

Guided Model Checking with a Bayesian Meta-heuristic

Kevin Seppi, Michael Jones* and Peter Lamborn

Department of Computer Science, 3328 TMCB

Brigham Young University, Provo, UT 84602, USA

kseppi@cs.byu.edu, jones@cs.byu.edu, pcl@email.byu.edu

Abstract. This paper presents a formal verification algorithm for finding errors in models of complex concurrent systems. The algorithm improves explicit guided model checking by applying the empirical Bayes method to revise heuristic estimates of the distance from a given state to an error state. Guided search using the revised estimates finds errors with less search effort than the original estimates.

1. Introduction

Concurrency increases the complexity of a protocol, which increases the probability that a protocol contains an error. Model checking is a verification technique that can establish either the presence and absence of errors. When used in conjunction with other verification techniques, model checking is most

*Corresponding author

useful for finding errors that occur infrequently after long sequences of unique state transitions. These kinds of errors often arise in concurrent protocols. As a result, model checking, and particularly explicit state enumeration model checking, has become a viable tool for protocol design and verification.

The work presented in this paper increases the error-finding capacity of enumeration model checkers by combining a well-known statistical method with guided model checking. In guided explicit state enumeration model checking, a heuristic guides the search toward regions of the transition graph more likely to contain errors. Given a state, the heuristic function estimates the cost of reaching an error by estimating the number of transitions needed to reach an error. This value is used to determine the order in which states will be expanded. Guided model checkers can find errors more quickly in incorrect designs and can achieve more useful partial coverage in resource-bound problems involving large designs.

Generally, heuristics are treated as point estimates (i.e., a single numeric value). This paper takes a slightly different interpretation of heuristic values. We treat heuristic values as random variables (i.e., functions that assign a real valued probability between 0 and 1 to each possible outcome of an event) [7, 14]. A probability density function (pdf) is used to characterize the distribution of inaccuracy in the heuristic. If the heuristic is an accurate estimate, then most of the probability will be close to the actual distance to the target. Interpreting the heuristic as a random variable allows us to assess and improve the quality of the heuristic using statistical methods. For the purpose of this paper, we use mean squared error to measure the quality of a heuristic.

Our Bayes heuristic search algorithm (“BHS’”) minimizes mean squared error using an Empirical Bayes [2, 11] meta-heuristic. In this paper, a meta-heuristic is a function which takes heuristic values as input and returns improved heuristic values (with reduced mean squared error). Our empirical Bayes meta-heuristic uses estimates from a set of sibling states to derive the confidence that should be attributed to each individual estimate. The confidence level is then used to proportionally revise the original estimate toward the mean of the sibling estimates. Estimates with low confidence are revised more severely than estimates with high confidence. We validate this approach theoretically using a Bayesian model and show that the resulting heuristic values have smaller total expected mean squared error.

We give experimental results in which the BHS algorithm finds errors in fewer states than conven-

tional heuristic search. Conventional heuristic search is a best-first search using an inadmissible (i.e., may overestimate the cost of reaching a target) property-dependent heuristic.

We also compare the performance of BHS and conventional heuristic search on 100 variants of a problem. This experiment measures the robustness of BHS on a family of related problems. The BHS algorithm found errors 18% faster. These results indicate that, for this problem, the performance improvement obtained by BHS is insensitive to the particular configuration of the search problem.

In the next section, we describe various heuristics that have been used in guided model checking and give a variation of a conventional heuristic which we will use as input to the BHS meta-heuristic. While we phrase our claims in terms of the conventional heuristic function presented in Section 2, we emphasize that BHS can be applied to a wide variety of heuristics which assign costs to states. Section 2 also gives a statistical model for the conventional heuristic. This model justifies distributional assumptions that make BHS simple to implement and facilitate the proof that BHS reduces expected mean squared error.

Section 3 describes the meta-heuristic and the supporting notation. Section 4 introduces the Bayesian model used by the BHS algorithm, and proves that the resulting heuristic values have lower expected mean squared error. Section 5 contains experimental results. Section 6 gives some concluding remarks and directions for future work.

2. Heuristics

The performance of a guided model checker depends critically on the heuristic used to guide the search. In this section, we discuss related work in designing heuristics for guided model checking then give the heuristic used in the rest of the paper. This section concludes with an empirical study of the inaccuracy (variance) observed in our heuristic. An approximate assessment of the inaccuracy in our heuristic is needed for the Bayesian meta-heuristic introduced in the next section.

2.1. Related Heuristics for Guided Model Checking

Several kinds of heuristics have been proposed for use in guided model checking. Yang and Dill used a combination of Hamming distance, preimage computation and approximation and user annotations

(called guideposts) in guided search [19]. A combination of user annotations and error state preimage computation reduced the number of states explored in 8 of 8 guided search problems. Our BHS does not require user annotations and does not perform preimage computation, but can be used to revise state rankings generated from Hamming distance and user annotations.

Bloem et. al. classify heuristics for guided model checking as property-dependent or system-dependent [1]. Other kinds of heuristics, such as structural-dependent [6] and specification-dependent [16] heuristics have also been proposed. Property-dependent heuristics estimate the cost of reaching a violation of a property from a given state and system-dependent heuristics attempt to exploit properties of the design under test to avoid computational bottlenecks. System-dependent heuristics apply only to symbolic model checking algorithms. Specification-dependent heuristics are like structural-dependent heuristics except that specification-dependent heuristics treat the specification as a black box. Treating the specification as a black box forces the heuristic to rank search priorities based on input/output pairs rather than the structure of the underlying (and hidden) transition graph.

