Technical Report

UCAM-CL-TR-883
ISSN 1476-2986

Number 883

UNIVERSITY OF CAMBRIDGE

Computer Laboratory

Survey propagation applied to weighted partial maximum satisfiability

Richard Russell, Sean B. Holden

March 2016

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

http://www.cl.cam.ac.uk/

ISSN 1476-2986

Survey propagation applied to weighted partial maximum satisfiability

Richard Russell
r.a.russell.04@cantab.net

Sean B Holden
sbh11@cam.ac.uk

Abstract

We adapt the survey propagation method for application to weighted partial maximum satisfiability (WPMax-SAT) problems consisting of a mixture of hard and soft clauses. The aim is to find the truth assignment that minimises the total cost of unsatisfied soft clauses while satisfying all hard clauses. We use fixed points of the new set of message passing equations in a decimation procedure to reduce the size of WPMax-SAT problems. We evaluate this strategy on randomly generated WPMax-SAT problems and find that message passing frequently converges to non-trivial fixed points, and in many cases decimation results in simplified problems for which local search solvers are able to find truth assignments of comparable cost faster than without decimation.

1 Introduction

Satisfiability problems are decision problems expressed in Boolean logic. A problem has a set of clauses, each with one or more literals. A solution is a truth assignment to the variables making at least one literal true in every clause. A problem is either *satisfiable* or *unsatisfiable* depending on whether a solution exists.

We address a subset of these problems known as random k-SAT. Here, we sample uniformly from the non-trivially satisfiable clauses of k literals for a fixed set of variables. Given such a formula, with M clauses and N variables, there is strong empirical evidence that the ratio $\alpha = M/N$ is an important indicator of satisfiability as $N \to \infty$; it is conjectured that there is a point, α_k^t, known as the *satisfiability threshold*, which strongly determines whether a randomly generated k-SAT problem is likely to be satisfiable. Problems in the region $\alpha < \alpha_k^t$ are under-constrained and almost always satisfiable, whereas those in the region $\alpha > \alpha_k^t$ are over-constrained and almost always unsatisfiable. Recent theoretical results for random 3-SAT give the following bounds: $3.52 \leq \alpha_3^t$ [1] and $\alpha_3^t \leq 4.4898$ [2]. The survey propagation method predicts that $\alpha_3^t \approx 4.267$ [3] and empirical observations suggest that $\alpha_3^t \approx 4.26$ [4].

In practice, algorithms for determining satisfiability often have the longest running times for problems close to α_k^t [5, 6]. Thus, problems sampled near α_k^t are a source of challenging benchmark instances, and critical for understanding and improving upon the shortcomings of existing algorithms.

A *maximum satisfiability (Max-SAT)* problem relaxes the requirement that all clauses be satisfied. Instead we seek a truth assignment maximizing the number of satisfied clauses. Furthermore, *weighted Max-SAT* problems specify numeric weights for the clauses and ask for the truth assignment maximizing the sum of weights of satisfied clauses.

In this paper, we address a variation of Max-SAT known as *weighted partial Max-SAT (WPMax-SAT)*. A WPMax-SAT problem has hard and soft clauses, where soft clauses have an integer-valued weight. An optimal solution is a truth assignment that satisfies all hard clauses and maximises the sum of weights of satisfied soft clauses.

Recently the *survey propagation* method [3, 7] has been proposed, based on ideas from statistical physics, and can be used to solve a higher proportion of the hardest satisfiability problems in random k-SAT. Survey propagation estimates, for each variable, the probability that it should take the value *true*, the value *false* or is unconstrained in a randomly selected satisfying truth assignment. These probabilities are used to identify variables much more likely to take one value than another. The most strongly biased variables are set to their more likely values in a process known as *decimation*. This yields a simplified problem that is hopefully easier to solve by a more conventional method.

The ideas underlying survey propagation can be adapted to Max-SAT, leading to an algorithm known as *SP(y)* [8]. Instead of treating all assignments as equal, SP(y) defines a distribution over truth assignments which peaks for those satisfying the greatest number of clauses. This alters the probability estimates in order to direct the decimation procedure towards truth assignments satisfying as many clauses as possible.

The SP(y) algorithm has two shortcomings when solving WPMax-SAT problems. First, it does not distinguish between *hard* and *soft* clauses: all clauses are regarded as soft. Second, all clauses are assumed to have weight 1, which limits expressiveness; for example it cannot model constraints having different importance.

Our main contributions in this paper are:

- An extension to the survey propagation equations applicable to WPMax-SAT problems with hard and soft clauses, together with soft clauses with non-uniform integer-valued costs.
- An empirical evaluation of a decimation procedure that uses this new set of equations on random WPMax-3SAT formulas. We find that decimation often yields simplified problems, in the sense that local search can find a truth assignment of comparable cost earlier than it can on an undecimated instance.

2 Survey propagation for WPMAX-SAT

We first describe the WPMax-SAT problem. Let $\mathbb{B} = \{0, 1\}$ denote Boolean values and $\mathbb{N}^+$ the positive integers. A WPMax-SAT problem with N variables $V = \{X_i\}_{i=1}^N$ uses a subset of the literals $L = V \cup \{\neg X_i \mid X_i \in V\}$ to form M clauses $C = \{C_a \subseteq L \mid \neg\exists i.\neg X_i \in C_a \text{ and } X_i \in C_a\}_{a=1}^M$. Each clause C_a has a weight $w_a \in \mathbb{N}^+ \cup \{\top\}$ denoting the cost of violating that clause. A clause C_a is *hard* if its weight $w_a = \top$; otherwise, it is a *soft* clause. A clause C_a is *satisfied* by a truth assignment $\mathbf{t}\colon V \to \mathbb{B}$ if there is at least one literal $\ell \in C_a$ for which $[\![\ell]\!]_\mathbf{t} = 1$ where for any i, $[\![X_i]\!]_\mathbf{t} = \mathbf{t}(X_i)$ and $[\![\neg X_i]\!]_\mathbf{t} = 1 - \mathbf{t}(X_i)$; otherwise, the clause is *unsatisfied* or *violated*. Furthermore, we define for all C_a

$$[\![C_a]\!]_\mathbf{t} = \begin{cases} 1, & \text{if } \exists \ell \in C_a.[\![\ell]\!]_\mathbf{t} = 1. \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

