Bounded Suboptimal Search: A Direct Approach Using Inadmissible Estimates

Jordan T. Thayer and **Wheeler Ruml**
Department of Computer Science
University of New Hampshire
Durham, NH 03824 USA
`jtd7`, `ruml` at `cs.unh.edu`

Abstract

Bounded suboptimal search algorithms offer shorter solving times by sacrificing optimality and instead guaranteeing solution costs within a desired factor of optimal. Typically these algorithms use a single admissible heuristic both for guiding search and bounding solution cost. In this paper, we present a new approach to bounded suboptimal search, Explicit Estimation Search, that separates these roles, consulting potentially inadmissible information to determine search order and using admissible information to guarantee the cost bound. Unlike previous proposals, it successfully combines estimates of solution length and solution cost to predict which node will lead most quickly to a solution within the suboptimality bound. An empirical evaluation across six diverse benchmark domains shows that Explicit Estimation Search is competitive with the previous state of the art in domains with unit-cost actions and substantially outperforms previously proposed techniques for domains in which solution cost and length can differ.

1 Introduction

When resources are plentiful, algorithms like A* [Hart *et al.*, 1968] or IDA* [Korf, 1985] can be used to solve heuristic search problems optimally. However, in many practical settings, we must accept suboptimal solutions in order to reduce the time or memory required for search. When abandoning optimality, we can still retain some control over the solutions returned by bounding their cost. We say a search algorithm is w-admissible if it is guaranteed to find solutions whose cost is within a specified factor w of optimal.

Weighted A*, for example, modifies the standard node evaluation function of A*, $f(n) = g(n)+h(n)$, where $g(n)$ is the cost of getting to n from the root and $h(n)$ is the estimated cost-to-go from n to the goal, into $f'(n) = g(n) + w \cdot h(n)$. Placing additional emphasis on $h(n)$ is a common technique for reducing the number of expansions needed to find solutions. This encourages the search algorithm to prefer states where there is little estimated cost remaining to the goal, as they tend to be closer to the goal.

In domains where different actions have different costs, this tendency is less pronounced. Cost estimates do not necessarily tell us anything about the potential length of solutions, as we could incur the same cost by taking many cheap actions or a single expensive one. A search algorithm can use an additional heuristic to determine which nodes are closer to goals. Several authors [Pearl and Kim, 1982; Ghallab and Allard, 1983; Pohl, 1973; Thayer and Ruml, 2009] have proposed techniques that rely on such estimates and have shown how to compute them.

A^*_ϵ [Pearl and Kim, 1982] is a search algorithm that relies on such estimates of solution length, but it has not seen wide adoption because it often performs poorly [Thayer *et al.*, 2009]. As we will explain in more detail later, this poor performance is a result of using lower bounds both to guide the search and to enforce suboptimality bounds. While lower bounds are needed to prove bounded suboptimality, admissibility is not required to guide the search towards solutions. Most previous proposals ignore this potential division of labor, but as we will see later in this paper, it can be the difference between solving problems and not.

We begin by showing that most previously proposed algorithms do not directly attempt to optimize solving time. We then propose a new search algorithm, Explicit Estimation Search, that directly attempts to find solutions within the desired suboptimality bound as quickly as possible. It does so by using unbiased estimates of solution cost as well as solution length estimates to guide search. Lower bounds are only used for ensuring that every expanded node is within the suboptimality bound. In a study across six diverse benchmark domains we demonstrate that this difference allows Explicit Estimation Search to perform much better than the current state of the art in domains where action costs can differ.

2 Previous Work

The objectives of bounded suboptimal search are, first, to find solutions within the user specified cost-bound, and second, to find them as quickly as possible. While it is not clear how to make search algorithms directly prefer those solutions that are fastest to find, we can make them prefer solutions that are estimated to be shorter. Intuitively, shorter solutions are likely to be easier to find. In order for heuristic search to find a solution, it must expand at least the path from the starting node to a goal node that represents that solution. Our fundamental assumption is that, all else being equal, shorter solutions will require fewer expansions to construct. While previous approaches to bounded suboptimal search satisfy the first objective by definition, most do not directly address the second:

weighted A* [Pohl, 1973] returns solutions of bounded quality, but it cannot prefer the shorter of two w-admissible solutions because it has no way of taking advantage of estimates of solution length when determining search order. Revised dynamically weighted A* [Thayer and Ruml, 2009] is an extension of weighted A* that dynamically adjusts the weight used based on a nodes estimated distance to goal. While this modification gives preferential treatment to nodes which appear to be close to a goal, it will not always attempt to pursue the shortest solution within the desired bound.

