Online Bayesian Moment Matching based SAT Solver Heuristics

Haonan Duan [*] [1] [2] Saeed Nejati [*] [1] George Trimponias [3] Pascal Poupart [1] [2] Vijay Ganesh [1]

Abstract

In this paper, we present a Bayesian Moment Matching (BMM) based method aimed at solving the *initialization problem* in Boolean SAT solvers. The initialization problem can be stated as follows: given a SAT formula ϕ, compute an initial order over the variables of ϕ and values/polarity for these variables such that the runtime of SAT solvers on input ϕ is minimized. At the start of a solver run, our BMM-based methods compute a posterior probability distribution for an assignment to the variables of the input formula after analyzing its clauses, which will then be used by the solver to initialize its search. We perform extensive experiments to evaluate the efficacy of our BMM-based heuristic against 4 other initialization methods (random, survey propagation, Jeroslow-Wang, and default) in state-of-the-art solvers, MapleCOMSPS and MapleLCMDistChronotBT over the SAT competition 2018 application benchmark, as well as the best-known solvers in the cryptographic category, namely, CryptoMiniSAT, Glucose, and MapleSAT. On the cryptographic benchmark, BMM-based solvers out-perform all other initialization methods. Further, the BMM-based MapleCOMSPS significantly out-perform the same solver using all other initialization methods by 12 additional instances solved and better average runtime, over the SAT 2018 competition benchmark.

1. Introduction

Over the last two decades, modern conflict-driven clause-learning (CDCL) Boolean satisfiability (SAT) solvers (Marques-Silva & Sakallah, 1999; Moskewicz et al., 2001) have had a transformative impact on many disciplines such as verification (Bradley, 2011), testing (Cadar et al., 2008), security (Avgerinos et al., 2011), and AI (Rintanen, 2009). The reason for this phenomenon is the dramatic improvement in the ability of SAT solvers in solving large real-world Boolean formulas with tens of millions of variables and clauses. While this performance is impressive, the demand for ever-more efficient solvers continues to grow unabated. CDCL solvers search the space of variable assignments exhaustively by starting with an empty assignment and repeatedly branching on variables to which truth values are assigned. When a conflict is found, the search backtracks by undoing one or several variable assignments. Additionally, a conflict analysis procedure finds a clause that is added to the formula to help the solver avoid the same conflict in the future. The search ends when a satisfying assignment is found or the space is proven to be unsatisfiable.

In the last 5 years, it has been shown that machine learning (ML) based heuristics for branching and restarts can dramatically improve SAT solver performance, ushering a new approach to solver design (Liang et al., 2016; 2017b; 2018). This impact can best be explained via the view that solvers are fundamentally proof systems, and machine learning methods are powerful ways of initializing, sequencing and selecting proof rules to optimally and adaptively solve formulas. Inspired by this success, we propose a Bayesian Moment Matching (BMM) based method to solve the initialization problem in SAT solvers.

The Initialization Problem in SAT Solvers. We define the initialization problem as follows: given a SAT formula ϕ, compute an *initial order* over the variables of ϕ and values/polarity for them such that the runtime of CDCL solvers on input ϕ is minimized. By initial order, we mean a total order over variables chosen by the CDCL solver S (and similarly, by initial value assignment we mean a mapping from variables to truth values) at the beginning of its search, i.e., before any variables have been branched upon by the solver S. Solver developers have known for a long time that the *initial order and value assignment to the variables of an input formula* can have a significant impact on the performance of CDCL SAT solvers.

BMM-based Method to Solve the Initialization Problem in SAT Solvers. The BMM method proposed in this paper is used as a pre-processor to a CDCL SAT solver. Our

[*]Equal contribution [1]University of Waterloo [2]Vector Institute [3]Huawei Noah's Ark Lab. Correspondence to: Pascal Poupart <ppoupart@uwaterloo.ca>, Vijay Ganesh <vijay.ganesh@uwaterloo.ca>.

Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020. Copyright 2020 by the author(s).

method takes as input a SAT formula ϕ and outputs a total order and assignment over the variables of ϕ. The method assigns a Bernoulli random variable to each variable of the input formula ϕ, associated with an unknown probability p of the variable being set to true (and $1 - p$ represents its probability of being false). For every clause C in the input formula ϕ, the belief about p is updated using Bayesian inference and moment matching. After our BMM method has scanned all the input clauses, it arrives at a posterior distribution that suggests an assignment that *ideally* satisfies most of the clauses (if not all of them).

The posterior distribution thus obtained is used to construct an assignment A that is most likely to satisfy the formula ϕ. One could treat such an assignment as a good guess for a satisfiable assignment to the formula ϕ (assuming it is satisfiable). Even if the formula is unsatisfiable, the hypothesis of our work is that the assignment A can be used as a good initial value (aka, polarity) selection for the variables in ϕ, as the CDCL solver starts its search. Further, the variables can be ranked in decreasing order based on the probability associated with their truth value in A (more certain the BMM is about a variable's value, the higher it is in the variable selection ranking). This ranking can be used as an initial variable selection order [1] by the CDCL SAT solver's branching heuristic.

An additional important point about our approach is that when the clause-learning method in the BMM-enhanced CDCL solver deductively learns a unit or a binary clause, it is used to update the posterior probability of the variables appropriately. The motivation behind this corrective feedback method from clause learning to the posterior probabilities of variables is that these BMM-based polarities are used to guide the solver's polarity/value selection heuristic during the run of the solver (not merely during the initialization), and thus get a further boost in performance.

We perform extensive experiments to test the efficacy of our BMM-based heuristics against state-of-the-art solvers. We show that BMM-based initialization of variable order and value selection in the context of CDCL SAT solvers can be effective for real-world instances obtained from verification, program analysis, software engineering, and cryptanalysis.