A truth assignment $\mathbf{t}^*$ is a *solution* if for all clauses $C_a \in C$ with $w_a = \top$, $[\![C_a]\!]_{\mathbf{t}^*} = 1$. Additionally, $\mathbf{t}^*$ is an *optimal* solution if for any other truth assignment $\mathbf{t}'$ that is also a solution,

$$\sum_{C_a \in C \text{ s.t. } w_a \neq \top} (1 - [\![C_a]\!]_{\mathbf{t}^*}) w_a \leq \sum_{C_a \in C \text{ s.t. } w_a \neq \top} (1 - [\![C_a]\!]_{\mathbf{t}'}) w_a.$$

2.1 Factor graph representation

We model a WPMax-SAT problem as a *factor graph* [9] G: a bipartite graph representation of a function $f\colon \mathbb{B}^N \to \mathbb{R}$, where f can be factorised into a product of smaller factors $\varphi_a\colon \mathbb{B}^{|C_a|} \to \mathbb{R}$ as $f(\mathbf{x}) = \prod_a \varphi_a(\mathbf{x}_a)$. Each variable $X_i \in V$ and factor φ_a is represented in G as a variable and factor node, respectively. As there is a one-to-one mapping between factor nodes and clauses, we refer to factor nodes as *clause* nodes henceforth. We also use i, j and k as placeholders for variable nodes and a, b and c as placeholders for clause nodes, unless otherwise stated; here, i denotes the variable node for X_i and a denotes the clause node for clause C_a.

Graph G only has edges between clause and variable nodes; namely, the undirected edges $\{(i,a) \mid X_i \in C_a \text{ or } \neg X_i \in C_a\}$. We use the notation of Mézard and Montanari [10] to denote the set of neighbouring variable nodes of clause a as ∂a and the set of neighbouring clauses of variable i as ∂i. We also split ∂i into four sets that are a function of a clause a connected to i:

1. $H_i^u(a)$ – the set of hard clauses in C where variable X_i has an opposite sign to that taken in C_a.
2. $H_i^s(a)$ – the set of hard clauses in C, excluding C_a, where variable X_i has the same sign to that taken in C_a.
3. $S_i^u(a)$ – the set of soft clauses in C where variable X_i has an opposite sign to that taken in C_a.
4. $S_i^s(a)$ – the set of soft clauses in C, excluding C_a, where variable X_i has the same sign to that taken in C_a.

G models the Gibbs-like function $g(\mathbf{x}) = \prod_a e^{-E_a(\mathbf{x}_a)}$, where a ranges over each clause in our formula such that

$$E_a(\mathbf{x}_a) = \begin{cases} 0 & \text{if } \mathbf{x}_a \text{ satisfies } C_a, \\ \infty & \text{if } \mathbf{x}_a \text{ does not satisfy } C_a \text{ and } w_a = \top, \\ w_a & \text{otherwise.} \end{cases} \tag{2}$$

For tree-structured (cycle-free) factor graphs the *max-product algorithm* can be applied to find a variable assignment maximizing the function. In general, we cannot rely on G being tree-structured; nevertheless, in practice max-product applied to non-tree structured graphs can yield useful results, and is then referred to as a *loopy* variant of the algorithm.

As the log function is monotonic, maximising $g(\mathbf{x})$ is equivalent to minimising $E(\mathbf{x}) = \sum_a E_a(\mathbf{x}_a)$. Thus, max-product can be rewritten as the *min-sum algorithm* [9], which is of interest as the survey propagation equations can be derived from it [10, Chapter 19]; however, survey propagation has a more intuitive explanation based on the related *warning propagation* procedure, which we describe first. In Section 2.3 we return to the min-sum algorithm, considering the effect of introducing non-uniform weights for soft clauses and how this affects exchanged messages.

2.2 Warning propagation

Warning propagation [7] aims to simplify SAT problems using the factor graph G. A *warning* is a message sent between a variable and a clause node. A variable i sends a warning to clause a to indicate that i cannot satisfy a based on the messages it has received from i's other neighbours. A clause a sends a warning to variable i to indicate that i must take the assignment satisfying a as the other neighbouring variable nodes of a cannot satisfy it.

Each edge in G connects a variable and a clause node, communicating a message along that edge in each direction. Messages are initialised randomly to either 0 or 1, where 1 indicates a warning and 0 the absence of a warning. Repeated passes are made along all edges, in a randomised order, computing whether a warning should be sent based upon the other warnings being received. This is repeated until convergence: when all edges can be visited with no change needed to the warnings being sent. This final set of warnings is a *fixed point* of the message passing equations and may be used in *warning inspired decimation*; here, in the absence of conflicting warnings, variables that receive warnings can be set to their appropriate values. This reduces the size of the problem, and what remains is passed to another satisfiability solving procedure to complete the solution.

Survey propagation assumes that there may be many fixed points of the warning propagation equations and seeks to estimate the probability, referred to as a *survey*, of a warning being exchanged between two nodes.

Warning propagation is a specialization of the more general *min-sum* algorithm for binary-valued variables when we only want to find the minimising truth assignment. We now discuss the connection of the min-sum equations with warning propagation and show how the survey propagation equations change for WPMax-SAT.

2.3 Min-sum equations

The min-sum equations are message passing equations, similar to those for warning propagation, used to find a truth assignment minimizing the objective function. Instead of sending warnings, messages exchanged over an edge of G estimate the lowest 'energy' achievable by a truth assignment in the subtree formed by removing the edge in consideration. The equations are as follows,

$$u_{i\to a}(x_i) = \sum_{b\in\partial i\setminus a} v_{b\to i}(x_i), \tag{3}$$

$$v_{a\to i}(x_i) = \min_{\mathbf{x}_{\partial a\setminus i}} \left[E_a(\mathbf{x}_{\partial a}) + \sum_{j\in\partial a\setminus i} u_{j\to a}(x_j) \right]. \tag{4}$$

If $\partial i \setminus a$ or $\partial a \setminus i$ are empty, the values of the messages are $u_{i\to a}(x_i) = 0$ or $v_{a\to i}(x_i) = E_a(x_i)$, respectively.

