Using Expectation Maximization to Find Likely Assignments for Solving CSP's

Eric I. Hsu and **Matthew Kitching** and **Fahiem Bacchus** and **Sheila A. McIlraith**
{eihsu, kitching, fbacchus, sheila}@cs.toronto.edu
Department of Computer Science
University of Toronto

Abstract

We present a new probabilistic framework for finding likely variable assignments in difficult constraint satisfaction problems. Finding such assignments is key to efficient search, but practical efforts have largely been limited to random guessing and heuristically designed weighting systems. In contrast, we derive a new version of Belief Propagation (BP) using the method of Expectation Maximization (EM). This allows us to differentiate between variables that are strongly biased toward particular values and those that are largely extraneous. Using EM also eliminates the threat of non-convergence associated with regular BP. Theoretically, the derivation exhibits appealing primal/dual semantics. Empirically, it produces an "EMBP"-based heuristic for solving constraint satisfaction problems, as illustrated with respect to the Quasigroup with Holes domain. EMBP outperforms existing techniques for guiding variable and value ordering during backtracking search on this problem.

1 Introduction

Any hard but satisfiable combinatorial problem represents some set of solutions. Elements of this set are not directly accessible—otherwise we could solve the problem immediately. On the other hand, it may still be tractable to estimate statistical information about such solutions, even without direct access. For example, Belief Propagation (BP) [16, 14] is a technique that, when applied to a constraint satisfaction problem (CSP), can estimate the percentage of solutions that contain a particular variable assignment. More generally, BP computes a *survey*, which is a vector of such estimates—one for each possible value of every variable in the CSP. Such surveys can then provide useful heuristic information for an underlying search framework.

Here we exploit the Expectation Maximization (EM) framework [7] to derive a new version of BP for CSP's, as exemplified by the Quasigroup with Holes (QWH) problem [12]. The yield of this process is threefold. First, the derivation shows how to find and set the variables that make up the densely constrained core of a problem instance. Further, it characterizes BP's poorly-understood dynamics as a special case of primal/dual optimization on the search and inference formulations of a constraint problem [5]. Finally, the derivation yields an "EMBP"-based heuristic that eliminates the convergence problems of BP. We use this heuristic to solve QWH problems, and demonstrate empirically that it usually outperforms both BP and standard heuristics.

In the remaining sections we define some necessary background concepts before proceeding to derive EMBP. After visiting related work and pragmatic issues, we proceed to demonstrate empirical gains and interesting behaviors. Finally, we consider theoretical insights and open questions that arise from the derivation and the experiments.

2 Background

Algebraic structures like quasigroups have generated great interest within various areas of Artificial Intelligence. Not only do their underlying combinatorics capture the essence of realistic problems like scheduling, coloring, error-correcting coding, and resource allocation, but as experimental benchmarks they also possess a number of intriguing properties [8]. Namely, their most difficult instances exhibit "heavy-tailed" behavior making them extremely sensitive to variable and value ordering heuristics [10]. This may be because QWH and its generalizations often exhibit small "backdoors": identifying a dense inner problem and solving it first can be the key to tractability [19, 1]. Below we formally define this domain.

2.1 The Quasigroup with Holes Problem

Definition 1 (Latin Square) *A* Latin Square *of order d is represented by a $d \times d$ array of cell variables. Each cell takes a value from $D = \{1, \ldots, d\}$ such that no value occurs twice in any row or column. (The array denotes the multiplication table for an algebraic quasigroup.)*

Definition 2 (Quasigroup with Holes (QWH)) *A (Balanced)* Quasigroup with Holes *problem is a completed Latin square that has had a given percentage of entries erased by the careful process detailed in [12]. To solve a problem means finding valid values for the erased cells.*

Definition 3 (CSP Formulation for QWH) *QWH can be formulated as a CSP in a number of ways. Here we associate a variable with every cell. Variables associated with hole cells have the domain of values $\{1, \ldots, d\}$, while variables associated with filled in cells have the specified value as the single element of their domain. Finally we have $2d$ "alldiff" [17] constraints, one over each row and column.*

2.2 Biases and Surveys

To solve CSP's, we develop a heuristic that estimates the most likely values for each of the variables in the otherwise

inaccessible space of satisfying assignments. That is, for every variable assignment $X = a$ we estimate the proportion of solutions where X is assigned the value a. Herein we will use the following QWH-specific terminology:

Definition 4 (Variable Bias) *A* Variable Bias $\theta_{a,b}(v)$ *for the cell at row* a *and column* b *estimates the probability that it holds value* v *in a randomly sampled solution to a given QWH instance. That is,* $\theta_{a,b}$ *represents a probability distribution over the values available for the variable associated with the cell in question—the term "survey" denotes a set of bias distributions, one for each variable.*

2.3 The Expectation Maximization Framework

In the past, the Belief Propagation algorithm has been used to compute surveys for QWH [11]. To derive a replacement technique with improved theoretical semantics, algorithmic behavior, and empirical performance, we employ the Expectation Maximization framework for parameter estimation in the face of incomplete data.

