Characterizing Propagation Methods for Boolean Satisfiability

Eric Hsu and Sheila A. McIlraith

University of Toronto, Canada
{eihsu,sheila}@cs.toronto.edu

Abstract. Iterative algorithms such as Belief Propagation and Survey Propagation can handle some of the largest randomly-generated satisfiability problems (SAT) created to this point. But they can make inaccurate estimates or fail to converge on instances whose underlying constraint graphs contain small loops–a particularly strong concern with structured problems. More generally, their behavior is only well-understood in terms of statistical physics on a specific underlying model. Our alternative characterization of propagation algorithms presents them as value and variable ordering heuristics whose operation can be codified in terms of the Expectation Maximization (EM) method. Besides explaining failure to converge in the general case, understanding the equivalence between Propagation and EM yields new versions of such algorithms. When these are applied to SAT, such an understanding even yields a slight modification that guarantees convergence.

1 Introduction

The Survey Propagation (SP) algorithm [1] is one of the most exciting current approaches to the Boolean Satisfiability problem, rapidly solving problems with millions of variables under the most critically constrained settings of the clauses-to-variables ratio. Other successful applications of SP include coding and learning [2–4], while the older Belief Propagation (BP) framework that SP extends has been applied to Constraint Satisfaction Problems (CSPs) [5, 6, 2]. Nonetheless, both SP and BP are subject to some shortcomings that make them best-suited to large, randomly generated SAT instances.

In particular, these propagation algorithms do not always converge, or if they do, can converge to inaccurate estimates that eliminate valid solutions–especially on smaller or structured problems that contain short feedback cycles in their underlying constraint graphs. In such cases SP and BP cannot provide useful information to a surrounding search framework, necessitating a random restart. More generally, their behavior on loopy graphs is not been well-understood outside of a statistical physics interpretation [7, 8] that is founded on Markov Random Fields. Within the constraint-based reasoning field, their behavior has been partially explained in terms of discrete inference for the special case of extreme values [9], but no more general understanding has emerged.

The contribution of this paper is to supplement existing mathematical presentations of propagation methods with informal intuition, elucidating connections to related concepts. The most significant product is an alternative derivation of BP and SP in terms of the Expectation Maximization (EM) framework, one that is not dependent on the groundwork of existing physical explanations. This allows a new version of the BP and SP algorithms that always converges.

These results are demonstrated in BP for clearer exposition, but can be expressed in SP via the transformation shown in [1]. To that end, Section 2 provides background to our approach to characterizing BP and SP, and Section 3 presents necessary notation. In Section 4 we explain the two algorithms based on the background provided in Section 2, supplementing the equations provided in [1]. Finally, in Section 5 we present the EM algorithm in standard form, and then consider a formulation for SAT in Section 6. This produces a convergent update rule for solving SAT instances. The results of implementing this process appear in Section 7, followed by discussion in Section 8.

2 Approach

In this paper, we characterize SP in alternative terms, translating existing descriptions into familiar concepts, founded on insights into its behavior in the context of SAT. Specifically, we contend that SP works best as a variable and value ordering heuristic within a simple search framework. It can be roughly understood as an extension of BP into ternary space, where variables are positive in some fraction of the solutions, negative in another fraction, and recognized as in a third proportion of solutions [1]. (The usefulness of such logics has been independently noted in completely different approaches [10].) Efforts in statistical physics to battle loops by clustering nodes together tend to parallel AI techniques for finding join-graphs, cluster-trees, etc. [11–13].

Familiar local search techniques find probabilistic variable settings Θ to maximize the probability $P(SAT|\Theta)$ that all clauses are satisfied, just as MAXSAT approximations target an expectation $E[SAT|\Theta]$ on the number of satisfied clauses [14–16]. In contrast, the BP and SP propagation algorithms can be viewed as estimators of $P(\Theta|SAT)$, the probability that the variables are configured a certain way given that all clauses are satisfied. Thus, SP can help detect the most prescient "backdoor" variables whose correct assignments trivialize the remaining problem, while BP settles for "backbone variables," which are constrained to be always positive or always negative in the majority of solutions [17, 18]. So, propagation methods serve as heuristics that guide the search; if they were always right (and always converged) the search would be backtrack-free.

Such insights enable a final conclusion: that propagation methods actually perform a slightly altered version of the well-known Expectation Maximization (EM) algorithm, which seeks out posterior likelihoods complicated by hidden interactions [19]. This understanding engenders further comparison with Lagrangian optimization approaches to SAT, through known connections to EM [20]. Similarly, it provides for the development of specialized propagation meth-

ods based on EM variants for sparse problems and partial optimizations [21, 22]. Perhaps the most interesting observation, though, is that EM always converges.

