Bounded-Cost Search Using Estimates of Uncertainty

Maximilian Fickert[*1] , **Tianyi Gu**[*2] , **Wheeler Ruml**[2]
[1]Saarland University, Saarland Informatics Campus, Germany
[2]University of New Hampshire, USA
fickert@cs.uni-saarland.de, {gu, ruml}@cs.unh.edu

Abstract

Many planning problems are too hard to solve optimally. In bounded-cost search, one attempts to find, as quickly as possible, a plan that costs no more than a user-provided absolute cost bound. Several algorithms have been previously proposed for this setting, including Potential Search (PTS) and Bounded-cost Explicit Estimation Search (BEES). BEES attempts to improve on PTS by predicting whether nodes will lead to plans within the cost bound or not. This paper introduces a relatively simple algorithm, Expected Effort Search (XES), which uses not just point estimates but belief distributions in order to estimate the probability that a node will lead to a plan within the bound. XES's expansion order minimizes expected search time in a simplified formal model. Experimental results on standard planning and search benchmarks show that it consistently exhibits strong performance, outperforming both PTS and BEES. We also derive improved variants of BEES that can exploit belief distributions. These new methods advance the recent trend of taking advantage of uncertainty estimates in deterministic single-agent search.

1 Introduction

In many real world applications, an optimal solution can be too hard to obtain within realistic constraints on CPU time and memory. In the field of suboptimal search, two main directions have been pursued. The first direction focuses on producing bounded suboptimal solutions that are within a given factor w of the optimal cost. The most famous algorithm in this group is weighted A^* [Pohl, 1970]. The second direction, called bounded-cost search [Stern *et al.*, 2011; Haslum, 2013], considers an alternative setting by providing an absolute cost bound C. The search algorithms are then designed to find a plan with cost less than or equal to C as quickly as possible. In this paper, we focus on the second direction and design a new bounded-cost search algorithm called Expected Effort Search (XES).

[*]These authors contributed equally to this work.

For bounded-cost search, one can take advantage of the problem setting and design heuristics that can guide the search to find solutions within the given cost bound as quickly as possible. For example, potential search (PTS) [Stern *et al.*, 2011] attempts to expand the node with the maximum probability of leading to a plan within the bound. However, PTS does not consider the search effort that would be required to find a goal under a search node. Bounded-cost Explicit Estimation Search (BEES) [Thayer *et al.*, 2012] attempts to improve upon PTS by incorporating distance-estimating heuristics. BEES attempts to expand the node, among those estimated to be within the bound, that is closest to a goal. However, BEES does not take the uncertainty of its estimate into account. A node with an expected cost estimate barely below the cost bound is considered equally important as a node with expected cost far less than the cost bound.

In this paper, we propose a new bounded-cost search algorithm called Expected Effort Search (XES), which takes both kinds of information into account. Like BEES, XES uses distance-to-go and inadmissible cost estimates, but it also uses belief distributions to model their uncertainty. XES estimates two quantities of a node n: 1) Estimated search effort $T(n)$, and 2) the probability $p(n)$ of finding a solution within the cost bound. It then performs best-first search on $T(n)/p(n)$. This idea was first proposed by Dobson and Haslum [2017] and implemented as an internal objective function for specific domain-independent planning heuristics. In this paper, we extend the idea to the general heuristic search setting, showing how such estimates can be obtained for arbitrary heuristics, and we prove its optimality in a simplified formal model. Our experimental results show that the XES approach consistently outperforms previous bounded-cost search methods. Furthermore, we devise new variants of BEES that make use of the belief distributions used in XES, which leads to improved performance on our benchmark set. These results further advance the trend of exploiting estimates of uncertainty, traditionally used in planning under uncertainty [Dearden *et al.*, 1998; McMahan *et al.*, 2005; Bellemare *et al.*, 2017], in methods for deterministic single-agent search.

2 Previous Work

A bounded-cost search problem can be defined as a six-tuple $\langle S, s_{init}, succ(s), c(s, s'), G, C\rangle$, where S is a set of states,

$s_{init} \in S$ is the initial state, $succ(s)$ is the transition function yielding the set of successor states of s, $c(s, s')$ is the cost function of the transition from state s to its successor $s' \in succ(s)$, $G \subseteq S$ is the set of goal states, and C is a given constant. The task is to find a path from s_{init} to a goal state (a *plan*) whose sum of transition costs is less than or equal to C. A node n represents a state in the search space with a corresponding path to it from the initial state; the cost of the path is denoted by $g(n)$, and $h(n)$ is a heuristic estimate of the path cost from the state represented by n to a goal (i.e, cost-to-go). Let $h^*(n)$ be the cost of an optimal plan. If $h(n) \leq h^*(n)$ holds on all states then h is called admissible. In that case, $f(n) = g(n) + h(n)$ is a lower bound on the minimum cost of any optimal plan to a goal with prefix n. Let $d(n)$ represent an estimate of the number of state transitions required to reach a goal (i.e., distance-to-go). The estimate d is often as easy to compute as h just by counting the number of actions in the solution to a relaxation of the original problem.