EM [7] is an iterative algorithm for estimating model parameters from data when that data is complicated by the presence of hidden variables. At a high level, the framework assumes that a vector Y of observations was generated by some probability model parameterized by the vector Θ. The goal is to find a setting of Θ to maximize the probability of the observation, $P(Y|\Theta)$. This would itself be straightforward, but EM deals with the case where we additionally posit some latent variables Z that helped generate Y, and that we did not get to observe. So we want Θ to maximize $P(Y, Z|\Theta)$, but we cannot marginalize over Z (as in $\sum_Z P(Y, Z|\Theta)$) since it is unobserved.

The solution is to bootstrap between hypothesizing an artificial distribution $Q(Z)$ to estimate $P(Z|Y, \Theta)$, and using this distribution to maximize the *expected* likelihood $P(Y, Z|\Theta)$ with respect to Θ. The first step is called the **E**xpectation Step, and uses the current value of Θ to update $Q()$; the second step, the **M**aximization Step, uses the current value of $Q()$ to update Θ under expectation. Finally, the two steps are typically combined into a single update rule that starts with randomized initial values and repeats to guaranteed convergence.

In the next section we will demonstrate how to derive such an update rule for estimating variable biases, by applying the EM methodology. The rule produces a series of surveys that can be used as a variable and value ordering heuristic within an overall CSP-solving framework. We call the rule "EMBP", indicating its origins within the general program of transforming traditional Belief Propagation-type algorithms to use EM [9].

Propagation-based update rules perform local search within the space of surveys, seeking a local maximum in survey likelihood. Using EM guarantees that we will never increase our distance from the nearest local maximum in likelihood, thus ensuring convergence. In contrast, regular BP does calculate the correct gradient for directing search toward the nearest maximum, but can take overly large steps that overshoot such maxima and enter new regions of the search space. Thus, from the same random seed, the two methods can return different local maxima—even when both manage to converge. In practice, the values returned by EMBP seem to work better, at least on heavy-tailed problems like QWH. An additional theoretical benefit of formulating BP within EM will be to characterize the algorithm's operation as an optimized alternation between primal and dual search and inference problems.

3 Deriving an EMBP Update Rule

To actually compute surveys for a CSP instance, we, in essence, deceive the EM framework. Namely, we claim that we have observed a state where all of the constraints are satisfied, but that we did not get to see how exactly they were supported by their constituent variables. The general process is shared across domains, but the derivation below focuses on QWH. Here EM's task is to find variable biases that best support our claimed solution sighting; in doing so it hypothesizes and refines a $Q()$ distribution to indicate which permutation of $D = \{1, \ldots, d\}$ satisfies each row and column constraint. In the language of duality, we are hypothesizing *extensions*, or satisfying tuples, for the constraints.

Vector	Interpretation	Domain
Y	whether constraints are satisfied (observed)	$\{0, 1\}^{2d}$
Z	support permutations for constraints (unobserved)	$\{\vec{r}\}^d \times \{\vec{c}\}^d$
Θ	variable biases (parameters)	$[0, 1]^{d^3}$

Table 1: EM Formulation for the QWH Problem.

To that end, Table 1 represents the QWH problem within the EM methodology. Y represents a $2d$ vector of ones and zeros indicating whether each row and column constraint is satisfied—it can be considered to be set to all ones, corresponding to the claimed observation that all constraints are satisfied. (We actually never have to explicitly represent Y in the derivation that follows). Similarly, Z represents satisfying tuples for a problem's constraints, but will take on various conceptual forms in the derivation. The main idea is that it represents hidden information on exactly how the constraints were satisfied. Initially, it will consist of d row vectors ($\vec{r}$) and d column vectors ($\vec{c}$). Each such vector is itself of dimension d and represents the permutation of $\{1, \ldots, d\}$ that instantiates the constraint. (So for instance, $\vec{r}_3[1] = 5$ indicates that the first column of the third row contains the entry 5.) Finally, the Θ parameterizing EM's $Q()$ distribution over Z is the vector of variable biases $\{\theta_{a,b}(v)\}$ that we wish to optimize. That is, for each cell indexed by row a, column b, and each value v that can be placed in that cell, we are asking EM for the bias $\theta_{a,b}(v)$ representing the probability that this cell takes the value v in a claimed solution.

