Applying Probabilistic Inference to Heuristic Search by Estimating Variable Bias

Eric I. Hsu and **Christian J. Muise** and **Sheila A. McIlraith** and **J. Christopher Beck**
{eihsu,cjmuise,sheila}@cs.toronto.edu and jcb@mie.utoronto.ca
University of Toronto

Abstract

Backbone variables have the same assignment in all solutions to a given constraint satisfaction problem; more generally, *bias* represents the proportion of solutions that assign a variable a particular value. Intuitively such constructs would seem important to efficient search, but their study to date has assumed a mostly conceptual perspective, in terms of indicating problem hardness or motivating and interpreting heuristics. In this work, we first measure the ability of both existing and novel probabilistic message-passing techniques to directly estimate bias (and identify backbones) for the specific problem of Boolean Satisfiability (SAT). We confirm that methods like Belief Propagation and Survey Propagation–plus Expectation Maximization-based variants–do produce good estimates with distinctive properties. We demonstrate the use of bias estimation within a modern SAT solver, and exhibit a correlation between accurate, stable, estimates and successful search. The same process also yields a family of search heuristics that can dramatically improve search efficiency for the hard random problems considered in this study.

1 Introduction

Probabilistic message-passing algorithms like Survey Propagation (SP) and Belief Propagation (BP), plus variants based on Expectation Maximization (EM), are state-of-the-art for solving large random SAT problems near the critically-constrained phase transition in problem hardness [2, 3, 9, 7]. (While the focus of this work is SAT, such techniques have also been applied to CSP in general [8].) This success would appear to result from the ability to implicitly sample from the space of solutions and thus estimate variable bias: the percentages of solutions that have a given variable set true or false. However, this bias estimation ability has never been measured directly, and its usefulness to heuristic search has also escaped systematic study–most of the techniques have never been implemented within a modern backtracking search system.

Similarly, backbones, or variables that must be set a certain way in any solution to a given problem, have also drawn a good deal of recent interest [15, 11]. Such variables typically serve as theoretical constructs for analyzing problem complexity [17, 2] or for motivating search heuristics [5, 20], but they have not been directly targeted for discovery within arbitrary problems. (Even bounded approximation of a backbone set is intractable unless $P = NP$ [11]). However, since backbones must have 100% positive or negative bias, finding bias generalizes the task of backbone identification. Thus any bias estimator can be used to identify backbones, though in the absence of bounds on approximation error, a method's accuracy can be arbitrarily bad.

Isolating the performance of probabilistic techniques as bias estimators will cast light on both the estimators and on bias itself, ultimately directing the implementation of a complete problem-solving system. Thus the first stage of our study compares the basic accuracy of six message-passing techniques and two control methods when applied to hard, random, satisfiable SAT problems as stand-alone bias estimators. An ancilliary contribution is that four of the probabilistic techniques are novel variations based on the EM method of maximum-likelihood parameter estimation.

For the second stage of study, we assess how such comparisons translate when we move the algorithms into the realm of full-featured search, by embedding them as variable/value ordering heuristics within the MiniSat solver [6]. While it is intuitive that bias should relate to how we set variables during search, it is not obvious that bias should be a key to efficiency in the presence of modern features like restarts, clause learning, and the like. Similarly, the best way to employ bias estimates is also non-obvious beyond some basic intuitions. Thus our focus extends beyond the basic accuracy of the different techniques, to the question of how they are best employed according to their unique characteristics. The resulting contribution is a deeper insight into the effective design and use of propagation algorithms, as realized by a practical solver that can dramatically improve MiniSat's speed in solving hard random problems.

The following section provides more background on propagation algorithms and how they can be applied to SAT. Next, Section 3 defines useful terminology and notation for understanding how they work. Sections 4 and 5 present the actual bias estimation methods and explain how they will be compared. With all of this background in hand, Section 6 finally presents our experimental comparison, and interprets the results. Lastly, Section 7 extracts overall conclusions and discusses future work.

2 Background

Message-passing algorithms (a.k.a. "propagation techniques") like BP, SP, and EM-based variants have been applied to a growing variety of discrete reasoning problems [2, 10, 8, 12], augmenting their traditional roles in probabilistic inference [16, 13, 4]. The methods all operate by propagating messages between a problem's variables, causing them to iteratively adjust their own values from some initial randomized settings. The techniques produce "surveys", representing, informally, the probability that each variable should be set a certain way if we were to assemble a satisfying assignment. That is, the survey reports a "bias" for each variable; in the case of SAT this tells us the probability that we would find a variable set to *true* or *false* if we were to somehow draw a sample from the set of solutions.

Importantly, then, a propagation algorithm does not output an outright solution to a SAT problem. Rather, applying a probabilistic method to search requires two interrelated design decisions: a means of calculating surveys, and a means of using the surveys to fix the next variable within an arbitrary search framework. A reasonable strategy for the latter is to pick the variable with the most extreme bias, and set it in the direction of that bias, i.e. "succeed-first" search. But by better understanding the characteristics of various survey techniques, we can explore more sophisticated approaches to variable and value ordering. Thus we will begin by studying survey techniques in isolation, and then examine their behavior within search by integrating with MiniSat.