One simple strategy for bounded-cost search is to prune all the nodes with $g(n) > C$ (or $f(n) > C$ if h is admissible) during a (weighted) A* or greedy search. However, this may waste work as the heuristic that guides node selection is insensitive to the bound. This has motivated the development of several specialized algorithms for the bounded-cost setting.

2.1 Potential Search

Stern *et al.* [2011] define the potential of a node as the probability that the node will be part of a solution whose cost is within the user-provided cost bound C: $P^C(n) = Pr(h^*(n) \leq C - g(n))$. PTS is a best-first search on $P^C(n)$ and thus guides the search toward nodes which have high probability of finding a solution within the cost bound. However, instead of explicitly calculating $P^C(n)$, which is not trivial to do, PTS constructs a function $f_{lnr}(n) = \frac{h(n)}{C-g(n)}$. Stern *et al.* [2011] show that $f_{lnr}(n)$ yields the same ordering as $P^C(n)$ under the assumption that the error between the true cost-to-go $h^*(n)$ and its estimate $h(n)$ is linear in the size of $h(n)$. Thayer *et al.* [2012] show that PTS can be enhanced by an inadmissible heuristic $\hat{h}(n)$ when one is available. Let $\widehat{\text{PTS}}$ denote a best-first search on $\hat{f}_{lnr}(n) = \frac{\hat{h}(n)}{C-g(n)}$.

The limitation of potential-based approaches is that they only consider the probability that a node is on a path to a goal under the bound and yet never take into account search effort, i.e. the time needed to find that solution. Potential on its own does not include a term encouraging the search to reach a goal. Since the task of bounded-cost search is to find a plan within the bound as quickly as possible, search effort estimates ought to be useful as another source of information to guide the search.

2.2 Bounded-cost Explicit Estimation Search

Building on Thayer and Ruml's [2011] EES algorithm for bounded suboptimal search, Thayer *et al.* [2012] introduced Bounded-cost Explicit Estimation Search (BEES) and Bounded-cost Explicit Estimation Potential Search (BEEPS). BEES uses three sources of estimates to guide the search: an admissible cost heuristic h, an inadmissible cost heuristic $\hat{h}$, and an inadmissible distance-to-go heuristic $\hat{d}$. In addition to the standard open list, $open^C$, which contains all nodes with $f(n) = g(n) + h(n) \leq C$ sorted by f, BEES also maintains a focal list, $\widehat{open}^C$, which contains all the nodes whose inadmissible estimate is within the cost bound, i.e., $\hat{f} = g(n) + \hat{h}(n) \leq C$, sorted by $\hat{d}$. Nodes in $\widehat{open}^C$ are expanded first, falling back on $open^C$ only if $\widehat{open}^C$ is empty. BEEPS is a variation of BEES, differing only in ordering the standard open list, $open^C$, on $\hat{f}_{lnr}(n)$.

2.3 Combining Cost and Distance Estimates

Dobson and Haslum [2017] attempt to improve BEEPS by merging cost and distance heuristics into a single combined heuristic rather than alternating between them. They point out that the probability that the node will be part of a solution whose cost is within the user-provided cost bound C, $P^C(n)$, indicates that we might expect to expand a total of $\frac{1}{P^C(n)}$ many n-like nodes whose subtrees are similar to n, before a solution is found. Given this observation, they design an *expected work* heuristic $H^C(n) = T^C(n)/P^C(n)$, where $T^C(n)$ is the prediction of the size of the C-bounded subtree rooted at n. They describe how pattern database heuristics [Culberson and Schaeffer, 1996] can be adapted to compute the expected work by modifying the internal objective function for the abstract plans, and show that it performs well on selected planning domains.

Dobson and Haslum [2017] approximate $P^C(n)$ with the potential value used in PTS (i.e, $f_{lnr}(n)$) and estimate $T^C(n)$ with distance-to-go. They point out the potential value could be a poor approximation for the probability and they speculatively propose that one could perhaps approximate $P^C(n)$ via online learning of heuristic errors. In this paper, we further study this idea, show how it can be implemented for arbitrary underlying heuristics, and compare it to the existing state-of-the-art techniques for bounded-cost search.

2.4 Search Using Belief Distributions

Recently, it has been shown that a heuristic search can benefit from being guided by belief distribution of cost estimates [O'Ceallaigh and Ruml, 2015; Mitchell *et al.*, 2019; Fickert *et al.*, 2020, inter alia]. For example, Nancy [Mitchell *et al.*, 2019] is a real-time search algorithm that uses distributions of cost-to-go estimates to choose its next action. The distributions are derived from online observations of the one-step error of the heuristic and distance [Thayer *et al.*, 2011]: for a search node n, Nancy creates a Gaussian distribution $N(\hat{f}(n), (\frac{\hat{f}(n)-f(n)}{2})^2)$ that is centered around the debiased total cost estimate $\hat{f}(n)$ and estimates the standard deviation as half of the difference between $\hat{f}(n)$ and $f(n)$.