At a higher level, we use EM to alternate between "soft," i.e., "probabilistic," or "non-integral," solutions to the primal and dual [5] representations of the CSP. (Here, "soft" means assigning weights to various possibilities via a probability distribution, rather than just choosing one outright.) In setting Θ, we weight the possible values of variables rep-

resenting the cells of a Latin square. In constructing $Q()$, we weight the possible extensions to constraints representing the rows and columns of the square; this amounts to choosing permutations of D to serve as satisfying tuples. Initializing with random values for Θ, we use its current value to construct $Q()$, then use $Q()$ to produce an updated version of Θ, repeating to guaranteed convergence.

3.1 E-Step: Hypothesizing Configurations

The first step of the EM derivation for QWH is to construct an artificial distribution function $Q(Z)$ over the total configuration Z of a Latin square. $Q(Z)$ represents each configuration's probability given the cells' biases and the observation that it is a valid one: $P(Z|Y,\Theta)$. We actually do not need a general expression for $Q(Z)$ itself, but its elaboration will clarify the intervening stages.

In particular, we decompose the variable Z into row vectors $\{\vec{r}_i\}$ and column vectors $\{\vec{c}_j\}$, each representing the permutation of $D = \{1,\ldots,d\}$ that instantiates the corresponding row or column constraint. (As such, i and j range from 1 to d). We will not have to worry about enforcing consistency over such hypotheses, for instance by specifying that $\vec{r}_a[b] = \vec{c}_b[a]$, because ultimately the entire probability is conditioned on our (hypothetical) observation Y of a satisfying configuration. In other words, the dependence between constraints will be enforced later, by the M-step.

Thus, we can compose $Q(Z)$ from a second operator, $q()$, that returns the probability of a given configuration for a particular row or column: $Q(Z) = \prod_{i,j=1}^{d} q(\vec{r}_i)q(\vec{c}_j)$. Recall that none of these constructions are explicitly represented within an implementation of EMBP. Rather, they serve to formalize its operation as optimal coordinate ascent over the primal (search) and dual (inference) problems. Here optimality means that EMBP derives exact step sizes that will never overshoot a local maximum in likelihood, thus guaranteeing convergence.

3.2 M-Step: Maximizing the Likelihood

At this point we fix the current Q-distribution and use it to update the cell biases $\theta_{a,b}(v)$. More specifically, we maximize the expected logarithm of the probability that Y given Θ. Using the logarithm ensures convergence via Jensen's Inequality: $\log E[p(x)] \geq E[\log p(x)]$, as elaborated, for example, in [15].

The upshot is that we must set Θ to maximize a Lagrangian function $\mathcal{L}(\Theta) = \mathcal{F}(\Theta) - \mathcal{P}(\Theta)$, where $\mathcal{F}$ represents the expectation in question, and the penalty function $\mathcal{P}(\Theta) = \sum_{i,j} \lambda_{i,j}(\sum_v \theta_{i,j}(v) - 1)$ introduces a Lagrange multiplier $\lambda_{i,j}$ for each cell, ensuring that the bias $\theta_{i,j}$ for that cell sums to 1 across all possible values. The main object of interest is the expected log-likelihood function, $\mathcal{F}$:

$$\begin{aligned} \mathcal{F}(\Theta) &= E_Q[\log P(Z,Y|\Theta)] \\ &= \textstyle\sum_Z Q(Z)\, \log P(Z,Y|\Theta) \end{aligned} \tag{1}$$

Because we have defined Z to range over only satisfying tuples, any of its values already implies the observation that $Y = 1$, allowing us to omit Y from the above probabilities. By proceeding to decompose Z into its constituent row and column configurations, we derive:

$$= \sum_{\substack{\vec{r}_1,\ldots,\vec{r}_d,\\ \vec{c}_1,\ldots,\vec{c}_d}} \left[\prod_{i,j=1}^{d} q(\vec{r}_i)q(\vec{c}_j)\ \log \prod_{i,j=1}^{d} p(\vec{r}_i;\Theta)p(\vec{c}_j;\Theta)\right] \tag{2}$$

The $p()$ distributions above correspond to the hypothesized $q()$ distributions, and express the true probability of a row or column configuration, given the current biases of the variables. So to recapitulate, (2) is a decomposed representation for the expected log-likelihood over all possible satisfying configurations, given a fixed setting of Θ. Our goal is to choose a Θ to maximize this likelihood, with the extra constraint $\mathcal{P}(\Theta)$ that it must represent valid probability distributions.