Optimistic Search [Thayer and Ruml, 2008] takes advantage of the fact that weighted A* tends to return solutions much better than the bound suggests. It does so by running weighted A* with a bound w_{opt} that is twice as suboptimal as the desired bound, $w_{opt} = (w - 1) \cdot 2 + 1$, and then expanding some additional nodes after a solution is found to ensure that the desired bound was met. Specifically, it repeatedly expands $best_f$, the node with the smallest $f(n) = g(n) + h(n)$ of all nodes on the open list, until $w \cdot f(best_f) \geq g(sol)$, where sol is the solution returned by the weighted A* search. Because $f(best_f)$ is a lower bound on the cost of an optimal solution, this proves that the incumbent is within the desired suboptimality bound. Because the underlying search order ignores the length of solutions, so does optimistic search.

Skeptical Search [Thayer and Ruml, 2011] replaces the aggressive weighting of optimistic search with a weighted search on an inadmissible heuristic, as in $\widehat{f}'(n) = g(n) + w \cdot \widehat{h}(n)$, where $\widehat{h}$ is some potentially inadmissible estimate of the cost-to-go from n to a goal. Then, as with optimistic search, $best_f$ is repeatedly expanded to prove that the incumbent solution is within the desired bound. Because skeptical search is only relying on cost-to-go estimates for guidance, it cannot prefer shorter solutions within the bound.

A^*_ϵ **[Pearl and Kim, 1982]** uses a distance-to-go estimate to determine search order. A^*_ϵ sorts its $open$ list on $f(n)$. It maintains the subset of all open nodes that it believes can lead to w-admissible solutions, called the *focal* list. These are the nodes where $f(n) \leq w \cdot f(best_f)$. From focal, it selects the node that appears to be as close to a goal as possible, the node with minimum $\widehat{d}(n)$, for expansion. $\widehat{d}$ is a potentially inadmissible estimate of the number of actions along a minimum-cost path from n to a goal. While this algorithm does follow the objectives of suboptimal search, it performs poorly in practice [Thayer *et al.*, 2009]. This is a result of using $f(best_f)$ to determine which nodes form *focal*. When we use a lower bound to determine membership in the focal list, it is often the case that the children of an expanded node will not qualify for inclusion in focal because $f(n)$ tends to rise along any path. As a result of f's tendency to rise, $best_f$ tends to have low depth and high $\widehat{d}$. This results in an unfortunate and inefficient thrashing behavior at low to moderate suboptimality bounds, where focal is repeatedly emptied until $best_f$ is finally expanded, raising $f(best_f)$ and refilling focal.

3 Explicit Estimation Search

The central contribution of this paper is a new bounded suboptimal search algorithm, Explicit Estimation Search (EES), that incorporates the objectives of bounded suboptimal search directly into its search order. Using $\widehat{h}$ and $\widehat{d}$ functions, it explicitly estimates the cost and distance-to-go for search guidance, relying on lower bounds solely for providing suboptimality guarantees. In addition to $best_f$, EES refers to $best_{\widehat{f}}$, the node with the lowest predicted solution cost, that is $\widehat{f}(n) = g(n) + \widehat{h}(n)$ and $best_{\widehat{d}}$, the node, among all those that appear w-admissible, that appears closest to a goal.

$$\begin{aligned} best_{\widehat{f}} &= \operatorname*{argmin}_{n \in open} \widehat{f}(n) \\ best_{\widehat{d}} &= \operatorname*{argmin}_{n \in open \wedge \widehat{f}(n) \leq w \cdot \widehat{f}(best_{\widehat{f}})} \widehat{d}(n) \end{aligned}$$

Note that $best_{\widehat{d}}$ is chosen from a *focal* list based on $best_{\widehat{f}}$ because $\widehat{f}(best_{\widehat{f}})$ is our current estimate of the cost of an optimal solution, thus nodes with $\widehat{f}(n) \leq w \cdot \widehat{f}(best_{\widehat{f}})$ are those nodes that we suspect lead to solutions within the desired suboptimality bound. At every expansion, EES chooses from among these three nodes using the function:

```
$selectNode$
1.  **if** $\widehat{f}(best_{\widehat{d}}) \leq w \cdot f(best_f)$ **then** $best_{\widehat{d}}$
2.  **else if** $\widehat{f}(best_{\widehat{f}}) \leq w \cdot f(best_f)$ **then** $best_{\widehat{f}}$
3.  **else** $best_f$
```