3 Expected Effort Search

Extending the line of work initiated by Dobson and Haslum [2017], we propose a new bounded-cost search algorithm called Expected Effort Search (XES). We design a combined expected effort heuristic that accounts for both the probability of finding a solution within the cost bound and the

effort required to do so. We first provide a simplified formal model to give a theoretical justification for this heuristic and then show how the probability estimate of finding a solution within the bound can be instantiated based on the belief distributions used by the Nancy algorithm [Mitchell *et al.*, 2019].

3.1 A Simple Formal Model

Let $p(n)$ be an estimate of a node n's potential, i.e., the probability that there is a solution under n within the cost bound. We will model expected search time using three assumptions. First (A1), we assume that the search procedure works exclusively in one subtree at a time. Let $T(n)$ be an estimate of the search effort to find a solution under n. Performing search below a node n can have two possible outcomes: either (a) a solution will be found under n with expected effort $T(n)$, or (b) the search does not find a solution under n. Second (A2), we assume that the subtrees are independent: not finding a solution in one subtree does not affect the probability of finding a solution in any other subtree. Third (A3), for case (b), we assume that the search abandons that subtree after having spent $T(n)$ time, the same as in case (a).

Now, let $\sigma = \langle n_0, n_1, \ldots, n_m \rangle$ be the ordering of the search nodes that are currently in the open list. We first have to consider the case where a solution is found below n_0, spending $T(n_0)$ time overall with a probability of $p(n_0)$. Otherwise, we need to next explore the search tree below n_1, where we would find a solution with overall probability $(1 - p(n_0))p(n_1)$ (considering the remaining probability of not finding a solution below n_0) and expected overall search time $(T(n_0) + T(n_1))$ since we explore both the search trees below n_0 and n_1, and so on. Abusing notation, let $T(n + m) = T(n) + T(m)$. Then, the expected overall search time for this ordering is

$$\begin{aligned} E(\sigma) =& p(n_0)T(n_0)+ \\ & (1 - p(n_0))p(n_1)(T(n_0 + n_1))+ \\ & (1 - p(n_0))(1 - p(n_1))p(n_2)(T(n_0 + n_1 + n_2))+ \\ & \ldots \end{aligned}$$

We note that there will be a final term in this expression representing the time in the case in which there is no solution; this quantity might be finite or infinite depending on the search tree and search procedure.

If we want to know whether we should prefer a node n over another node m, we can, without loss of generality, compare the open list orderings $\sigma_n = \langle n, m, \ldots \rangle$ and $\sigma_m = \langle m, n, \ldots \rangle$. Note that all but the first two terms of the corresponding expected search times are identical. We will use this to show that for ordering the open list, it is sufficient to use this shortened expression for expected search effort:

$$xe(n) = T(n)/p(n)$$

Theorem 1. *Given two open list orderings $\sigma_n = \langle n, m, \ldots \rangle$ and $\sigma_m = \langle m, n, \ldots \rangle$, $E(\sigma_n) < E(\sigma_m) \Leftrightarrow xe(n) < xe(m)$.*

Proof. We start with $E(\sigma_n) < E(\sigma_m)$. Removing equal terms from both sides yields

$$\begin{aligned} & p(n)T(n) + (1 - p(n))p(m)(T(n + m)) < \\ & p(m)T(m) + (1 - p(m))p(n)(T(m + n)). \end{aligned}$$

Multiplying out, we get

$$\begin{aligned} & p(n)T(n) + p(m)(T(n + m)) - p(n)p(m)(T(n + m)) < \\ & p(m)T(m) + p(n)(T(m + n)) - p(m)p(n)(T(m + n)). \end{aligned}$$

Simplifying by removing equal terms yields

$$\begin{aligned} & p(n)T(n) + p(m)(T(n + m)) < \\ & p(m)T(m) + p(n)(T(m + n)). \end{aligned}$$

Then by multiplying out again we get

$$\begin{aligned} & p(n)T(n) + p(m)T(n) + p(m)T(m) < \\ & p(m)T(m) + p(n)T(m) + p(n)T(n) \end{aligned}$$

and after removing equal terms again:

$$p(m)T(n) < p(n)T(m).$$

Finally, dividing by $p(n)$ and $p(m)$ yields

$$T(n)/p(n) < T(m)/p(m). \qquad \square$$

3.2 Estimating Expected Search Effort

XES is a best-first search on the expected search effort $xe(n) = T(n)/p(n)$. The key question is how to obtain the estimates $T(n)$ and $p(n)$. In this work, we instantiate $T(n)$ with the debiased distance estimate $\hat{d}(n) = \frac{d(n)}{1-\epsilon_d}$, where ϵ_d is the mean one-step error in d [Thayer *et al.*, 2011], and we derive $p(n)$ from Nancy-style probability distributions.

For a search node n, we use the same Gaussian distribution as Nancy, but truncated at a lower bound of $g(n)$ ($f(n)$ if h is admissible). The probability of finding a solution under n within the cost bound C is then just the area below the distribution up to C, as illustrated in Figure 1. This can be calculated directly using the cumulative density function for truncated Gaussian distributions [Hald, 1952].