Note also that integrating with a solver means computing a new survey each time we fix a single variable, a standard practice for addressing correlations between variables [9]. A single survey might report that v_1 is usually *true* within the space of solutions, and that v_2 is usually *false*, even though the two events happen simultaneously with relative infrequency. Instead of trying to fix multiple variables at once, then, we will fix a single variable first and simplify the resulting problem. In subsequent surveys the other variables' biases are thereby conditioned on this first assignment.

With this background in hand, it is now possible to precisely define our study of how well probabilistic methods estimate variable bias, and ultimately how they behave within a complete backtracking search system.

3 Definitions

Definition 1 (SAT instance) *A (CNF)* **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^- comprise the variables appearing positively and negatively in a clause c, respectively. The sets C_v^+ and C_v^- comprise the clauses that contain positive and negative literals for variable v, respectively. $C_v = C_v^+ \cup C_v^-$ denotes the whole of v's clauses.*

Definition 2 (Bias, Profile) *For a satisfiable SAT instance $\mathcal{F}$, the* **bias distribution** *ϕ_v of a variable v represents the fraction of solutions to $\mathcal{F}$ wherein v appears positively or negatively. Thus it consists of a* **positive bias** *ϕ_v^+ and a* **negative bias** *ϕ_v^-, where $\phi_v^+, \phi_v^- \in [0,1]$ and $\phi_v^+ + \phi_v^- = 1$. A vector of bias distributions, one for each variable in a theory, will be called a* **bias profile**, *denoted $\Phi(\mathcal{F})$.*

Equivalently, it can be useful to think of a variable's bias as the probability of finding the variable set positively or negatively when choosing a solution uniformly at random from the space of satisfying assignments.

Definition 3 (Backbone Variable) *A* **backbone variable** *for a given SAT instance is one whose bias is concentrated entirely on one polarity: $\phi_v^+ = 1$ or $\phi_v^- = 1$*

Definition 4 (Survey) *Given a SAT instance $\mathcal{F}$, a bias estimation algorithm outputs a* **survey** *$\Theta(\mathcal{F})$ trying to match the true bias profile $\Phi(\mathcal{F})$. Accordingly, Θ will contain an estimated bias distribution θ_v, representing θ_v^+ and θ_v^-, for each variable v in $\mathcal{F}$.*

Less formally, it is useful to describe a variable as "**positively biased**" with respect to a true or estimated bias distribution. This means that under the given distribution, its positive bias exceeds its negative bias. Similarly the "**strength**" of a bias distribution indicates how much it favors one value over the other, as defined by the maximum difference between its positive or negative bias and 0.5.

4 Probabilistic Methods for Estimating Bias

In this section we present six distinct propagation methods for measuring variable bias: Belief Propagation (BP), EM Belief Propagation-Local/Global (EMBP-L and EMBP-G), Survey Propagation (SP), and EM Survey Propagation-Local/Global (EMSP-L and EMSP-G). These methods represent the space of algorithms defined by choosing between BP and SP and then employing one of them either in original form, or by applying a local- or global-consistency transformation based on the Expectation Maximization framework. The EM-based rules were newly derived for this work; implementations and derivations are available online[1].

On receiving a SAT instance $\mathcal{F}$, any of the propagation methods begins by formulating an initial survey at random. For instance, the positive bias can be randomly generated, and the negative bias can be set to its complement: $\forall v, \theta_v^+ \sim \mathcal{U}[0,1]$; $\theta_v^- \leftarrow 1 - \theta_v^+$. Each algorithm proceeds to successively refine its estimates, over multiple iterations. An iteration consists of a single pass through all variables, where the bias for each variable is updated with respect to the other variables' biases, according to the characteristic rule for a method. If no bias has changed between two successive iterations, the process ends with convergence; otherwise an algorithm terminates by timeout or some other parameter. EM-type methods are "convergent", or guaranteed to converge naturally, while regular BP and SP are not [9].

The six propagation methods are discussed elsewhere in greater theoretical detail than space permits here [9, 2]. But for a practical understanding, they can be seen as update rules that assign weights (ω_v^+ and ω_v^-) toward a variable's positive and negative biases–plus a third weight (ω_v^*) for the "joker bias" (explained below) in the case of SP-based methods. The rules will make extensive use of the formula $\sigma(v,c)$ in Figure 1(a). In so doing they express the probability that variable v is the "sole-support" of clause c in an implicitly sampled satisfying configuration: all other variables that appear in the clause are set unsatisfyingly. From a generative statistical perspective, the probability of this event is the product of the negative biases of all (other) variables that appear in the clause as positive literals, and the positive biases of all (other) variables that are supposed to be negative.