3 Problem and Notation

Definition 1. *A* SAT instance *is a set C of m clauses, constraining a set V of n Boolean variables. Each clause $c \in C$ is a disjunction of literals built from the variables in V. An assignment $X \in \{0,1\}^n$ to the variables satisfies the instance if it makes at least one literal true in each clause. The sets V_c^+ and V_c^- contain the indexes of the variables appearing positively and negatively in a clause c, respectively. The sets C_v^+ and C_v^- contain the indexes of the clauses that contain positive and negative literals for variable v, respectively.*

Definition 2. *A variable* bias $\theta_v \in \mathbb{R}$, $0 \leq \theta_v \leq 1$, *represents the probability that variable v will be positive rather than negative. $\Theta \in \mathbb{R}^n$ denotes a vector of biases for all n variables.*

In general, capitalized variables will correspond to vectors, and lower-case variables with subscript indexes will be their components.

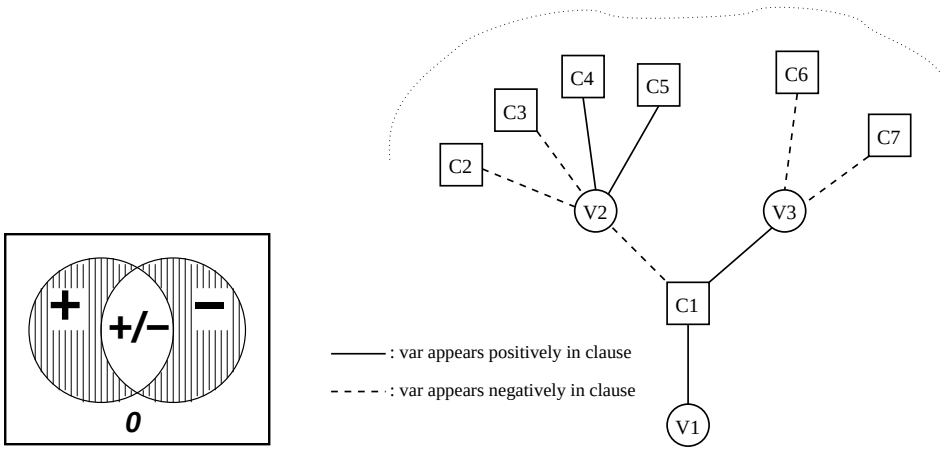
4 The BP and SP Algorithms

BP and SP are message-passing algorithms that attempt to sample from the space of satisfying assignments. Here we explain the algorithms at an intuitive level, to supplement the formulas in [1]. The only concrete artifact that is necessary for our purposes is the BP update rule appearing in equation (1).

Imagine a listing of all solutions to a given SAT problem. Clearly the chance to simply read from that list is wishful thinking for a polynomial algorithm, as this ability would instantly provide a solution if one existed. But what if it were possible to compile statistics over the contents of that list, despite being blind to the list itself? This would be very useful for guiding search, and comprises the goal of propagation methods as applied to SAT. BP and SP attempt this by repeatedly updating estimated biases, hopefully until convergence to a local maximum in likelihood. For BP this means a bias θ_v for each variable, indicating the estimated proportion of solutions in which the variable must be positive rather than negative. SP extends this space with a third state where a variable is not constrained to take either value; with BP the mass for this case ends up proportionately distributed between the positive and negative.

4.1 Sample Space for Determining Bias

In this section, we differentiate the sample space for determining the variable biases that BP and SP are estimating. To borrow terminology from Section 6.2, BP codifies a simplifying assumption that in any satisfying assignment, every variable is the "sole support" of some clause. That is, each variable believes that



(a) (b)

Fig. 1. (a) Probability Space for Variable Bias under BP; (b) Factor Graph Fragment.

it is satisfying at least one clause that would otherwise be left totally unsatisfied by all of its other variables.

Figure 1(a) is a Venn diagram depicting the bias space for a given variable, as the shaded region. The area of the diagram as a whole spans the space of all satisfying assignments to the variables. The left circle, labeled "+", denotes those assignments where there exists some clause that is wholly dependent on the variable being positive. That is, the considered variable appears positively in some clause whose other literals all hold unsatisfactory values under the current assignment. Likewise, the right circle indicates that some clause requires the variable to be negative. Their intersection, labeled "+/−", is eliminated from the probability space, as no satisfying assignment could require a single variable to be both positive and negative.

Thus BP's goal is to determine, for each variable, the proportion of solutions in which it lies in the positive half of the shaded area, versus the negative half. In comparison, the power of SP lies in additionally considering region "0", where all clauses are already satisfied by variables other than the one under consideration. Stated differently, BP determines the bias of each variable, in terms of the chances that it would appear positively or negatively if a satisfying assignment were randomly drawn from the otherwise inaccessible list of solutions.