We first consider $best_{\widehat{d}}$, as pursuing nearer goals should lead to a goal fastest, satisfying the second objective of bounded suboptimal search. $best_{\widehat{d}}$ is selected if its expected solution cost can be shown to be within the suboptimality bound. If $best_{\widehat{d}}$ is unsuitable, $best_{\widehat{f}}$ is examined. We suspect that this node lies along a path to an optimal solution. Expanding this node can also expand the set of candidates for $best_{\widehat{d}}$ by raising the estimated cost of $best_{\widehat{f}}$. We only expand $best_{\widehat{f}}$ if it is estimated to lead to a solution within the bound. If neither $best_{\widehat{f}}$ nor $best_{\widehat{d}}$ are thought to be within the bound, we return $best_f$. Expanding it can raise our lower bound $f(best_f)$, allowing us to consider $best_{\widehat{d}}$ or $best_{\widehat{f}}$ in the next iteration.

Theorem 1 *If $\widehat{h}(n) \geq h(n)$ and $g(opt)$ represents the cost of an optimal solution, then for every node n expanded by EES, it is true that $f(n) \leq w \cdot g(opt)$, and thus EES returns w-admissible solutions.*

Proof: $selectNode$ will always return one of $best_{\widehat{d}}$, $best_{\widehat{f}}$ or $best_f$. No matter which node n is selected we will show that $f(n) \leq w \cdot f(best_f)$. This is trivial when $best_f$ is chosen. When $best_{\widehat{d}}$ is selected:

$$\begin{aligned} \widehat{f}(best_{\widehat{d}}) &\leq w \cdot f(best_f) && \text{by } selectNode \\ g(best_{\widehat{d}}) + \widehat{h}(best_{\widehat{d}}) &\leq w \cdot f(best_f) && \text{by def. of } \widehat{f} \\ g(best_{\widehat{d}}) + h(best_{\widehat{d}}) &\leq w \cdot f(best_f) && \text{by } h \leq \widehat{h} \\ f(best_{\widehat{d}}) &\leq w \cdot f(best_f) && \text{by def. of } f \\ f(best_{\widehat{d}}) &\leq w \cdot g(opt) && \text{by admissible } h \end{aligned}$$

When $best_{\widehat{d}}$ is a solution, $h(best_{\widehat{d}}) = 0$ and $f(best_{\widehat{d}}) = g(best_{\widehat{d}})$, thus the cost of the solution represented by $best_{\widehat{d}}$ is within a bounded factor w of the cost of an optimal solution. The $best_{\widehat{f}}$ case is analogous. $\square$

3.1 The Behavior of EES

Explicit Estimation Search takes both the distance-to-go estimate as well as the cost-to-go estimate into consideration when deciding which node to expand next. This allows it to prefer finding solutions within the bound as quickly as possible. It does this by operating in the same spirit as A^*_ϵ, using both an open and a focal list. However, unlike A^*_ϵ, EES consults an unbiased estimate of the cost-to-go for nodes in order to form its focal list. $\widehat{f}(n)$ is less likely to rise along a path than $f(n)$, considering that $f(n)$ can only rise or stay the same by the admissibility of $h(n)$. Relying on an unbiased estimate tempers the tendency for estimated costs to always rise and avoids the thrashing behavior of A^*_ϵ.

Like optimistic and skeptical search, EES uses a cleanup list, a list of all open nodes sorted on $f(n)$, to prove its suboptimality bound. Rather than doing all of these expansions after having found a solution, EES interleaves cleanup expansions with those directed towards finding a solution. We can see this in the definition of $selectNode$, where EES may either raise the lower bound on optimal solution cost by selecting $best_f$, or it may instead pursue a solution by expanding either of the other two nodes. It can never run into the problem of having an incumbent solution that falls outside of the desired suboptimality bound because every node selected for expansion is within the bound.

Much like A^*_ϵ or weighted A*, EES will become faster as the bound is loosened. Like A^*_ϵ, EES becomes a greedy search on $\widehat{d}$. This contrasts with weighted A*, skeptical search, and optimistic search which focus exclusively on cost estimates and thus do not minimize search time.

3.2 Alternative Node Selection Strategies

Although the formulation of $selectNode$ is directly motivated by the stated goals of bounded suboptimal search, it is natural to wonder if there are other strategies that may have better performance. We consider two alternatives, one conservative and one optimistic. First, a more conservative approach that prefers to do the bound-proving expansions, those on $best_f$, as early as possible and so it first considers $best_f$:

```
$selectNode_{con}$
1.  **if** $\widehat{f}(best_{\widehat{f}}) > w \cdot f(best_f)$ **then** $best_f$
2.  **else if** $\widehat{f}(best_{\widehat{d}}) \leq w \cdot f(best_f)$ **then** $best_{\widehat{d}}$
3.  **else** $best_{\widehat{f}}$
```

If $best_f$ isn't selected for expansion, it then considers $best_{\widehat{d}}$ and $best_{\widehat{f}}$ as before. This expansion order produces a solution within the desired bound by the same argument as that for $selectNode$. While this alternate select node seems quite different on the surface, it turns out that it is identical to the original rules. If we were to strengthen the rules for selecting each of the three nodes by including the negation of the previous rules, we would see that the two functions are identical.