Using a combination of property and system dependent heuristics, Bloem et. al. obtained a reduction in time and space requirements in 6 of 8 reported search problems. Edelkamp et. al. have developed a property-dependent heuristic for use in SPIN, an explicit model checker [5]. This heuristic is the basis for the heuristic defined next in Section 2.2. Edelkamp's heuristic is intended to be admissible (i.e. never overestimates the cost of reaching an error) and relies on specific limitations of queue behavior in SPIN to generate better estimates. For example, SPIN queues do not allow reordering so that a message stored i places from the head of a queue can be received in no fewer than i transitions. Edelkamp et. al. used best-first search with this heuristic to explore fewer states than depth-first search in 14 of 20 problems.

Edelkamp et. al. have developed a property-dependent heuristic for use in SPIN, an explicit model checker [5]. This heuristic is the basis for the heuristic defined next in Section 2.2. Edelkamp's heuristic is intended to be admissible (i.e. never overestimates the cost of reaching an error) and relies on specific limitations of queue behavior in SPIN to generate better estimates. For example, SPIN queues do not allow reordering so that a message stored i places from the head of a queue can be received in no fewer than i transitions. Edelkamp et. al. used best-first search with this heuristic to explore fewer states than

depth-first search in 14 of 20 problems.

Although our primary interests in [1] and [5] are the heuristic functions, the primary contributions of [5] and [1] are extensions of guided model checking algorithms to linear temporal logic (LTL) and computational tree logic (CTL) properties, respectively. Our contribution is a new algorithm for computing a meta-heuristic that improves the performance of a heuristic in guided model checking. The meta-heuristic can be applied within the context of any guided model checking algorithm that ranks states using estimates of numerical values.

2.2. A Property-dependent Heuristic

The empirical results in this paper are obtained using a property-dependent heuristic inspired by Edelkamp [5]. Table 1 contains the heuristic in both inadmissible and admissible forms. The admissible heuristic is included for comparison. Given a state s and a boolean formula, the heuristic C estimates the number of transitions between s and a state that satisfies the formula. In the table, r is a variable quantified over finite domain R for some predicate p , a and b are boolean expressions, x and y are arithmetic expressions and v is a boolean variable. Negations are restricted to boolean variables and constants.

The admissible heuristic always underestimates the distance between s and a state that satisfies the formula because the heuristic assumes that any predicate can be satisfied in one step. This is true of all transition systems because a transition can always move s into a new state s' which satisfies an arbitrary set of predicates—including the predicate of interest. The inadmissible heuristic relaxes this requirement, and loses admissibility, by making assumptions about the number of transitions needed to satisfy a predicate based on the values in the predicate. These assumptions are reasonable for some transition systems, but not others. A more detailed discussion of the admissible heuristic in Table 1 can be found in [5].

Note that the admissible heuristic presented here differs from the Edelkamp heuristic because it accounts for the possibility of satisfying more than one operand of an operator with a single-step modification of a single variable. This can occur when an expression of the form $a \wedge b$ can be satisfied in a single step. For example, the trivial expression $v_1 \wedge v_1$ is satisfied by setting v_1 to “true” in a single step.

Boolean Formula	Inadmissable	Admissible
$C(s, \exists r : R.p(r))$	if($\exists r : R.p(r)$) then 0 else $avg_{\forall q \in R}(C(s, p(q)))$	if($\exists r : R.p(r)$) then 0 else $min_{\forall q \in R}(C(s, p(q)))$
$C(s, \forall r : R.p(r))$	$\Sigma_{\forall q \in R}(C(s, p(q)))$	$max_{\forall q \in R}(C(s, p(q)))$
$C(s, a \vee b)$	if($a \vee b$) then 0 else $avg(C(s, a), C(s, b))$	if($a \vee b$) then 0 else $min(C(s, a), C(s, b))$
$C(s, a \wedge b)$	$C(s, a) + C(s, b)$	if($a \wedge b$) then 0 else $max(C(s, a), C(s, b))$
$C(s, x \otimes y)$ where $\otimes \in (<, >)$	if($x \otimes y$) then 0 else $ x - y + 1$	if($x \otimes y$) then 0 else 1
$C(s, x \otimes y)$ where $\otimes \in (\geq, \leq, =)$	if($x \otimes y$) then 0 else $ x - y $	if($x \otimes y$) then 0 else 1
$C(s, \neg v)$	if (v) then 1 else 0	if (v) then 1 else 0
$C(s, v)$	if(v) then 0 else 1	if(v) then 0 else 1
$C(s, T)$	0	0
$C(s, F)$	∞	∞

Table 1. Inadmissible and admissible heuristics defined recursively on boolean formulas.

Such expressions could be re-written to factor out common variables or subexpressions, but for many models, including those used in our tests below, model builders rarely find and remove such redundancy. A similar problem occurs when the values of more than one variable are changed in a single atomic transition (e.g., multiple assignments within atomic transitions in Promela or rules in Mur ϕ).