4.2 Algorithmic Framework

At a high level, the BP and SP algorithms accomplish the described task by passing messages over a given SAT problem's factor graph representation, as depicted in Figure 1(b).

Nodes representing variables connect to nodes representing clauses in which they appear. Edges can be distinguished, conceptually, by whether the variables

appear as positive or negative literals in the clauses. The edges carry clause-to-variable messages in one direction, and variable-to clause messages in the other.

Each variable is randomly seeded with an initial bias, and informs all of its clauses by passing variable-to-clause messages along the edges. The clauses compile such reports and determine whether they are poorly supported–that is, they calculate the probability that their variables will jointly end up failing to satisfy them. From here they signal each variable as to whether they need their support by passing messages back along the edges, in the opposite direction. The variables weigh such requests, and begin a new iteration by updating and reporting their new biases. Equations (13-16) in [1] represent this process. A crucial detail is that a variable tells a clause what its bias would be *in the absence of that clause*. Likewise, clauses do not broadcast identical distress messages along all their outgoing edges. Rather, along each edge they report whether they are unlikely to be satisfied *in the absence of the corresponding variable*. This is what makes the algorithm exactly the same as Pearl's original BP, also known as the Sum-Product algorithm [5, 6].

Thus, consider a state for Figure 1(b) where c_6 and c_7 have both informed v_3 that they are unlikely to be supported by their other variables. Consequently, v_3 sees that they need v_3's help, and tells c_1 that in its absence, v_3 would probably have to be negative. Likewise, if v_2 is getting stronger messages from c_4 and c_5 than from c_2 and c_3, then v_2 can report to c_1 that v_2 will also not be of much help. Thus, c_1 will send a strong message to v_1, *whether or not* v_1 *is already positively biased*. This message could be interpreted as either "I need you to come support me" or "don't listen to your other clauses, I'm highly dependent on you" depending on the strength of v_1's existing bias toward the positive.

The graph and messages are conceptual, though. After much derivation, the entire dynamics can be operationalized as a single update rule. Clause-to-variable messages can be bypassed by expressing variable-to-clause messages in terms of other variable-to-clause messages, two edges removed. (In fact, the original SP derivation and code happen to employ the opposite clause-to-variable representation, shown as Equations (17) and (18) [1, 23].) The variables' incoming messages can themselves be represented as changes to the bias, culminating in the update rule for Θ shown below as Equation (1).

Recall that Θ is a vector containing biases for each of the variables, and undergoes this update once for each individual bias θ_v. The entire process is repeated in hopes of eventual convergence. The products expressed in terms of i's and j's represent the probability that a clause c will be left unsatisfied by all of its variables outside of v. Subtracting from 1 creates the negation of this proposition. So the numerator represents the chance that none of the variable's negative clauses require it. In other words, it is the inverse of the proposition that some negative clause requires the variable. Thus the rule can be understood in terms of the diagram in Figure 1(a). Here the numerator represents the inverse of the entire circle labeled "−." Because the "+/-" region is excluded by sampling only satisfying assignments, and the "0" region is excluded by the BP assumption, the inverse yields the left moon of the shaded sample space.

Thus, the numerator of the update rule denotes the area of the positive shaded region, while the sum in the denominator constructs the complete sample space.

$$\theta'_v \leftarrow \frac{\prod_{c \in C_v^-} \left[1 - \prod_{i \in V_c^+} (1 - \theta_i) \prod_{j \in V_c^- \setminus v} \theta_j \right]}{\prod_{c \in C_v^-} \left[1 - \prod_{i \in V_c^+} (1 - \theta_i) \prod_{j \in V_c^- \setminus v} \theta_j \right] + \prod_{c \in C+_v} \left[1 - \prod_{i \in V_c^+} (1 - \theta_i) \prod_{j \in V_c^- \setminus v} \theta_j \right]} \quad (1)$$

So we represent two states with one parameter; the probability that variable v must be negative is just $1 - \theta_v$. In the case of SP, we use two equations to represent three states. The rule for positive variables is identical to (1), but with an added term in the denominator for including "0" in the outcome space, and extra factors in the numerator for excluding it from the sample space. A second rule will represent a variable's negative bias, flipping the products for $c \in C^+$ for those over $c \in C^-$. Finally, the "0" state where a variable is unbiased is left to marginalization; it is just one minus the first two probabilities.

4.3 Applying BP/SP via Unit Decimation

Propagation algorithms cannot solve SAT instances on their own. Rather, they can be embedded within a simple search framework that consults them when deciding which variable to fix next. Originally this was tied to a decimation algorithm, where blocks of several variables are fixed all at once. This is risky because the probabilities are not conditional: perhaps v_1 and v_2 both are positive in most satisfying assignments, but often are not positive at the same time.