Consider Equation 4. As x_i values are binary, $|v_{a\to i}(0) - v_{a\to i}(1)| \leq w_a$. To see this note that if for any $j \in \partial a \setminus i$ either

1. $u_{j\to a}(0) = u_{j\to a}(1)$,
2. or $X_j = \arg\min_{x_j} [u_{j\to a}(x_j)]$ satisfies a,

then $\min$ will select an x_j that satisfies a and thus $E_a(\mathbf{x}_{\partial a}) = 0$ and $v_{a\to i}(0) = v_{a\to i}(1)$; otherwise, either (1) for some $j^* \in \partial a\setminus i$ we have $|u_{j^*\to a}(0) - u_{j^*\to a}(1)| = w^* \leq w_a$ and $\min$ can use j^* to satisfy clause a and keep $|v_{a\to i}(1) - v_{a\to i}(0)| \leq w^*$ or (2) for all $j \in \partial a\setminus i$ we have that $|u_{j\to a}(0) - u_{j\to a}(1)| > w_a$ and $X_j = \arg\min_{x_j} [u_{j\to a}(x_j)]$ does not satisfy a, which means that $\min$ chooses an $\mathbf{x}_{\partial a\setminus i}$ that does not satisfy a resulting in $|v_{a\to i}(1) - v_{a\to i}(0)| = w_a$.

Consequently, choosing variable assignments to minimise $E(\mathbf{x}) = \sum_a E_a(\mathbf{x}_a)$ does not require knowledge of the absolute values of the messages exchanged, but rather whether those values differ and by how much between $x_i = 0$ and $x_i = 1$. Furthermore, $v_{a\to i}(0) \neq v_{a\to i}(1)$ (by Equation 4) only if all variables $j \in \partial a \setminus i$ have messages such that $X_j = \arg\min_{x_j} [u_{j\to a}(x_j)]$ does not satisfy clause a.

This relates to the ideas of Section 2.2 in that a warning is sent from clause a to variable i when $v_{a\to i}(0) \neq v_{a\to i}(1)$. If this warning is ignored then at most w_a extra cost is incurred, otherwise no extra cost is incurred. Therefore, we could adopt the following worst-case assumption: whenever a variable i ignores a warning from a soft clause a then w_a is added to the cost of the truth assignment.

As in SAT, warnings from hard clauses must be heeded; however, for WPMax-SAT, in the absence of warnings from hard clauses a variable i must resolve potentially conflicting warnings from neighbouring soft clauses to decide whether to send a warning to clause a. To do this, it should calculate the *cavity field*

$$h = \sum_{b\in S_j^u(a)} |v_{b\to j}(1) - v_{b\to j}(0)| - \sum_{b\in S_j^s(a)} |v_{b\to j}(0) - v_{b\to j}(1)|\,. \tag{5}$$

Under our worst-case assumption, a warning from clause $b \in S_i^u(a)$ contributes w_b and a warning from clause $c \in S_i^s(a)$ contributes $-w_c$ to h. If $h > 0$, then a warning should be sent from i to a as the cost incurred from its other neighbours by an assignment not satisfying a is less than if it takes the assignment satisfying a.

3 Survey propagation equations

Survey propagation assumes that there are many fixed-point solutions to Equations 3 and 4 having a Gibbs distribution. Belief propagation then estimates the marginal probabilities of this distribution—those estimating the probability of a warning message being sent between two nodes for a randomly chosen fixed-point solution. The details can be found in [10, Chapter 19]; we now adapt that derivation for WPMax-SAT and explain the intuitions behind the resulting equations.

Let $\eta_{i\to a}$ be the probability of a warning from variable i to clause a and let $\hat{\eta}_{a\to i}$ be the probability of a warning from clause a to variable i. Hard clauses are subject to the survey propagation equations for SAT. Warnings from hard clauses must be obeyed and we assume that in a fixed-point solution to the min-sum equations we receive no conflicting warnings from hard clauses. The behaviour for hard clauses can be described using the quantities:

$$\mathcal{H}_i^u(a) = \left[1 - \prod_{b\in H_i^u(a)} (1-\hat{\eta}_{b\to i})\right] \prod_{b\in H_i^s(a)} (1-\hat{\eta}_{b\to i}), \tag{6}$$

$$\mathcal{H}_i^s(a) = \left[1 - \prod_{b\in H_i^s(a)} (1-\hat{\eta}_{b\to i})\right] \prod_{b\in H_i^u(a)} (1-\hat{\eta}_{b\to i}), \tag{7}$$

$$\mathcal{H}_i^0(a) = \prod_{b\in H_i^u(a)\cup H_i^s(a)} (1-\hat{\eta}_{b\to i}). \tag{8}$$

Here, $\mathcal{H}_i^u(a)$ and $\mathcal{H}_i^s(a)$ are the probabilities clause a receives a warning from variable i indicating that i is constrained by warnings from hard clauses to take a value that will, respectively, violate or satisfy clause a. $\mathcal{H}_i^0(a)$ is the third probability that i is not constrained by its neighbouring hard clauses other than a to take any value.

With no warnings from hard clauses, consider the probability of i receiving warnings from other neighbouring soft clauses that constrain i to take a value that will satisfy or violate clause a. Let $P_{i,a}(h)$ be the probability that the set of incoming warnings to i from clauses $b \in \partial i \setminus a$ according to Equation 5 is equal to h. Also, define

$$\mathcal{S}_i^u(a) = \sum_{h=1}^{\infty} P_{i,a}(h), \qquad \mathcal{S}_i^s(a) = \sum_{h=-1}^{-\infty} P_{i,a}(h). \tag{9}$$

Here, $\mathcal{S}_i^u(a)$ and $\mathcal{S}_i^s(a)$ are the probabilities that variable i receives warnings from soft clauses indicating that i is constrained under the decision criteria described to take a value that will, respectively, violate or satisfy a.

