Message-Passing and Local Heuristics as Decimation Strategies for Satisfiability

Lukas Kroc Ashish Sabharwal Bart Selman
Department of Computer Science, Cornell University Ithaca NY 14853-7501, U.S.A.
{kroc,sabhar,selman}@cs.cornell.edu

ABSTRACT

Decimation is a simple process for solving constraint satisfaction problems, by repeatedly fixing variable values and simplifying without reconsidering earlier decisions. We investigate different decimation strategies, contrasting those based on local, syntactic information from those based on message passing, such as statistical physics based Survey Propagation (SP) and the related and more well-known Belief Propagation (BP). Our results reveal that once we resolve convergence issues, BP itself can solve fairly hard random k-SAT formulas through decimation; the gap between BP and SP narrows down quickly as k increases. We also investigate observable differences between BP/SP and other common CSP heuristics as decimation proceeds, exploring the hardness of the decimated formulas and identifying a somewhat unexpected feature of message passing heuristics, namely, unlike other heuristics for satisfiability, they avoid unit propagation as variables are fixed.

Categories and Subject Descriptors

I.2.8 [**Computing Methodologies**]: Artificial Intelligence – *problem solving, control methods, and search*

Keywords

Belief propagation, constraint reasoning, heuristics, SAT, survey propagation

1. INTRODUCTION

In 2002, Mézard, Parisi, and Zecchina (2002) introduced the Survey Propagation (SP) method to solve hard Boolean satisfiability (SAT) problem instances. The method is remarkably effective, capable of solving certain million variable instances that were far beyond the reach of previous techniques. Even more importantly, SP, which can be viewed as a form of belief propagation (BP), is the first successful example of the use of a probabilistic reasoning technique to solve a purely combinatorial search problem. SP is based on the so-called 1-RSB cavity method from statistical physics. The method is still far from well-understood, but in recent years, we are starting to see results that provide important insights into its workings.

The SAT problem consists of a formula F representing a set of constraints over n Boolean variables, which must be set so as to satisfy all constraints. The SP method solves the problem using the "decimation" process, which assigns a truth value to one variable (or a few variables) of F and simplifies F, obtaining a smaller formula on $n-1$ variables. SP repeatedly decimates the formula in this manner, until a simplified instance is obtained that is easily solved by existing SAT solvers, such as `Walksat` (Selman et al., 1996). The decimation process can fail in that it may assign a variable the wrong value, inadvertently eliminating all remaining satisfying assignments. The remarkable property of SP is that it can take a million variable, hard random 3-SAT instance and set 40%-50% of the variables, without making any "mistake." The resulting (satisfiable) subformula is then easily solved by `Walksat`. How does SP select a variable for decimation? Intuitively speaking, it considers the space of all satisfying assignments, and estimates the marginal probability of each variable being TRUE (or FALSE). It then selects the variable and the value assignment that has the highest marginal probability, thus, in a sense, preserving the largest number of satisfying assignments in the subproblem. As it turns out, SP does not directly consider the space of satisfying assignments but rather the space of "covers," which are special combinatorial objects representing clusters of solutions. It was shown by Braunstein and Zecchina (2004) and Maneva et al. (2007) that the SP method can be viewed as a form of belief propagation (BP) on a new constraint satisfaction problem derived from the original SAT instance, in which the objects of interest correspond to the covers of the original SAT problem.

The introduction of SP with its remarkable effectiveness has created the impression that to solve hard combinatorial instances via decimation, SP is required and BP would have little success. This was not explicitly stated in the literature but has become a fairly common view. However, we show that BP can be almost as effective as SP. In particular, we show that for random k-SAT formulas over a wide range of parameter settings, BP works just as well as SP when performing decimation. For a preview of these results, see Figure 2. (To remedy some of the convergence difficulties of BP on loopy factor graphs, we use a damped version BP$^\kappa$.) This finding is good news in terms of the use of probabilistic techniques for solving combinatorial problems. SP equations

SAC'09 March 8-12, 2009, Honolulu, Hawaii, U.S.A.

need to be developed in a specialized manner for each individual combinatorial problem. For example, different equations have been developed for k-SAT, graph coloring, and vertex cover problems over a series of papers. At this point there is no generic recipe for writing the SP equations for arbitrary combinatorial problems, and one needs an understanding of the 1-RSB cavity method from statistical physics to do this for each problem. The BP equations on the other hand are quite generic and can be written down directly from the factor graph representing the combinatorial problem in question. In practical terms, our first message is that in further exploring probabilistic methods for solving combinatorial problems, BP holds a promise that is quite similar to that of SP. In particular, in the development of hybrid search methods, e.g., to boost SAT solvers, one should not rule out BP methods before going to SP.