2.3. Properties of the Heuristic

Both the admissible and inadmissible heuristics given in Table 1 *estimate* the cost of reaching a state that satisfies a given predicate. These estimates can be interpreted as random variables with pdf's that describe the distribution of inaccuracy in the estimates. For a given true remaining cost, the heuristic estimate varies over some range of values and according to some density. Functions describing the error density can be estimated by collecting heuristic and actual path costs over a large number of states.

The error density shown in Figure 1 was obtained from the true and estimated costs from an exhaustive search of an instance of the atomix model described later in section 5 (outliers removed). As is often the case, the range of heuristic values (1 to 6, with many ties) is much smaller than the range of actual costs (1 to 53, with fewer ties). In Figure 1 the actual costs are scaled to the range of the heuristic values, before computing the error.

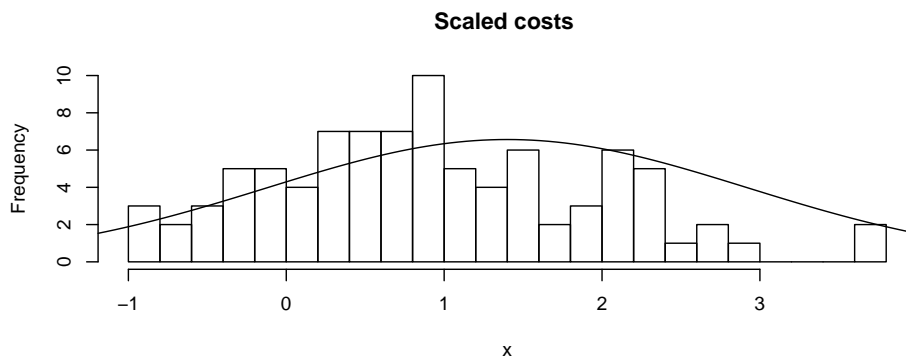


Figure 1. Observed distribution of error in the inadmissible heuristic and the normal distribution with the same mean and variance as the observed error.

The error density for our inadmissible heuristic follows a roughly normal distribution. The curve

overlaid in Figure 1 is the normal distribution with mean and variance taken from the points in the histogram. Thus we have characterized how well the heuristic estimates the remaining path length using a normal distribution with experimentally derived variance.

Note also that the true path costs observed in this study of the heuristic error are not used in the Meta-heuristic, only the variance of error density. We do *not* use information from this study to bias our algorithm toward a path we already know to be best. In this sense BHS is *not* a statistical learning algorithm.

3. Meta-heuristic

Bayes heuristic search differs from conventional heuristic search procedure in that it uses a Bayes meta-heuristic to improve estimates. Before establishing the mathematical properties of the Bayes meta-heuristic, we describe the rationale for the meta-heuristic with an example and present the algorithm for computing it.

The purpose of the meta-heuristic, and Empirical Bayes method in general, is to improve estimates using information from related sets of estimates.

Bayes heuristic search is designed of use in models where the true cost to a error sibling states are closer than the costs for a random set of states. The distribution of heuristic values for a set of sibling states suggests the confidence that should be given to those values. We assert that this is a reasonable working assumption. This assumption is illustrated in the following example.

Consider the states and heuristic values shown in Figure 2. States a and r each have five children each. The heuristic estimates for each child are mapped to the first number line. The placement of states on the number line corresponds to their order in the priority queue of states to be expanded. State a 's children are clustered between 6 and 9, while state r 's children are spread between 5 and 12. Although it is entirely possible that the heuristic values of the children of state a are just as believable as those for the children of r , we subjectively argue that siblings with tightly clustered estimates (e.g., states b through f) are more plausible than siblings with widely spread estimates (e.g., states s through w).

According to this reasoning, the estimates for children of state a are more believable than the es-

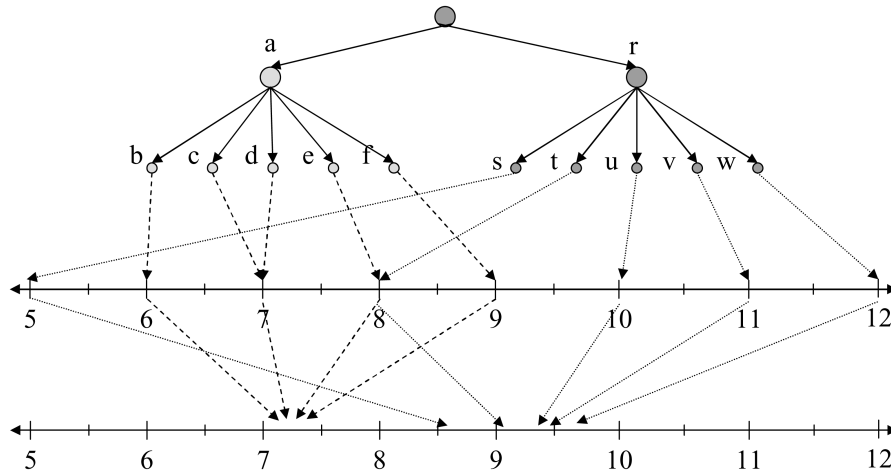


Figure 2. Original and revised heuristic estimates for two sets of sibling states.

estimates for children of state r . Some unusual property of either the heuristic or the children of state r must be causing the heuristic to give widely varying and inaccurate values for the children of state r . Note that we are not arguing that the children of a node are always tightly clustered, only that one would generally expect siblings to have *closer* heuristic values than non-siblings (two arbitrary nodes in the search tree). This assertion is always true for models in which the transition graph allowed all transitions to be reversed. In this case the heuristic estimate for two siblings should never differ by more than two: one transition back to the parent and one transition to a sibling.