We can now write the survey propagation message passing equations for WPMax-SAT:

$$\eta_{i\to a} = \frac{\mathcal{H}_i^u(a) + \mathcal{H}_i^0(a)\mathcal{S}_i^u(a)}{\mathcal{H}_i^u(a) + \mathcal{H}_i^s(a) + \mathcal{H}_i^0(a)} \tag{10}$$

$$\hat{\eta}_{a\to i} = \prod_{j\in\partial a\setminus i} \eta_{i\to a}. \tag{11}$$

Here, $\hat{\eta}_{a\to i}$ is the probability that all other variables connected to a are sending warnings that they cannot satisfy a. This matches the original survey propagation equations; however, the equations differ in the handling of $\eta_{i\to a}$. This is the sum of the probabilities of two events: (1) that it receives at least one warning from another hard clause $b \in H_i^u(a)$ and (2) that it receives no warnings from hard clauses and a net warning from soft clauses that i appears in.

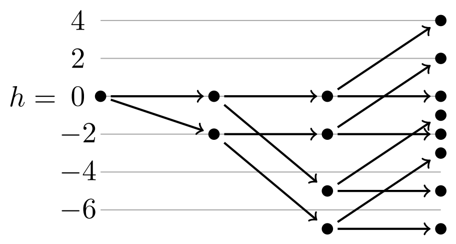

Figure 1: Possible energy levels from incoming warnings for variable X_i sending a message to clause a when $S_i^u(a)$ has a single clause of weight 4 and $S_i^s(a)$ has two clauses with weights 2 and 5.

3.1 Calculating $P_{i,a}(h)$

We use dynamic programming to efficiently compute $P_{i,a}(h)$ (Algorithm 1). Figure 1 provides an example. Without loss of generality, we consider clauses in $S_i^s(a)$ followed by clauses in $S_i^u(a)$; thus Figure 1 shows horizontal and downwards movements initially, then horizontal and upwards movements.

As fixed-point solutions to the min-sum equations are Gibbs distributed, the computation of $P_{i,a}(h)$ includes a term e^{-yw_b} for each clause $b \in \partial i \setminus a$ that contributed to h and had its warning ignored. The term y is an *inverse pseudo-temperature* parameter used to control the distribution. In the for loop on line 3, warnings received agree on the sign that X_i should take; however, from line 8 onwards, warnings received may conflict and if ignored can result in a clause being violated and so require the penalty term e^{-yw_b}. Where $h < 0$, warnings received from clauses in $S_i^u(a)$ will be ignored, but as we cross into the region $h > 0$ those warnings will be respected and instead the warnings already received from clauses $S_i^s(a)$ will be ignored; thus, penalty terms already included in $\tilde{P}_{i,a}^{(t)}(h)$ are now attributed to clauses in $S_i^s(a)$. This is the effect of the term $\theta(h, w_b)$, where

$$\theta(h,w) = \begin{cases} 0 & \text{if } h > 0, \\ w & \text{if } h + w < 0, \\ -h & \text{otherwise.} \end{cases} \tag{12}$$

3.2 Calculating bias and decimation

Once a fixed point of Equations 10 and 11 is found, we use the $\hat{\eta}_{a\to i}$ to calculate each variable's bias. Let $H_i^+ = \{a \mid X_i \in C_a \text{ and } w_a = \top\}$ and $H_i^- = \{a \mid \neg X_i \in C_a \text{ and } w_a = \top\}$. Slightly modified equations are then used to compute the bias of a variable.

$$\mathcal{H}_i^+ = \left[1 - \prod_{a \in H_i^+} (1 - \hat{\eta}_{a\to i})\right] \prod_{a \in H_i^-} (1 - \hat{\eta}_{a\to i}), \tag{13}$$

$$\mathcal{H}_i^- = \left[1 - \prod_{a \in H_i^-} (1 - \hat{\eta}_{a\to i})\right] \prod_{a \in H_i^+} (1 - \hat{\eta}_{a\to i}), \tag{14}$$

$$\mathcal{H}_i^0 = \prod_{a \in H_i^+ \cup H_i^-} (1 - \hat{\eta}_{a\to i}). \tag{15}$$

Algorithm 1 can be adjusted to calculate the distribution $P_i(h)$ in line 13 instead of $P_{i,a}(h)$: replace $S_i^s(a)$ and $S_i^u(a)$ in lines 3 and 8 with $S_i^+ = \{a \mid X_i \in C_a \text{ and } w_a \neq \top\}$ and $S_i^- = \{a \mid \neg X_i \in$

Algorithm 1: Calculating $P(h)$

```
Result: The distribution $P_{i,a}(h)$ of the net sum of weights $h$ of soft clauses in $S_i^s(a) \cup S_i^u(a)$
        sending warnings.
1  begin
2      $t \leftarrow 1$ ; $\tilde{P}^{(1)}(0) \leftarrow 1$
3      for $b \in S_i^s(a)$ do
4          for $h \in \{x \mid \tilde{P}^{(t)}(x) \neq 0\}$ do
5              $\tilde{P}^{(t+1)}(h) \leftarrow \tilde{P}^{(t+1)}(h) + (1 - \hat{\eta}_{b\to i})\tilde{P}^{(t)}(h)$
6              $\tilde{P}^{(t+1)}(h - w_b) \leftarrow \tilde{P}^{(t+1)}(h - w_b) + \hat{\eta}_{b\to i}\tilde{P}^{(t)}(h)$
7          $t \leftarrow t + 1$
8      for $b \in S_i^u(a)$ do
9          for $h \in \{x \mid \tilde{P}^{(t)}(x) \neq 0\}$ do
10             $\tilde{P}^{(t+1)}(h) \leftarrow \tilde{P}^{(t+1)}(h) + (1 - \hat{\eta}_{b\to i})\tilde{P}^{(t)}(h)$
11             $\tilde{P}^{(t+1)}(h + w_b) \leftarrow \tilde{P}^{(t+1)}(h + w_b) + \hat{\eta}_{b\to i}\tilde{P}^{(t)}(h)e^{-y\theta(h,w_b)}$
12         $t \leftarrow t + 1$
13     $Z \leftarrow \sum_{h=-\infty}^{\infty} \tilde{P}^{(t)}(h)$ ; $P_{i,a}(h) \leftarrow \frac{1}{Z}\tilde{P}^{(t)}(h)$
```