Conceptually and in practice, one expedient is to consider a more extreme version of this methodology, where only one variable is fixed at a time (at a greater cost in terms of number of surveys). The conditional probabilities are essentially produced by re-running the survey on simplified problems where the previous choices have already been fixed. More concretely, the rest of this paper will consider BP and SP within the following framework:

Algorithm 1 BP/SP with Unit Decimation

BP/SP(*SAT-instance*, $t_{timeout}$)
  repeat
    survey ← SP(*SAT-instance*, $t_{timeout}$) or BP(*SAT-instance*, $t_{timeout}$).
    assignment ← CHOOSE-ASSIGNMENT(*SAT-instance*, *survey*).
    SAT-instance ← FIX(*SAT-instance*, *assignment*)
    If all are clauses satisfied, return solution.
  until all variables fixed
  Return failure.

On each iteration, BP or SP produces a survey estimating the biases of the variables, in either binary or ternary space respectively. If the $t_{timeout}$ parameter is reached before convergence, the entire algorithm fails. With a survey

in hand, the algorithm uses CHOOSE-ASSIGNMENT to identify a single variable to fix, and whether to fix it to true or false. One straightforward rule is to choose the variable with the most extreme positive or negative bias, and fix it in that direction. With SP, the extra "0" space allows more choices, such as fixing the variable with the smallest such bias. Next, the assignment is simplified to reflect the assignment and the process repeats. This process can incorporate unit-propagation or other such inference processes. The algorithm terminates if enough assignments have been made to satisfy the problem, or if all variables are assigned yet there is still an unsatisfied clause.

Viewed as such, the surveys serve as simultaneous variable and value ordering heuristics. If we only require a single solution, and the heuristics are always correct, then we have a complete reasoning process. For if a survey returns any bias whatsoever for a variable, then some percentage of the solutions features the variable at that value. Thus there is at least one solution remaining. If a variable must not be fixed a certain way, it must have zero bias in that direction.

In practice, the algorithm fixes the most important variables in the first few iterations, and then the maximum bias drops below a certain level. At that point the simplified problem can be passed to a regular solver like WalkSAT for improved speed [14]. Also, completeness is not guaranteed; the algorithms converge to local maxima in terms of survey correctness. Finally, BP and SP may simply fail to converge. For tree networks like the excerpt in Figure 1(b) it is clear that the algorithms converge. But for factor graphs with cycles, it is easy to visualize the algorithms' incompleteness, as feedback loops of messages being passed around and around. Alternatively, such structure can cause the algorithm to converge, but to the wrong answer: randomly initializing biases via a uniform distribution can tilt the optimization process away from endless loops, but only by immediately jumping into local maxima. Without a uniform understanding of the algorithms, such behavior has historically been difficult to characterize.

5 The EM Algorithm

In this section we consider the general EM algorithm [19], so that we can later exploit its transformation into BP and derive an improved way to calculate surveys, one that always converges. At a high level, EM accepts a vector of observations Y, and determines the model parameters Θ that maximize the likelihood of having seen Y. Maximizing $log\, P(Y|\Theta)$ would ordinarily be straightforward, but for the additional complication that we posit some latent variables Z that contributed to the generation of Y, but that we did not get to observe. That is, we want to set Θ to maximize $log\, P(Y, Z|\Theta)$, but cannot marginalize on Z.

So, we bootstrap by constructing $\tilde{P}(Z)$ to estimate $P(Z|Y, \Theta)$ and then use this distribution to maximize the expected likelihood $P(Y, Z|\Theta)$ with respect to Θ. The first step is called the E-Step, and the second is the M-Step. The two are repeated until convergence, which is guaranteed.

6 Transformation from BP to EM Approaches for SAT

Operationally, the BP and the EM algorithm appear to share nothing more than their dualized iterative dynamics. Yet even here there are differences: BP can actually be expressed in terms of just one set of messages (either function-to-variable or variable-to-function) while EM cannot (unless it is used stochastically by updating one variable at a time.) On convergence, neither algorithm can promise more than a local maximum, but EM is guaranteed to converge, while BP is not in the case of graphs with cycles.

6.1 Free-Energy Characterizations of BP and EM

It turns out, though, that the two approaches are actually derived from equivalent energy minimization equations, called "Variational Free Energy" and "Gibbs Free Energy" in the EM and BP literature, respectively. Such expressions arise from trying to minimize the Kullback-Leibler distance between two probability distributions, meaning that the distributions are made to give similar predictions across a common domain of outcomes. The following is a high-level overview of this equivalence, while less germane details can be found in [24].

In the case of BP, we want our belief $b(x)$ to match a factorized approximation of the truth $p(x)$, across all values of x [8]. With EM, there are two distinct steps in getting $\tilde{P}(Z)$, our estimated distribution on Z, to match the true probability $P(Z|Y,\Theta)$. During E, we adjust $\tilde{P}(Z)$, and during M, we adjust Θ [21].