A mathematical argument to justify this reasoning is presented in Section 4.

We give the states under a preference using the following algorithm, which can be integrated into any guided search that uses a heuristic to rank states:

- Compute heuristic values, y_i , for each child, i , of a given state. For example, in Figure 2, states b through f have heuristic values y_b through y_f .
- Compute the mean, θ_{sib} , and the variance, σ_{sib}^2 , of the heuristic values of the children.
- For each child i : create a revised estimate, θ_i^* , using the weighted average of the original value, y_i ,

and the mean of the siblings, θ_{sib} . For example, if $i = b$ then the revised estimate for state b is

$$\theta_b^* = (1 - B)y_b + B\theta_{sib} \quad (1)$$

where $B = \sigma_y^2 / \sigma_{sib}^2$ and σ_y^2 is the general variance of the heuristic which was discussed above in Section 2.3.

In Figure 2, the values on the second number line show new state ordering resulting from the revised estimates. The low variance of the children of state a implies that their original estimates are more plausible and require little modification. However, the estimates for state r 's children have a higher variance and require more modification.

The significance of this example is, that although state s has a lower estimate than state b , the higher variance and mean of state s 's siblings lead us to revise the estimate for state s to 8.6 while state b 's estimate is revised to 7.4. Hence, the more believable state b will be expanded before state s .

4. Bayesian Justification for the Meta-heuristic

In this section we map the meta-heuristic described in the preceding section to a Bayesian model and prove the resulting adjustment lowers the expected mean squared error. The details of this section may be skipped on the first reading.

The meta-heuristic presented in the prior section relies on Equation 1 to adjust heuristic values. Equation 1 is called the James-Stein estimator [18] and is used for estimating parameters in a group of parameters with normal distributions. This estimator can best be understood in the context of a Bayesian model.

4.1. Bayesian Notation

In order to introduce empirical Bayes and to motivate our meta-heuristic, we need some basic Bayesian notation. Table 2 summarizes the notation for and definitions of distributions used in this discussion.

Suppose we wish to make a decision, such as which state we should expand first in a transition graph, based on an unknown, random parameter θ (i.e., the true remaining path length to the goal,). We use a

Distribution	Description	Notation
Prior	Characterization of our understanding of some unknown random parameter θ	$g(\theta)$
Sampling	Distribution of observed values for a given θ	$f(y \theta)$
Posterior	New understanding of θ based on an observed y and the prior distribution	$\bar{g}(\theta y)$
Marginal	Density of probable sampling values, y , given the uncertainty in θ (from the Prior distribution) and the uncertainty inherent in the Sampling distribution	$\bar{f}(y)$

Table 2. Summary of distributions used in Bayes method.

prior distribution, or just *prior*, to characterize our belief about the most likely value of θ . For example, see “prior” in Figure 3. The notation $\theta \sim g(\theta)$ indicates that a pdf g models our understanding of θ .

Suppose that we are permitted to observe y , an imprecise estimate of θ . Also, suppose that the distribution of the random variable y is a function of θ . In other words, the distribution of y is more related to θ than a uniform random distribution. We call the distribution of y given θ the *sampling distribution* and describe it with the notation: $y|\theta \sim f(y|\theta)$ where f is the pdf of y given θ . The sampling distribution describes how far off the estimate y might be from θ as if we knew θ . For example see “sample” in Figure 3.

Bayes law allows us to create a *posterior distribution* which combines our prior understanding of θ (which is modeled by the pdf $g(\theta)$) with the new understanding of θ that can be *inferred* by observation of y . This yields a new distribution over θ , given the data we have observed:

$$\theta|y \sim \bar{g}(\theta|y) = \frac{f(y|\theta)g(\theta)}{\bar{f}(y)}, \quad (2)$$

where \bar{g} is the pdf of the new distribution on the random variable $\theta|y$. That is, our belief about the parameter θ given both our prior belief and the observed data y obtained from the heuristic. See the “posterior” in Figure 3.

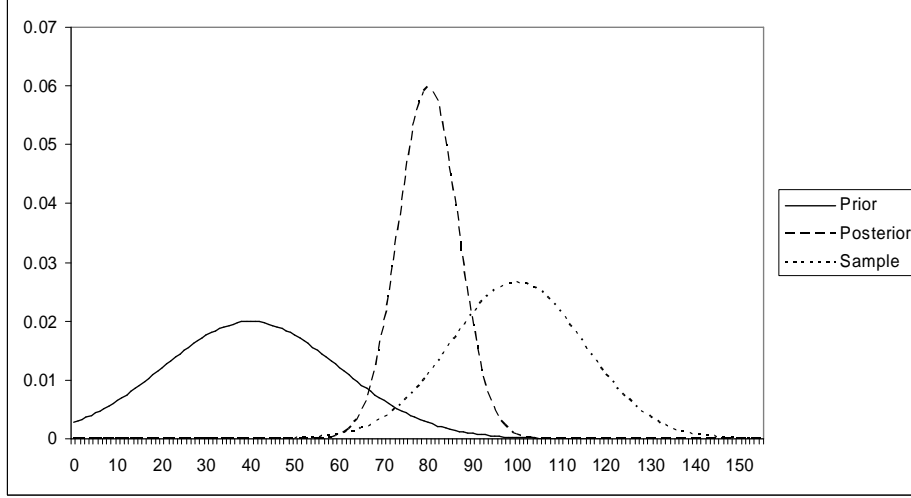


Figure 3. Prior, posterior and sample distributions in Bayes method.