We also compare the decimation strategies of BP, BP$^\kappa$, and SP, with those found in more traditional approaches towards SAT solving. A highly successful class of modern SAT solvers is based on the so-called DPLL procedure (Davis and Putnam, 1960, Davis et al., 1962). Such solvers perform essentially a backtrack search through the space of all truth assignments searching for a satisfying one. In the search, heuristics are used to select which variable to set next and to what value. Each branch of the backtrack search tree corresponds to a decimation strategy. Our experiments show that the heuristics employed in modern SAT solvers can in fact solve some random k-SAT instances through pure decimation (no backtracking) but the process, as one might expect, quickly fails on harder instances. We thus obtain a strict order in terms of decimation strategies, from the least effective one, which is a random variable setting, to local heuristics used in modern SAT solvers, to BP, BP$^\kappa$, and finally, SP.

As noted above, the decimation process can fail because one may inadvertently create an unsatisfiable subproblem. Consider, for example, the Boolean constraint C defined as $(x_1 \texttt{ or } \neg x_2 \texttt{ or } x_3)$. If the decimation heuristic assigns a value to both x_1 and x_2, C is either satisfied and disappears, or reduced to the "unit" clause x_3. If another clause C' similarly creates the unit clause $\neg x_3$, the subproblem becomes unsatisfiable. This observation suggests that unit clauses may play an important role in determining the success or failure of a decimation strategy. Indeed, as we show, what distinguishes message passing based heuristics from standard search heuristics on the hardest random 3-SAT instances is that the former virtually avoid the creation of unit clauses.

Finally, we do see that despite its promising performance, BP$^\kappa$ doesn't quite match SP's performance on the hardest set of random instances. We investigate this behavior in terms of an indirect measure: the evolution of the hardness of the resulting formula as decimation proceeds. Our results reveal an interesting phenomenon, where both BP$^\kappa$ and SP start by gradually making the formula easier, but while SP continues to do so, BP$^\kappa$ soon makes the formula much harder to solve than it originally was. Such studies have not been performed earlier to our knowledge, and shed light into the not-so-well understood differences between BP and SP for solving CSPs.

2. BACKGROUND

We are concerned with the Boolean satisfiability problem (SAT), which is well-known to be NP-complete and asks the following question: Given a formula F over n Boolean variables $x_1, \ldots, x_n$, is there a TRUE/FALSE (equivalently, 1/0) assignment to each x_i such that F evaluates to TRUE? Such an assignment is referred to as a satisfying assignment for or solution of F. If no such truth assignment exists, F is called unsatisfiable.

We are interested in formulas in the Conjunctive Normal Form (CNF), where F is specified as a conjunction (AND, $\wedge$) of clauses, each clause is a disjunction (OR, $\vee$) of literals, and each literal is a variable x_i or its negation $\neg x_i$. An example of a CNF formula is $F = (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$. A clause containing k literals is called a k-clause. 1-clauses are often referred to as unit clauses. When every clause of a formula is a k-clause, we have a k-CNF formula, and the corresponding satisfiability problem is referred to as the k-SAT problem.

Our study is of random k-SAT formulas F, characterized by an important parameter α defined as the ratio of the number of clauses to the number of variables in F. A random k-SAT formula over n variables at ratio α is generated by selecting αn clauses uniformly at random from the set of all clauses over the n variables. The resulting distribution is often denoted by $\mathbb{F}(n, \alpha n)$. Random 3-SAT formulas, in particular, have been studied extensively in both theory and practice. They empirically exhibit a phase transition at the ratio $\alpha_c \approx 4.26$ in the sense that nearly all formulas below this threshold are found to be satisfiable, while those above are found to be unsatisfiable. This phase transition also coincides with a sharp peak in the computational complexity of finding solutons using state-of-the-art methods around this critical ratio. An interesting and well-researched area, then, is to device techniques that push the limit of feasibility of various solution methods as close to the phase transition region as possible.

2.1 Solving SAT by Decimation

The computational hardness of SAT coupled with its numerous applications have led to the development of dozens of progressively faster SAT solvers, which essentially fall into two categories: branch-and-backtrack search (i.e., DPLL-based methods) and local search. Both of these work with partial or sub-optimal candidate solutions, and attempt to iteratively improve these candidate solutions until a satisfying assignment is found. A key aspect of both is the ability to make possibly wrong decisions early on in the search and later rectify the mistakes as the search progresses.

A conceptually much simpler method of finding a solution to a (satisfiable) formula F is **decimation**: select a literal ℓ of F according to some heuristic, set $\ell =$ TRUE, simplify F by removing all clauses containing ℓ and shrinking those containing $\neg\ell$, and repeat. Decimation is said to *fail* if one ends up with a subformula F' of F that contains an empty (and thus unsatisfiable) clause. To be of any interest, the heuristic used for decimation must, in particular, satisfy the property that if there is a unit clause ℓ in F', then the heuristic suggests that ℓ (or some other unit literal) be set to TRUE. Since there is no "repair" mechanism in decimation, its success as a solution technique for SAT relies critically on the correctness of every single variable setting made during the process until a solution is found. As one might expect, decimation does not work very well in solving many formulas of interest. However, when one considers random 3-SAT formulas, the picture is quite positive. Analysis of decimation with various

local heuristics has provided formal lower bounds on the satisfiability threshold (e.g. Achlioptas et al., 2005, Hajiaghayi and Sorkin, 2003, Kaporis et al., 2006). Further, decimation with SP as the guiding heuristic (Mézard et al., 2002) has turned out to not only work surprisingly well for 3-SAT, it is the best method that we know of for this problem.