To that end, the next step is to transform the logarithmic factor in (2) by making logs of products into sums of logs and expressing the two probabilities directly in terms of Θ:

$$\begin{aligned} &\log \prod_{i,j=1}^{d} p(\vec{r}_i;\Theta)p(\vec{c}_j;\Theta) = \\ &\quad \sum_{i=1}^{d}\sum_{k=1}^{d} \log \theta_{i,k}(\vec{r}_i[k]) + \sum_{j=1}^{d}\sum_{k=1}^{d} \log \theta_{k,j}(\vec{c}_j[k]) \end{aligned} \tag{3}$$

That is, the probability that Θ bears out a specific row (resp. column) configuration $\vec{r}_i$ is just the chance that the bias $\theta_{i,k}$ for each cell in the row produces the value $\vec{r}_i[k]$ specified by that configuration. Upon substituting this expression back into (2), we can optimize the entire Lagrangian $\mathcal{L}(\Theta)$ by taking its derivative with respect to each individual bias:

$$\begin{aligned} \tfrac{d\mathcal{L}}{d\theta_{a,b}(v)} \quad = \quad & \sum_{\substack{\vec{r}_1,\ldots,\vec{r}_d,\\ \vec{c}_1,\ldots,\vec{c}_d\\ \text{s.t. } \vec{r}_a[b]=v}} \prod_{i,j=1}^{d} q(\vec{r}_i)\, q(\vec{c}_j) \cdot (\theta_{a,b}(v))^{-1} \\ + \quad & \sum_{\substack{\vec{r}_1,\ldots,\vec{r}_d,\\ \vec{c}_1,\ldots,\vec{c}_d\\ \text{s.t. } \vec{c}_b[a]=v}} \prod_{i,j=1}^{d} q(\vec{r}_i)\, q(\vec{c}_j) \cdot (\theta_{a,b}(v))^{-1} \\ - \quad & \lambda_{a,b} \end{aligned} \tag{4}$$

In other words, $\mathcal{L}$ varies linearly with $(\theta_{a,b}(v))^{-1}$, weighted by the probabilities of those total configurations that have put v in the correct position (b) for the row (a) in question, along with those that have placed it correctly at position a for column b. From this more holistic perspective, we can revert to denoting these total configurations by Z, and optimize by

setting the derivative to zero:

$$\frac{d\mathcal{L}}{d\theta_{a,b}(v)} = 0 \quad \Rightarrow \quad \sum_{Z:\, \vec{r}_a[b] = v} Q(Z) \cdot (\theta_{a,b}(v))^{-1} + \sum_{Z:\, \vec{c}_b[a] = v} Q(Z) \cdot (\theta_{a,b}(v))^{-1} = \lambda_{a,b} \tag{5}$$

Solving for $\theta_{a,b}(v)$ yields an update rule that is expressed in terms of the hypothesized Q-distribution on configurations:

$$\Rightarrow \theta_{a,b}(v) = \frac{\sum_{Z:\, \vec{r}_a[b] = v} Q(Z) \;+ \sum_{Z:\, \vec{c}_b[a] = v} Q(Z)}{\mathcal{N}} \tag{6}$$

Here $\mathcal{N} \triangleq \lambda_{a,b}$ serves as a normalizing constant, and is defined to equal the summation of the expression in the numerator over all values of v. (This is readily confirmed by summing both sides of (6) over v and recalling that the biases for a cell must add up to 1.)

Intuitively, this equation tells us the optimal way to take the variable representing cell (a, b) in a Latin square and bias it toward its various possible values. Namely, we consult our $Q()$ distribution over satisfying tuples for the row and column constraints, and score each possible value by additively combining the likelihoods of tuples where it appears in this cell. By rescaling these scores by their sum we produce a distribution $\theta_{a,b}$ representing our desired biases.

3.3 Combining Steps: The EMBP Update Rule

To complete the derivation, we can now unite the two steps of EMBP into a single update rule. Recall that for the E-Step we declined to specify a fixed form for $Q(Z)$, the estimated probability that the cells are in a particular satisfying configuration. Now, we can choose an expression well-suited for substitution into (6).

Theoretically, an initial, most extreme approach is for Z to range over explicitly represented solutions to the entire Latin square. Clearly, this presupposes the solution to the problem at hand, and is thus trivially impractical. However, such "total consistency" serves to define one extreme in a spectrum of inferential power. Below we define three increasingly pragmatic approaches, the last two of which are actually implemented.