1.1. Contributions

1. **BMM-based Initialization Method.** We present the design and implementation of a novel BMM-based initialization method to address the "initialization problem" for value selection and variable order in CDCL SAT solvers. The key idea is to use clauses in the input formula as evidence to update a probability distribution of value assignment for each variable in the input formula. Our method can incrementally update and improve the posterior probability during the search by taking into account unit and binary learnt clauses in a corrective feedback loop. (Section 4)

2. **Evaluation on Cryptographic Instances.** We perform an apple-to-apple comparison of BMM-based versions of CryptoMiniSAT, MapleSAT, and Glucose against their respective configurations using 4 other initialization methods on a set of hard cryptographic benchmarks encoding round reduced SHA-1 inversion attacks, with a timeout of 4 hours. We used these solvers since they are among the best solvers for hard cryptographic instances. More precisely, for each solver, we compared our BMM-based method against 4 other initialization methods (namely, default, random, Jeroslow-Wang (Jeroslow & Wang, 1990), and Survey-propagation (Braunstein et al., 2005)). Our BMM-based method significantly outperforms all other methods, where BMM-based MapleSAT inverts all of the given targets and BMM-based CryptoMiniSAT solves the instances 50% faster on average. (Sec. 6.1)

3. **Evaluation on SAT 2018 Application Instances[2].** We further compare the efficacy of BMM-based versions of MapleLCMDistChronoBT (winner of SAT 2018 competition) and MapleCOMSPS (Gold/Silver medalist in SAT 2016/2017 competition), against the corresponding respective versions with 4 other initialization methods (listed above). We observe that our BMM-based method outperforms all other versions with 12 additional instances solved and an average runtime speedup of 15.2%, compared to the next best method, namely, Jeroslow-Wang. (Section 6.2)

4. **BMM Consistency Proof in Naïve Bayes Model.** While BMM has been successful in many settings (Omar, 2016; Hsu & Poupart, 2016; Jaini & Poupart, 2016; Rashwan et al., 2016), the consistency of BMM is still an open problem. We prove the consistency of BMM in the naïve Bayes model by formulating it as a stochastic approximation (SA) problem. Even though BMM for SAT is unarguably more complicated than for the naïve Bayes model, they do share the important property that the moment matching method is applied to the same distribution family; furthermore, they can both be formulated as a SA problem. Therefore, a formal proof of the consistency of BMM on the naïve Bayes model based on SA theory gives us

[1]The term *variable selection or branching method* refers to a procedure that computes a total order over the variables of an input formula and chooses the highest-ranked variable to assign a value to, during the run of the solver. The term *value or polarity selection method* refers to computing a mapping from variables of an input formula to truth values during the run of a solver.

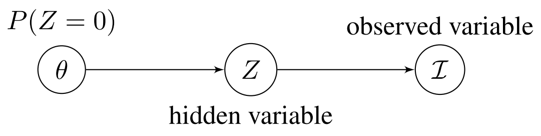
[2]`http://sat2018.forsyte.tuwien.ac.at/benchmarks/`

some motivation and confidence for applying BMM for the SAT problem from a theoretical perspective (see Section 2 and the supplemental material).

2. Bayesian Moment Matching

In this section, we give an overview of the Bayesian moment matching algorithm with an application to the naïve Bayes model and provide a consistency proof of BMM in this fundamental setting. BMM for mixture models was proposed to prevent the exponential growth of mixture components in online Bayesian learning (Jaini & Poupart, 2016; Rashwan et al., 2016). A distribution belonging to the same family as the prior is used to approximate the posterior by matching the sufficient moments, in order to reduce the complexity of posterior distributions. BMM has been successful in the context of topic modelling (Omar, 2016; Hsu & Poupart, 2016), hidden Markov models (Jaini et al., 2016) and sum product networks (Rashwan et al., 2016).

Problem Setup. We use Z to represent a binary hidden variable and $\mathcal{I}$ to represent a binary observable variable. Furthermore, we assume that the conditional distribution of $\mathcal{I}|Z$ is fully known. c_1 and c_2 are used to denote $P(\mathcal{I}=0|Z=0)$ and $P(\mathcal{I}=0|Z=1)$ respectively. Let θ represent the unknown probability of the hidden variable, i.e., $P(Z=0)$, the quantity we wish to infer from $i.i.d.$ observations $\{\mathcal{I}_1, \mathcal{I}_2, \dots\}$ in an online and Bayesian fashion.



A beta distribution $Beta(\theta;\alpha_0,\beta_0)$ is chosen as the initial prior over θ, i.e., $P(\theta) = Beta(\theta;\alpha_0,\beta_0) = \frac{1}{B(\alpha_0,\beta_0)}\theta^{\alpha_0-1}(1-\theta)^{\beta_0-1}$, where $B(\alpha_0,\beta_0)$ represents the beta function of α_0 and β_0 (Johnson et al., 1995). We choose a beta distribution because its support is a probability simplex and it is also conjugate with the likelihood (MacKay, 2002). The posterior after observing the first evidence $\mathcal{I}_1$ is:

$$\begin{aligned}
P(\theta|\mathcal{I}_1=0) &\propto P(\theta)P(\mathcal{I}_1=0|\theta) \\
&\propto \theta^{\alpha_0-1}(1-\theta)^{\beta_0-1}[\theta c_1 + (1-\theta)c_2] \\
&\propto c_1\theta^{\alpha_0}(1-\theta)^{\beta_0-1} + c_2\theta^{\alpha_0-1}(1-\theta)^{\beta_0} \\
P(\theta|\mathcal{I}_1=1) &\propto (1-c_1)\theta^{\alpha_0}(1-\theta)^{\beta_0-1} \\
&\quad + (1-c_2)\theta^{\alpha_0-1}(1-\theta)^{\beta_0}
\end{aligned}$$

The equations above suggest that the posterior is a mixture of two beta distributions after the first point is observed. Therefore, the number of mixture components in the posterior distributions will grow exponentially by a factor of two for each new observation, which makes inference intractable. To solve this problem, BMM approximates the mixture posterior with a single Beta distribution $\tilde{P}(\theta_1) = Beta(\theta_1;\alpha_1,\beta_1)$ by matching the first and second moments. Concretely, α_1, β_1 can be obtained by solving:

$$\begin{aligned}
\mathbb{E}_{\theta_1\sim Beta(\theta_1;\alpha_1,\beta_1)}[\theta_1] &:= \mathbb{E}_{\theta\sim P(\theta|\mathcal{I}_1)}[\theta] \\
\mathbb{E}_{\theta_1\sim Beta(\theta_1;\alpha_1,\beta_1)}[\theta_1^2] &:= \mathbb{E}_{\theta\sim P(\theta|\mathcal{I}_1)}[\theta^2],
\end{aligned}$$

where $\mathbb{E}_{\theta\sim P(\theta|\mathcal{I}_1)}[f(\theta)] = \int f(\theta)P(\theta|\mathcal{I}_1)d\theta$.