Next, consider an optimistic approach, analogous to that used by optimistic search and skeptical search:

```
$selectNode_{opt}$
1.  **if** no incumbent **then** $best_{\widehat{d}}$
2.  **else if** $\widehat{f}(best_{\widehat{d}}) \leq w \cdot f(best_f)$ **then** $best_{\widehat{d}}$
3.  **else if** $\widehat{f}(best_{\widehat{f}}) \leq w \cdot f(best_f)$ **then** $best_{\widehat{f}}$
4.  **else** $best_f$
```

In its initial phase, EES$_{opt}$ finds the shortest solution it believes to be within the bound. Then, after finding an incumbent solution, it behaves exactly like EES. The search ends when either $w \cdot f(best_f)$ is larger than the cost of the incumbent solution or when a new goal is expanded, because in either case EES$_{opt}$ has produced a w-admissible solution. We will examine both EES and EES$_{opt}$ in our evaluation below.

3.3 Implementation

EES is structured like a classic best-first search: insert the initial node into $open$, and at each step, we select the next node for expansion using $selectNode$. To efficiently access $best_{\widehat{f}}$, $best_{\widehat{d}}$, and $best_f$, EES maintains three queues, the *open* list, *focal* list, and *cleanup* list. These lists are used to access $best_{\widehat{f}}$, $best_{\widehat{d}}$, and $best_f$ respectively. The *open* list contains all generated but unexpanded nodes sorted on $\widehat{f}(n)$. The node at the front of the *open* list is $best_{\widehat{f}}$. *focal* is a prefix of the *open* list ordered on $\widehat{d}$. *focal* contains all of those nodes where $\widehat{f}(n) \leq w \cdot \widehat{f}(best_{\widehat{f}})$. The node at the front of *focal* is $best_{\widehat{d}}$. *cleanup* contains all nodes from *open*, but is ordered on $f(n)$ instead of $\widehat{f}(n)$. The node at the front of *cleanup* is $best_f$. We need to be able to quickly select a node at the front of one these queues, remove it from all relevant data structures, and reinsert its children efficiently. To accomplish this, we implement *cleanup* as a binary heap, *open* as a red-black tree, and *focal* as a heap synchronized with a left prefix of *open*. This lets us perform most insertions and removals in logarithmic time except for transferring nodes from *open* onto *focal* as $best_{\widehat{f}}$ changes. This requires us to visit a small range of the red-black tree in order to put the correct nodes in *focal*.

3.4 Simplified Approaches

With three queues to manage, EES has substantial overhead when compared to other bounded suboptimal search algorithms. One way to simplify EES would be to remove some of the node orderings. It is clear that we can not ignore the *cleanup* list, or we would lose bounded suboptimality. If we were to eliminate the *open* list, and synchronize *focal* with *cleanup* instead, we would obtain A^*_ϵ. If we were to ignore *focal*, and instead only expand either $best_f$ or $best_{\widehat{f}}$, we would be left with a novel interleaved implementation of skeptical search, but this would not be able to prefer the shorter of two solutions within the bound. We can conclude that EES is only as complicated as it needs to be. In the next section, we will see that its overhead is worthwhile.

4 Empirical Evaluation

From the description of the algorithm, we can see that EES was designed to work well in domains where solution cost and length can differ and it should do particularly well on problems with computationally expensive node expansion functions and heuristics. We do not know how much of an advantage using distance estimates will provide, nor do we know how much the overhead of EES will degrade performance in domains where expansion and heuristic computation are cheap. To gain a better understanding of these aspects of the algorithm, we performed an empirical evaluation across six benchmark domains. All algorithms were implemented in Objective Caml and compiled to native code on 64-bit Intel Linux systems with 3.16 GHz Core2 duo processors and 8 GB of RAM. Algorithms were run until they solved the problem or until ten minutes had passed.

We examined the following algorithms:

weighted A*(**wA***) [Pohl, 1970] For domains with many duplicates and consistent heuristics, it ignores duplicate states as this has no impact on the suboptimality bound and typically improves performance [Likhachev *et al.*, 2003].