Less ambitious approaches than total consistency might decompose Z into row and column constraints, and enforce various degrees of consistency amongst and within these entities. Moving down the spectrum of constraint complexity, then, a first practical idea might be to explicitly represent Z as a series of tuples representing rows and columns, as in the derivation. We would then achieve consistency between such row and column possibilities by using their $Q()$ distribution values to update Θ via (6), and then using Θ to update $Q()$. It turns out, though, that this first approach is still intractable. If u is the number of unassigned variables in a row or column constraint, then the resulting space of $\mathcal{O}(u!)$ possible tuples is unmanageable for large problems.

However, we can exploit specific knowledge about the *alldiff* constraint underlying the rows and columns in order to reduce the complexity of exactly computing such a $Q(Z)$. That is, we can implicitly calculate the probability of each possible tuple without explicitly representing one such probability for every possible tuple. We have implemented a $\mathcal{O}(d \cdot 2^d)$ dynamic programming algorithm for doing so exactly. Using this technique yields a second approach, a completed EMBP rule that we will call "**EMBP-e**," but we will omit its details from this presentation.

Instead, we show how to complete the rule via a further approximation that is still more efficient in practice. The main idea remains the same: rather than summing over the probabilities of all satisfying configurations, after filtering out those whose rows or columns do not match the given bias, express the sum itself as a decomposed probability.

Note that the first summation in (6) constructs the probability that row a places value v in position b, conditioned on the space of all satisfying total configurations. Since $\vec{r}_a$ is supposed to range exclusively over satisfying configurations, this can be approximated by the probability that none of the other entries in the row has chosen v. With analogous reasoning for columns, we can derive a second completed update rule, and our third overall approach, "**EMBP-a**":

$$\theta_{a,b}(v) = \frac{\prod_{b' \in D/b} (1 - \theta_{a,b'}(v)) \;+ \prod_{a' \in D/a} (1 - \theta_{a',b}(v))}{\mathcal{N}} \tag{7}$$

Here again, $\mathcal{N}$ is a normalizing constant equal to the sum of the expression in the numerator over all values of v.

Within the spectrum of inferential power, this approach enforces pairwise (arc) consistency between the variable in question and the other variables in its rows and columns, producing $\mathcal{O}(u)$ complexity. In contrast, **EMBP-e** is founded on global consistency between *every* pair of variables in the constraint, and the initial, degenerately impractical approach assumes total consistency across the entire problem. Under experimentation, Equation (7) turns out to be the most effective compromise between the complexity of its computation and the accuracy of its result.

In summary, we base a cell's bias on the probability that it is realized by the hidden configuration Z of its row (resp. column) constraint (6). We express this probability via the current bias of neighboring cells and substitute to produce a single combined update rule (7). This framework generalizes to arbitrary CSP's; any derivation would proceed directly to an expression like (6), expressing bias as a sum of realizations over whatever sorts of constraints contain a given variable—here, a pair of *alldiff*'s. The only domain-specific engineering task is to efficiently represent such $Q(Z)$'s for substitution into completed update rules.

4 Discussion

Regular BP Applying traditional Belief Propagation produces a very similar family of update rules, but by a completely different route. For instance, replacing the addition in the numerator of (7) with a multiplication actually yields an expression that could have been derived from regular BP. This modified update rule would ambitiously couple a cell's duties to its two constraints by means of a geometric average. In contrast, EMBP's probabilistic underpinnings result

in a more conservative arithmetic average. This is the difference that guarantees convergence for EMBP but not for BP, even as both approaches seek local maxima of the same (negative) free energy equations.

More generally, using EM eliminates a variety of lesser problems inherent to regular BP. Formally, BP's operation on CSP's is not well-understood outside of a limited scenario based on extreme values [6]. Given that the algorithm is not guaranteed to converge, its overall performance is highly sensitive to a parameter governing when to give up on trying to compute a survey and just use the current value. In fact, even when it converges, BP can arrive at different answers (local maxima in survey likelihood) than EMBP's.

Other Heuristics In general terms, almost all existing variable/value-ordering heuristics attempt to identify the "important" variables together with promising values for them. As such, they often perform similar operations to BP, just more heuristically than probabilistically. For instance, one of the most successful recent approaches to variable ordering is to score variables using a system of weights on the constraints in which they appear [3]. The weights are devised to measure the constraints' risks of being unsatisfied based on search experience and the current settings of the variables. This mirrors EMBP's alternation between precisely biasing variables to satisfy constraints most at risk of unsatisfaction, and probabilistically measuring the levels of risk for each constraint in terms of the current biases.