However, the proof of equivalence, based on [8]'s Markov Random Fields representation of factor graphs, is not necessarily constructive. In particular, pushing a straightforward SAT formulation from BP through to EM is liable to produce an inoperable restatement of the problem. Typical interpretations of such formulas produce English phrasings like "bias all variables toward 0 or 1 (by avoiding entropy) while still satisfying all clauses (by avoiding free energy.)" Furthermore, the generic Random Field structure relies on an approximation that breaks any guarantee of convergence. Despite this, a BP-inspired, yet convergent, SAT solution method can be reverse-engineered into the EM framework, by essentially lying to the algorithm.

6.2 SAT Formulation for EM

The trick is to tell EM that we have seen that all the clauses are satisfied, but not how exactly they went about choosing satisfying variables for support. We ask the algorithm to find the variable biases that will best support our claimed sighting, via some hypothesized support configuration. This produces the desired $P(\Theta|SAT)$. In this section explicitly derive this formulation from first principles, resulting in a modification of the update rule (1), reflected in (9).

First we set the EM variables as in Table 1. Y will be a vector of all 1's, meaning that all clauses are satisfied, while each $\theta_v \in \Theta$ represents variable v's probability of being positive. Finally, Z contains some value $s_{c,v}$ for each clause c, denoting that "clause c is solely-supported by variable v".

Vector	Status	Interpretation	Domain
Y	Observed	whether clauses are SAT	$\{0,1\}^m$
Z	Unobserved	support configurations for clauses	$\{s_{c,v}\}^m$ $c \in C, v \in V$
Θ	Parameters	variable biases	$(0,1)^n$

Table 1. SAT Formulation for EM

Definition 3. *A variable v is the* sole support *to a clause c when its current assignment satisfies the clause, and none of the clause's other variables satisfy it under the current assignment.*

The Z terms invite elaboration on the space of possible configurations for a given clause. Under a particular variable assignment, a clause is either satisfied by multiple variables, or else by exactly one, or else by none–in which case it is unsatisfied. In order to sample only from the space of satisfying assignments (or more presciently, because $Y = [1]^m$) we eliminate the last case from the probability space. Further, by the simplifying assumption of BP, we eliminate the first case: all constraints are tight in that all supporting variables think that they are the only supports. Reinstating this case yields SP, and a more unreadable derivation. In short, v is the sole support of c when it satisfies c and all of c's other variables do not. For each c, exactly one $s_{c,v}$ must hold–these are the hidden values that EM will weight with its artificial distribution.

6.3 Deriving a SAT Algorithm from EM

For the E-Step we derive said distribution $\tilde{P}(Z)$ for the latent variables, decomposing it into a single $\tilde{p}_c(z_c) = p(z_c|y_c, \Theta)$ for each clause:

$$\tilde{p}_c(s_{c,v}) = \prod_{w \in V_c^+ \setminus v} (1 - \theta_w) \prod_{w \in V_c^- \setminus v} \theta_w \tag{2}$$

The equation states that for $s_{c,v}$ ("v is the sole support of c") to hold, all variables outside of v that were supposed to be positive turned out negative, and vice versa–so v is the sole support. It might seem that v's own support (θ_v if $v \in V_c^+$, $1 - \theta_v$ if $v \in V_c^-$) should appear as a factor above. But it is precisely its exclusion that guarantees that we sample from a space of satisfied clauses, maintaining consistency with the conditioned $y_c = 1$.

In the M-Step we use this distribution to get a lower bound on the logarithm of the expected probability of Y. The log is crucial for maintaining convergence via Jensen's Inequality: $log\, E[p(x)] \geq E[log\, p(x)]$. It also allows us to decompose the set of data (clauses) into terms in a sum. So in short we will set $\Theta \leftarrow argmax_\Theta\ F(\Theta)$, where:

$$F(\Theta) = E_{\tilde{P}}[log\, P(Y, Z|\Theta)] \tag{3}$$

$$= \sum_c E_{\tilde{p}_c}[log\, p(z_c|\Theta)] \tag{4}$$

This is effected by making logs of products into sums of logs, and by observing that any valid z_c, i.e. one that is given any weight by the corresponding $\tilde{p}_c$, already signifies that its clause is satisfied by definition. In other words, z_c implies y_c, enabling its removal from the joint probability.