Consistency Analysis. The above process is repeated for each new observation. The approximate Beta posterior from the current step serves as the prior over θ for the next step. Even though some information is lost when we use a single beta distribution to approximate a mixture of betas, we prove the consistency of BMM by formulating it as a stochastic approximation (SA) problem.

If we apply BMM n consecutive times with observations $\{\mathcal{I}_1,\dots,\mathcal{I}_n\}$, our n^{th} estimate θ_n for θ will be by induction distributed according to $Beta(\theta_n;\alpha_n,\beta_n)$, for suitable α_n,β_n. Let $\mu_n = \frac{\alpha_n}{\alpha_n+\beta_n}$ denote the mean of θ_n, $\tau_n = \alpha_n+\beta_n$ the precision of θ_n, and $\mathcal{I}_{n+1}$ the binary random variable for a new instance. By matching the first two moments of the exact posterior $P(\theta_n|\mathcal{I}_{n+1})$ at step $n+1$ with the moments of a Beta distribution $Beta(\theta_{n+1};\alpha_{n+1},\beta_{n+1})$, we get two update equations, one for α_{n+1} and another for β_{n+1}. Equivalently, we can write the update equations in terms of μ_{n+1} and τ_{n+1} using $\alpha_n,\beta_n,\mu_n,\tau_n$ (μ_n,τ_n are functions of α_n,β_n). For instance, the update equation for μ_{n+1} is:

$$\begin{aligned}
\mu_{n+1} =& \mu_n + \frac{1}{\tau_n+1}\Big[\big(\frac{c_1\alpha_n}{c_1\alpha_n+c_2\beta_n} - \mu_n\big)(1-\mathcal{I}_{n+1}) \\
&+ \big(\frac{(1-c_1)\alpha_n}{(1-c_1)\alpha_n+(1-c_2)\beta_n} - \mu_n\big)\mathcal{I}_{n+1}\Big].
\end{aligned} \tag{1}$$

We can similarly write the update equation for τ_{n+1}.

The following theorem asserts that our BMM scheme has the property that the mean of the posterior distribution in Eq. (1) eventually converges to the true θ w.p. 1.

Theorem 1. *When performing BMM with the first and second moment in the naïve Bayes model, the update in Eq. (1) for the first moment converges almost surely to the true underlying θ:* $\Pr(\lim_{n\to\infty}\mu_n = \theta) = 1$.

The proof of the theorem is technical and included in the supplemental material. Briefly, we show that Eq. (1) satisfies the four sufficient conditions for consistency of an SA problem described in (Chen & Ryzhov, 2020).

3. BMM for SAT

We introduce a novel Bayesian perspective to solve the SAT problem. In our Bayesian formulation, each variable in

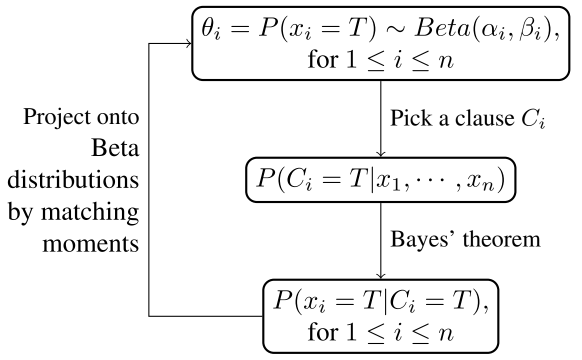

$\theta_i = P(x_i = T) \sim Beta(\alpha_i, \beta_i)$, for $1 \le i \le n$
Pick a clause C_i
$P(C_i = T | x_1, \cdots, x_n)$
Bayes' theorem
$P(x_i = T | C_i = T)$, for $1 \le i \le n$
Project onto Beta distributions by matching moments

Figure 1. A Beta prior is assigned to each variable in the beginning. The posteriors are then calculated each time when encountering a new clause. We project the posteriors back to Beta distributions using BMM, which serves as priors for the next clause.

the SAT formula is a Bernoulli random variable with an unknown probability being assigned to T (true) and each clause is treated as evidence. Our objective is to learn the unknown probability associated with each variable by BMM, which we illustrate with a toy SAT instance:

$$\begin{aligned} &C_1 : x \vee y \vee \neg z \\ &C_2 : x \vee y \vee z \\ &C_3 : x \vee \neg y \vee z \\ &C_4 : \neg x \vee \neg y \vee \neg z \end{aligned}$$

We use $\theta_x, \theta_y, \theta_z$ to denote $P(x = T), P(y = T), P(z = T)$ respectively. To estimate $\theta_x, \theta_y, \theta_z$ by Bayesian inference, we assume that each of them is initially distributed according to a Beta prior and that they are mutually independent. Concretely, the prior for the joint distribution is:

$$P(\theta_x, \theta_y, \theta_z) = \prod_{i=x,y,z} Beta(\theta_i; \alpha_i, \beta_i).$$

An instance is satisfiable if all of its clauses are satisfied. To satisfy a clause at least one of the literals needs to be satisfied, which can be done in many different ways if we have many literals in a clause. However, there is only one way to falsify the clause. Therefore, we define our likelihood function as the complement probability of falsifying the observed clause. For example, to falsify clause C_1, we should have $x = F, y = F, z = T$, and thus: $P(C_1|\theta_x, \theta_y, \theta_z) = 1 - (1 - \theta_x) \cdot (1 - \theta_y) \cdot \theta_z$. The posterior after seeing the first clause C_1 is:

$$\begin{aligned} &P(\theta_x, \theta_y, \theta_z | C_1) \propto P(\theta_x, \theta_y, \theta_z) P(C_1 | \theta_x, \theta_y, \theta_z) \\ &\propto P(\theta_x, \theta_y, \theta_z)[1 - (1 - \theta_x)(1 - \theta_y)\theta_z] \\ &\propto Beta(\theta_x; \alpha_x, \beta_x) \cdot Beta(\theta_y; \alpha_y, \beta_y) \cdot Beta(\theta_z; \alpha_z, \beta_z) \\ &- \frac{\beta_x}{\alpha_x + \beta_x} \frac{\beta_y}{\alpha_y + \beta_y} \frac{\alpha_z}{\alpha_z + \beta_z} Beta(\theta_x; \alpha_x, \beta_x + 1) \\ &\cdot Beta(\theta_y; \alpha_y, \beta_y + 1) \cdot Beta(\theta_z; \alpha_z + 1, \beta_z) \end{aligned}$$