Indeed, the adoption of EMBP and its clearer primal/dual semantics demonstrates that one step of the algorithm is performing soft search, while the other step does soft inference in the same sense that an operation like arc consistency or constraint propagation performs hard inference. This helps to explain the observation that empirically, EMBP alone performs better than any such heuristics alone. (At the same time this does not unilaterally raise one method above all others—they can be used in partnership, as will be described in Section 5.)

Primal/Dual Semantics Recall that the primal version of a CSP consists of selecting values for variables, subject to constraints, as usual. In the dual problem, the constraints become variables that range over satisfying tuples drawn from the constraints' extensions. At the same time, the variables define a new set of constraints: the selected tuples must be consistent with one another. That is, when two constraints share a variable in the primal problem, the two tuples chosen to instantiate them in the dual must use the same value for this variable.

While the primal and dual versions of our CSP have lent intuitions to the derivation, the exact correspondence to EM still bears elaboration. In particular, the EMBP framework is performing optimal primal/dual (or perhaps, "dual/primal") solving in a soft assignment space. When the E-Step weights the most likely row and column permutations according to the current biases, it is solving the *dual* problem by choosing tuples that are born out by the current (probabilistic) variable settings. Conversely, when the M-Step sets the biases to reflect the hypothesized tuples, it operates in the *primal* space by (softly) choosing variable values to meet the dictates of the constraints. The claim of "optimal" solving stems from the same property of EM that underlies its convergence. On each E-Step or M-Step, EM receives a fixed solution to the dual or primal problem, respectively. The guarantee is that it will construct the new $Q()$ distribution or Θ that maximizes progress toward a local maximum, given what is already fixed. At least in a softened space, then, EMBP has struck a heretofore elusive balance between search and inference for solving CSP's.

5 Using EMBP in Practice

For QWH, the practical yield of the derivation is an update rule (7) that expresses the bias of a cell variable in terms of the biases of the other cells in the same row or column. If we sequentially update the biases of all the variables over all their values, we are guaranteed to converge to a local maximum in likelihood. This survey information is used as a variable and value ordering heuristic, directing a CSP solver toward the most extremely biased variables first, using their most likely values. In practice, we do this by simply choosing the variable whose maximal bias is itself maximal across all such biases for all variables. By branching on such a variable and first assigning it its maximally biased value, we seek to enter a subtree that probably contains a solution (as we have estimated that a large proportion of the solutions contain this assignment). The hope is that we will never have to backtrack out of this subtree, a wish that comes true surprisingly often in our experiments.

In this context there remain two more algorithmic details of interest. First is a parameter governing the size of a single *decimation* [4]. Upon computing a survey, we can decide to fix a certain number of the most highly-biased variables before computing a new survey on the reduced domains of the remaining variables. Each such step is called a decimation, and in practice the best decimation size is 1 (i.e., fixing only one variable at a time). On the one hand, this entails paying the computational cost of computing a new survey every time we fix a variable. On the other hand, the new surveys are more accurate in that they are conditioned upon the variables fixed so far, and thus avoid the problem of correlations between variable biases within a single decimation block.

A second parameter governs the *threshold* for turning off the survey module. In practice, the first few surveys for a problem return strong bias information for certain "important" variables. As these variables are fixed, subsequent surveys become weaker and weaker, converging toward uniform biases across all values for the remaining variables. Eventually, we wish to stop computing surveys—both because they are no longer paying returns in useful information, and because the remaining problem is now easily handled by a regular solver. At an intuitive level, this roughly corresponds to the notion that problems have a few "backdoor" [19, 13] variables that, when set correctly, render the remaining problem polynomially solvable by a simpler method. We have experimented with various definitions of a threshold that measure the level of entropy in bias estimations. However, in practice the best approach appears to be simply stopping the computation of surveys after instantiating a fixed percentage (5%) of the variables in a problem.

6 Experimental Results

To illustrate the practical use of EMBP, we compare the proposed EMBP methods with other commonly used variable ordering heuristics for the QWH problem. In particular, we compare making variable assignment choices using *EMBP-e* (the more exact form of EM survey computation discussed above), *EMBP-a* (the less expensive but more approximate EM survey computation of Eq. (7)), *BP-e* (the Belief Propagation algorithm [11] using the same distribution function $Q(Z)$ as used by the EMBP-e algorithm), *BP-a* (the Belief Propagation algorithm using the same approximate distribution function $Q(Z)$ used by the EMBP-a algorithm), *dom/deg* (picking the variable with lowest current domain/degree ratio), and a highly successful weighted constraint approach *dom/wdeg* proposed in [3]. The EMBP and BP approaches ran decimations of size one up to a threshhold where 5% of the variables were assigned. At that point *dom/wdeg* was used as the ordering heuristic.