C_a and $w_a \neq \top\}$, respectively. We can then calculate

$$\mathcal{S}_i^- = \sum_{h=1}^{\infty} P_i(h), \qquad \mathcal{S}_i^+ = \sum_{h=-1}^{-\infty} P_i(h). \tag{16}$$

Together, these terms can be used to calculate the bias, *Bias*$(i) = \mathcal{B}_i^+ - \mathcal{B}_i^-$, of variable X_i, where

$$\mathcal{B}_i^+ = \frac{\mathcal{H}_i^+ + \mathcal{H}_i^0\mathcal{S}_i^+}{\mathcal{H}_i^+ + \mathcal{H}_i^- + \mathcal{H}_i^0}, \qquad \mathcal{B}_i^- = \frac{\mathcal{H}_i^- + \mathcal{H}_i^0\mathcal{S}_i^-}{\mathcal{H}_i^+ + \mathcal{H}_i^- + \mathcal{H}_i^0}. \tag{17}$$

When a variable is selected for assignment, it should be *true* (1) if *Bias*$(i) > 0$ and *false* (0) otherwise. If no variable's absolute bias exceeds a minimum threshold, it may be that further assignments using the bias estimates are not worthwhile and we should switch to a different search strategy.

The decimation algorithm is typical of one based on survey propagation. First construct G for the problem and initialise all clause to variable messages randomly in $[0, 1]$. Visit the clause nodes in random order and calculate incoming surveys from neighbouring variables using Equation 10 with values $\hat{\eta}_{b\to i}$. Use the results with Equation 11 to compute outgoing surveys to neighbouring variable nodes. Repeat until all clause nodes can be visited in one pass with outgoing warnings unchanged. This fixed point is used to set the most strongly biased variables to their favoured value. The process can stop when it becomes difficult to find a fixed point with strongly biased variables.

4 Experimental results

We allowed up to 50 rounds of decimation. In each round, up to 100 variables with the highest absolute bias > 0.006 had their values fixed, and up to 1000 iterations of message passing were performed. At the end of each round unit propagation was performed to simplify the remaining problem. Within a round, a fixed point was detected if no survey changed by more than 0.0001. If at the end of a round either no fixed point had been found or one was found but no variables had an absolute bias > 0.006 then decimation terminated.

		percentage reduction in truth assignment cost by cutoff value (%)				
problem		10^4	10^5	10^6	10^7	10^8
1	worst	91.3 ± 2.1	99.9 ± 6.5	99.4 ± 43.9	26.2 ± 4.2	-14.0 ± 4.1
	best	98.8 ± 2.0	100.0 ± 6.4	99.6 ± 43.9	55.1 ± 5.4	34.3 ± 6.5
2	worst	70.8 ± 2.1	95.4 ± 4.9	99.7 ± 20.7	85.5 ± 100.7	40.0 ± 5.5
	best	84.7 ± 2.1	99.8 ± 4.6	99.8 ± 20.7	87.9 ± 102.0	46.2 ± 5.0
3	worst	52.9 ± 2.0	79.9 ± 5.3	96.8 ± 19.8	93.9 ± 59.5	37.6 ± 59.4
	best	56.0 ± 2.1	83.3 ± 5.5	98.7 ± 19.5	94.1 ± 59.5	41.1 ± 60.9
4	worst	29.5 ± 2.1	51.8 ± 5.0	75.6 ± 16.1	87.0 ± 45.4	64.4 ± 102.3
	best	30.6 ± 2.1	54.5 ± 5.1	80.9 ± 16.2	95.5 ± 42.9	65.9 ± 103.1
5	worst	69.9 ± 2.1	94.1 ± 5.8	99.7 ± 25.5	79.1 ± 90.3	41.0 ± 6.0
	best	74.2 ± 2.1	97.7 ± 5.7	99.7 ± 25.5	80.9 ± 91.0	43.7 ± 5.9
6	worst	93.0 ± 2.3	100.0 ± 4.9	99.7 ± 31.8	49.7 ± 4.7	28.1 ± 5.6
	best	99.1 ± 2.2	100.0 ± 4.9	99.8 ± 31.8	57.6 ± 5.0	40.4 ± 6.2
7	worst	59.8 ± 2.3	90.4 ± 5.8	99.7 ± 20.5	93.9 ± 75.3	42.4 ± 6.1
	best	85.2 ± 2.3	99.8 ± 5.5	99.8 ± 20.4	95.1 ± 75.7	47.8 ± 5.7
8	worst	93.5 ± 2.4	100.0 ± 6.7	99.4 ± 33.7	31.8 ± 5.6	4.3 ± 5.8
	best	99.0 ± 2.3	100.0 ± 6.7	99.6 ± 33.7	50.3 ± 5.9	27.2 ± 6.1
9	worst	48.8 ± 2.1	76.8 ± 6.0	97.3 ± 20.4	83.1 ± 82.3	26.7 ± 5.8
	best	60.4 ± 2.2	87.5 ± 6.0	99.5 ± 19.9	85.2 ± 83.0	33.9 ± 5.7
10	worst	87.4 ± 2.1	99.6 ± 5.4	99.6 ± 28.8	53.0 ± 64.5	24.6 ± 4.4
	best	96.3 ± 2.0	100.0 ± 5.2	99.7 ± 28.8	58.2 ± 66.7	36.4 ± 5.4

Table 1: Percentage reduction in mean cost found by Walksat after decimation for $(\alpha, \beta) = (4.09, 0.2)$.