Since the likelihood $1 - (1 - \theta_x)(1 - \theta_y)\theta_z$ can also be expressed as the sum of joint probabilities, we can see that the posterior is a mixture of products of Beta distributions. The number of mixture components will grow exponentially as more clauses are encountered. To solve this intractability issue, we approximate the true mixture $P(\theta_x, \theta_y, \theta_z | C_1)$ by a single product of Beta distributions using BMM:

$$\tilde{P}(\tilde{\theta}_x, \tilde{\theta}_y, \tilde{\theta}_z) = \prod_{i=x,y,z} Beta(\tilde{\theta}_i; \tilde{\alpha}_i, \tilde{\beta}_i)$$

The parameters $\tilde{\alpha}_x, \tilde{\beta}_x$ are then computed by matching the first and second moments of the marginal distribution of θ_x:

$$\begin{aligned} \mathbb{E}_{\tilde{\theta}_x \sim Beta(\tilde{\theta}_x; \tilde{\alpha}_x, \tilde{\beta}_x)}[\tilde{\theta}_x] &:= \mathbb{E}_{\theta_x \sim P_{\theta_x}(\theta_x | C_1)}[\theta_x] \\ \mathbb{E}_{\tilde{\theta}_x \sim Beta(\tilde{\theta}_x; \tilde{\alpha}_x, \tilde{\beta}_x)}[\tilde{\theta}_x^2] &:= \mathbb{E}_{\theta_x \sim P_{\theta_x}(\theta_x | C_1)}[\theta_x^2], \end{aligned}$$

where $P_{\theta_x}(\theta_x|C) = \int_0^1 \int_0^1 P(\theta_x, \theta_y, \theta_z | C) d\theta_y d\theta_z$. The parameters $\tilde{\alpha}_y, \tilde{\beta}_y, \tilde{\alpha}_z, \tilde{\beta}_z$ are computed similarly. Subsequently, $\tilde{P}(\tilde{\theta_x}, \tilde{\theta_y}, \tilde{\theta_z})$ is used as the prior when C_2 is observed. During one epoch, the above update is repeated once for each clause. Fig. 2 shows how $P(\theta_x), P(\theta_y), P(\theta_z)$ change as the number of epochs increases for this example.

The BMM algorithm for a general SAT instance is discussed in more detail in the supplemental material.

4. Bayesian Moment Matching as a SAT Solver Component

The learned probabilities collectively represent an assignment to the variables that maximizes the number of satisfied clauses. If all of the clauses are satisfied, the assignment represents a solution. There is no guarantee that the BMM algorithm converges to a solution. Furthermore, BMM cannot determine if an input formula is unsatisfiable (does not have any solutions). Stand-alone BMM is an incomplete solver. To make it a complete solver, we embed the BMM as a component inside a complete solver (e.g. a CDCL SAT solver). In this section, we discuss how we connect BMM as a component to a CDCL SAT solver. In short, we run BMM as a pre-processor component and initialize the search of our CDCL solver.

To initialize the search procedure of a SAT solver, we targetted the branching heuristic. Modern CDCL SAT solvers commonly use *look-back* branching heuristics, which means that they collect statistics about their variables during the search, and maintain scores (a.k.a activities) for them so that they can make educated guesses on which variable to pick next in the future. For example, we know that CDCL solvers (e.g., VSIDS or LRB branching heuristics) tend to pick branches that are more likely to prove a subspace unsatisfiable faster than methods that don't maintain such statistics (Liang et al., 2015). At each decision step, SAT

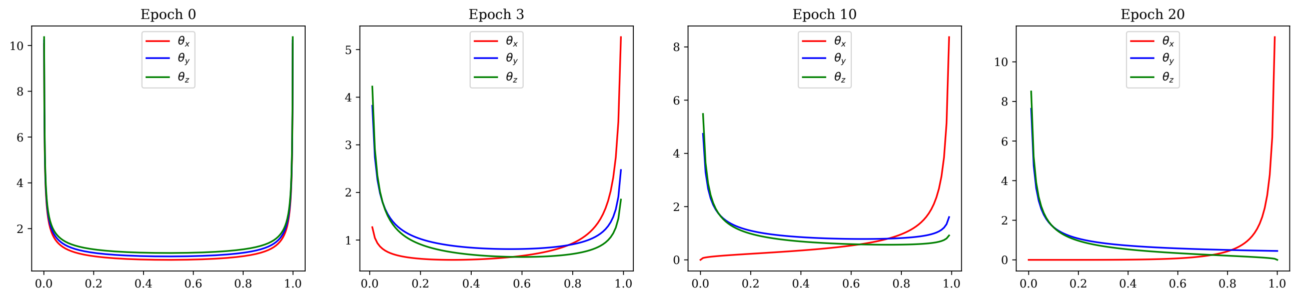


Figure 2. The probability density functions of θ_x, θ_y and θ_z for the example in section 3 using BMM update. As the number of epochs increases, the densities of θ_y and θ_z are shifting towards 0 while θ_x is shifting towards 1. This suggests an assignment $x := T, y := F, z := F$.