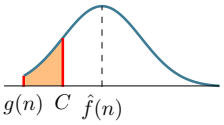

Figure 1: Estimating $p(n)$.

XES weighs the potential of a node against its estimated search effort. If the cost bound is very generous, then $p(n)$ should be close to 1 for most nodes and XES converges to a best-first search on $\hat{d}$, effectively guiding the search to a goal as quickly as possible. We prune nodes with $h(n) = \infty$ (dead ends) and nodes with $g(n) > C$, but not nodes with $T(n)/p(n) = \infty$ as this case can occur when the heuristic overestimates significantly (so $p(n)$ is very low and may round to zero). XES satisfies the usual completeness guarantee from a best-first search, as all nodes that may satisfy the cost bound are eventually expanded until a solution is found.

4 BEES with Explicit Probability Estimates

BEES and BEEPS predict whether a node will lead to a solution within the cost bound using the criterion $\hat{f}(n) \leq C$, in which case n is inserted into the prioritized focal list instead of the standard open list. The explicit probability estimates derived from the Nancy-style belief distributions allow us to introduce new variants of these algorithms, BEES95 and BEEPS95, where we instead check if the probability is sufficiently high: $p(n) \geq 0.95$.

5 Experimental Evaluation

Although XES would appear to be appealing when the simplifying assumptions A1–A3 hold, it remains to be seen how it performs in practice. We empirically evaluate XES and the new BEES variants on both planning and heuristic search benchmarks. We follow Thayer *et al.* [2012] and implement PTS with $p^C(n) = \frac{h(n)}{1-g(n)/C}$ as opposed to $\frac{h(n)}{C-g(n)}$, which not only makes it clearer that PTS and $\widehat{\text{PTS}}$ converge to GBFS for large cost bounds, but also avoids precision issues for large C values. To compute the debiased heuristic $\hat{h}(n)$ for $\widehat{\text{PTS}}$, BEES, and XES, we use $\hat{h}(n) = h(n) + \bar{\epsilon_h} \cdot \hat{d}(n)$ where $\bar{\epsilon_h}$ is the mean one-step error in h [Thayer *et al.*, 2011]. We initialize ϵ_h with 100 virtual samples to avoid a large fluctuation of $\hat{h}$ values at the beginning of the search. The initial value is set to make $\hat{h}$ optimistic, using an initial one-step error of zero on the search domains (where the heuristics are admissible), and -0.5 on the planning domains (where the heuristic is inadmissible).

5.1 Planning Domains

We use the benchmark set from the bounded-cost track of the 2018 International Planning Competition (IPC), which contains 180 instances from 9 domains, and the instances of the satisficing tracks of all previous IPCs, where we use the upper bounds from Planning.Domains [Muise, 2016] as the cost bounds, omitting instances where no bound is available, leaving 2147 instances from 48 domains. These bounds correspond to best found solutions for these instances and 1218 of them are known to be optimal.

We implemented the bounded-cost search algorithms in Fast Downward [Helmert, 2006].[1] The experiments were run using the lab framework [Seipp *et al.*, 2017] on a cluster of 2.2 GHz Intel Xeon E5-2660 CPUs. The time and memory limits were set to 30 minutes and 4 GB, respectively. We use the popular FF heuristic [Hoffmann and Nebel, 2001] as the heuristic h and use the length of the relaxed plans as the search distance estimator d. All algorithms use a dual queue for preferred operators.

IPC'18 Results

Table 1 compares XES, BEES95, and BEEPS95 to previous bounded-cost search algorithms. We also include GBFS (with pruning on g), which was used by most planners in the IPC.

XES performs best overall with a coverage of 66, followed by BEES95 (64) and the other algorithms of the BEES family (60). The potential-based algorithms PTS and $\widehat{\text{PTS}}$ have significantly lower coverage, and the statistics on the number of expansions show that they are not as effective in guiding the search to a goal. GBFS is not competitive with the specialized bounded-cost search algorithms overall, though it can sometimes beat them if it quickly finds a cost-effective path to the goal (e.g. in Agricola and Caldera-split). On five of the nine domains, XES has the highest coverage of the considered algorithms. Across the commonly solved instances, XES also needs the fewest expansions on average, and its search time is only beaten by GBFS (which solves significantly fewer instances, but those that it can solve are solved quickly).

[1] See https://github.com/fickert/fast-downward-xes.

Coverage	GBFS	PTS	$\widehat{\text{PTS}}$	BEES	BEEPS	BEES95	BEEPS95	XES
Agricola (20)	**1**	0	0	0	0	0	0	0
Caldera (20)	8	10	11	10	10	12	11	**13**
Caldera-split (20)	**4**	2	2	2	2	2	2	2
DataNetwork (20)	2	0	0	3	3	3	3	**4**
Nurikabe (20)	4	10	7	10	10	**11**	9	9
Settlers (20)	4	5	7	10	10	**11**	**11**	**11**
Snake (20)	4	**5**	**5**	4	**5**	4	**5**	**5**
Spider (20)	7	**11**	9	10	10	10	9	9
Termes (20)	11	9	6	11	10	11	10	**13**
Sum (180)	45	52	47	60	60	64	60	**66**
Exp. ($*10^3$)	1.93	3.93	6.75	2.10	2.62	2.25	2.24	**1.77**
Search time (s)	**1.69**	4.11	7.20	2.14	2.79	2.32	2.39	1.91

Table 1: Coverage on the IPC'18 bounded-cost instances, and geometric means of the expansions (multiply by 10^3) and search time on commonly solved instances (last two rows).