We generated 10 random problems for each pair $(\alpha = \frac{M_h}{N}, \beta = \frac{M_s}{N})$ where M_h and M_s are the number of hard and soft clauses. We did this for the pairs $(4.09, 0.2), (4.14, 0.15), (4.19, 0.10)$ and $(4.24, 0.05)$, with the number of variables set to $N = 10000$. By keeping α below 4.25 we expect these problems to be in the satisfiable region, but $\alpha + \beta$ is held at 4.29 so we expect that some clauses will not be satisfiable. All clauses had three distinct literals sampled without replacement from the set of variables then negated with probability 0.5. Soft clauses were given an integer weight chosen uniformly from the range $[1, 100]$.

For each problem we performed decimation for values $y \in \{0.5, 1.0, 1.5, 2.0, 2.5, 3.0\}$ yielding 70 problems for each (α, β) pair, 10 of which were the original problems with the remaining 60 corresponding to decimations for the 6 values of y.

We used UBCSAT [11] to explore local search algorithms on the randomly generated problems and their decimated counterparts. We tested the stochastic algorithms 'crwalk', 'novelty', 'novelty+', 'irots', 'walksat' and 'walksat-tabu'; these were chosen as they had weighted variants applicable to our problems. For each problem, the hard clause weight was set to the sum of weights of soft clauses in the problem so that truth assignments satisfying all hard clauses had lower cost than any which violated at least one hard clause. UBCSAT takes a cutoff argument that limits the number of steps taken in searching for a solution; once exceeded, the lowest cost truth assignment found so far is returned. We gathered 100 runs for each algorithm/problem/cutoff combination and took the mean and standard error, repeating each for the cutoff values $\{10^4, 10^5, 10^6, 10^7, 10^8\}$.

The experiments were run on virtual machines in a batch processing cluster, so the hardware could differ between problems. Thus the times measured are a guideline and cannot be reliably used for comparisons. Of the algorithms used, Walksat most effectively found the lowest cost truth assignments; due to space limitations, we only present our results for Walksat from the total gathered. Table 1 shows the reduction in mean truth assignment cost found running Walksat on the decimated problem compared

	Hard/soft clause ratios (α, β)			
	$(4.09, 0.2)$	$(4.14, 0.15)$	$(4.19, 0.1)$	$(4.24, 0.05)$
worst	-14.0 ± 4.1	17.7 ± 4.9	93.8 ± 35.1	61.2 ± 11.9
best	34.3 ± 6.5	39.9 ± 6.1	97.2 ± 32.5	65.4 ± 12.2
worst	40.0 ± 5.5	44.2 ± 72.7	87.1 ± 23.5	41.0 ± 10.4
best	46.2 ± 5.0	65.9 ± 84.2	92.7 ± 22.9	44.8 ± 10.2
worst	37.6 ± 59.4	53.7 ± 80.1	87.8 ± 26.3	35.6 ± 10.2
best	41.1 ± 60.9	68.8 ± 87.5	95.1 ± 23.8	49.9 ± 9.9
worst	64.4 ± 102.3	84.5 ± 66.6	94.7 ± 32.9	93.0 ± 16.9
best	65.9 ± 103.1	87.3 ± 65.2	96.9 ± 32.2	96.1 ± 16.8
worst	41.0 ± 6.0	15.2 ± 4.2	41.1 ± 17.8	48.2 ± 10.6
best	43.7 ± 5.9	36.6 ± 5.4	50.1 ± 17.5	51.5 ± 11.0
worst	28.1 ± 5.6	70.6 ± 92.6	89.9 ± 76.1	50.1 ± 14.5
best	40.4 ± 6.2	72.8 ± 93.6	91.3 ± 76.6	56.7 ± 14.3
worst	42.4 ± 6.1	85.7 ± 73.0	95.4 ± 29.3	52.0 ± 11.0
best	47.8 ± 5.7	87.6 ± 73.6	98.0 ± 27.6	59.1 ± 11.1
worst	4.3 ± 5.8	63.9 ± 96.6	34.5 ± 16.9	66.1 ± 11.3
best	27.2 ± 6.1	71.2 ± 100.6	45.4 ± 17.6	68.5 ± 11.2
worst	26.7 ± 5.8	48.0 ± 71.0	84.0 ± 27.7	32.7 ± 8.7
best	33.9 ± 5.7	54.0 ± 73.5	87.6 ± 27.1	36.3 ± 8.5
worst	24.6 ± 4.4	37.6 ± 5.3	96.8 ± 35.7	76.5 ± 14.1
best	36.4 ± 5.4	55.5 ± 6.4	97.9 ± 34.3	78.6 ± 14.7

Table 2: Percentage reduction in mean cost found by Walksat after decimation at cutoff 10^8 across ten problems.

to the undecimated instance. The 'worst' and 'best' are taken from the reductions obtained across all y values, choosing those yielding the least and greatest reduction of mean truth assignment cost after decimation. The errors are a 95% confidence interval using Student's t-distribution with 99 degrees of freedom. Initially errors increase with cutoff value, but it is likely that errors will reduce as this value increases further, as given a sufficiently high cutoff value most Walksat runs should converge to equal cost. The increase in error is most likely due to Walksat finding solutions in a larger proportion of runs as cutoff value is increased, and to a large difference in truth assignment cost between solutions and non-solutions leading to large variance when results from these two classes mix in the runs.

Table 2 shows the reduction in mean truth assignment cost found by Walksat after decimation, for cutoff 10^8 across the (α, β) pairs. The variance across the 100 runs of Walksat is larger for the middle values $(4.14, 0.15)$ and $(4.19, 0.1)$ than for the values $(4.09, 0.2)$ and $(4.24, 0.05)$. For $(4.14, 0.15)$ the errors are too large to make a meaningful statement; however, despite the large error for $(4.19, 0.1)$, decimation consistently makes a significant reduction by over 50% in seven out of ten problems. For $(4.24, 0.05)$, eight out of ten problems show a significant cost reduction by over 30%, whereas for $(4.09, 0.2)$ six out of ten problems reduce cost by over 20%. There is only one decimation significantly increasing cost: by $14 \pm 4\%$ for $(4.09, 0.2)$.