2.2 Belief Propagation & Survey Propagation

Belief propagation (BP) (Pearl, 1988) is an iterative algorithm for computing marginal probabilities of the nodes of a graphical model such as a Bayesian network. It works by iteratively solving a set of mutually recursive equations on variables that represent *messages* among the nodes of the graphical model. In the context of SAT, one can think of a CNF formula F with n variables and m clauses as a factor graph G_F with n variable nodes taking value in $\{0, 1\}$ and m factors. Each variable node of G_F corresponds naturally to a variable of F. Each factor f_C corresponds to a clause C of F and is connected to the variable nodes corresponding to the variables appearing in C. f_C evaluates to 1 for a certain valuation of the variable nodes iff C evaluates to TRUE for the corresponding truth assignment to the variables.

We are interested, for example, in computing $\Pr[x_1 = 1]$ for our example formula F: this represents the marginal probability of x_1 being 1 when all solutions to F are chosen with equal probability. Equivalently, it is the fraction of solutions of F that have $x_1 = 1$. We use the term **magnetization** to refer to the difference between the marginal probabilities of a variable or literal being TRUE and it being FALSE. We refer the reader to standard texts (e.g. Pearl, 1988) for details of the iterative equations used to compute such marginal probabilities.

The BP iterations have been proved to converge and provide accurate answers essentially only when applied to problems with no circular dependencies, such as SAT instances with no loops in the factor graph. Empirically, BP has been shown to provide very good approximations even in some domains which do not satisfy this condition (Murphy et al., 1999). Unfortunately, the number of domains where this is true is rather small, and BP fails on many problems of interest as well as on hard random k-SAT instances. Making progress in this direction is an open problem with active interest (e.g., Hsu and McIlraith, 2006, Pretti, 2005).

Survey propagation (SP) is a similar iterative method designed specifically for solving hard SAT instances, and is in fact the most successful method when dealing with random k-SAT instances. While introduced initially from a very different perspective, namely the cavity method in statistical physics (Mézard et al., 2002), it was later shown to be an instance of BP applied to a modified problem (Braunstein and Zecchina, 2004, Maneva et al., 2007). This modified problem is that of finding "covers" of the formula rather than satisfying assignments. The notion of covers (Maneva et al., 2007) is based on generalized assignments to Boolean variables, i.e., assignments in $\{0, 1, *\}$, and each such cover is supposed to represent a whole set of assignments that are close to each other in Hamming distance (loosely called "clusters"). The iterative equations of SP thus compute marginal probabilities of a variable being 0, 1, or $*$ in the covers of F. Covers have many interesting properties; we refer the reader to Maneva et al. (2007), Kroc et al. (2007) for details.

2.3 Convergent BP

As mentioned above, BP equations often do not converge to a fixpoint for hard enough problem instances. Moreover, problem instances that we study have tens or hundreds of thousands of variables and clauses, which is several orders of magnitude more than what off-the-shelf probabilistic inference programs can handle. Even improvements of the belief propagation technique that allow it to be used on a wider set of problems, such as Generalized Belief Propagation (Yedidia et al., 2000) or Loop Corrected Belief Propagation (Mooij et al., 2007), are not scalable enough for our purposes. The problem of very slow convergence on hard instances seems to plague also approaches based on other methods for solving the BP equations than the simple iteration scheme, such as the convex-concave procedure introduced by Yuille (2002). Finally, in our context, the speed requirement is accentuated by the need to use marginal estimation repeatedly during the decimation process.

We consider a parameterized variant of BP which is guaranteed to converge when this parameter is small enough, and which imposes no additional computational cost over the traditional BP[1]. As we will shortly see, this "damped" variant of BP provides much more useful information than BP iterations terminated without convergence, and, surprisingly, performs very well as a decimation heuristic for random k-SAT. We refer to this particular way of damping the BP equations as $\mathbf{BP}^{\kappa}$, where $\kappa \geq 0$ is a real parameter that controls the extent of damping in the iterative equations. Knowing the exact details of the corresponding update equations is not essential for understanding the rest of this paper. The interested reader is referred to Figure 5 in the Appendix for the update equations for SAT. The same variant of BP was used earlier (Kroc et al., 2008) in a different context; we include it here for completeness and provide some further insights.