We experimented with a range of QWH problems of size 15×15 and 17×17 and with a varying percentage of holes. We tested 100 randomly generated instances at each percentage. The experiments measured the time taken to solve the total set of 100 instances (in CPU seconds), the total number of nodes searched during the solving episode, and the total number of failures (within a timeout of 500 sec.). All experiments were preformed on 2.20 GHz Opteron machines.

QWH 15×15	EMBP-e	EMBP-a	BP-e	BP-a	dom/wdeg	dom/deg
QWH(45%)	3119	**37**	674	764	208	38
QWH(50%)	14109	**111**	2517	2411	3710	2002
QWH(55%)	22743	**546**	5574	3223	4813	15506
QWH(60%)	7795	868	8604	**567**	7376	19025
QWH(65%)	9056	935	15695	**263**	4266	13805
QWH(Total)	56823	**2499**	33066	7230	20375	50376

Table 2: Time in seconds of QWH 15×15 (100 instances with 500 second timeout). Best times in bold.

QWH 15×15	EMBP-e	EMBP-a	BP-e	BP-a	dom/wdeg	dom/deg
QWH(45%)	**0**	**0**	**0**	**0**	**0**	**0**
QWH(50%)	1	**0**	2	1	**0**	1
QWH(55%)	12	**0**	3	5	7	27
QWH(60%)	**0**	1	4	**0**	2	34
QWH(65%)	0	1	3	**0**	3	25
QWH(Total)	13	**2**	12	6	12	87

Table 3: Timeouts on QWH 15×15 (100 instances with 500 second timeout)

In Tables 2 and 4, we see that EMBP-a solve the complete set of benchmarks in the fastest time, despite the expense of computing surveys. We see a substantial improvement over the *dom/wdeg* algorithm in both time and number of solutions found for all benchmarks. Although the EMBP-e algorithm did not solve as many problems as EMBP-a, it often outperformed EMBP-a with respect to nodes searched. Even when it expanded fewer nodes, the time to solve a problem was substantially greater than EMBP-a. This increased time is due to the considerable overhead in performing exact surveys during search. There is often a significant gap in the time taken to solve "easy instances".

QWH 17×17	EMBP-e	EMBP-a	BP-e	BP-a	dom/wdeg	dom/deg
QWH(45%)	3663	**172**	5063	8383	208	4429
QWH(50%)	15767	**2065**	13080	4471	3710	27586
QWH(55%)	12780	4338	15839	**3850**	4813	31610
QWH(60%)	4885	4708	28034	**1951**	7376	30501
QWH(65%)	5191	**1909**	29616	1927	4266	22026
QWH(Total)	42287	**13217**	91129	20585	20375	116154

Table 4: Time in seconds on QWH 17×17 (100 instances with 500 second timeout). Best times in bold.

QWH 17×17	EMBP-e	EMBP-a	BP-e	BP-a	dom/wdeg	dom/deg
QWH(45%)	**0**	**0**	2	6	**0**	2
QWH(50%)	5	**2**	17	6	6	48
QWH(55%)	**1**	7	13	5	7	58
QWH(60%)	6	8	18	**3**	13	57
QWH(65%)	8	3	53	2	7	42
QWH(Total)	**20**	**20**	102	22	34	207

Table 5: Timeouts on QWH 17×17 (100 instances with 500 second timeout)

Heuristic	**QWH(50%)**		**QWH(55%)**	
	Time	**Nodes**	**Time**	**Nodes**
EMBP-e	12508	**2.6***10^6	10925	**2.0***10^6
EMBP-a	**1047**	7.0*10^6	**837**	5.0*10^6

Table 6: Time and nodes visited of instances solved by both EMBP versions

To further illustrate this point, consider the comparisons between EMBP-e and EMBP-a on instances solved by both algorithms. Although the number of nodes searched is far fewer using EMBP-e, the time to solve the instances is greater. This shows the large overhead involved in running the exact EMBP and BP algorithms.