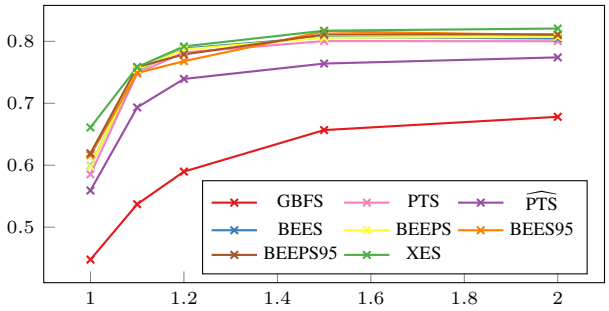

Figure 2: Normalized coverage as the cost bound is increased.

Satisficing Results

Table 2 shows the results on the instances of the satisficing tracks using the cost bounds from Planning.Domains. Consistent with the results on the IPC'18 bounded-cost instances, XES clearly outperforms the other considered algorithms: on 14 of the 48 domains it solves strictly more instances than any other algorithm, and on further 14 domains XES has the shared best coverage. Our new BEES variants BEES95 and BEEPS95 show small but relatively consistent improvements over their respective base algorithms. While PTS and $\widehat{\text{PTS}}$ show good performance in some individual domains (e.g., Floortile and Freecell), they again lag behind the other bounded-cost search algorithms overall.

Figure 2 shows the normalized coverage as the cost bounds are multiplied by increasing factors. The relative strength of the algorithms remains mostly consistent across the different bounds, with XES being the best performing algorithm for all considered values, and the specialized bounded-cost search algorithms still have a significant advantage over GBFS with pruning even as the cost bound is relaxed.

Coverage	GBFS	PTS	$\widehat{\text{PTS}}$	BEES	BEEPS	BEES95	BEEPS95	XES
Airport (49)	26	24	24	31	31	**32**	**32**	31
Assembly (30)	17	18	18	25	24	**30**	29	**30**
Barman (40)	5	0	0	6	6	9	9	**10**
Blocks (35)	19	26	26	30	26	30	26	**31**
Cavediving (8)	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**
Childsnack (6)	1	0	0	0	0	0	0	**3**
CityCar (13)	1	4	4	4	4	**5**	**5**	4
Depot (21)	6	13	13	**14**	13	12	13	13
DriverLog (20)	10	**19**	18	18	18	18	**19**	15
Elevators (39)	7	12	12	22	22	**24**	**24**	**24**
Floortile (27)	6	**20**	**20**	8	8	9	8	9
Freecell (80)	20	**65**	57	37	47	39	58	41
GED (20)	0	**1**	0	0	0	**1**	**1**	**1**
Grid (5)	1	3	3	3	3	3	3	**4**
Gripper (20)	7	8	8	7	8	7	8	**12**
Hiking (18)	8	**18**	16	15	17	14	17	16
Logistics (63)	33	46	38	51	50	52	**53**	52
Maintenance (17)	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Miconic (439)	342	362	427	403	426	397	426	**437**
Movie (30)	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**
Mprime (35)	29	**34**	33	33	33	**34**	33	33
Mystery (19)	17	**19**	**19**	**19**	**19**	**19**	**19**	**19**
Nomystery (18)	4	16	16	**18**	14	**18**	14	17
Openstacks (98)	33	31	41	53	54	56	56	**58**
Opt. Telegr. (4)	**4**	2	2	2	2	**4**	**4**	**4**
Parcprinter (40)	26	25	20	**29**	27	24	20	24
Parking (40)	4	**21**	10	11	12	16	13	15
Pathways (57)	13	20	**27**	22	22	25	22	24
Pegsol (35)	32	**35**	**35**	34	**35**	34	**35**	**35**
Philosophers (45)	**45**	11	5	11	11	11	11	14
Pipes-notank (46)	20	41	42	**43**	**43**	42	42	**43**
Pipes-tank (45)	15	29	28	28	32	26	31	**33**
PSR (116)	107	111	110	111	110	111	110	**113**
Rovers (40)	12	20	18	25	25	27	25	**30**
Satellite (36)	20	20	20	**23**	21	21	**23**	**23**
Scanalyzer (28)	20	16	17	20	18	**22**	17	21
Schedule (150)	36	51	47	62	73	60	75	**89**
Sokoban (30)	24	**29**	**29**	**29**	**29**	**29**	**29**	**29**
Storage (28)	14	19	20	20	20	16	18	**22**
Tetris (18)	**7**	4	3	4	4	2	2	3
Thoughtful (20)	6	10	10	8	10	9	9	**12**
Tidybot (17)	3	11	11	**12**	**12**	11	11	**12**
TPP (30)	10	16	13	20	19	15	18	**24**
Transport (53)	7	9	10	15	15	**19**	**19**	17
Trucks (31)	18	30	30	30	29	**31**	**31**	30
VisitAll (37)	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Woodw. (31)	18	**24**	11	19	19	21	19	22
Zenotravel (20)	8	14	13	13	13	**16**	**16**	14
Sum (2147)	1098	1344	1361	1425	1461	1438	1490	**1550**
Normalized (%)	44.8	58.5	55.9	59.9	59.8	61.6	61.9	**66.1**
Expansions	1961	1135	1787	401	491	**365**	485	369
Search time (s)	0.53	0.40	0.57	0.19	0.23	**0.18**	0.24	**0.18**