When $\kappa = 1$, BP^{κ} is identical to regular belief propagation. On the other hand, when $\kappa = 0$, the equations surely coverge in one step to a unique fixed point and the marginal estimates obtained from this fixed point have a clear probabilistic interpretation in terms of a local property of the variables (we defer formal details on this to the Appendix; see Proposition 2 and the related discussion). The κ parameter thus allows one to interpolate between regimes where the equations correspond to the original BP providing global information if the iterations converge ($\kappa = 1$), and regime where the iterations surely converge but provide only local information (for $\kappa = 0$).

The resulting marginals for $\kappa \neq 1$ do not correspond to the actual BP marginal estimates, but we believe that the information obtained retains relevance to the true marginals. This is shown in Figure 1, where the quality of the estimate of solution marginals obtained by running standard BP and cutting it off because of non-convergence (left pane) is compared to the quality of marginals as obtained by BP^{κ} for $\kappa = 0.9$ (right pane), for a hard random 3-SAT formula with $5,000$ variables and $21,000$ clauses. The scatter-plots in both cases contrast the computed marginals with marginals obtained by *sampling* satisfying assignments for the formula and computing empirical marginals. What is actually plotted in Figure 1 is the magnetization for each variable, i.e.,

[1]Similar but distinct parametrization was proposed by Pretti (2005).

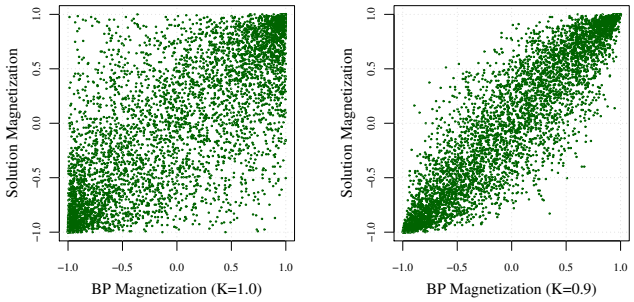

Figure 1: Quality of solution marginals computed by BP (left) and BP$^\kappa$ (right).

the difference between the probability of a variable being TRUEand the probability of it being FALSE. If the computed marginals did correspond exactly to the sampled solution marginals, the plot would be a diagonal line. The further away from the diagonal the points lie, the less accurate the estimate is. The figure shows that BP$^\kappa$ is more accurate in estimating the solution marginals than unconverged BP. This is especially true in the *extreme magnetization* regions at the top-right and bottom-left of the scatter-plots, which is the region of interest when using these estimates as a decimation heuristic. We refer the reader to Kroc et al. (2007) for a similar comparison between BP and SP, where SP is shown to also compute very good marginal estimates in the extreme magnetization regions.

3. DECIMATION STRATEGIES

In this section, we consider various strategies that can be used as a heuristic for the decimation process for solving SAT instances. All of these heuristics give the highest priority to setting to TRUE any literal appearing in a *unit clause*; such a literal much be satisfied in any case in order to obtain a solution. When no unit clauses are present, we consider five alternatives:

1. Random: Choose a variable x uniformly at random and set it to TRUE or FALSE uniformly at random. We note that on random formulas, this is equivalent to the heuristic employed by the state-of-the-art SAT solver `Minisat` (Eén and Sörensson, 2005) before it encounters its first contradiction.

2. DPLL: Follow the heuristic used by the DPLL-based SAT solver `zChaff` (Moskewicz et al., 2001) before the first backtrack. This heuristic prefers the variable that occurs the most often, and sets it to TRUE or FALSE based on majority occurrence. If positive and negative occurrences are equally numerous, it goes for the polarity that creates the *most* unit literals.

3. BP: Compute variable magnetizations using BP (report failure if it does not converge). Set the variable with the most extreme magnetization accordingly.

4. BP$^\kappa$: Similar to BP, but with magnetizations obtained from BP$^\kappa$ (we used $\kappa = 0.9$).

5. SP: Similar to BP, but with magnetizations obtained from SP. In case SP finds the formula in a "paramagnetic" state where it can provide no useful information (see Mézard et al., 2002), the formula is easily solved by the local search SAT solver `Walksat`.

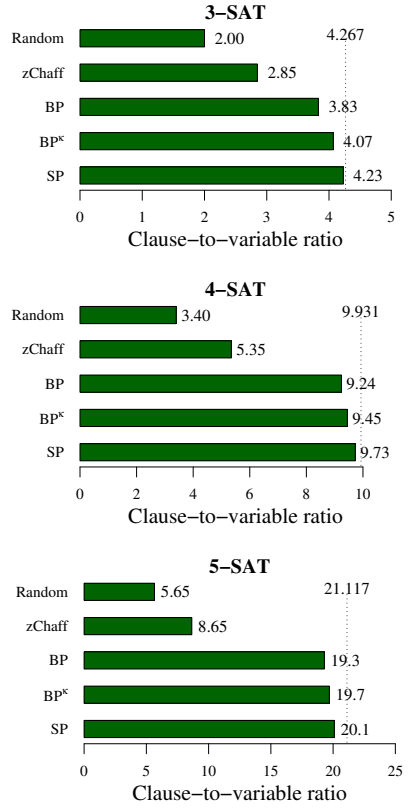

Figure 2: Various decimation heuristics on random 3-SAT, 4-SAT, and 5-SAT formulas