By similar reasoning to that in (2) we derive, after some simplification,

$$F(\Theta) = \sum_c \sum_{s_{c,v}} \tilde{p}_c(s_{c,v}) \left[\sum_{i \in V_c^+ \setminus v} log\,(1-\theta_i) \; + \sum_{j \in V_c^- \setminus v} log\,\theta_j \right] \tag{5}$$

To optimize, we take the first derivative with respect to each variable v:

$$\frac{dF}{d\Theta_v} = \sum_{c \in C_v} \sum_{\substack{s_{c,w} \\ w \neq v}} \tilde{p}_c(s_{c,w}) \left[\begin{cases} \frac{1}{\theta_v - 1} & \text{if } v \in V_c^+ \\ \frac{1}{\theta_v} & \text{if } v \in V_c^- \end{cases} \right] \tag{6}$$

Because the various support profiles $s_{c,v}$ partition the space of possible configurations for c, we can marginalize out the probability that the clause is solely supported by some variable other than v:

$$\sum_{\substack{s_{c,w} \\ w \neq v}} \tilde{p}_c(s_{c,w}) = 1 - \tilde{p}_c(s_{c,v}) \tag{7}$$

By substituting into (6), and splitting v's clauses into those where it appears positively and negatively, we obtain:

$$\begin{array}{c} \frac{dF}{d\Theta_v} = \alpha \cdot \frac{1}{\theta_v} + \beta \cdot \frac{1}{\theta_v - 1} \\ \text{where} \quad \alpha = \sum_{c \in C_v^-} (1 - \tilde{p}_c(s_{c,v})) \text{ and} \\ \beta = \sum_{c \in C_v^+} (1 - \tilde{p}_c(s_{c,v})) \end{array} \tag{8}$$

Finally, by setting the derivative to zero, and substituting (2) for $\tilde{p}_c(s_{c,v})$, we arrive at an EM update rule for the θ's:

$$\frac{dF}{d\Theta_v} = 0 \quad \Rightarrow \quad \alpha \cdot (\theta_v - 1) = -\beta \cdot (\theta_v) \quad \Rightarrow \quad \theta_v = \frac{\alpha}{\alpha + \beta} \quad \Rightarrow$$

$$\theta'_v \leftarrow \frac{\displaystyle\sum_{c \in C_v^-} \left[1 - \prod_{i \in V_c^+} (1-\theta_i) \prod_{j \in V_c^- \setminus v} \theta_j \right]}{\displaystyle\sum_{c \in C_v^-} \left[1 - \prod_{i \in V_c^+} (1-\theta_i) \prod_{j \in V_c^- \setminus v} \theta_j \right] + \sum_{c \in C+_v} \left[1 - \prod_{i \in V_c^+} (1-\theta_i) \prod_{j \in V_c^- \setminus v} \theta_j \right]} \tag{9}$$

6.4 Comparison with BP

Thus we exhibit a transformation between this EM formulation for SAT, and previous approaches based on BP; the above is almost identical to (1). The sole difference, the replacement of products by sums, is the crux of ensuring

convergence. A high-level syntactic understanding is that the logarithms allow us to treat each clause as a separate term in a sum, making the update rule into a (log) odds expression rather than a standard probability. A high-level operational understanding is that when walking toward local maxima, we want to avoid large steps that can overshoot a given peak, resulting in a sort of orbiting nonconvergence. Ratios of sums produce gentler steps than those of products.

In fact, the steps here are bounded in such a way that they are guaranteed never to increase our distance from the nearest local maximum. This is a general property of EM, and a more detailed (and rigorous) explanation can be found in most introductions to the algorithm, or to related variational methods [22]. In essence, the use of Jensen's Inequality in formulating (3) ensures that we have a lower bound on the local maximum, i.e. an admissible heuristic. Further, the fact that the E step fully optimizes the energy equation ensures tightness, meaning that we can only raise this lower bound between alternations of E and M.

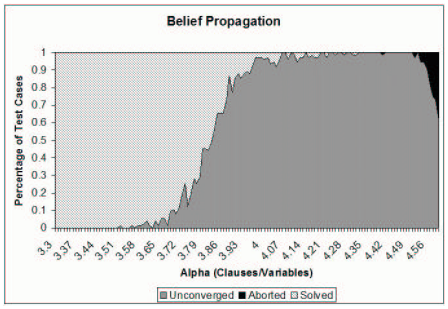
7 Implementation and Empirical Results

We have examined such theoretical claims by implementing the EM-derived version of BP ("EMBP") within existing SP code [23]. The code also implements regular BP, allowing comparison of the three approaches within a common infrastructure. (An EM version of SP is also possible, but was not implemented.) As expected, EMBP always proceeded directly to single local maximum in likelihood, and thus always converged. A second question of interest, though, concerns the quality of such maxima. Although the EM formulation always converges, it can still fail to find a solution when one exists. Indeed, even on convergence, all three algorithms arrive at only a *local* maximizer for the log likelihood of $P(\Theta|SAT)$; this peak might not correctly reflect the truth. Thus, even under unit decimation it is still possible to make an incorrect decision that eliminates all remaining solutions. Though they were not implemented in the proof of concept, backtracking or restarts would be necessary at this point.