Table 3 shows average CPU time taken to complete decimation over the 10 problems for each y. The time dramatically increases as we increase the soft clause ratio. This is expected, as computing the distribution of warnings from soft clauses is an expensive part of the calculation, and needs to be performed more often as the soft clause ratio increases. There appears to be no significant change in the time to perform decimation as y is altered for fixed clause ratios.

Figure 2 shows a histogram of the number of variables set during decimation across problem sets for

	y					
(α, β)	0.5	1.0	1.5	2.0	2.5	3.0
(4.09,0.2)	68.9 ± 37.6	93.6 ± 62.2	73.5 ± 41.7	67.7 ± 22.9	70.8 ± 36.8	96.8 ± 28.5
(4.14,0.15)	38.2 ± 6.9	39.3 ± 18.3	53.9 ± 18.3	82.7 ± 6.1	44.2 ± 13.5	31.9 ± 3.1
(4.19,0.1)	11.3 ± 1.0	10.7 ± 1.3	10.9 ± 1.1	10.9 ± 0.9	9.1 ± 0.8	9.1 ± 0.6
(4.24,0.05)	3.7 ± 0.3	3.7 ± 0.3	3.5 ± 0.3	3.6 ± 0.3	3.7 ± 0.3	3.3 ± 0.3

Table 3: Average time taken to perform decimation step (hours).

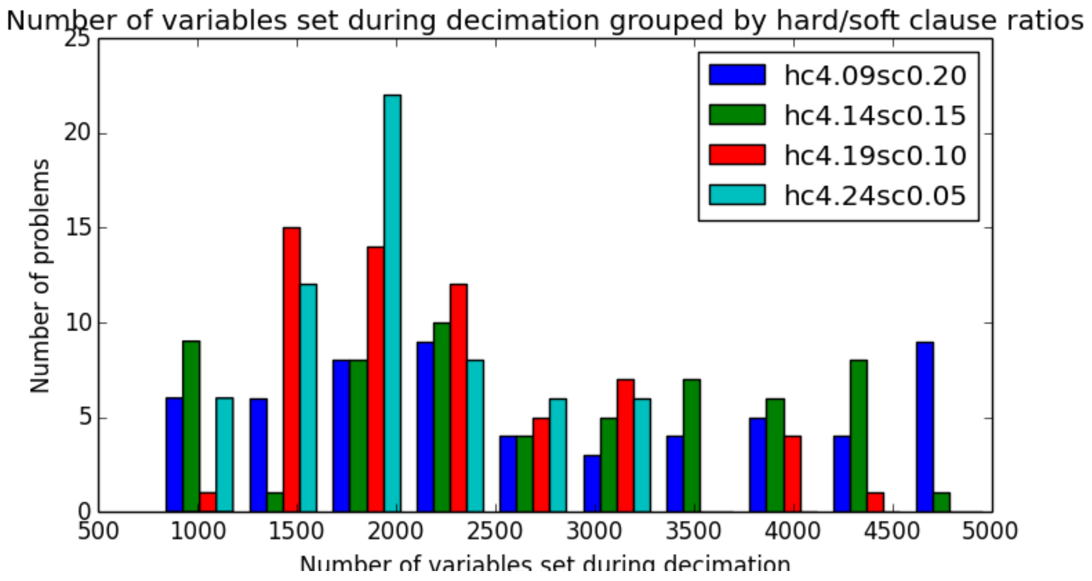

Figure 2: Histogram of the number of variables set during decimation across all problem sets grouped by clause ratio pairs (α, β).

different clause ratios. The problems with lowest hard clause ratio appear to promote the setting of the most variables during decimation. No problem with hard clause ratio 4.24 had more than 3,500 variables set during decimation.

During decimation, as variables are set we may infer that some soft clauses have become false. The sum of weights of these clauses is the cost incurred by decimation. Figure 3 shows a histogram of this metric over all 10 problems and y values, grouped for clause ratio pair (α, β). Across all problem sets the highest cost was 1043, occurring when 5000 variables were set during decimation in the context of a problem having a sum of weights of soft clauses of 100,934. No hard clauses were violated during decimation across all experiments.

5 Discussion

Unlike in other experiments testing SP(y) [8], the value y did not strongly affect the mean truth assignment cost after decimation for WPMax-SAT. SP(y) showed a distinction between truth assignment costs as y changed, the lowest costs being found as y increases. This may be because SP(y) is applied to Max-SAT problems with only soft clauses, so the y term affects the calculation of surveys from every variable node. In WPMax-SAT, there may be variable nodes connected to a small number of soft clauses or only to hard clauses; in such cases, y has little or no effect on the survey calculation. Decimation is randomised, so we would need many decimation experiments, for a fixed y, to identify any statistically