[1] http://www.cs.toronto.edu/~eihsu/VARSAT/

$$\sigma(v,c) \triangleq \prod_{i\in V_c^+\setminus\{v\}} \theta_i^- \prod_{j\in V_c^-\setminus\{v\}} \theta_j^+$$

(a) $\sigma(v,c)$: v is the sole support of c.

$$\theta_v^{+\prime} \leftarrow \frac{\omega_v^+}{\omega_v^+ + \omega_v^-} \qquad \theta_v^{-\prime} \leftarrow \frac{\omega_v^-}{\omega_v^+ + \omega_v^-}$$

(b) Bias normalization for BP methods.

$$\theta_v^{+\prime} \leftarrow \frac{\omega_v^+}{\omega_v^+ + \omega_v^- + \omega_v^*} \qquad \theta_v^{-\prime} \leftarrow \frac{\omega_v^-}{\omega_v^+ + \omega_v^- + \omega_v^*} \qquad \theta_v^{*\prime} \leftarrow \frac{\omega_v^*}{\omega_v^+ + \omega_v^- + \omega_v^*}$$

(c) Bias normalization for SP methods.

Figure 1: Formula for "sole-support"; normalizing rules for BP and SP families of bias estimators.

$$\omega_v^+ = \prod_{c\in C_v^-} (1-\sigma(v,c))$$
$$\omega_v^- = \prod_{c\in C_v^+} (1-\sigma(v,c))$$

(a) Regular BP update rule.

$$\omega_v^+ = |C_v| - \sum_{c\in C_v^-} \sigma(v,c)$$
$$\omega_v^- = |C_v| - \sum_{c\in C_v^+} \sigma(v,c)$$

(b) EMBP-L update rule.

$$\omega_v^+ = |C_v^-| \left[\prod_{c\in C_v^-} (1-\sigma(v,c))\right] + |C_v^+|$$
$$\omega_v^- = |C_v^+| \left[\prod_{c\in C_v^+} (1-\sigma(v,c))\right] + |C_v^-|$$

(c) EMBP-G update rule.

Figure 2: Update rules for the Belief Propagation (BP) family of bias estimators.

Thus the six sets of update rules weight each variable's biases according to the current biases of its surrounding variables. The weights are normalized into proper probabilities as depicted in Figures 1(b) and 1(c), depending on whether we are dealing with the two states characteristic of BP-based methods, or with the three states of an SP-based method. This creates a new bias distribution for each variable, completing a single iteration of the algorithm. The update rules appear in Figures 2 and 3.

BP can first be viewed as generating the probability that v should be positive according to the odds that one of its positive clauses is completely dependent on v for support. That is, v appears as a positive literal in some $c \in C_v^+$ for which every other positive literal i turns out negative (with probability θ_i^-), and for which every negative literal $\neg j$ turns out positive (with probability θ_j^+). This combination of unsatisfying events would be represented by the expression $\sigma(v,c)$. However, a defining characteristic of BP is its assumption that every v is the sole support of at least one clause. (Further, v cannot simultaneously support both a positive and a negative clause since we are sampling from the space of *satisfying* assignments.) Thus, we should view Figure 2(a) as weighing the probability that no negative clause needs v (implying that v is positive by assumption), versus the probability that no positive clause needs v for support.

EMBP-L is the first of a set of update rules derived using the EM method for maximum-likelihood parameter estimation. This statistical technique features guaranteed convergence, but requires a bit of invention to be used as a bias estimator for constraint satisfaction problems [4, 8]. Resulting rules like EMBP-L are variations on BP that calculate a milder, arithmetic average by using summation, in contrast to the harsher geometric average realized by products. This is one reflection of an EM-based method's convergence versus the non-convergence of regular BP and SP. All propagation methods can be viewed as energy minimization techniques whose successive updates form paths to local optima in the landscape of survey likelihood [9]. By taking smaller, arithmetic steps, EMBP-L (and EMBP-G) is guaranteed to proceed from its initial estimate to the nearest optimum; BP and SP take larger, geometric steps, and can therefore overshoot optima. This explains why BP and SP can explore a larger area of the space of surveys, even when initialized from the same point as EM-based methods, but it also leads to their non-convergence. Empirically, EMBP-L and EMBP-G usually converge in three or four iterations for the examined SAT instances, whereas BP and SP typically require at least ten or so, if they converge at all.

Intuitively, the equation in Figure 2(b) (additively) reduces the weight on a variable's positive bias according to the chances that it is needed by negative clauses, and vice-versa. Such reductions are taken from a smoothing constant representing the number of clauses a variable appears in overall; highly connected variables have less extreme bias estimates than those with fewer constraints.

EMBP-G is also based on smoother, arithmetic averages, but employs a broader view than EMBP-L. While the latter is based on "local" inference, namely generalized arc-consistency, the derivation of EMBP-G uses global consistency across all variables. In the final result, this is partly reflected by the way that Figure 3(c) weights a variable's positive bias by going through each negative clause (in multiplying by $|C_v^-|$) and *uniformly* adding the chance that *all* negative clauses are satisfied without v. In contrast, when EMBP-L iterates through the negative clauses, it considers their satisfaction on an individual basis, without regard to how the clauses' means of satisfaction might interact with one another. So local consistency is more sensitive to in-

$$\omega_v^+ = \prod_{c \in C_v^-} (1-\sigma(v,c)) \cdot \rho \left[1 - \prod_{c \in C_v^+} (1-\sigma(v,c))\right]$$

$$\omega_v^- = \prod_{c \in C_v^+} (1-\sigma(v,c)) \cdot \rho \left[1 - \prod_{c \in C_v^-} (1-\sigma(v,c))\right]$$

$$\omega_v^* = \prod_{c \in C_v} (1-\sigma(v,c))$$

(a) Regular SP update rule.