The density \bar{f} used above, is the *marginal* or *unconditional distribution* of y and is defined as

$$y \sim \bar{f}(y) = \int_{\Theta} f(y|\theta)g(\theta)d\theta. \quad (3)$$

This distribution models the density of possible values of y given that there is uncertainty both in “ y given θ ” and in θ —upon which y depends.

In BHS we model these distributions with the normal distribution in which the prior distribution is

$$\theta \sim Normal(\theta_0, \sigma_0^2) \quad (4)$$

and the sampling distribution is:

$$y|\theta \sim Normal(\theta, \sigma_y^2). \quad (5)$$

In the case of BHS this is the quality of our heuristic, given the actual distance to a goal state. This is the pdf which we estimated empirically in Section 2.

Applying Bayes law to these two distributions yields the posterior distribution:

$$\theta|y \sim Normal((1 - B)y + B\theta_0, \sigma_y^2(1 - B)) \quad (6)$$

where $B = \frac{\sigma_y^2}{\sigma_y^2 + \sigma_0^2}$ and the marginal distribution is

$$y \sim Normal(\theta_0, \sigma_y^2 + \sigma_0^2). \quad (7)$$

See [3] for a more detailed development of these relationships.

4.2. Empirical Bayes

Thus far, the origin of the prior distribution has not been discussed. The generation of prior distribution when humans are available to render opinions is problematic, but is even more difficult in the context of software because computers do not have prior opinions. To solve this problem, we create a prior opinion for the computer in an algorithmic fashion. The approach is based on an Empirical Bayes philosophy. In Empirical Bayes, we focus on the interpretation and mathematical form of the marginal distribution referenced in Equations 3 and 7. As referenced above, this is the distribution one would observe if a sample of y 's were drawn unconditionally on θ , or expressed more algorithmically, first draw a θ from $g(\theta)$ then draw a y from $f(y|\theta)$. We will refer to this as the Empirical Bayes sample.

In the BHS meta-heuristic, sibling states are treated as the Empirical Bayes sample. Each child state has its own unknown true path length, θ_i , to the target but we observe only the y_i s, which are uncertain measures of these path lengths.

Given this set of sibling states with corresponding heuristic values, y_i , we estimate the mean (θ_{sib}) and variance (σ_{sib}^2) of the marginal distribution in Equation 7. From this equation we set θ_0 to θ_{sib} . Likewise, the observed sample variance σ_{sib}^2 is set equal to the variance equation $\sigma_y^2 + \sigma_0^2$ found in 7. Since σ_y^2 was derived experimentally in Section 2.3, we solve for the unknown $\sigma_0^2 = \sigma_{sib}^2 - \sigma_y^2$. Given a set of siblings, we have now empirically derived a shared prior distribution specified by σ_0^2 and θ_0 . Using these shared parameters and an individual heuristic value y_i , we can compute the parameters of the posterior as shown in Equation 6.

For the purposes of determining the next state to expand we retain only the mean, θ_i^* , of each of these posterior distributions which, from Equation 6, is

$$\theta_i^* = (1 - B)y_i + B\theta_0 \quad (8)$$

with $B = \sigma_y^2 / (\sigma_y^2 + \sigma_0^2)$ as before. Re-writing the equation for B using the fact that $\sigma_y^2 + \sigma_0^2 = \sigma_{sib}^2$ we obtain Equation 1, the equation used by the BHS meta-heuristic. Equation 8 is the James-Stein estimator which will have lower mean squared error than using the y_i values alone [18, 13, 12]. More formally, for

$n > 2$,

$$\sum_{i=1}^n E[(\theta_i^* - \theta_i)^2] \leq \sum_{i=1}^n E[(y_i - \theta_i)^2]. \quad (9)$$

We assumed in section 3 that siblings were more likely to have similar heuristic values than non-siblings. Interestingly 9 holds even if this assumption does not hold.

Readers familiar with Bayesian methods may prefer a pure Bayesian approach over the “Empirical Bayes” approach used here. Note, however, that we assume all information available to the creator of the heuristic would already be factored into the heuristic, including any information from prior knowledge (using Bayes law in the usual way). The BHS meta-heuristic would be applied *after* all such information has been incorporated into the heuristic. We use an empirical Bayes model to motivate the structure of our meta-heuristic, and properties of the James-Stein estimator that allow us to achieve heuristics with lower total expected mean squared error.

A more sophisticated Hierarchical Bayes model could also have been used as the basis for the meta-heuristic. The current Empirical Bayes version adds very little overhead to the search procedure since it requires only a few floating point computations per sibling. A sophisticated Hierarchical Bayes model would likely take many more operations and could significantly slow the computation of the meta-heuristic. Such an approach would almost certainly improve our assessment of the posterior variance, but since we discard the posterior variance anyway, it seems unlikely to be worth the computational effort. We plan to consider this issue more in future work.