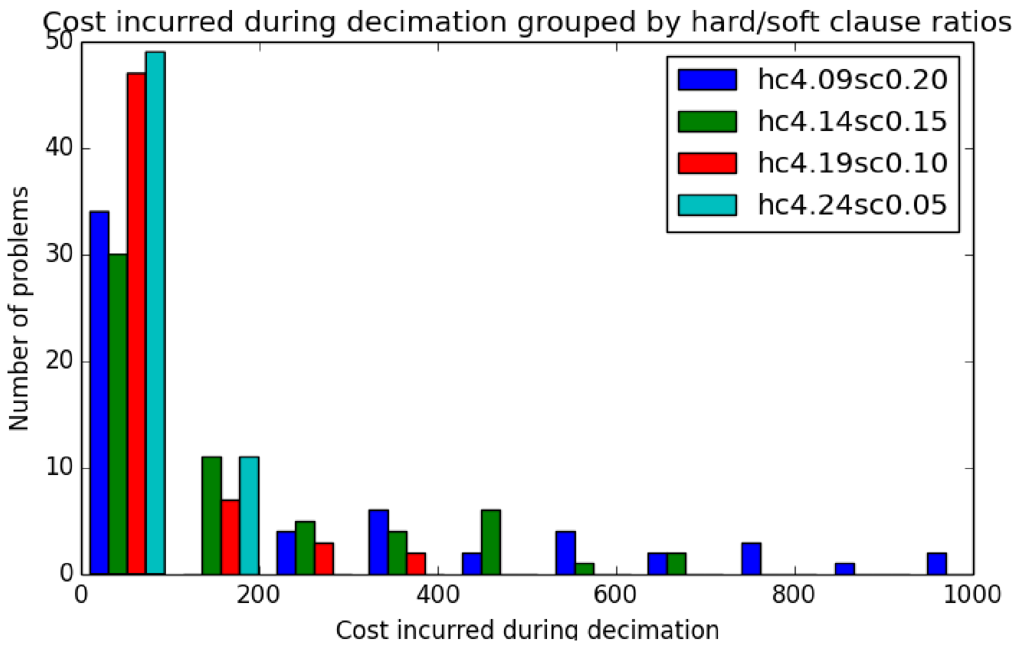

Figure 3: Histogram of the cost incurred during decimation across all problem sets grouped by clause ratio pairs (α, β).

significant difference between values of y, and its magnitude if it exists.

Of our 60 decimation experiments, three failed due to a bug which is now fixed. [1] All other decimation experiments converged to non-trivial fixed points. Two experiments set fewer than 1000 variables, instead setting 800 and 900 variables. There were six instances, all with $(\alpha, \beta) = (4.09, 0.2)$, where decimation set the imposed maximum of 5000 variables. This may suggest that decimation cannot continue simplifying the problem given extra computing resources. Another concern is the costly running times of decimation. No randomised restart strategy was used to improve our chances of convergence, nor did we retry if a fixed point was found containing no variables with sufficient bias to continue decimation. Experimenting with these may increase the amount of simplification decimation can achieve and improve aggregate running times by avoiding poor random seeds. It should be possible to parallelize message updates but the effect on convergence properties remains to be seen.

6 Related work

Belief propagation has been applied to other problems with loops in the factor graph; for example, in decoding low density parity check codes, with performance close to the Shannon limit [12]. Experiments on smaller graphs with loops, for which exact marginals can be found, indicate that when belief propagation converges its marginal estimates are surprisingly accurate [13].

An explanation for the success of loopy belief propagation is that fixed points of the belief propagation equations correspond to stationary points of the Bethe free energy [14]. This provides a fix when belief propagation does not converge: minimise the Bethe free energy, which is guaranteed to terminate. Empirical data suggests that this yields similar results to belief propagation [15].

This link between belief propagation and the Bethe approximation has inspired *generalised belief propagation* [16] methods, based on increasingly accurate approximations to the free energy, known as

[1]Numerical results were unaffected, but the bug sometimes caused a crash when there were multiple variables with the same bias.

Kikuchi approximations. These group variables into *regions* and sum the free energies of each region together with terms correcting for over-counted regions. Messages are then passed from regions to their subregions. The accuracy improves with region size until the approximation is exact when a region surrounds the entire graph; however, complexity increases exponentially. On graphs with many small loops, it may be possible to eliminate most of the error from belief propagation by surrounding these loops with small regions whilst limiting the increase in complexity this causes.

7 Conclusion

We have extended the survey propagation equations allowing them to be applied to WPMax-SAT problems: those consisting of (1) both hard and soft clauses and (2) arbitrary integer weights for soft clauses. We demonstrated the convergence of these equations across randomly generated WPMax-3SAT formulas with hard clause ratios close to the satisfiability threshold. We found that in most cases, decimation using fixed-points of these equations appears to simplify the problem, allowing local search solvers to find a truth assignment of comparable cost faster than without decimation.

The computational cost of message passing is high and increases with the ratio of soft clauses; work will be needed to turn this into a practical problem-solving method, possibly by updating multiple messages in parallel. Whether this affects convergence or the resulting fixed points remains to be seen.

References

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

[2] J. Díaz, L. Kirousis, D. Mitsche, and X. Pérez-Giménez. On the satisfiability threshold of formulas with three literals per clause. *Theoretical Computer Science*, 410(30-32):2920–2934, 2009.

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

[4] James M Crawford and Larry D Auton. Experimental results on the crossover point in random 3-sat. *Artificial intelligence*, 81(1):31–57, 1996.

[5] P. Cheeseman, B. Kanefsky, and W.M. Taylor. Where the really hard problems are. In *Proc. IJCAI*, 1991.

[6] D. Mitchell, B. Selman, and H. Levesque. Hard and easy distributions of SAT problems. In *Proc. AAAI*, 1992.

[7] Alfredo Braunstein, Marc Mézard, and Riccardo Zecchina. Survey propagation: An algorithm for satisfiability. *Random Structures & Algorithms*, 27(2):201–226, 2005.

[8] D. Battaglia, M. Kolář, and R. Zecchina. Minimizing energy below the glass thresholds. *Physical Review E*, 70(3):036107, 2004.

[9] Frank R Kschischang, Brendan J Frey, and H-A Loeliger. Factor graphs and the sum-product algorithm. *Information Theory, IEEE Transactions on*, 47(2):498–519, 2001.

[10] Marc Mezard and Andrea Montanari. *Information, physics, and computation*. Oxford University Press, 2009.

[11] Dave A. D. Tompkins and Holger H. Hoos. UBCSAT: An implementation and experimentation environment for SLS algorithms for SAT and MAX-SAT. In *Proc. SAT*, 2004.

[12] D.J.C. MacKay and R.M. Neal. Near Shannon limit performance of low density parity check codes. *Electronics letters*, 32(18):1645, 1996.

[13] K. Murphy, Y. Weiss, and M.I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proc. UAI*, 1999.

[14] J.S. Yedidia, W.T. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. In G. Lakemeyer and B. Nebel, editors, *Exploring artificial intelligence in the new millennium*, chapter 8, pages 239–270. Morgan Kaufmann Publishers, 2003.

[15] M. Welling and Y.W. Teh. Belief optimization for binary networks: A stable alternative to loopy belief propagation. In *Proc. UAI*, 2001.

[16] J.S. Yedidia, W.T. Freeman, and Y. Weiss. Generalized belief propagation. In *Proc. NIPS*, 2000.