief propagation. Variable nodes tell function nodes if they must take a certain value in order to satisfy some otherwise unsatisfiable function nodes. Note that if they do not need to assume some value in order to satisfy a particular neighboring clause, then they do not output a probability. Then the function nodes, or the clauses, report to the variable nodes if they must be frozen to some state in order to satisfy them. If a clause is satisfied by variable j, then to all its other neighbors it sends no warnings, allowing them to remain unfrozen.

It is important to remark that like belief propagation, warning propagation is exact on tree-structured factor graphs. The arguments that show this are quite similar to those for BP.

3.4.2 Survey Propagation

Much like belief propagation, warning propagation works well for tree structured graphs, and also in the region where α is far from α_c, where the random instances are underconstrained, and the set of satisfiable assignments is connected. To adapt the algorithm to the hard region, we want to take surveys of the warning propagation over all satisfiable clusters. We let $\mathcal{N}$ be the number of clusters of satisfying solutions. Then we can write down the "survey" of the messages from warning propagation, over all the clusters . Let $Q_{a\to i}(u)$ denote the survey over all clusters of the variables $u_{a\to i}$. More generally, if U is some collection of such variables, let $Q(U)$ denote its survey. Then we have:

$$Q_{a\to i}(u) = \frac{1}{\mathcal{N}} \sum \delta(u, u_{a\to i}),$$

where $u_{a\to i}$ is the message $u_{a\to i}$ in cluster . Since u can take only two possible values, we can parameterize Q by a single number: the fraction of clusters where $u_{a\to i} = 1$. Call this value $\eta_{a\to i}$. Then we have

$$Q_{a\to i}(u) = (1 - \eta_{a\to i})\delta(u, 0) + \eta_{a\to i}\delta(u, 1).$$

As we had for warning propagation, and for belief propagation, we want a set of recursive closed equations which define the updates of the survey propagation messages. Towards this end, we note that

$$\begin{aligned}
\eta_{a\to i} &= \text{fraction of clusters where } u_{a\to i} = 1 \\
&= \text{probability that } u_{a\to i} = 1 \\
&= \text{fraction of clusters where } h_{j\to a} J_j^a > 0,\ \forall j \in V(a) \setminus i \\
&= \text{probability that } \forall j \in V(a) \setminus i, \text{ we have } h_{j\to a} J_j^a > 0.
\end{aligned}$$

Now, let $\rho_{j\to a}$ denote the probability that $h_{j\to a} J_j^a > 0$. If $J_j^a = 1$, then

$$\begin{aligned}
\rho_{j\to a} &= P(h_{j\to a} > 0) \\
&= P\left(\left(\sum_{V_+(j)\setminus a} u_{b\to j} - \sum_{V_-(j)\setminus a} u_{b\to j}\right) > 0\right) \\
&= P\left(\sum_{V_+(j)\setminus a} u_{b\to j} > 0\right) \\
&= \sum_U Q(U)\,\theta\left(\sum_{V_+(j)\setminus a} u_{b\to j}\right).
\end{aligned}$$

In the above, the second to last equality holds because we are taking surveys only over the satisfying clusters. This allows us to assume that we do not have constradictions. In other words, if some clauses are telling variable j that it must assume, say, value $+1$, then there cannot be any clauses informing j that it must do the contrary. Here $U = \{u_{b\to j}\}$ and $Q(U)$ is the survey. Note that in the above, if $J_j^a = -1$, then V_+ and V_- switch roles. Recall again that in order to unify notation, we have the sets $V_a^s(j)$ and $V_a^u(j)$ of function nodes that encourage or discourage variable j to satisfy node a, respectively. So, if variable j appears negated in clause c, then we can say that clause c is encouraging j to satisfy clause a, and hence $c \in V_a^s(j)$. Thus, regardless of the sign of J_j^a, we can write

$$\rho_{j\to a} = \sum_U Q(U)\,\theta\left(\sum_{b\in V_a^u(j)\setminus a} u_{b\to j}\right).$$

In order to obtain a system of closed equations, we assume (as would be true on any tree structured distribution) that the incoming messages to a function node are independent, and thus their joint distribution factorizes.

Thus we can write:

$$\begin{aligned}
\rho_{j\to a} &= \sum_U Q(U)\,\theta\left(\sum_{b\in V_a^u(j)} u_{b\to j}\right) \\
&= C_{j\to a} \sum_{\{u_{b\to j}\}} \left[\prod_{b\ inV(j)\setminus a} Q_{b\to j}(u_{b\to j})\right] \theta\left(\sum_{b\in V_a^u(j)} u_{b\to j}\right) \prod_{b\in V_a^s(j)} \delta(u_{b\to j},0).
\end{aligned}$$

This last equation needs some explaining. The factorization of the survey $Q(U)$ occurs thanks to our independence assumption. However, this comes at a cost, since it is precisely this correslation across several variable and function nodes that guarantees that we will not receive contradictory messages. Instead, we need to implement this constraint explicitly in the final product in the above expression. Because we introduce this, we also need to renormalize, since we want this to be a probability.