$$\omega_v^+ = |C_v| - \sum_{c \in C_v^-} \sigma(v,c)$$

$$\omega_v^- = |C_v| - \sum_{c \in C_v^+} \sigma(v,c)$$

$$\omega_v^* = |C_v| - \sum_{c \in C_v} \sigma(v,c)$$

(b) EMSP-L update rule.

$$\omega_v^+ = |C_v^-| \prod_{c \in C_v^-} (1-\sigma(v,c)) + |C_v^+| \left[1 - \prod_{c \in C_v^+} (1-\sigma(v,c))\right]$$

$$\omega_v^- = |C_v^+| \prod_{c \in C_v^+} (1-\sigma(v,c)) + |C_v^-| \left[1 - \prod_{c \in C_v^-} (1-\sigma(v,c))\right]$$

$$\omega_v^* = |C_v| \prod_{c \in C_v} (1-\sigma(v,c))$$

(c) EMSP-G update rule.

Figure 3: Update rules for the Survey Propagation (SP) family of bias estimators.

dividual clauses in that it will subtract a different value for each clause from the total weight, instead of using the same value uniformly. At the same time, the uniform value that global consistency does apply for each constraint reflects the satisfaction of all clauses at once.

SP can be seen as a more sophisticated version of BP, specialized for the SAT problem. To eliminate the assumption that every variable is the sole support of some clause, it introduces the possibility that a variable is not constrained at all in a given satisfying assignment. Thus, it uses the three-weight normalization equations in Figure 1(c) to calculate a three-part bias distribution for each variable: θ_v^+, θ_v^-, and θ_v^*, where '*' indicates the unconstrained "joker" state. Thus, in examining the weight on the positive bias in Figure 3(a), it is no longer sufficient to represent the probability that no negative clause needs v. Rather, we explicitly factor in the condition that *some* positive clause needs v, by complementing the probability that *no* positive clause needs it. This acknowledges the possibility that no negative clause needs v, but no positive clause needs it either. As seen in the equation for ω_v^*, such mass goes toward the joker state. ($\rho = 0.95$ is an optional smoothing parameter explained in [14].)

For the purpose of estimating bias/finding backbones, any probability mass for θ_v^* is evenly distributed between θ_v^+ and θ_v^- when the final survey is compiled. This reflects how the event of finding a solution with v labeled as unconstrained indicates that there exists one otherwise identical solution with v set to *true*, and another where v is set to *false*. So while the "joker" state affects a variable's bias between iterations, the final result omits it from the task of bias estimation. (One point of future interest is to examine the prevalence of lower "joker" bias in *backdoor* variables [18].)

EMSP-L and **EMSP-G** are analogous to their BP counterparts, extended to weight the third '*' state where a variable may be unconstrained. So similarly, they can be understood as convergent versions of SP that take a locally or globally consistent view of finding a solution, respectively.

4.1 Control Methods for Estimating Bias

For experimental comparison, we additionally created two simple control methods.

LC ("Literal Count") simplifies a heuristic that was not explicitly designed to rigorously infer bias, but still comprises the core of a highly successful system for refuting *unsatisfiable* SAT instances [5]. The heuristic measures the effect of setting a variable v to be positive or negative by directly counting the degrees of freedom for the remaining literals in v's clauses. As such, information propagates between interconnected variables much as update messages travel through the probabilistic methods. If repeated recursively to no end, the heuristic simply solves the problem directly; instead LC performs a single iteration of analysis for use as an experimental control.

CC ("Clause Count") is an even simpler baseline method that just counts the number of clauses containing a given variable as a positive literal, and the number wherein it appears negatively. The ratio of these two counts serves to estimate the variable's bias distribution: $\theta_v^+ \leftarrow |C_v^+|/|C_v|$ and $\theta_v^- \leftarrow |C_v^-|/|C_v|$.

5 Experimental Methods

This section details the test regimes that were used to evaluate the eight bias estimators on their own and as heuristics within backtracking search. In general, the selected problems are small enough to allow the exact calculation of true bias profiles, or to allow repeated experiments in a reasonable amount of time. Further, they are randomly generated so as to isolate the approaches' average-case behavior. Finally, the problems are drawn from near the phase transition, with $\alpha \triangleq \frac{m}{n}$ set to 4.11, to exclude trivially easy instances.

While the results presented in the next section are limited to this fixed window of problem types, we have observed consistent behavior when scaling the problems and backbones to larger sizes, and when increasing α beyond 4.11 within the space of satisfiable problems. For structured problems, though, we have only anecdotal reports of success, and leave the systematic study of such less-regular domains to future research.

5.1 Phase I: Stand-Alone Bias Estimation Ability

To isolate each approach's ability to estimate bias, we relied on a library of 100-variable satisfiable instances with controlled backbone size [17]. By alternately fixing each variable to one value then the other, and passing the resulting problems to a model-counter, we were able to exhaustively calculate the true bias distribution ϕ_v for every variable in each problem. To guard against variance, each of the bias estimators was run on 100 instances each of problems with backbones of size 10, 30, 50, 70, and 90. Section 6 will compare the resulting bias estimates with the known true biases.