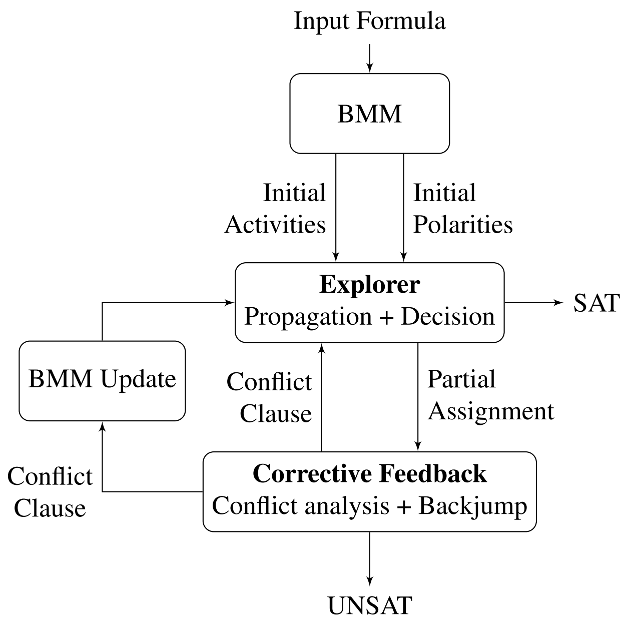


Figure 3. Overview of BMM as a component in a SAT Solver.

solvers ask two questions: which unassigned variable to pick (Variable order) and what value to assign to that variable (Polarity/Value selection). It is well known that these heuristics have a huge impact on the performance of a SAT solver. An important question as described in the introduction is how to initialize the variable values and the variable selection order at the start of the search where there are no previously seen data. This question is generally referred to as the *initialization problem*. We use BMM as a pre-processor that scans the clauses and computes the preferred initialization values before the search starts. We use 10 epochs for application instances (each clause is seen ten times) and empirically observed that it results in a good initial point while being efficient.

BMM-based Initial Value Selection: As the learned probabilities collectively represent an assignment to the variables of the input formula that satisfies most of the clauses, it is natural to hypothesize that they can be used as initial preferred values. We simply set the preferred value of a variable to True if the first moment $\mathbb{E}(\theta_x) > 0.5$ and False otherwise.

BMM-based Initial Variable Selection: Successful branching heuristics like VSIDS (Moskewicz et al., 2001) and LRB (Liang et al., 2016) that are widely used in modern SAT solvers, keep a score for each variable, called activity, which represents how much that variable was involved in conflict analysis recently. The variable with the highest score will be picked as the decision variable. At the start of the search, we do not have any information about the variables and which one is preferred over the others. Therefore it is very common to start from zero scores for all variables and build the ranking of variables based on the search statistics. However, having the learned probabilities, we can prioritize the variables before the search starts. In our experiments, we give higher priority to the variables with less uncertainty about a polarity (high probability of being either True or False). For each variable x, we define the $score(x)$ to be a number in the range [0.5, 1] as follows:

$$score(x) = \begin{cases} 1 - \mathbb{E}(\theta_x), & \mathbb{E}(\theta_x) < 0.5 \\ \mathbb{E}(\theta_x), & \mathbb{E}(\theta_x) \geq 0.5 \end{cases}$$

This is the same as saying for a variable x with the $Beta(\alpha_x, \beta_x)$ distribution: $score(x) = \max(\alpha_x, \beta_x)/(\alpha_x + \beta_x)$. The score will be 1, if BMM is certain that the variable x is False ($\mathbb{E}(\theta_x) = 0$), or True ($\mathbb{E}(\theta_x) = 1$), in a satisfying assignment.

Updating Posterior During Search (BMM Update). During a CDCL search, the solver might reach a conflicting state, where the partial assignment to the variables cannot be extended to a full assignment. At that point, the solver analyzes the root cause of this conflict and encodes this information as a clause (*conflict clause*). Conflict clauses are implied by the original formula, so they can be added to the original formula. The conflict clauses can thus be treated as new evidence. In the case that we use BMM probabilities to initialize polarities, the partial assignment that led to a conflict is derived from the BMM posterior distribution. This means that the new evidence has the necessary information to fix an inaccurate posterior. We update the posterior using this corrective feedback. However, we do this only for unit and binary clauses to keep the overhead low. We directly update the polarity of variables in the conflict clause.

Figure 3 shows a high level block diagram of where BMM fits in as a component in a CDCL SAT solver. The "Propagation + Decision" block is responsible for expanding the partial assignment by assigning values to unassigned variables and propagating this information to other variables. This block receives initial values for the order of variables and their preferred values from BMM. The "Conflict analysis + Backjump" is responsible for correcting the mistakes made by the explorer component. The BMM Update unit gets a copy of the conflict clause returned by this component and updates the probabilities. In other words, an approximate solution proposed by BMM is checked on the formula (by propagation), and if it does not satisfy the formula, the conflict analysis component gives corrective feedback about the inaccuracy of the probabilities. The variable order computed by our method is used only as an initial order for the solver's branching heuristic (VSIDS and LRB in the case of MapleCOMSPS). When a conflict clause is learned, the VSIDS or LRB scores will be updated for variables in that clause, thus the branching heuristics are dynamic. These scores (representing the order) are kept after each restart. We do not change the restart policy of the baseline solvers.

5. Description of Other Initialization Methods

Default. Most CDCL solvers simply initialize the activity scores of variables with zeroes and set the preferred polarity of variables to false. In this work whenever we say default or do not explicitly mention the initialization method, we mean the all zero and all false initialization.

Random. To verify that our proposed initialization method indeed improves the search and not randomly shuffles the variables and values, we compare with random initialization as a control experiment. In this method, polarities are randomly picked with 0.5 probability between true and false, and activity scores are set to a number between 0 and 1 picked uniformly at random.

Survey Propagation. Survey propagation is a message passing algorithm that was designed to find solutions for random k-SAT problems (Braunstein et al., 2005). They are mostly believed to be the hardest to solve when their clause to variable ratio is close to the experimental threshold of SAT-UNSAT regions. Survey propagation works over the factor graph representation of SAT instances. It generates messages that surveys over clusters of ordinary messages, and then uses these surveys to fix variables and simplify the problem by decimation.

Jeroslow-Wang. Jeroslow and Wang proposed a static branching heuristic (Jeroslow & Wang, 1990), which in some modern solvers is used for computing initial scores for literals. It assigns a score to each variable such that the variables that appear in shorter clauses get a higher score. The intuition is that these variables when assigned by the solver, create unit clauses sooner than others and allow unit propagation to imply many other literals. The score for each variable is computed as $score(x) = \sum_{x \in C, C \in \phi} 2^{-|C|}$, where ϕ is the input formula and C is a clause in ϕ.