We evaluate these heuristics on random 3-SAT, 4-SAT, and 5-SAT problems, parameterized by the clause-to-variable ratio α. Recall that these instances become progressively harder as α increases; the computationally hardest instances are found near the α value of 4.26 for 3-SAT, 9.93 for 4-SAT, and 21.12 for 5-SAT (Mertens et al., 2006). For each heuristic and each $k = 3, 4, 5$, we measure the value of α up to which decimation with this heuristic solves at least 50% of the random formulas generated (we use formulas with $10,000$ variables). Figure 2 gives a summary of the results.

As one would expect, all five heuristics can solve these formulas for small enough α, but start to break down as α grows. The three propagation-based methods — BP, BP$^\kappa$, and SP — go much farther than the two local information heuristics — random and zChaff. While the zChaff heuristic solves 3-SAT instances for $\alpha \leq 2.85$, BP pushes this to 3.83 and SP as far as 4.23. A surprising finding is that κ-damping improves the performance of BP considerably, bringing it half-way from BP to SP. As k grows, i.e., one goes from 3-SAT to 4-SAT and 5-SAT, the difference in performance between propagation-based methods and local information methods becomes even more extreme. In particular, the random and zChaff heuristics stop working relatively much earlier.

Among various propagation-based heuristics themselves, BP interestingly appears to get closer to SP in performance as k increases. BP$^\kappa$ is able to push the limits of BP signif-

icantly further, for example, to $\alpha \leq 4.07$ for 3-SAT. Nevertheless, in all cases, SP is able to solve instances harder than the hardest instances solved by any other technique, such as those at $\alpha = 4.2$ for 3-SAT. This naturally raises the question: what is it that SP is doing differently than these other heuristics? In the next section we will focus on identifying measures that distinguish SP from the other techniques, in particular, from BP$^\kappa$.

4. OBSERVABLE DIFFERENCES IN DECIMATION STRATEGIES

In this section we focus on identifying key differences between various heuristics as decimation progresses. Despite the obvious difference that one strategy solves a given instance while other does not, it is surprisingly difficult to identify features of the decimation process that would clearly distinguish between the strategies. For example, simple measures like the ratio of 2-clauses to 3-clauses in the decimated formula, or the frequency of positive vs. negative occurrences of variables, do not show a significant difference, especially amongst various message passing heuristics. We identify two measures that do show a clear difference, namely, how many unit clauses the heuristic generates and how hard-to-solve does the decimated formula become.

4.1 BP and SP Avoid Creating Unit Clauses

Quite surprisingly, our experiments reveal that a key difference between decimation based on message passing heuristics (both SP and BP) and other heuristics considered is that message passing heuristics avoid the creation of unit clauses. Since unit clauses allow for simplification of the formula, DPLL solvers and decimation strategies perform *unit propagation*, that is setting literals in unit clauses to true, possibly creating new unit clauses, until all unit clauses are resolved. This chaining effect may lead to a contradiction, and while this is fine for a DPLL solver with a possibility of backtracking, the increased chance of contradiction may be fatal for decimation strategies. Minimizing the number of unit clauses is thus a desirable property of decimation strategies. Figure 3 shows the cumulative number of unit clauses generated as decimation progresses, for a random 3-SAT formula with $10,000$ variables at ratio $\alpha = 4.2$. The decimation performed by randomly setting variables is not careful about avoiding unit clauses, and indeed after setting 10% of the variables the decimation is dominated by unit propagation, which can be seen by observing that the slope of the "Random" curve in the plot is one. That there is actually a contradiction (i.e., the decimated formula has already become unsatisfiable) is detected only after nearly 50% of the variables have been set, marked by the cross in the figure where the "Random" curve ends. The situation is similar for the DPLL heuristic, `zChaff`, but the contradiction is detected much sooner. BP inspired decimation results in a significantly lower number of unit clauses (BP$^\kappa$ was used for $\kappa = 0.9$ because standard BP does not converge for such α), at least till roughly 40% of the variables have been set, which is the point where SP usually hands the formula over to `Walksat`. Of course, when enough variables are fixed, it becomes nearly impossible to avoid unit propagation using any heuristic, and BP$^\kappa$ shows this trend after 40% of the variables are set. Finally, SP inspired decimation results in virtually no unit propagation. There were only three unit clauses created, and this number diminishes for larger formulas. Note that the ordering of the curves from left to right corresponds to the order in Figure 2, confirming that avoiding unit propagation is desirable when solving formulas by decimation (in contrast to solving by search with backtracking).