5.2 Phase II: Using Bias Estimates within Search

For the final phase, we embedded each algorithm as a variable and value ordering heuristic within MiniSat. MiniSat is a modern solver that integrates efficient constraint propagation, clause learning, and restarts with built-in variable ordering based on VSIDS [6]. More specifically, MiniSat uses unit propagation by "watch literals" to efficiently identify and fix variables that have become sole supports; it learns "conflict clauses" upon backtracking in order to focus search and skip future conflicts; it applies a theoretically and empirically useful strategy of periodically restarting itself with a different random seed; and in lieu of the bias estimators it has a default variable-ordering heuristic based on the recent activity of variables in the context of the current search.

For this phase we used larger problems ($n = 250$) because the original ones were too small to challenge any of the resulting approaches. The results presented in the next section generalize even more dramatically to still larger problems, but the chosen n allowed for a larger number of repeated trials within a reasonable amount of time.

Directed by the results of the first phase, we tested several branching strategies for using surveys as variable- and value-ordering heuristics. In addition to the "conflict-avoiding" strategy of setting the most strongly biased variable to its stronger value, we also tried to fail first or streamline a problem via the "conflict-seeking" strategy of setting the strongest variable to its weaker value [1]. Other approaches involved different ways of blending the two. While there are many more sophisticated strategies to try in the future, so far all heuristics performed best when using the original (and presumably most intuitive) conflict-avoiding approach.

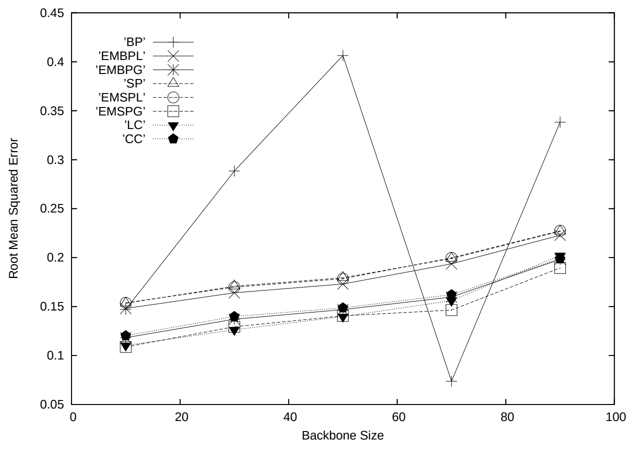

Figure 4: RMS error of all bias estimates over 500 instances of increasing backbone size.

A final complication when integrating with a conventional solver is that any of the eight methods can be governed by a *"threshold"* parameter expressed in terms of the most strongly biased variable in a survey. For instance, if this parameter is set to 0.6, then we only persist in using surveys so long as their most strongly biased variables have a gap of size 0.6 between their positive and negative bias. As soon as we receive a survey where the strongest bias for a variable does not exceed this gap, then we deactivate the bias estimation process and revert to using MiniSat's default variable and value ordering heuristic until the solver triggers a restart. (Note that setting this parameter to 0.0 is the same as directing the solver to never deactivate the bias estimator.) The underlying motivation is that problems should contain a few important variables that are more constrained than the rest, and that the rest of the variables should be easy to set once these few have been assigned correctly. For various theoretical reasons this thought to be of special relevance within the phase-transition region in problem hardness.

We conclude our experiments by systematically determining the best value of the threshold parameter for running each technique on the entire test set as a whole. By determining the overall utility of each technique within backtracking search, we can confirm a relationship between the quality of bias estimates and the efficiency of search.

6 Experimental Results

The experiments produced many interesting results, the bulk of which are available online. Here we summarize the results most relevant to the overall goals of measuring bias estimation ability in isolation, and in backtracking search.

6.1 Phase I: Accuracy of Stand-Alone Estimates

Figure 4 shows root-mean-squared error for bias estimates on all variables in all runs described in Section 5: $\sqrt{\sum_i (\theta_i^+ - \phi_i^+)^2 / n}$. Aside from BP, which is discussed in Section 6.2, the remaining methods are grouped into two bands of linearly increasing error.

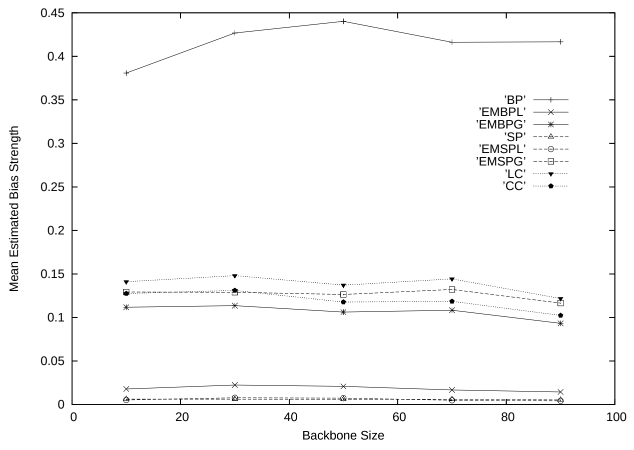

Figure 5: Average strength of estimated bias, over same 500 instances.