6. Experimental Results

In this section we present and discuss the experimental evaluation of our BMM-based initialization method and compare it against 4 other initialization methods described in Section 5. We implemented the initialization methods in all of the solvers in a modular way. In other words, we kept all the implementation of solvers intact except for the initialization methods, so we can have an apple-to-apple comparison, between different versions of one solver.[3]

Cactus plots: The plots presented in this section are cactus plots. Each data point (X, Y) shows that X instances are solved under Y seconds. This means that solvers that are further to the right are solving more instances and solvers that are further to the bottom are solving instances faster.

For each combination of (solver, initialization method), we experimented with 3 configurations: initializing 1) polarities only, 2) activities only and 3) both of polarity and activity. Due to lack of space, for each combination we only report the best performing configuration. We observed that generally the third configuration performs the best, with the exception of survey propagation, where initializing polarity only performs better than the other two configurations.

6.1. Evaluation over Hard Cryptographic Instances

Experimental Setup. All jobs were run on Intel Xeon E5-2667 CPUs at 3.20GHz and 8GB of RAM. We used cryptographic instances encoding preimage of round reduced SHA-1 hash function. We encoded 22 rounds of SHA-1 and used 50 randomly generated hash values to be inverted. Time and memory limit for cryptographic instances was 4 hours and 8GB respectively.

Solver Descriptions. The solvers we used were MapleSAT (Liang et al., 2016), Glucose-4 (Audemard & Simon, 2018) and CryptoMiniSAT-5 (Soos, 2018). From the experiments

[3]The source code of our implementations is available at `https://github.com/saeednj/BMMSAT`.

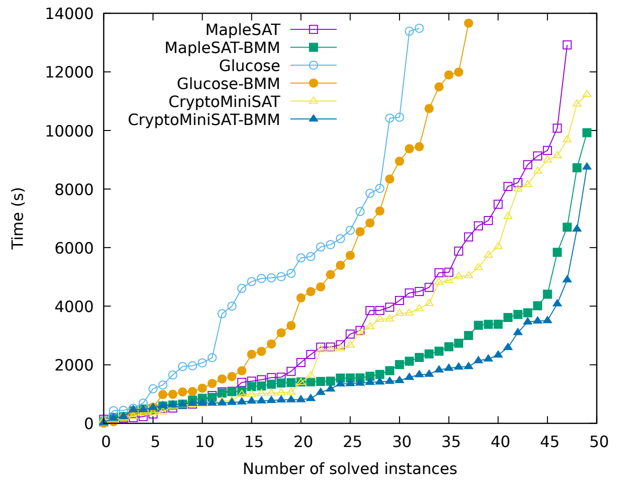


Figure 4. Performance comparison of MapleSAT, Glucose and CryptoMiniSAT solvers with default, and BMM initialization methods on hard cryptographic benchmarks.

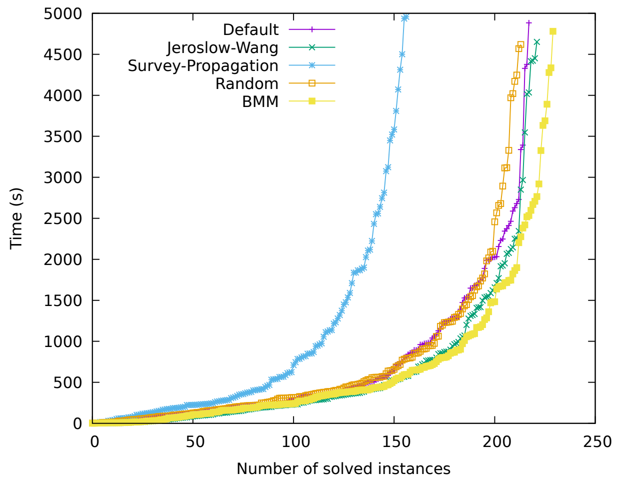


Figure 5. Performance comparison of different version of MapleCOMSPS on SAT competition 2018 benchmark.

performed on SHA-1 instances in the literature (Nossum, 2012; Nejati et al., 2017a;b), we know that these solvers are top performing solvers in this benchmark. We used 100 epochs for pre-processing and 1 epoch for updating BMM posterior.

Results. Table 1 gives details on the number of solved instances out the 50 hard cryptographic instances, where it can be seen that BMM version of MapleSAT is the only variant of MapleSAT that is able to solve all of the instances with much lower average runtime compared to other initialization methods. Also BMM version of CryptoMiniSAT solves the instances around 50% faster than the default version on average. Figure 4 shows how MapleSAT-BMM and CryptoMiniSAT-BMM have a clear advantage over other versions of these three solvers.

Table 1. Number of solved instances out of 50 hard cryptographic instances and average runtime (in seconds) of MapleSAT, Glucose and CryptoMiniSAT with different initialization methods.

	Initialization	Total	Avg. time
MapleSAT	Default	48	3645.08
	Random	48	3180.42
	Survey Propagation	40	3405.20
	Jeroslow-Wang	47	3389.27
	BMM	50	2238.85
Glucose	Default	33	4817.69
	Random	32	5741.74
	Survey Propagation	30	5386.74
	Jeroslow-Wang	32	6334.71
	BMM	38	4563.08
CryptoMiniSAT	Default	50	3475.06
	Random	50	3223.48
	Survey Propagation	41	3501.00
	Jeroslow-Wang	49	5387.20
	BMM	50	1706.63

6.2. Evaluation over SAT Competition 2018 Application Instances

Experimental Setup. All jobs were run on StarExec environment with Intel(R) Xeon(R) CPU E5 at 2.40GHz (Stump et al., 2014). We used the main track of the SAT competition 2018, which contains 400 instances coming from a variety of real-world application domains, like verification, graph problems, scheduling and combinatorics (Heule et al., 2019). Time limit for solving each instance was 5000 seconds (the same as SAT competitions) and memory limit was 8GB.