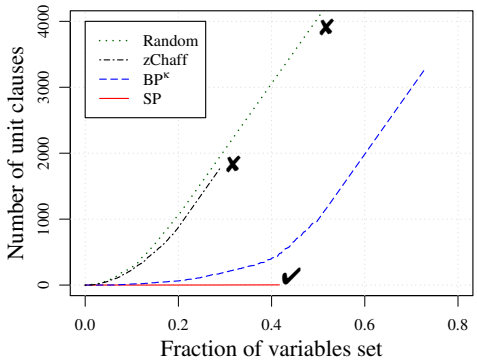

Figure 3: Cumulative number of unit clauses generated during decimation

Creating no unit clauses means that no trivial contradiction (i.e., two unit clauses requiring a variable to be set opposing ways) is derived. But how do BP$^\kappa$ and SP manage to avoid unit propagation for such a long time during decimation? The following proposition studies this question in the ideal case, and shows that if the message passing method was able to compute marginals exactly, unit propagation would be naturally avoided. Since BP/SP inspired decimation always fixes variables with the highest magnetization, Proposition 1 suggests that unit clauses are highly likely to be avoided. Of course, marginals computed by BP and SP on complex problems are not exact, but as remarked in Section 2.3, both BP$^\kappa$ and SP do a very good job of estimating the extreme marginals.

Proposition 1. *Let μ be the exact solution marginals of a formula F. That is, for each variable x, $\mu(x = 0) = P[\sigma(x) = 0 \mid \sigma$ is a solution$]$, where σ is some variable valuation with a uniform prior, and analogously for $\mu(x = 1)$. If the maximum marginal for 0 or 1 is unique, i.e., the most magnetized literal is unique, then setting this literal to* TRUE *will not create any unit clauses.*

PROOF. Recall that magnetization of a literal is the difference between its marginal probabilities of being TRUE and FALSE. Let x be the unique most magnetized literal of F. For contradiction, assume that setting $x = 1$ creates a unit clause (y). This means that there must have been a binary clause $(\neg x \vee y)$ in F. Now any solution σ of F must, by definition, have the property that every clause has at least one satisfying literal under valuation σ. In particular, because of our binary clause, it must be the case that $\sigma(x) = 1 \Rightarrow \sigma(y) = 1$. But this means that $\mu(x = 1) \leq \mu(y = 1)$ and $\mu(y = 0) \leq \mu(x = 0)$, and thus $\mu(y = 1) - \mu(y = 0) \geq \mu(x = 1) - \mu(x = 0)$. This is a contradiction with x being the unique most magnetized literal. □

An analogous proposition can be stated for any Constrain Satisfaction Problem, not only SAT. In particular, it follows by a similar analysis that decimation by exact cover

marginals tries to avoid creation of unit clauses. This, coupled with the observation in Kroc et al. (2007) that SP is remarkably accurate in computing the cover marginals, explains why SP creates so few unit clauses.

4.2 Evolution of Problem Hardness

So far we have seen that quite clearly, decimation based on probabilistic reasoning is better than decimation based on standard DPLL heuristics in the domain of random SAT instances. Furthermore, compared to BP$^\kappa$, SP inspired decimation is able to solve formulas much closer to the phase transition threshold, where the problem hardness quickly increases with α. We now focus on what happens during the decimation process on these hard formulas, digging deeper into the difference between BP$^\kappa$ and SP.

In order to understand the gradual effect of decimation as more and more variables are set, we study the evolution of the problem hardness in terms of how difficult it is to solve the decimated formula using `Walksat`, a local search SAT solver suitable for random SAT instances. The hardness is measured as the average number of variable-flips required by `Walksat` to solve the decimated formula generated by SP or BP$^\kappa$, with $\kappa = 0.9$. (Standard BP equations did not converge, and the other heuristics quickly made the problem much harder.) Figure 4 summarizes the general trend through the results on a 5000 variable 3-SAT formula at $\alpha = 4.2$. The plot shows the hardness measure on the y-axis, and the fraction of variables set on the x-axis. The solid lines correspond to decimation by BP$^\kappa$ and SP only, while the dashed lines correspond to first setting 20% of the variables using one strategy, and then continuing with the other strategy.

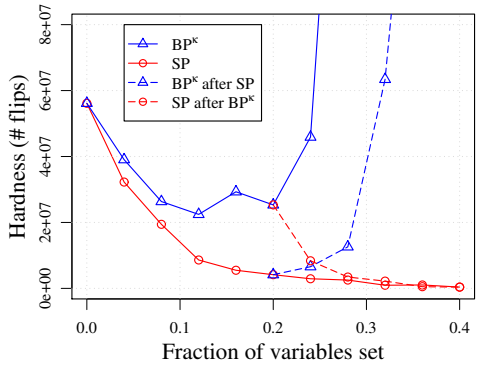

Figure 4: Hardness of subformulas created by BP$^\kappa$ and SP

We see that the decimated formulas generated by BP$^\kappa$ and SP behave quite differently with respect to their computational hardness. Initially, both SP and BP$^\kappa$ keep making the formula easier to solve, in this case till approximately 12% of the variables have been set. After this point, SP continues to make the formula easier to solve, while BP$^\kappa$ starts increasing the hardness, which eventually surpasses the hardness of the original formula. Finally, the dashed lines show that this behavior is quite robust: running BP$^\kappa$ on an SP-decimated formula quickly makes the formula much harder, while running SP on a BP$^\kappa$-decimated formula reduces the hardness, and eventually solves the formula.