Let us consider the possible non-contradictory messages that SP could pass. For these, if

$$\sum_{b\in V_a^u(j)} u_{b\to j} > 0,$$

then variable j gets at least one warning that it *must* take a value that will not satisfy clause a, in order to ensure that some other clause is satisfied. In this situation, we do not want variable j to receive a message from a function node (other than a) demanding that it take the opposite value, as this would be a contradiction. Thus we have three cases:

(i) We have $\sum_{b\in V_a^u(j)} u_{b\to j} > 0$ and j must be fixed at a value that does not satisfy a,

(ii) $\sum_{b\in V_a^s(j)} u_{b\to j} > 0$ and j must be fixed at a value that satisfies a,

(iii) We have $u_{b\to j} = 0$ for all $b \in V_a(j)$.

Calling these classes $u, s, 0$ respectively, we can write down the probabilities of each class:

$$\begin{aligned}
\Pi^u_{j\to a} &= \sum_{\{u_{b\to j}\}} \left[\prod_{b\in V(j)\setminus a} Q_{b\to j}(u_{b\to j})\right] \theta\left(\sum_{b\in V_a^u(j)} u_{b\to j}\right) \prod_{b\in V_a^s(j)} \delta(u_{b\to j},0) \\
\Pi^s_{j\to a} &= \sum_{\{u_{b\to j}\}} \left[\prod_{b\in V(j)\setminus a} Q_{b\to j}(u_{b\to j})\right] \theta\left(\sum_{b\in V_a^s(j)} u_{b\to j}\right) \prod_{b\in V_a^u(j)} \delta(u_{b\to j},0) \\
\Pi^0_{j\to a} &= \sum_{\{u_{b\to j}\}} \left[\prod_{b\in V(j)\setminus a} Q_{b\to j}(u_{b\to j})\right] \left[\prod_{b\in V(j)\setminus a} \delta(u_{b\to j},0)\right].
\end{aligned}$$

This gives us the normalization factor above, as

$$C_{j\to a} = (\Pi^u_{j\to a} + \Pi^u_{j\to s} + \Pi^0_{j\to a})^{-1}.$$

Thus we have a closed family of equations for the parameters of our surveys, $\eta_{a\to i}$:

$$\begin{aligned}
\eta_{a\to i} &= \prod_{j\in V(a)\backslash a}\left[\frac{\Pi^u_{j\to a}}{\Pi^u_{j\to a}+\Pi^s_{j\to a}+\Pi^0_{j\to a}}\right]\\
\Pi^u_{j\to a} &= \left[1-\prod_{b\in V^u_a(j)}(1-\eta_{b\to j})\right]\prod_{b\in V^s_a(j)}(1-\eta_{b\to j})\\
\Pi^s_{j\to a} &= \left[1-\prod_{b\in V^s_a(j)}(1-\eta_{b\to j})\right]\prod_{b\in V^u_a(j)}(1-\eta_{b\to j})\\
\Pi^0_{j\to a} &= \prod_{b\in V(j)\backslash a}(1-\eta_{b\to j}).
\end{aligned}$$

These four equations define our update mechanism for survey propagation. The first parameter, $\eta_{a\to i}$, is the probability that node i receives a warning from function node j. Meanwhile, the three probabilities $\{\Pi^u_{j\to a},\Pi^s_{j\to a},\Pi^0_{j\to a}\}$, are the probabilities that variable node j is constrained to something that will not satisfy a, or to something that will satisfy a, or to nothing at all, respectively.

3.4.3 Decimation Process

We have given above a set of closed equations to provide updates for the messages of survey propagation. Provided that the algorithm terminates having converged to a fixed point of the above update mechanism, we decimate the problem, reducing its size and repeating the procedure on the smaller problem. We want to compute the probability that in a random cluster of satisfying assignments, variable i is frozen to some value, x_i. We call this the *bias* of the variable i. The positive bias, i.e. for $x_i = 1$, we have, letting $W_i^{(+)}$ denote positive bias,

$$\begin{aligned}
W_i^{(+)} &= P\left(\sum_{a\in V(i)} u_{a\to i} > 0\right)\\
&= C_{i\to a}\sum_{\{u_{a\to i}\}}\left[\prod_{a\ inV(i)} Q_{a\to i}(u_{a\to i})\right]\theta\left(\sum_{a\in V_+(i)} u_{a\to i}\right)\prod_{a\in V_-(i)}\delta(u_{a\to i},0),
\end{aligned}$$

where we have used our independence assumption to factorize the distribution. Now writing the quantities for the converged weights $\{\eta^*_{a\to i}\}$,