Solver Descriptions. The solvers that we used to incorporate BMM were MapleCOMSPS (gold/silver medalist of SAT competition 2016/2017) (Liang et al., 2017a) and MapleLCMDistChronoBT (winner of SAT competition 2018) (Ryvchin & Nadel, 2018). We used 10 epochs to compute the posterior in the pre-processing phase and 1 epoch for each learned unary and binary clause. MapleLCMDistChronoBT, switches between Distance, VSIDS and LRB branching heuristics. We initialized activity scores of all of these heuristics. Similarly we initialized both VSIDS and LRB in MapleCOMSPS.

Results. Table 2 shows the number of solved instances out of 400 instances by the two solvers described above, comparing BMM with other methods. Figure 5 depicts that BMM-based initialization beats all other methods, by solving more instances, and having lower average runtime on the solved instances. The closest performing method is the Jeroslow-Wang, which solves 4 more than default, but still, BMM solves 8 more instances than Jeroslow-Wang. In case of MapleLCMDistChronoBT, BMM-based initialization does not improve the number instances, however, it solves the instances 15% faster on average.

Table 2. Number of solved instances (out of 400) and average runtime (in seconds) of MapleCOMSPS and MapleLCMDistChronoBT and their variations on SAT competition 2018 benchmark. SAT column shows how many of the solved instances were satisfiable.

	Initialization	Total	SAT	Avg. time
MapleCOMSPS	Default	218	124	674.43
	Random	214	121	678.09
	Survey Propagation	157	100	862.30
	Jeroslow-Wang	222	128	654.05
	BMM	230	136	646.18
MapleLCMDist	Default	240	138	769.85
	Random	232	131	673.02
	Survey Propagation	173	109	885.50
	Jeroslow-Wang	235	134	655.98
	BMM	240	139	652.80

6.3. Discussion of Experimental Results

SAT vs. UNSAT. The posterior distribution that BMM learns, is supposed to form a solution to the input formula. Therefore we expect to see better performance in satisfiable instances rather than unsatisfiable instances, and in fact that is what we have observed in our experiments. Table 2 shows that the BMM-initialized MapleCOMSPS, solves 12 more satisfiable instances compared to the vanilla MapleCOMSPS, and solving the same number of unsatisfiable instances. All instances in our hard cryptographic benchmark are satisfiable (there exists a preimage to each hash target, and the task is to find it), and we specifically wanted to study this benchmark as an important class of satisfiable instances.

Impact of BMM Update. As described in Section 4, we also update the posterior with the new evidence (conflict clauses that are implied by the formula) that the solver generates. This update, had a positive impact on the performance, although not a significant impact. On average the solving times are reduced by 11.2% in application benchmark, but no additional instances were solved. The results in the tables and figures are with using BMM update.

Sub-category Analysis for SAT 2018 Application Instances. We analyzed categories of problems in the SAT competition benchmark (Heule et al., 2019), to further study which types of problems, BMM is more effective, and in which types it is less effective. The categories that we extracted from this benchmark were: Combinatorics, Cryptography, Graph theory, Verification, Number theory, Scheduling and Hard 3-SAT. In most categories, the BMM-based version of MapleCOMSPS performs on par with the default version. However, it solves one more instance in the verification category and one less instance in hard 3-SAT and scheduling problems, and a large leap of 16 more instances in the cryptography category.

Computational Overhead. In each epoch, all clauses are processed and for each clause, all of the literals in the clause are linearly processed, which means that the overall complexity is linearly proportional to the total number of literals appearing in the formula. 10 epochs over the largest formula in our benchmarks with 12 million clauses and 2 million variables, takes 80 seconds. On average BMM pre-processing constitutes 6% of the total running time of MapleCOMSPS on the SAT 2018 benchmark. This number is negligible in hard cryptographic instances even with 100 epochs.

7. Related Work

Unfortunately the initialization problem has not been studied as extensively as other components of the SAT solvers. Jeroslow-Wang (Jeroslow & Wang, 1990) proposed a scoring for each literal based on the length of the clauses that the literal appears in, where the literals that are appearing in shorter clauses are preferred. Initially this was proposed as a static branching heuristic, but this was later also used as a way of giving an initial preference to the literals. However, as the Boolean formula gets larger and more complicated, it might not capture the information about the underlying structure. In contrast, BMM updates the prior hypothesis with the target of satisfying all of the clauses and does not use a proxy for guessing a good measure. Despite being an approximation, BMM takes us to a relatively useful starting point. Most of the modern solvers, set the polarities and activities either to a fixed value (all zero, all False or all True), or a random value and let the search engine explore the search space. Some solvers in their initial phase of exploring, use a different branching heuristic (e.g. Distance (Xiao et al., 2019)) to get to a fruitful state and then use the main branching heuristics. However, all such solvers that use hybrid branching methods, only get to that desired state by collecting conflict clauses and do not re-use the intermediate activity scores. Kibria & Li (2006) proposed a genetic programming approach to find initialization of activities that minimizes runtime, where they had mixed results on a small set of electronic design automation instances.

8. Conclusion

We present the design of a novel BMM-based algorithm for the initialization problem of value selection (polarity) and variable order (branching) heuristics in conflict-driven clause-learning SAT solvers. We implemented our methods alongside other initialization methods (random, survey propagation, Jeroslow-Wang and default) in state-of-art solvers such as MapleCOMSPS, MapleLCMDistChronoBT, MapleSAT, Glucose and CryptoMiniSAT, and showed significant improvement over these already leading solvers. We evaluated our methods on the main track benchmark of SAT competition 2018, consisting of real-world application instances, as well as a set of hard cryptographic instances (inversion attacks) obtained from a round-reduced version of SHA-1 hash function. The BMM-enhanced version of MapleCOMSPS with both value selection and value order initialized, solves 12 more instances with lower average runtime compared to the baseline version, and is also faster than the random, survey propagation and Jeroslow-Wang initializations. Furthermore, the BMM-enhanced version of MapleSAT solves all of the hard cryptographic instances encoding preimage attacks on SHA-1 in our benchmark, and BMM-based CryptoMiniSAT solves them around 50% faster on average than the default version. We also gave a consistency proof of the Bayesian moment matching in the naïve Bayes model, which to the best of our knowledge is the first theoretical analysis of BMM.

Acknowledgments

Resources used in preparing this research were provided by Compute Canada, NSERC, the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute[4].