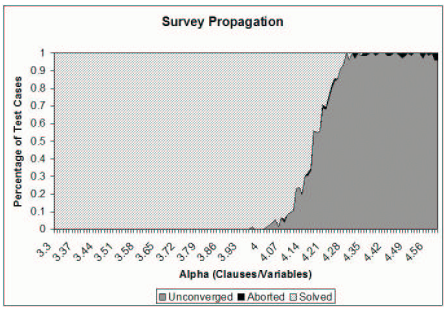
Figure 2 addresses such issues over both random and structured problems. Across one hundred trials per test suite, the three approaches either solved (satisfiable) instances, failed to converge at some point during execution, or else aborted upon fixing all variables without finding an assignment (either because of inaccurate local maxima, or because an instance was indeed unsatisfiable.) The three graphs represent the relative proportion of these cases on random problems as the clause-to-variable ratio α crosses the phase transition area, while the table represents these percentages over the entirely satisfiable "Inductive Inference," "Logistics," "Parity," and "Quasigroup Completion" test suites of the Satlib benchmark library. In short, EMBP always converges, but it appears to give worse answers than BP and SP on random problems, and better answers on the selected structured problems.

As the random problems become more constrained, the traditional propagation techniques encounter increasing risk of non-convergence, essentially on unsatisfiable instances. Further on, they begin to recapture the ability to con-

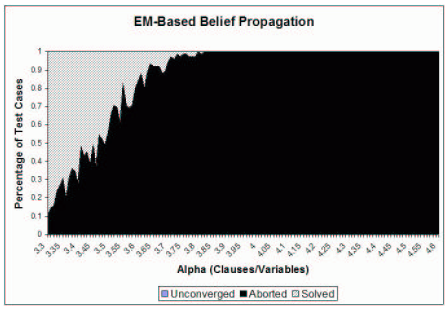
verge, and only abort due to incorrect maxima. (The maxima must be incorrect, as these are unsatisfiable instances.) While EMBP always converges, it will begin aborting with an earlier threshold than BP and SP. This is consistent with the hypothesis that by overshooting their targets, traditional propagation methods are able to sample a larger space of local maxima than EM methods, but at the risk of failing to converge. All three approaches remain practical, though, with the use of restarts–so long as there is a non-negligible probability of success, repeated attempts will eventually cure single-mindedness and wanderlust alike. Similarly, with the exception of logistics, the structured results are positive de-



(a)



(b)



(c)

Suite	Method	Unconverged	Aborted	Solved
ii	BP	2.7	73.5	23.8
	SP	0	68.0	32.0
	EMBP	0	60.5	39.5
logistics	BP	0	100	0
	SP	0	100	0
	EMBP	0	100	0
parity	BP	0	99.8	0.2
	SP	0	99.9	0.1
	EMBP	0	97.0	3.0
qg	BP	0	100	0
	SP	0	100	0
	EMBP	0	95.4	4.6

(d)

Fig. 2. Outcomes for (a) BP, (b) SP, and (c) BP-EM Propagation on Random Problems; Outcomes on (d) Structured Problems.

spite relatively low success rates. Recall that the framework is backtrack-free: each run is first randomly initialized and then continues on to success only by making an entirely correct string of decisions for fixing variables. (It is this initialization that makes BP and SP fail by abortion rather than non-convergence.) Still, within the restart framework, EMBP is superior to BP/SP on these structured problems–it is significantly more likely to find a solution for inductive

inference and parity problems, and is the only approach with any chance of solving a quasi-group completion problem.

8 Summary and Discussion

The main contribution of this paper is to provide a clearer understanding of the BP and SP algorithms by relating them to the EM algorithm. This exposition provides deeper insight into the differing performance of these algorithms on structured and unstructured problems. It also enables development of variants of these algorithms that were guaranteed to converge. A secondary contribution of this paper is to provide an intuitive but nonstandard explanation of BP and SP by characterizing unit decimation as a variable/value heuristic and relating these algorithms' purpose to finding backbones and backdoors.

We hope that relating BP and SP to EM will allow more tangible gains through the application of related ideas. EM is widely used in statistically-inclined research communities and features many variants suggested by theoretical works like [21]. There are incremental versions that converge more quickly and enable online processing of new clauses. Sparse versions can handle near-zero probabilities symbolically, another expedient that has been used in similar form [23] with propagation algorithms, but not systematically. Other variants alter the artificial distribution $\tilde{P}(Z|Y,\Theta)$, for instance by encouraging it to give more mass to fewer possibilities; the commonly-used K-Means algorithm is an extreme example of this idea. Finally, variational methods can be crudely viewed as developing less exact or less convergent techniques to more efficiently operate on the Markov Random Fields underlying BP's energy equations [22].