Finally, in Table 7 we show the number of backtracks over nodes that use the EM surveys. Table entries represent the total number of EM surveys performed, and the total number of backtracks over these nodes for all 100 instances of each benchmark. The number of backtracks is small compared to the total number of nodes performing the surveys, meaning that the predictions of variable assignments made by the EM algorithm were correct a large percentage of the time. In the case of EMBP-e, we only backtrack past 8% of all assignments. The percentage of backtracks is highest on the QWH(60%) benchmark at 17%.

Version	**EMBP-a**	
	Surveys	**Backtrack**
QWH(45%)	497	33
QWH(50%)	502	21
QWH(55%)	526	61
QWH(60%)	557	97
QWH(65%)	500	15
QWH(Total)	2582	227

Table 7: Total number of backtracks past EM nodes

7 Conclusions and Future Work

In this work we have solved the QWH problem by estimating the likelihoods that its variables hold their various possible values in a satisfying assignment. To do so, we derived an EMBP update rule by re-characterizing BP within the EM framework. Besides providing a clear semantics founded on the duality between search and inference, this yields a heuristic that is guaranteed to converge, and that can outperform existing approaches to the problem. The bulk of this process generalizes to arbitrary domains, but future applications involving different types of constraints will have to explore a spectrum of inferential approximations for each such constraint.

Experimenting with new domains will also broaden the comparison between the sorts of local maxima found by EMBP and regular BP. Characterizing the properties of maxima that attract convergent versus overstepping local searches would serve to explain differences in performance across domains. We would also like to rigorously compare the variables identified by our heuristic with those that comprise known backdoors and backbones.

In addition, estimating the fraction of solutions where a variable is fixed a certain way directly corresponds to model counting. Thus methods from this area can be used in a similar way to our technique, and vice versa. Likewise, there is a Bayesian correspondence between computing surveys via parameter estimation and finding soft solutions directly via local search. If Y is the event of a solution and Θ is a probabilistic setting for the variables in a problem, then the former maximizes $P(\Theta|Y)$ and the latter, $P(Y|\Theta)$.

A final future direction is to learn online from search experience, as per some of the latest advances in systematic search [2, 18]; on its own the methodology of computing surveys does not perform any such operation.

References

[1] Carlos Ansótegui, Ramón Béjar, César Fernàndez, Carla Gomes, and Carles Mateu. The impact of balancing on problem hardness in a highly structured domain. In *Proc. of 9th International Conference on Theory and Applications of Satisfiability Testing (SAT '06), Seattle, WA*, 2006.

[2] J. Christopher Beck. Multi-point constructive search. *Forthcoming, Journal of Artificial Intelligence Research*, 2007.

[3] Frédéric Boussemart, Fred Hemery, Christophe Lecoutre, and Lakhdar Sais. Boosting systematic search by weighting constraints. In *Proc. of 16th European Conference on Artificial Intelligence (ECAI '04), Valencia, Spain*, 2004.

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

[5] Rina Dechter. *Constraint Processing*. Morgan Kaufmann, San Mateo, 2003.

[6] Rina Dechter and Robert Mateescu. A simple insight into properties of iterative belief propagation. In *Proc. of 19th International Conference on Uncertainty in Artificial Intelligence (UAI '03), Acapulco, Mexico*, 2003.

[7] 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.

[8] Carla Gomes and David Shmoys. Completing quasigroups or latin squares: A structured graph coloring problem. In *Proc. of Computational Symposium on Graph Coloring and its Generalizations (COLOR '02), Ithaca, NY*, 2002.

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

[10] Tudor Hulubei and Barry O'Sullivan. Heavy-tailed runtime distributions: Heuristics, models, and optimal refutations. In *Proc. of 12th International Conference on Constraint Programming (CP '06), Nantes, France*, 2004.

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

[12] Henry Kautz, Yongshao Ruan, Dimitris Achlioptas, Carla Gomes, Bart Selman, and Mark Stickel. Balance and filtering in structured satisfiable problems. In *Proc. of 17th International Joint Conference on Artificial Intelligence (IJCAI '01), Seattle, WA*, 2001.

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

[14] 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.

[15] Radford Neal and Geoffrey Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M.I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. Kluwer Academic Publishers, 1998.

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

[17] Jean-Charles Régin. A filtering algorithm for constraints of difference in CSP's. In *Proc. of 12th Conference on Artificial Intelligence (AAAI '94), Seattle, WA*, pages 362–367, 1994.

[18] Meinolf Sellmann and Carlos Ansótegui. Disco–novo–gogo: Integrating local seach and complete search with restarts. In *Proc. of 21st National Conference on Artificial Intelligence (AAAI '06), Boston, MA*, pages 1051–1056, 2006.

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