[4]www.vectorinstitute.ai/#partners

References

Audemard, G. and Simon, L. Glucose and syrup: Nine years in the sat competitions. *Proc. of SAT Competition*, pp. 24–25, 2018.

Avgerinos, T., Cha, S. K., Hao, B. L. T., and Brumley, D. Aeg: Automatic exploit generation. In *Network and Distributed System Security Symposium*, February 2011.

Bradley, A. R. Sat-based model checking without unrolling. In *International Workshop on Verification, Model Checking, and Abstract Interpretation*, pp. 70–87. Springer, 2011.

Braunstein, A., Mézard, M., and Zecchina, R. Survey propagation: An algorithm for satisfiability. *Random Structures & Algorithms*, 27(2):201–226, 2005.

Cadar, C., Ganesh, V., Pawlowski, P. M., Dill, D. L., and Engler, D. R. EXE: Automatically Generating Inputs of Death. *ACM Transactions on Information and System Security (TISSEC)*, 12(2):10, 2008.

Chen, Y. and Ryzhov, I. O. Technical note–consistency analysis of sequential learning under approximate bayesian inference. *Operations Research*, 68(1):295–307, 2020.

Heule, M. J., Järvisalo, M., and Suda, M. Sat competition 2018. *Journal on Satisfiability, Boolean Modeling and Computation*, 11(1):133–154, 2019.

Hsu, W.-S. and Poupart, P. Online bayesian moment matching for topic modeling with unknown number of topics. In *Advances in Neural Information Processing Systems*, pp. 4536–4544. Curran Associates, Inc., 2016.

Jaini, P. and Poupart, P. Online and distributed learning of gaussian mixture models by bayesian moment matching. *arXiv preprint arXiv:1609.05881*, 2016.

Jaini, P., Chen, Z., Carbajal, P., Law, E., Middleton, L., Regan, K., Schaekermann, M., Trimponias, G., Tung, J., and Poupart, P. Online bayesian moment matching for topic modeling with unknown number of topics. In *International Conference on Learning Representations*, 2016.

Jeroslow, R. G. and Wang, J. Solving propositional satisfiability problems. *Annals of mathematics and Artificial Intelligence*, 1(1-4):167–187, 1990.

Johnson, N., Kotz, S., and Balakrishnan, N. *Continuous univariate distributions*, volume 2. Wiley & Sons, 2nd edition, 1995.

Kibria, R. H. and Li, Y. Optimizing the initialization of dynamic decision heuristics in dpll sat solvers using genetic programming. In *European Conference on Genetic Programming*, pp. 331–340. Springer, 2006.

Liang, J. H., Ganesh, V., Zulkoski, E., Zaman, A., and Czarnecki, K. Understanding vsids branching heuristics in conflict-driven clause-learning sat solvers. In *Haifa Verification Conference*, pp. 225–241. Springer, 2015.

Liang, J. H., Ganesh, V., Poupart, P., and Czarnecki, K. Learning rate based branching heuristic for sat solvers. In *International Conference on Theory and Applications of Satisfiability Testing*, pp. 123–140. Springer International Publishing, 2016.

Liang, J. H., Oh, C., Ganesh, V., Czarnecki, K., and Poupart, P. Maple-comsps lrb vsids and maplecomsps chb vsids. *Proc. of SAT Competition*, pp. 20–21, 2017a.

Liang, J. H., VK, H. G., Poupart, P., Czarnecki, K., and Ganesh, V. An empirical study of branching heuristics through the lens of global learning rate. In *International Conference on Theory and Applications of Satisfiability Testing*, pp. 119–135. Springer, 2017b.

Liang, J. H., Oh, C., Mathew, M., Thomas, C., Li, C., and Ganesh, V. Machine learning-based restart policy for cdcl sat solvers. In *International Conference on Theory and Applications of Satisfiability Testing*, pp. 94–110. Springer, 2018.

MacKay, D. J. C. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, 2002.

Marques-Silva, J. P. and Sakallah, K. A. GRASP: A Search Algorithm for Propositional Satisfiability. *Computers, IEEE Transactions on*, 48(5):506–521, 1999.

Moskewicz, M. W., Madigan, C. F., Zhao, Y., Zhang, L., and Malik, S. Chaff: Engineering an Efficient SAT Solver. In *Proceedings of the 38th annual Design Automation Conference*, pp. 530–535. ACM, 2001.

Nejati, S., Liang, J. H., Gebotys, C., Czarnecki, K., and Ganesh, V. Adaptive restart and cegar-based solver for inverting cryptographic hash functions. In *Working Conference on Verified Software: Theories, Tools, and Experiments*, pp. 120–131. Springer, 2017a.

Nejati, S., Newsham, Z., Scott, J., Liang, J. H., Gebotys, C., Poupart, P., and Ganesh, V. A propagation rate based splitting heuristic for divide-and-conquer solvers. In *International Conference on Theory and Applications of Satisfiability Testing*, pp. 251–260. Springer, 2017b.

Nossum, V. Sat-based preimage attacks on sha-1. Master's thesis, 2012.

Omar, F. *Online Bayesian Learning in Probabilistic Graphical Models using Moment Matching with Applications*. PhD thesis, University of Waterloo, 2016.

Rashwan, A., Zhao, H., and Poupart, P. Online and distributed bayesian moment matching for parameter learning in sum-product networks. In *Artificial Intelligence and Statistics*, pp. 1469–1477, 2016.

Rintanen, J. Planning and SAT. *Handbook of Satisfiability*, 185:483–504, 2009.

Ryvchin, V. and Nadel, A. Maple_lcm_dist_chronobt: Featuring chronological backtracking. *Proc. of SAT Competition*, pp. 29–29, 2018.

Soos, M. The cryptominisat 5.5 set of solvers at the sat competition 2018. *Proc. of SAT Competition*, pp. 17–18, 2018.

Stump, A., Sutcliffe, G., and Tinelli, C. Starexec: A cross-community infrastructure for logic solving. In *International joint conference on automated reasoning*, pp. 367–373. Springer, 2014.

Xiao, F., Li, C.-M., Luo, M., Manyà, F., Lü, Z., and Li, Y. A branching heuristic for sat solvers based on complete implication graphs. *Science China Information Sciences*, 62(7):72103, 2019.