5. Results ¹

We have implemented guided search using the inadmissible heuristic (referred to as just “inadmissible” henceforth) in Table 1 and the empirical Bayes meta-heuristic based on the inadmissible heuristic (referred to as BHS) in the Mur ϕ model checker [4]². This section contains the results of a series of ex-

¹To the reviewer: this paper is the full version of a paper of the same title that appeared in the proceedings of ACSD 2004. The results in this paper are significantly different, and more plausible, than the results in the ACSD publication. While preparing data for this manuscript, we discovered a pernicious coding error that caused the BHS algorithm to perform disproportionately well. We have fixed that error, thoroughly reviewed our entire implementation, and give the correct results here.

²The model checker and all models used in this paper are available at <http://vv.cs.byu.edu>

periments that compare the number of states explored before finding an error using breadth-first search (BFS), inadmissible and BHS.

A comparison with the admissible heuristic is not given because we were unable to define an admissible heuristic sufficiently precise to effectively guide the search. In [5], a comparison of best-first search with an inadmissible heuristic and A* search with an admissible heuristic does not clearly endorse either technique. Our admissible heuristic is less precise than the heuristic in [5] because our heuristic can not exploit properties of queues in SPIN models and must allow multiple variable assignments in an atomic step. Indeed, preliminary experiments with our admissible heuristic required exploring many times more states than best first search with our inadmissible heuristic.

Recall that Equation 1, upon which BHS depends, requires:

1. The mean heuristic value of the sibling nodes in the search (θ_{sib})
2. The variance of the heuristic values of the sibling nodes (σ_{sib}^2)
3. The heuristic value to be adjusted (y)
4. The variance of the heuristic in general (σ_y^2)

All of these are easily available in the context of the search except σ_y^2 . We treat σ_y^2 as a tunable parameter and discuss tuning σ_y^2 in more detail later.

We first give results for a synthetic problem before giving results for other verification problems. The synthetic problem is designed to approximate the assumptions used to justify the Bayesian meta-heuristic. Deploying the meta-heuristic on a synthetic problem will calibrate the performance improvement we can expect on other problems.

The synthetic model consists of n bounded up/down counters. In any state, the value of any counter can be incremented or decremented by $1..m$ as long as the new value is within the bounds. The model has an average branching factor of $2mn$. An “error” state is a state in which all counters have a specified value v . The heuristic determines the minimum distance to an error by summing the difference between

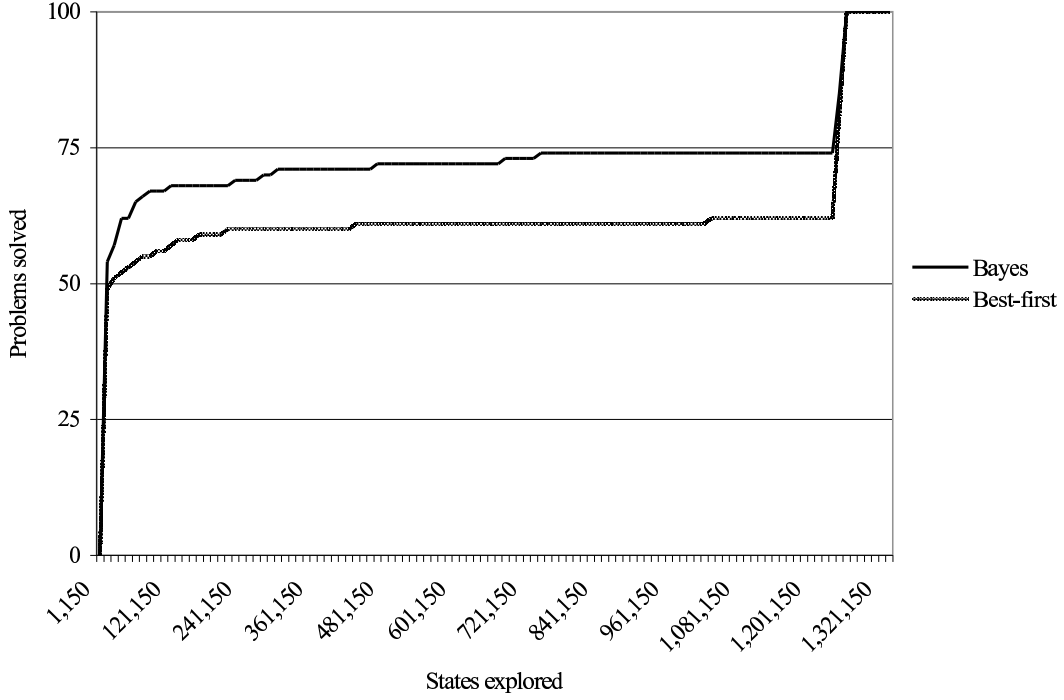


Figure 4. States explored before error discovery in one hundred trials of a synthetic model with uniformly distributed error.

each counter value, c_i , and v dividing by m :

$$H(s) = \sum_{i=1}^n ((c_i) - v)/m.$$

We add a uniformly distributed random variable with known variance and mean 0 to the heuristic value.

The synthetic model satisfies two key assumptions:

- Large branching factor.
- Known variance for error in the heuristic, σ_y^2 .