Another avenue for future work is to consider $P(SAT|\Theta)$ in lieu of $P(\Theta|SAT)$. While there are no clear semantics for the priors $P(\Theta)$ and $P(SAT)$, the two conditional probabilities are proportional via Bayes' rule. The unit decimation framework suggests an alternate employment of local search for finding solutions to SAT and MAXSAT. Instead of using searches as walks to maxima, each walk can be considered a sample to use as a variable and value ordering heuristic.

References

1. Braunstein, A., Mezard, M., Zecchina, R.: Survey propagation: An algorithm for satisfiability. Random Structures and Algorithms **27** (2005) 201–226
2. Kask, K., Dechter, R., Gogate, V.: Counting-based look-ahead schemes for constraint satisfaction. In: Proc. of 10th International Conference on Constraint Programming (CP '04), Toronto, Canada. (2004)
3. Wang, Y., Zhang, J., Fossorier, M., Yedidia, J.: Reduced latency iterative decoding of LDPC codes. In: IEEE Conference on Global Telecommunications (GLOBECOM). (2005)
4. Braunstein, A., Zecchina, R.: Learning by message passing in networks of discrete synapses. Physics Review Letters **96**(5) (2006)
5. Pearl, J.: Probabilistic Reasoning in Intelligent Systems. Morgan Kaufmann, San Mateo (1988)

6. Kschischang, F.R., Frey, B.J., Loeliger, H.A.: Factor graphs and the sum-product algorithm. IEEE Transactions on Information Theory **47**(2) (2001)
7. Braunstein, A., Zecchina, R.: Survey propagation as local equilibrium equations. Journal of Statistical Mechanics: Theory and Experiments **PO6007** (2004)
8. Yedidia, J., Freeman, W., Weiss, Y.: Understanding belief propagation and its generalizations. In Nebel, B., Lakemeyer, G., eds.: Exploring Artificial Intelligence in the New Millennium. Morgan Kaufmann (2003) 239–256
9. Dechter, R., Mateescu, R.: A simple insight into properties of iterative belief propagation. In: Proc. of 19th International Conference on Uncertainty in Artificial Intelligence (UAI '03), Acapulco, Mexico. (2003)
10. Lardeux, F., Saubion, F., Hao, J.K.: Three truth values for the SAT and MAX-SAT problems. In: Proc. of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI '05), Edinburgh, Scotland. (2005)
11. Yedidia, J., Freeman, W., Weiss, Y.: Constructing free-energy approximations and generalized belief propagation algorithms. IEEE Transactions on Information Theory **51**(7) (2005) 2282–2312
12. Dechter, R., Kask, K., Mateescu, R.: Iterative join-graph propagation. In: Proc. of 18th International Conference on Uncertainty in Artificial Intelligence (UAI '02), Edmonton, Canada. (2002) 128–136
13. Kask, K., Dechter, R., Larrosa, J., Pfeffer, A.: Cluster-tree decompostitions for reasoning in graphical models. Artificial Intelligence **166**(1-2) (2005)
14. Selman, B., Kautz, H., Cohen, B.: Local search strategies for satisfiability testing. DIMACS Series in Discrete Mathematics and Theoretical Computer Science **26** (1996)
15. Goemans, M., Williamson, D.: New 3/4-approximation algorithms for the maximum satisfiability problem. SIAM Journal on Discrete Mathematics **7** (1994) 656–666
16. Goemans, M., Williamson, D.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. Journal of the ACM (42) (1995) 1115–1145
17. Williams, R., Gomes, C., Selman, B.: Backdoors to typical case complexity. In: Proc. of 18th International Joint Conference on Artificial Intelligence (IJCAI '03), Acapulco, Mexico. (2003)
18. Gomes, C., Selman, B., Crato, N., Kautz, H.: Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. Journal of Automated Reasoning **24**(1-2) (2000) 67–100
19. Dempster, A., Laird, N., Rubin, D.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society **39**(1) (1977) 1–39
20. Shang, Y., Wah, B.: A discrete Lagrangian-based global-search method for solving satisfiability problems. Journal of Global Optimization **12**(1) (1998) 61–99
21. Neal, R., Hinton, G.: A view of the EM algorithm that justifies incremental, sparse, and other variants. In Jordan, M., ed.: Learning in Graphical Models. Kluwer Academic Publishers (1998) 355–368
22. Jordan, M., Ghahramani, Z., Jaakkola, T., Saul, L.: An introduction to variational methods for graphical models. In Jordan, M., ed.: Learning in Graphical Models. MIT Press (1998)
23. Braunstein, A., Leone, M., Mezard, M., Weigt, M., Zecchina, R.: Sp-1.3 survey propagatrion implementation. (http://www.ictp.trieste.it/~zecchina/SP/)
24. Hsu, E., McIlraith, S.: Characterizing loopy belief propagation as expectation maximization (2006) Manuscript in preparation.