$$\begin{aligned}
\hat{\Pi}_i^+ &= \left[1-\prod_{a\in V_+(i)}(1-\eta^*_{a\to i})\right]\prod_{a\in V_-(i)}(1-\eta^*_{a\to i})\\
\hat{\Pi}_i^- &= \left[1-\prod_{a\in V_-(i)}(1-\eta^*_{a\to i})\right]\prod_{a\in V_+(i)}(1-\eta^*_{a\to i})\\
\hat{\Pi}_i^0 &= \prod_{a\in V(i)}(1-\eta^*_{a\to i}),
\end{aligned}$$

we have an expression for the positive, negative, and unfrozen bias:

$$\begin{aligned}
W_i^{(+)} &= \frac{\hat{\Pi}_i^+}{\hat{\Pi}_i^+ + \hat{\Pi}_i^- + \hat{\Pi}_i^0}\\
W_i^{(-)} &= \frac{\hat{\Pi}_i^-}{\hat{\Pi}_i^+ + \hat{\Pi}_i^- + \hat{\Pi}_i^0}\\
W_i^{(0)} &= 1 - W_i^{(+)} - W_i^{(-)}.
\end{aligned}$$

Once we have computed these biases, we reduce the size of the problem by computing the variables that are "most frozen" by computing which variables maximize the difference $|W_i^{(+)} - W_i^{(-)}|$.

Let us recall the BP equations given above. We can see then from the above SP equations, that SP is merely BP, where in addition to the two values 0 and 1, we take into account the possibility that a variable is unconstrained, or unfrozen. Therefore we have explored the similarities between SP and BP from two directions: first passing to the survey over the clusters, and then making more coarse, and the reverse: first making BP more coarse (by converting it to warning propagation) and then taking surveys over all satsifying clusters.

4 Conclusion and Some Questions

In this paper we discuss an interative message passing technique for solving constraint satisfaction problems. To a large extent, an analysis of the physical phenomena in the solution space, near the critical values, has been the key. First, it explains why conventional local search algorithms, like BP or WP, fail, and the in addition it motivates the development of the survey propagation algorithm. Empirically, it seems like this does quite well, solving problems of large size, very close to the phase transition. It has been observed that this method is successful even as high as $\alpha = 4.252$. Nevertheless, its practical success aside (or perhaps better to phrase it as "with its practical success in mind") many questions still remain, of which we present only a few.

(i) The main question which we must ask, given that survey propagation seems to do so well in practice, is where does the power of survey propagation come from? After all, since it is simply belief propagation with an additional state, it is by definition a local method. Are there further generalizations here? What happens if we don't simply add one joker state, but many? The first derivation makes it quite clear that we can add many more if we like, thus making the coarsening less dramatic.

(ii) With generalized belief propagation, it is very clear how we bridge the gap from local to global methods (and of course at the same time pay the price for that in terms of computational effort). Is survey propagation to be considered as a "tighter relaxation" in some sense, than belief propagation? What would the next relaxation be? Is it ever exact?

(iii) In the context of the LP results of Feldman et al., the standard LP relaxation of Sherali and Adams, correspond to generalized belief propagation. Can we understand the addition of the joker state in an optimization framework?

(iv) Is survey propagation appropriate for problems that may have only a unique optimal solution, rather than clusters of optimal solutions? In particular, is SP appropriate for decoding linear codes? Mezard, in [6], suggests that it is, but how best to do this is not clear.

(v) And a related question: Is there a connection to list decoding?

References

[1] Kschischang, F.R.; Frey, B.J., Loeliger, H.-A. "Factor Graphs and the Sum-Product Algorithm," IEEE Trans. Infor. Theory 47, 498 (2002).

[2] Parisi, G. "On the survey propagation equations for the random K-satisfiability problem," arXiv:cs.CC/0212009 v1, Sept. 2003.

[3] Parisi, G. "A backtracking survey propagation algorithm for K-satisfiability," arXiv:cond-mat/0308510 v1, August 2003.

[4] Braunstein, A.; Mézard, M.; Zecchina, R. "Survey Propagation: an algorithm for satisfiability." arXiv:cs.CC/0212002 v2 20 Jan 2003.

[5] Braunstein, A.; Mézard, M.; Zecchina, R. “Constraint Satisfaction by Survey Propagation.” arXiv:cond-mat/0212451 v2 23 Jan 2003.

[6] Mezard, M. ”Passing messages between disciplines,” Science, vol. 301, pp. 1685-1686, 19 Sept. 2003.

[7] Yedidia, J.S.; Freeman, W.T.; Weiss, Y. in “Exploring Artificial Intelligence in the New Millenium,” G. Lakemeyer, B. Nebel Eds. Morgan Kaufman, San Mateo, CA 2003, pp. 239-256.

[8] Yedidia, J.S.; Freeman, W.T.; Weiss, Y. “Generalized Belief Propagation,” in Advances in Neural Information Processing Systems 13 eds, T.K. Leen, T.G. Diettrich, V. Tresp, MIT Press 2001, 689-695.

[9] Wainwright, M. “Stochastic Processes on Graphs with Cycles: Geometric and Variational Approaches,” Ph.D. Thesis, MIT January 2002.