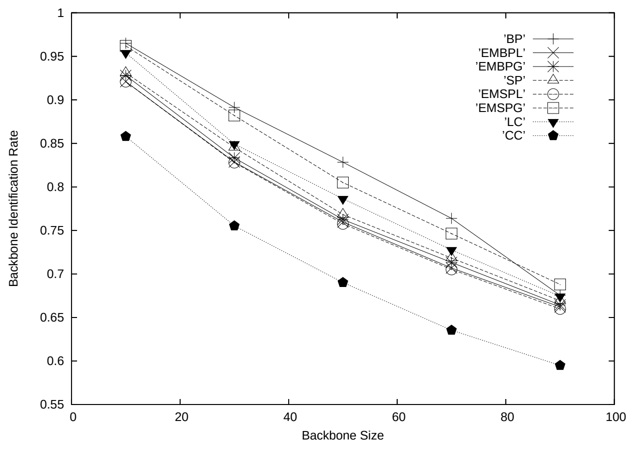

Figure 6: Average success rate in predicting the correct sign for backbone variables.

The best band in terms of average accuracy contains the two global methods, EMBP-G and EMSP-G, along with the two control methods, LC and CC. SP and the local EM methods comprise the less accurate band. Prior study indicates that problems with larger backbones are more constrained and usually harder to solve. Here it appears that their biases are harder to estimate as well. However, the flat lines in Figure 5 demonstrate that the methods are bound to make predictions of constant average strength regardless of problem constrainedness. In other words, the problems in the two figures contain increasingly strong biases; the increase in error in the first figure can be at least partially explained by the methods' inability to make stronger predictions in turn.

Recall that the strength of a variable's bias is the distance between its positive (or negative) bias and 0.5; for backbones this is 0.5, and evenly split variables have strength 0. So the plot averages the formula $\sum_v (max(\theta_v^+, \theta_v^-) - 0.5)/n$ across runs, and we again see familiar groupings of algorithms. BP is the most wildly inaccurate, and also makes the strongest predictions by far. Again the global and control methods form one group, of moderate estimators, and SP joins the local methods in making the most conservative estimates.

Figure 6 considers the special case of backbone identification. Because the estimators are not prone to offer many predictions of 100% bias strength, we instead measure whether they correctly predict the polarity of known backbone variables. Looking ahead to the context of search, it is not critical to set backbones early, but it is certainly critical to set them correctly (during conflict-avoidance search). Here the same groupings of bias estimators are compressed into a more mixed band of linearly decreasing success rates; when there are more backbones (up to 90 out of 100 variables) it becomes harder to get the same percentage of them correct.

The uncharacteristically poor performance of CC reveals an extra insight into the difficulty of predicting backbones. Because it always biases a variable according to the proportion of its positive and negative clause counts, all of CC's incorrect predictions and at least some of its correct ones involve variables that appear in the theory more often than not

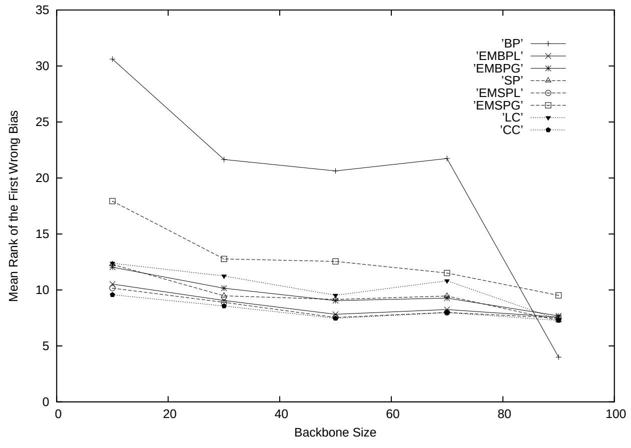

Figure 7: Average strength ranking of the variable most strongly biased toward wrong the polarity.

with the opposite sign from the one they are constrained to hold! For instance, in a problem that contains 50% backbone variables, at least 30% of those appear more often as literals with the wrong polarity than with the right one.

A final quantity of interest when anticipating full search is the strength rank of the first wrong bias. If we are fixing the most strongly biased variable to its stronger value, then a survey's accuracy on the other, weaker variables is irrelevant. Further, when we fix a variable to some polarity, we can never eliminate all solutions unless its true bias is 100% in the direction of the opposite polarity. In this light, Figure 7 averages the results of ranking the variables in a survey by strength, and reading down from the top to find the most highly ranked variable that was actually set in the wrong direction. Here BP exhibits excellent average performance; for backbones of size 10, for instance, its 30 strongest estimates typically turn out to predict the correct sign, before the 31st is biased the wrong way. As we will see, outside of BP and SP, the ordering between methods here closely follows their overall effectiveness when embedded within MiniSat.