Table 2: Coverage on the IPC satisficing instances. The normalized overall coverage corrects for the different numbers of instances per domain. The last two rows show the geometric means of the expansions and search time on commonly solved instances.

5.2 Search Domains

We also empirically compare XES and BEES95 to PTS, $\widehat{\text{PTS}}$, and BEES on four classic heuristic search benchmarks. We do not include A*, GBFS, speedy search, or BEEPS because Thayer *et al.* [2012] found them to be dominated by BEES and $\widehat{\text{PTS}}$ (we confirmed this in preliminary experiments). Domain-specific solvers were implemented in C++[2] and run on 64-bit Linux systems with 3.16 GHz Intel E8500 Core2 duo processors and 8 GB of RAM. Algorithms were cut off at 7 GB RAM use. Beforehand, we solve each problem instance optimally, recording the cost. Then, in the tests, we set the cost bound of each instance to a factor of the optimal solution cost.

Sliding-Tile Puzzle We consider four variants of the well-known 15-puzzle: uniform cost, heavy cost (cost=tile#), inverse cost (cost=1/#), and square-root cost (cost=$\sqrt{\#}$). We use the Manhattan Distance as the heuristic h, weighting the components as appropriate [Thayer and Ruml, 2011], using the classic 100 start states published by Korf [1985].

Vacuum World The vacuum world domain was introduced by Thayer and Ruml [2011], following the first state space presented in Russell and Norvig's [1995] textbook. We use 60 solvable instances of size 200×200, each cell having a 35% probability of being blocked. We again consider two variants with uniform and heavy costs, using the minimum spanning tree heuristic for h on the uniform-cost version and an adaptation thereof for the heavy-cost variant [Thayer and Ruml, 2011]. The robot and the dirt piles are randomly placed in unblocked cells (10 piles for unit, 6 for heavy).

Pancake The objective of the pancake problem [Kleitman *et al.*, 1975; Gates and Papadimitriou, 1979; Heydari and Sudborough, 1997] is to sort a sequence of pancakes through a minimal number of prefix reversals. We use the GAP heuristic [Helmert, 2010] for all the algorithms. We also consider Heavy Pancake, where the action cost is the sum of the indices of all pancakes being flipped. We use 100 randomly generated instances.

Racetrack The Racetrack domain [Barto *et al.*, 1995] is similar to the grid pathfinding problem with inertia. We use the track map that was created by Hansen and Zilberstein [2001]. Each action modifies the acceleration of the agent by -1, 0, or 1 in both the horizontal and vertical directions, making for a total of 9 distinct actions. The admissible heuristic function is the maximum, either horizontally or vertically, of the distance to the goal divided by an estimate of the maximum achievable velocity in that dimension. Both a precomputed Dijkstra distance function and a weaker Euclidean distance function are used. We created 25 instances with starting positions chosen randomly among those cells that were at least 90% of the maximum distance from a goal.

Results

Figure 3 shows the most important measure of performance, CPU time. In the tile puzzle, we sort the variants by their difficulty from left to right, with the easiest uniform-cost tile on the left. As the problems get harder, XES BEES95 and

[2] See https://github.com/gtianyi/boundedCostSearch.