We analyzed the meta-heuristic on a model with $n = 10$ and $m = 3$ and added a uniformly distributed random error with variance 6 (σ_y^2). We conducted two sets of experiments with the synthetic model. The first experiment measured the impact of the meta-heuristic. For these experiments we treat σ_y^2 as known and use the actual variance of 6. Figure 4 shows the cumulative number of “errors” found, or problems

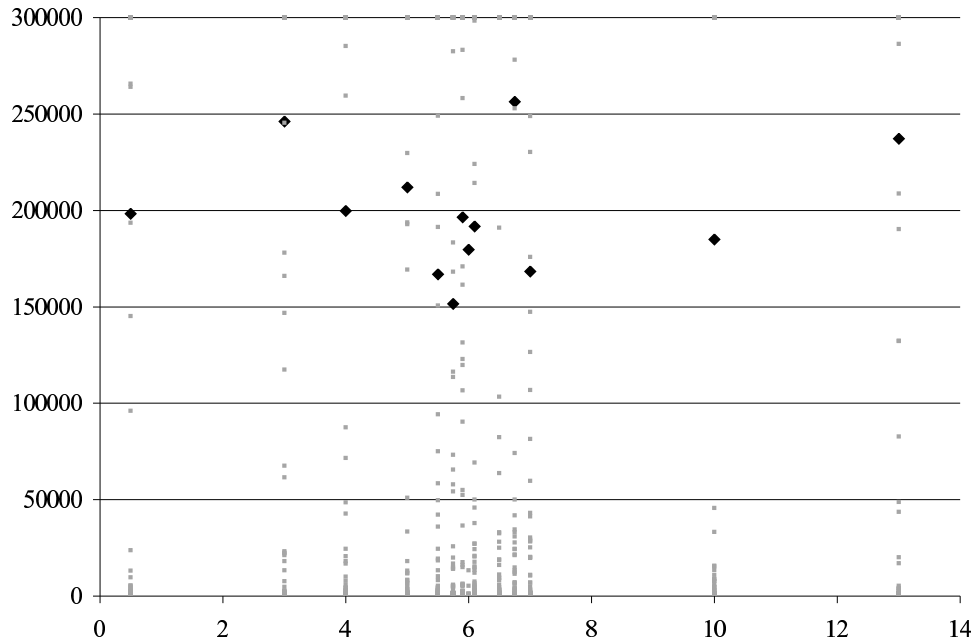


Figure 5. Actual and average ($n=100$) number of states explored before discovering an error in a synthetic model with σ_y^2 ranging over 2.0 to 11.0. The actual σ_y^2 is 6.0. Boxes represent individual experiments and diamonds represent the average for a given variance. Data points greater than 300,000 are truncated in the graph but included in the averages

solved, as a function of the number of states explored for 100 repetitions of the same problem. The search exhausts 2 GB of available memory and fails at 1.24 M states expanded. The Bayesian meta-heuristic solves 70 of 100 problems at 277,000 states expanded while the best-first search solves 60 problems with the same number of states expanded. The margin of improvement remains fairly constant until memory is exhausted.

The second experiment measured the effects of different error variance values (σ_y^2). The purpose of this experiment is to determine the sensitivity of the meta-heuristic to changes in the global variance estimate. The synthetic model was verified 100 times at each of 12 different global variances, σ_y^2 . The results of these verifications are shown in Figure 5. The success of BHS depends on a reasonably accurate approximation of σ_y^2 .

Note that the Bayesian model which supports BHS assumes normally distributed errors. In practice



Figure 6. States explored before error discovery in one hundred mutations of atomix.

BHS does not seem sensitive to this assumption. In simulation studies, using uniformly distributed error, the performance of BHS fell only 11% [17] when compared to simulated results using normal distributed error as specified in the Bayesian model. The results obtained using our synthetic $\text{Mur}\phi$ model also indicate that the error and actual cost distributions need not be strictly normal for the approach to work.

Figure 6 is obtained from a set of 200 experiments in which the inadmissible search and BHS algorithms locate error states, encoded as winning states, in 100 instances of atomix. This experiment measures how many states must be explored before error discovery in a given percentage of models using each search technique. Atomix is a single player game for which finding solutions is PSPACE-complete [9, 8]. In atomix, a player moves atoms and must use them to construct a target molecule. The playing field contains barriers as well as atoms. When an atom is moved in any direction it continues to move until it reaches the border of the board, a barrier or another atom. The player wins when the molecule is correctly assembled. The one hundred variations were created using a four by four board

containing one barrier (more barriers reduce reachable states and search times). The target molecule contained five atoms that must be arranged in the correct positions. The models were generated with random placement of both the atoms and barrier. There are 5,765,760 such models. Models that did not contain a winning state were excluded. For these experiments we let $\sigma_y^2 = 0.5$. The improvement depicted in Figure 6 is less than but is consistent with the improvement shown in Figure 4. We expect less improvement with the atomix game because we do not know σ_y^2 and heuristic error may not be uniformly (or normally) distributed.

We also experimented with other models and obtain the results shown in Figure 3. The importance of high branching factors for BHS is clear from these results. Models with a branching factor greater than 10, on average have better performance with BHS. The one exception is bulls& cows. The reason for this anomaly is the inadmissible heuristic turns out to be perfect for bulls& cows. BHS can never improve upon exploration order.

6. Conclusion

Interpreting heuristic estimates as random variables rather than point values allows the application of statistical methods to improve the expected behavior of guided search. More specifically, the BHS algorithm uses the empirical Bayes method to improve the accuracy of the heuristic estimates. In our experiments, the BHS algorithm achieved the best reduction for model checking problems with average branching factor greater than 10 when an accurate estimate of the variance of the error in the heuristic is known.