As seen in the figure, BP$^\kappa$ starts going "wrong" when approximately 12% of the variables are set. At that point, it would be helpful to be able to identify variables in the formula that SP sets, making the formula yet easier. It turns out that the proper variables are indistinguishable by several measures commonly used in local heuristics, such as the number of positive and negative occurrences in 2- and 3-clauses. This suggests that local heuristics, looking no deeper than these easy-to-obtain statistics, are unlikely to follow SP's path in making the formula easier by further decimation. On the other hand, the SP- and BP$^\kappa$-decimated formulas do eventually get more and more different in terms of their computational hardness. At this point it is unclear exactly why this happens.

5. CONCLUSION

SP was introduced by statistical physicists as a fairly complex technique for solving random k-SAT instances very efficiently, and was later understood as essentially a form of BP, albeit on somewhat more complex combinatorial objects than solutions. This raised a natural question: can the original BP framework, more well-known to computer scientists, itself provide a good solution technique for hard random k-SAT instances? Our experimental results provide an affirmative answer to this question. Simply terminating the iterative equations of BP when they don't converge and using the current values as estimates of true marginals already pushes the limit of decimation much further than local information based heuristics commonly employed in backtrack search SAT solvers. By using a parameterized convergent form of BP, the limit is pushed even further. The gap between BP and SP narrows as k increases, and heuristics based on message-passing in general begin to perform substantially better than other heuristics.

Acknowledgments

This work was supported by IISI, Cornell University (AFOSR grant FA9550-04-1-0151), DARPA (REAL grant FA8750-04-2-0216), and NSF (grants 0514429 and 0829861).

References

D. Achlioptas, A. Naor, and Y. Peres. Rigorous location of phase transitions in hard optimization problems. *Nature*, 435:759–764, 2005.

A. Braunstein and R. Zecchina. Survey propagation as local equilibrium equations. *J. Stat. Mech.*, P06007, 2004. URL http://lanl.arXiv.org/cond-mat/0312483.

M. Davis and H. Putnam. A computing procedure for quantification theory. *CACM*, 7:201–215, 1960.

M. Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *CACM*, 5:394–397, 1962.

N. Eén and N. Sörensson. MiniSat: A SAT solver with conflict-clause minimization. In *8th SAT*, St. Andrews, U.K., June 2005.

M. Hajiaghayi and G. B. Sorkin. The satisability threshold of random 3-SAT is at least 3.52. Technical Report RC22942, IBM Research Report, 2003. http://arxiv.org/pdf/math.CO/0310193.

E. I. Hsu and S. A. McIlraith. Characterizing propagation methods for boolean satisfiabilty. In *9th SAT*, volume 4121 of *LNCS*, pages 325–338, Seattle, WA, Aug. 2006.

A. C. Kaporis, L. M. Kirousis, and E. G. Lalas. The probabilistic analysis of a greedy satisfiability algorithm. *Random Structures and Algorithms*, 28(4):444–480, 2006.

L. Kroc, A. Sabharwal, and B. Selman. Survey propagation revisited. In *23rd UAI*, pages 217–226, Vancouver, BC, July 2007.

L. Kroc, A. Sabharwal, and B. Selman. Leveraging belief propagation, backtrack search, and statistics for model counting. In *5th CPAIOR*, volume 5015 of *LNCS*, pages 127–141, Paris, France, May 2008.

E. Maneva, E. Mossel, and M. J. Wainwright. A new look at survey propagation and its generalizations. *J. Assoc. Comput. Mach.*, 54(4):17, July 2007.

S. Mertens, M. Mézard, and R. Zecchina. Threshold values of random K-SAT from the cavity method. *Random Struct. and Alg.*, 28(3):340–373, 2006.

M. Mézard, G. Parisi, and R. Zecchina. Analytic and algorithmic solution of random satisfiability problems. *Science*, 297(5582): 812–815, 2002.

J. M. Mooij, B. Wemmenhove, H. J. Kappen, and T. Rizzo. Loop corrected belief propagation. In *Proc. 11th Intl. Conf. on AI and Statistics (AISTATS-07)*, 2007.

M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient SAT solver. In *38th DAC*, pages 530–535, Las Vegas, NV, June 2001.