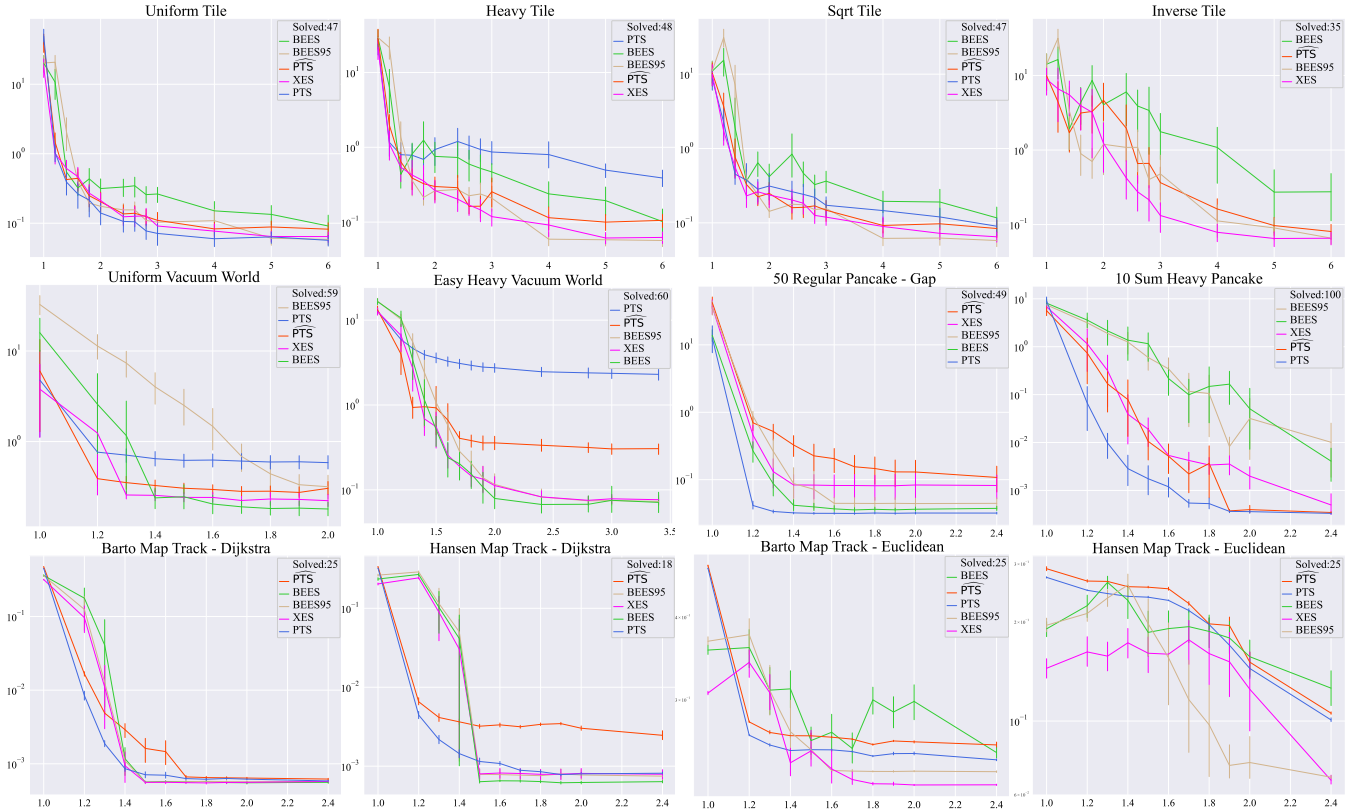

Figure 3: CPU time (in seconds) as a function of the cost bound (factor of optimal). Error bars show 95% confidence intervals on the mean across the commonly solved instances (number shown in legend). The legends are sorted by the geometric mean across all cost bounds.

$\widehat{\text{PTS}}$ start to outperform BEES and PTS. To have a reasonable number of commonly solved instances, we exclude PTS in the inverse cost plot due to its low coverage for tight bounds.

In the uniform vacuum problem, XES behaves competitive to BEES, PTS and $\widehat{\text{PTS}}$. BEES95 performs poorly, especially when the bounds are tight. In heavy vacuum, XES and BEES95 are very close to BEES and outperform $\widehat{\text{PTS}}$ and PTS by one and two orders of magnitude respectively.

For regular pancake, PTS performs well while $\widehat{\text{PTS}}$ performs poorly, with BEES, BEES95, and XES in between. This is the worst domain for XES, but its performance is not far from the best algorithm. For heavy pancake, XES performs similarly to $\widehat{\text{PTS}}$ and slightly worse than PTS, while BEES and BEES95 are much worse.

On both the Barto and Hansen maps of the Racetrack domain, most algorithms scale similarly when using the accurate Dijkstra heuristic, with PTS and $\widehat{\text{PTS}}$ being slightly better when the bound is very tight and $\widehat{\text{PTS}}$ being worse when the bound is loose. However, using the weaker Euclidean heuristic, XES and BEES95 perform better than other algorithms.

5.3 Experiment Summary

The results on benchmarks from both planning and heuristic search show that XES performs consistently well across almost all tested domains. It is often better than previous state-of-the-art bounded-cost search algorithms. The distributionally-enhanced BEES variants BEES95 and BEEPS95 improve over BEES and BEEPS in most domains. One advantage of XES over these BEES variants is that it is less reliant on the probability estimate being accurate: If the probability estimate consistently over- or underestimates, then the expected effort values are affected equally and the ordering of the open list does not change much, while the focal list is either over- or underutilized in the BEES algorithms. There are some domains where PTS and $\widehat{\text{PTS}}$ work well, but their performance is less robust to weaker heuristics.

6 Conclusions

XES optimizes search effort in a simple formal model and empirically appears more robust than other state-of-the-art bounded-cost algorithms in both classical planning and heuristic search benchmarks. This work advances the recent trend of taking advantage of distributional information in heuristic search, showing that, even in single-agent deterministic domains, representing an algorithm's uncertainty about its estimates of properties of the unexplored portions of the search space can be a valuable tool.