A recent study of the structural properties of state spaces found that toy examples (such as the atomix game used in this paper) tend to have greater branching factors than industrial examples (such as the td problem) [15]. Industrial examples have low branching factors because abstraction techniques are typically applied to industrial problems to reduce their branching factor in an effort to make the problem tractable. A particularly useful application of BHS is error discovery for industrial problems in which abstraction techniques do not effectively reduce the branching factor.

Avenues for future work include devising ways to efficiently estimate the error in a heuristic for a

Problem	BFS	Inadmiss- able path cost	BHS path cost	Inad- missable	BHS	Average Branching Factor
atomix1	1,573,909	502,342	269,753	8,806	15,755	11.34
atomix3	9,570,478*	100,545	54,141	20,051	24,415	10.73
atomix4	11,685,798*	780,415	778,596	303,211	247,686	10.43
atomixe	1,573,909	502,342	221,127	8,806	15,787	9.99
bulls&cows	7,354	96	1,393	96	96	36
dense	1,380,098	176	176	176	176	6.09
down	10,957	10,957	10,957	167	237	4.35
queens8	3,696,597	452,391	751,081	1,264,965	380,902	8
td	2,768,951	247	247	247	247	5

Table 3. States explored before error discovery in breadth first search, guided search using the inadmissible heuristic and BHS with and without path costs. An * indicates that the search exhausted 2 GB of memory, and failed, before locating an error.

given problem and applying the meta-heuristic to structural heuristics in addition to property heuristics. We expect similar results since the reduction obtained by the meta-heuristic does not depend on the structure of the heuristic. The meta-heuristic can be incorporated into our parallel algorithm for concentrating search effort on regions likely to contain errors in large transition graphs [10]. This parallel algorithm currently uses random walk to find regions likely to contain errors. While random walk is surprisingly effective in some cases, replacing random walk with guided search may improve the coverage in other cases.

References

- [1] Bloem, R., Ravi, K., Somenzi, F.: Symbolic guided search for CTL model checking, *Design Automation Conference (DAC'00)*, 2000.
- [2] Bradley P. Carlin, T. A. L.: *Bayes and Empirical Bayes Methods for Data Analysis*, Chapman & Hall, 1996.
- [3] DeGroot, M. H.: *Optimal Statistical Decisions*, McGraw-Hill, 1970.
- [4] Dill, D. L.: The Mur ϕ Verification System, *Computer-Aided Verification, CAV '96* (R. Alur, T. A. Henzinger, Eds.), 1102, Springer-Verlag, New Brunswick, NJ, July/August 1996.
- [5] Edelkamp, S., Lluch-Lafuente, A., Leue, S.: Directed Explicit Model Checking with HSF-SPIN, *8th International SPIN Workshop on Model Checking Software*, number 2057 in Lecture Notes in Computer Science, Springer-Verlag, 2001.
- [6] Groce, A., Visser, W.: Model Checking Java Programs using Structural Heuristics, *International Symposium on Software Testing and Analysis*, July 2002.
- [7] Hart, P., Nilsson, N., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths, *IEEE Transactions on Systems, Science and Cybernetics*, **SSC-4**(2), 1968, 100–107.
- [8] Holzer, M., Schwoon, S.: *Assembling molecules in atomix is hard*, Technical Report TUM-I0101, Institut für Informatik, Technische Universität München, June 2001.
- [9] Hueffner, F., Edelkamp, S., Fernau, H., Niedermeier, R.: Finding optimal solutions to Atomix, *Proceedings of the Joint German/Austrian Conference on Artificial Intelligence, 24th German / 9th Austrian Conference on Artificial Intelligence (KI 2001)*, number 2174 in Lecture Notes in Artificial Intelligence, Springer, 2001.

- [10] Jones, M. D., Sorber, J.: Parallel Search for LTL Violations, *Software tools for technology transfer*, 2004, To appear.
- [11] Maritz, J.: *Empirical Bayes Methods*, Methuen, 1970.
- [12] Morris, C. N.: Parametric empirical Bayes inference: theory and applications. With discussion, *Journal of the American Statistical Association*, **78**(381), 1983, 47–65.
- [13] Morris, C. N., Efron, B.: Empirical Bayes Estimators on Vector Observations - An Extension of Stein's Method, *Biometrika*, **59**, 1972, 335–347.
- [14] O. Hansson, A. M.: Probabilistic heuristic estimates, *Annals of Mathematics and Artificial Intelligence*, **2**, 1990, 209–220.
- [15] Pelanek, R.: Typical Structural Properties of State Spaces, *International SPIN Workshop on Software Model Checking (SPIN'04)*, number 2989 in LNCS, Springer, Barcelona, Spain, March 2004.
- [16] Pyhälä, T., Heljanko, K.: Specification Coverage Aided Test Selection, *Proceeding of the 3rd International Conference on Application of Concurrency to System Design (ACSD'2003)* (J. Lilius, F. Balarin, R. J. Machado, Eds.), IEEE Computer Society, June 2003.
- [17] Seppi, K.: *Empirical Bayes Assisted Probabilistic Search*, Technical Report VV-0301, Brigham Young University Department of Computer Science, 2003.
- [18] Stein, C.: Inadmissibility of the usual estimator for the mean of a multivariate normal distribution, *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability 1*, University of California Press, 1955.
- [19] Yang, C., Dill, D.: Validation with guided search fo the state space, *35th Design Automation Conference (DAC98)*, 1998.