K. Murphy, Y. Weiss, and M. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *15th UAI*, pages 467–475, Sweden, July 1999.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

M. Pretti. A message-passing algorithm with damping. *J. Stat. Mech.*, P11008, 2005.

B. Selman, H. Kautz, and B. Cohen. Local search strategies for satisfiability testing. In D. S. Johnson and M. A. Trick, editors, *Cliques, Coloring and Satisfiability: the Second DIMACS Implementation Challenge*, volume 26 of *DIMACS Series in DMTCS*, pages 521–532. Amer. Math. Soc., 1996.

J. S. Yedidia, W. T. Freeman, and Y. Weiss. Generalized belief propagation. In *NIPS*, pages 689–695, 2000.

A. L. Yuille. CCCP algorithms to minimize the Bethe and Kikuchi free energies: Convergent alternatives to belief propagation. *Neural Comput.*, 14(7):1691–1722, 2002.

APPENDIX

Convergent BP: The iterative update equations for the convergent form of belief propagation, BP^κ, are given in Figure 5. The only difference from the normal BP equations is the exponent κ in the updates for the Π messages.

The role of the parameter κ is to damp oscillations of the message values by pushing the variable-to-clause messages, Π, closer to 1. Intuitively speaking, the damping is realized by the function $y = x^\kappa$ for $\kappa < 1$. For inputs x that are positive and less than one, the function increases their value, or sets them to 1 in the case of $\kappa = 0$. As a result, after normalization, the Π values are less extreme. For $\kappa = 0$, we can obtain a probabilistic interpretation of the algorithm reminiscent of a local heuristic for SAT solving:

Proposition 2. *The system of BP^κ equations for $\kappa = 0$ converges in one iteration for any starting point, and the following holds for the resulting values μ_i (see Figure 5 for notation):*

$$\mu_i(1) \propto \prod_{b \in C^-(i)} \left(1 - 2^{-(|V(b)|-1)}\right)$$

$$\mu_i(0) \propto \prod_{b \in C^+(i)} \left(1 - 2^{-(|V(b)|-1)}\right)$$

Proof. For any initial starting point η_0 (with values in $[0,1]$), the first iteration sets all $\Pi^u = 1$ and $\Pi^s = 1$. This means $\eta_{a\to i} = (\frac{1}{2})^{|V(a)|-1}$ for all clauses a containing variable i. This is the fixed point η^*, because applying the updates again yields the same values. The rest follows directly from the form of the μ_i equations in Figure 5. □

Notation used in the figure: $V(a)$ is the set of all variables in clause a. For a variable i appearing in clause a, $C^u_a(i)$ is the set of all clauses in which i appears with the *opposite* sign than it does in a. $C^s_a(i)$ is the set of all clauses in which i appears with the *same* sign as it does in a except for a (thus $C^u_a(i)$, $C^s_a(i)$ and $\{a\}$ are disjoint and their union is the set of all clauses where i appears). $C^+(i)$ is the set of all clauses in which i appears unnegated. $C^-(i)$ is the set of all clauses in which i appears negated.

Messages from clauses to variables:

$$\eta_{a\to i} = \prod_{j \in V(a)\setminus i} \left[\frac{\Pi^u_{j\to a}}{\Pi^u_{j\to a} + \Pi^s_{j\to a}}\right]$$

Messages from variables to clauses:

$$\Pi^u_{i\to a} = \left(\prod_{b \in C^s_a(i)} (1 - \eta_{b\to i})\right)^\kappa$$

$$\Pi^s_{i\to a} = \left(\prod_{b \in C^u_a(i)} (1 - \eta_{b\to i})\right)^\kappa$$

Computing marginals from a fixed point η^* of the message equations:

$$\mu_i(1) \propto \prod_{b \in C^-(i)} (1 - \eta^*_{b\to i})$$

$$\mu_i(0) \propto \prod_{b \in C^+(i)} (1 - \eta^*_{b\to i})$$

$\mu_i(1)$ is the probability that variable i is positive in a random satisfying assignment, and $\mu_i(0)$ is the probability that it is negative.

Figure 5: Modified belief propagation update equations.

The intuitive interpretation of the values of μ_i in Proposition 2 is the following: assuming independence of variable occurrences in clauses, the value $2^{-(|V(b)|-1)}$ can be interpreted as the probability that clause $b \in C(i)$ is unsatisfied by a random truth assignment if variable i is not considered. $\mu_i(1)$ is thus the probability that all clauses b in which variable i appears negated are satisfied, and analogously for $\mu_i(0)$. Since clauses not considered in the expressions for μ_i are satisfied by i itself, the resulting values of μ_i are proportional to the probability of all clauses where i appears being satisfied with the particular setting of variable i, when all other variables are set randomly. This is very local information, and depends only on what clauses the variable appears in negatively or positively. The parameter κ can thus be used to tune the tradeoff between the ability of the iterative system to converge and the locality of the information obtained from the resulting fixed point.