7 Acknowledgements

This work was partially funded by DFG grant 389792660 as part of TRR 248 – CPEC (see https://perspicuous-computing.science) and NSF-BSF via grant 2008594.

References

[Barto *et al.*, 1995] Andrew G Barto, Steven J Bradtke, and Satinder P Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72(1-2):81–138, 1995.

[Bellemare *et al.*, 2017] Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *Proceedings of ICML*, pages 449–458, 2017.

[Culberson and Schaeffer, 1996] Joseph C Culberson and Jonathan Schaeffer. Searching with pattern databases. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pages 402–416. Springer, 1996.

[Dearden *et al.*, 1998] Richard Dearden, Nir Friedman, and Stuart Russell. Bayesian Q-learning. In *Proceedings of AAAI*, pages 761–768, 1998.

[Dobson and Haslum, 2017] Sean Dobson and Patrik Haslum. Cost-length tradeoff heuristics for bounded-cost search. In *Proceedings of the ICAPS-17 HSDIP Workshop*, 2017.

[Fickert *et al.*, 2020] Maximilian Fickert, Tianyi Gu, Leonhard Staut, Wheeler Ruml, Jörg Hoffmann, and Marek Petrik. Beliefs we can believe in: Replacing assumptions with data in real-time search. In *Proceedings of AAAI*, pages 9827–9834, 2020.

[Gates and Papadimitriou, 1979] William H Gates and Christos H Papadimitriou. Bounds for sorting by prefix reversal. *Discrete mathematics*, 27(1):47–57, 1979.

[Hald, 1952] A. Hald. *Statistical Theory with Engineering Applications*. Probability and Statistics Series. Wiley, 1952.

[Hansen and Zilberstein, 2001] Eric A Hansen and Shlomo Zilberstein. LAO*: A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*, 129(1-2):35–62, 2001.

[Haslum, 2013] Patrik Haslum. Heuristics for bounded-cost search. In *Proceedings of ICAPS*, 2013.

[Helmert, 2006] Malte Helmert. The Fast Downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.

[Helmert, 2010] Malte Helmert. Landmark heuristics for the pancake problem. In *Proceedings of SoCS*, 2010.

[Heydari and Sudborough, 1997] Mohammad H Heydari and I Hal Sudborough. On the diameter of the pancake network. *Journal of Algorithms*, 25(1):67–94, 1997.

[Hoffmann and Nebel, 2001] Jörg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

[Kleitman *et al.*, 1975] DJ Kleitman, Edvard Kramer, JH Conway, Stroughton Bell, and Harry Dweighter. Elementary problems: E2564–E2569. *The American Mathematical Monthly*, 82(10):1009–1010, 1975.

[Korf, 1985] Richard E. Korf. Iterative-deepening-A*: An optimal admissible tree search. In *Proceedings of IJCAI-85*, pages 1034–1036, 1985.

[McMahan *et al.*, 2005] H Brendan McMahan, Maxim Likhachev, and Geoffrey J Gordon. Bounded real-time dynamic programming: RTDP with monotone upper bounds and performance guarantees. In *Proceedings of ICML*, pages 569–576, 2005.

[Mitchell *et al.*, 2019] Andrew Mitchell, Wheeler Ruml, Fabian Spaniol, Joerg Hoffmann, and Marek Petrik. Real-time planning as decision-making under uncertainty. In *Proceedings of AAAI-19*, 2019.

[Muise, 2016] Christian Muise. Planning.Domains. In *ICAPS Demonstrations*, 2016.

[O'Ceallaigh and Ruml, 2015] Dylan O'Ceallaigh and Wheeler Ruml. Metareasoning in real-time heuristic search. In *Proceedings of SoCS*, 2015.

[Pohl, 1970] Ira Pohl. Heuristic search viewed as path finding in a graph. *Artificial Intelligence*, 1(3-4):193–204, 1970.

[Russell and Norvig, 1995] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, first edition, 1995.

[Seipp *et al.*, 2017] Jendrik Seipp, Florian Pommerening, Silvan Sievers, and Malte Helmert. Downward Lab. https://doi.org/10.5281/zenodo.790461, 2017.

[Stern *et al.*, 2011] Roni Tzvi Stern, Rami Puzis, and Ariel Felner. Potential search: A bounded-cost search algorithm. In *Proceedings of ICAPS*, 2011.

[Thayer and Ruml, 2011] Jordan Thayer and Wheeler Ruml. Bounded suboptimal search: A direct approach using inadmissible estimates. In *Proceedings of IJCAI*, volume 2011, pages 674–679, 2011.

[Thayer *et al.*, 2011] Jordan T. Thayer, Austin Dionne, and Wheeler Ruml. Learning inadmissible heuristics during search. In *Proceedings of ICAPS*, 2011.

[Thayer *et al.*, 2012] Jordan Tyler Thayer, Roni Stern, Ariel Felner, and Wheeler Ruml. Faster bounded-cost search using inadmissible estimates. In *Proceedings of ICAPS*, 2012.