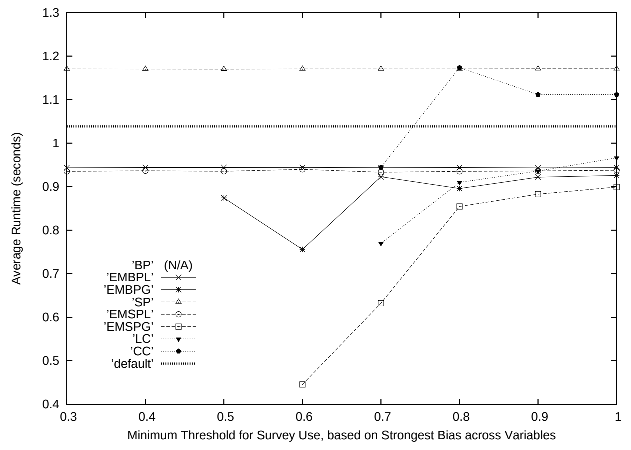

Figure 8: Adjusting the threshold parameter: average runtime for various settings.

6.2 Phase II: Full-Featured Search Using Bias

At last, these insights allow us to embed the bias estimators within a real solver. As mentioned in Section 5, it turned out to be better to pick the most strongly-biased variables first, and to assign them to their stronger values. Figure 8 summarizes the process of tuning the deactivation threshold parameter for each survey method. The dashed level line represents the fixed running time of the default MiniSat heuristic without any bias estimation. Each remaining line represents change in average runtime when a given heuristic is run with different threshold values. Lines that are discontinuous to the left represent problems that could not solve some problem in less than three minutes when the the threshold was set too low. (BP timed out on some problem at every threshold setting, and thus does not appear on the graph.)

EMBP-L, EMSP-L, and SP make mild estimates and thus are not sensitive to the threshold parameter within the range covered by the graph. As the parameter increases, the remaining approaches first reach feasibility when they deactivate themselves early enough to prevent wrong decisions that cause timeouts. Their lines then dip to some optimal setting before rising somewhat as MiniSat increasingly ignores them when it shouldn't. Also observe that most of the methods are still below the default line when the threshold is set to 1.0–if we only follow the bias estimators when they claim to have found a backbone variable, then search performance still improves slightly.

Finally, Figure 9 compiles the lowest points from Figure 8 to compare the average runtimes of the various heuristics when using their best settings. Additionally, the bars are broken down to show the proportion of runtime that was devoted to computing surveys. The bar labels also indicate the average number of surveys that each method wound up producing at its optimal threshold setting. The most important general observation about the relative performance of these methods is that it roughly corresponds to their accuracy as bias estimators. This supports our hypothesis of a correlation between bias and efficient search: better bias estimators tended to produce better SAT-Solvers when employed

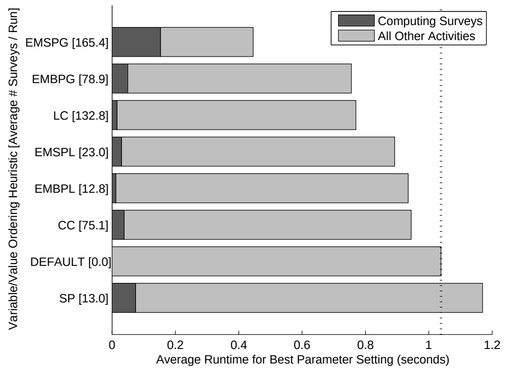

Figure 9: Total/Survey runtimes averaged over 100 random problems, $n = 250$ and $\alpha = 4.11$.

as variable and value ordering heuristics. Besides average accuracy, the results concerning rank seemed to suggest the next-strongest criterion for a good heuristic, as CC's poor performance might explain why it is barely better than the default when it actually had pretty good average accuracy. However, good rankings are not enough, as BP timed out so frequently as to be omitted from the figures.

The erratic behavior of BP and its subsequent failure as a search heuristic suggest future investigation into variance across surveys: a quick check of our data showed that it suffered much greater variance than the other approaches in almost every assessment of accuracy; this is consistent with its non-convergence (observed on about 5% of runs), and the strong biases engendered by its strong assumption that every variable is a sole-support to some clause.

At this point, though, the experiments indicate that accurate, temperate, bias estimators that are right about their strongest beliefs make for the best search heuristics. Overall they confirm the importance of bias in efficient search.

6.3 An Improved SAT-Solving System

The experiments summarized above comprise an iterative process of assessing bias estimators' properties, determining which properties are most useful to search when used appropriately, and repeating from the beginning. Though this was not a fundamental goal, the cycle also yields a recommended "best-performing" SAT-solving combination, at least for random satisfiable 3-SAT with α near 4.11. In particular, by using EMSP-G as the variable and value ordering heuristic, and employing the conflict-avoiding strategy with a deactivation threshold of 0.6, we more than halve MiniSat's runtime when $n = 250$. Cursory trials show that if problem size doubles to 500 and we remain focused on satisfiable instances, then this configuration of EMSP-G within MiniSat typically finds a solution in about one minute, while regular MiniSat will either take about thirty minutes, or timeout by failing to find a solution within two hours.

7 Conclusions and Future Work

The main findings of these experiments indicate that probabilistic message-passing techniques are comparatively successful at estimating variable bias, and that successful bias estimation has a positive effect on heuristic search by a modern solver. Secondary contributions include a novel family of EM-based bias estimators, and a series of design insights culminating in a fast solver for hard random problems.

However, many important issues remain. For instance, "real-world" and unsatisfiable instances have not yet been considered by this bias estimation framework. This may require a finer-grained analysis of estimate quality that considers variance across multiple runs with various random seeds. Further, the best way to use surveys for variable ordering cannot be settled conclusively by the limited span of branching strategies that have been studied to date. For instance, we waste some of the SP framework's power when we split the probability mass for the joker '*' state between positive and negative bias; future branching strategies might favor variables with low joker probabilities.

There are interesting abstract similarities with other problem-solving methodologies for constraint satisfaction. Bias measures the probability of a variable setting given satisfiability, while many local search methods maximize the probability of satisfiability given a certain variable setting [20]. Thus, the two targets are directly proportional via Bayes' Rule and techniques for one can be applied to the other. Another line of similar research calculates exact solution counts for individual constraints as a means of ordering variables and values [19]. Fundamentally, such an approach represents an exact and localized version of the approximate and interlinked techniques studied here.

Future applications of bias estimation include query answering and model counting. In the case of model counting, one detail omitted from discussion is that the normalization value $\omega_v^+ + \omega_v^-$ in Figure 1(b) (in fact, the log-partition function of a specific Markov Random Field) is proportional to the number of solutions for a given problem.

References

[1] J. Christopher Beck, Patrick Prosser, and Richard J. Wallace. Trying again to fail-first. *Recent Advances in Constraints, LNAI*, 3419, 2005.

[2] A. Braunstein, M. Mezard, and R. Zecchina. Survey propagation: An algorithm for satisfiability. *Random Structures and Algorithms*, 27:201–226, 2005.

[3] Rina Dechter, Kalev Kask, and Robert Mateescu. Iterative join-graph propagation. In *Proc. of 18th Int'l Conference on Uncertainty in A.I. (UAI '02), Edmonton, Canada*, pages 128–136, 2002.

[4] Arthur Dempster, Nan Laird, and Donald Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–39, 1977.

[5] Olivier Dubois and Gilles Dequen. A backbone-search heuristic for efficient solving of hard 3-SAT formulae. In *Proc. of 17th International Joint Conference on Artificial Intelligence (IJCAI '01), Seattle, WA*, 2001.

[6] Niklas Eén and Niklas Sörensson. An extensible SAT-solver. In *Proc. of 6th Int'l Conference on Theory and Applications of SAT (SAT '03), Portofino, Italy*, 2003.

[7] Carla Gomes, Bart Selman, Nuno Crato, and Henry Kautz. Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. *Journal of Automated Reasoning*, 24(1-2):67–100, 2000.

[8] Eric Hsu, Matthew Kitching, Fahiem Bacchus, and Sheila McIlraith. Using EM to find likely assignments for solving CSP's. In *Proc. of 22nd National Conf. on AI (AAAI '07), Vancouver, Canada*, 2007.

[9] Eric Hsu and Sheila McIlraith. Characterizing propagation methods for boolean satisfiability. In *Proc. of 9th International Conference on Theory and Applications of SAT (SAT '06), Seattle, WA*, 2006.

[10] Kalev Kask, Rina Dechter, and Vibhav Gogate. Counting-based look-ahead schemes for constraint satisfaction. In *Proc. of 10th Int'l Conf. on Constraint Programming (CP '04), Toronto, Canada*, 2004.

[11] Philip Kilby, John Slaney, Sylvie Thiébaux, and Toby Walsh. Backbones and backdoors in satisfiability. In *Proc. of 20th National Conference on A.I. (AAAI '05), Pittsburgh, PA*, 2005.

[12] Lukas Kroc, Ashish Sabharwal, and Bart Selman. Survey propagation revisited. In *Proc. of 23rd International Conference on Uncertainty in A.I. (UAI '07), Vancouver, Canada*, 2007.

[13] Frank R. Kschischang, Brendan J. Frey, and Hans-Andrea Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2), 2001.

[14] Elitza Maneva, Elchanan Mossel, and Martin Wainwright. A new look at survey propagation and its generalizations. *Journal of the ACM*, 54(4):2–41, 2007.

[15] Rèmi Monasson, Riccardo Zecchina, Scott Kirkpatrick, Bart Selman, and Lidror Troyansky. Determining computational complexity from characteristic phase transitions. *Nature*, 400(7):133–137, 1999.

[16] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

[17] Josh Singer, Ian Gent, and Alan Smaill. Backbone fragility and the local search cost peak. *Journal of Artificial Intelligence Research*, 12:235–270, 2000.

[18] Ryan Williams, Carla Gomes, and Bart Selman. Backdoors to typical case complexity. In *Proc. of 18th International Joint Conference on A.I. (IJCAI '03), Acapulco, Mexico*, 2003.

[19] Alessandro Zanarini and Gilles Pesant. Solution counting algorithms for constraint-centered search heuristics. In *13th Int'l Conf. on Constraint Programming (CP '07), Providence, RI*, 2007.

[20] Weixiong Zhang. Configuration landscape analysis and backbone guided local search. Part I: Satisfiability and maximum satisfiability. *Artificial Intelligence*, 158(1):1–26, 2004.