A^*_ϵ(**A* eps**) [Pearl and Kim, 1982] using the base distance-to-go heuristic d to sort its focal list.

Optimistic search as described by Thayer and Ruml [2008].

Skeptical search with $\widehat{h}$ and $\widehat{d}$ estimated from the base h and d using the on-line single step heuristic corrections presented in Thayer and Ruml [2011]. The technique calculates the mean one step error in both h and d along the current search path from the root to the current node. This measurement of heuristic error is then used to produce a more accurate, but potentially inadmissible heuristic.

EES using the same estimates of $\widehat{h}$ and $\widehat{d}$ as skeptical.

EES Opt is EES using $selectNode_{opt}$.

We evaluated these algorithms on the following domains:

Dock Robot We implemented a dock robot domain inspired by Ghallab *et al.* [2004] and the depots domain from the International Planning Competition. Here, a robot must move containers to their desired locations. Containers are stacked at a location using a crane, and only the topmost container on a pile may be accessed at any time. The robot may drive between locations and load or unload itself using the crane at the location. We tested on 150 randomly configured problems having three locations laid out on a unit square and ten containers with random start and goal configurations. Driving between the depots has a cost of the distance between them, loading and unloading the robot costs 0.1, and the cost of using the crane was 0.05 times the height of the stack of containers at the depot. h was computed as the cost of driving between all depots with containers that did not belong to them in the goal configuration plus the cost of moving the deepest out of place container in the stack to the robot. d was computed similarly, but 1 is used rather than the actual costs.

We show results for this domain in the leftmost plot of Figure 1. All plots are laid out similarly, with the x-axis representing the user-supplied suboptimality bound and the y-axis representing the mean CPU time taken to find a solution (often on a log_{10} scale). We present 95% confidence intervals on the mean for all plots. In dock robots, we show results for suboptimality bounds of 1.2 and above. Below this, no algorithm could reliably solve the instances within memory. Here we see that the techniques that rely on both inadmissible heuristics and estimates of solution length outperform weighted A* and optimistic search. A^*_ϵ performs particularly poorly in this domain, with run times that are hundreds of times larger than the other search algorithms, a direct result of the thrashing behavior described previously. Further we see that both variants of EES are significantly faster than skeptical search, with EES$_{opt}$ being slightly faster than EES for most suboptimality bounds. Both variants take about half the time of skeptical search to solve problems at the same bound. Not only are both EES variants consistently faster than skeptical search, their performance is more consistent, as noted by the tighter confidence intervals around their means.

Vacuum World In this domain, which follows the first state space presented in Russell and Norvig, page 36 [2010], a robot is charged with cleaning up a grid world. Movement is in the cardinal directions, and when the robot is on top of a pile of dirt, it may clean it up. We consider two variants of this problem, a unit-cost implementation and another more realistic variant where the cost of taking an action is one plus the number of dirt piles the robot has cleaned up (the weight from the dirt drains the battery faster). We used 50 instances that are 200 by 200, each cell having a 35% probability of being blocked. We place ten piles of dirt and the robot randomly in unblocked cells and ensure that the problem can be solved. For the unit cost domain, we use the minimum spanning tree of the dirt piles and the robot for h and for d we estimate the cost of a greedy traversal of the dirt piles. That is, we make a freespace assumption on the grid and calculate the number of actions required to send the robot to the nearest pile of dirt, then the nearest after that, and so on. For h on the heavy vacuum problems, we compute the minimum spanning as before, order the edges by greatest length first, and then multiply the edge weights by the current weight of the robot plus the number of edges already considered. d is unchanged.

The center panel of Figure 1 shows the relative performance of the algorithms on the unit-cost vacuum problem. We see that there is very little difference between EES and EES Opt. which both outperform the other search algorithms about an order of magnitude, solving the problems in tenths of seconds instead of requiring several seconds. Again, what these two algorithms have in common that differs from other approaches is their ability to rely on inadmissible cost-to-go estimates and estimates of solution length for search guidance. The performance gap between EES and skeptical search is not as large here as it is in domains with actions of differing costs. In unit cost domains like this, searches that become greedy on cost-to-go behave identically to those that become greedy on distance-to-go. Removing the distinction between solution cost and solution length removes one of the advantages that EES holds over skeptical search.

The right most panel of Figure 1 shows the performance of the algorithms for the heavy vacuum robot domain. Of the domains presented here, this has the best set of properties for use with EES; the heuristic is relatively expensive to compute and there is a difference between solution cost and solution length. For very tight bounds, the algorithms all perform similarly. As the bound is relaxed, both versions of EES clearly dominate the other approaches, being between one and two orders of magnitude faster than other approaches. Both vari-

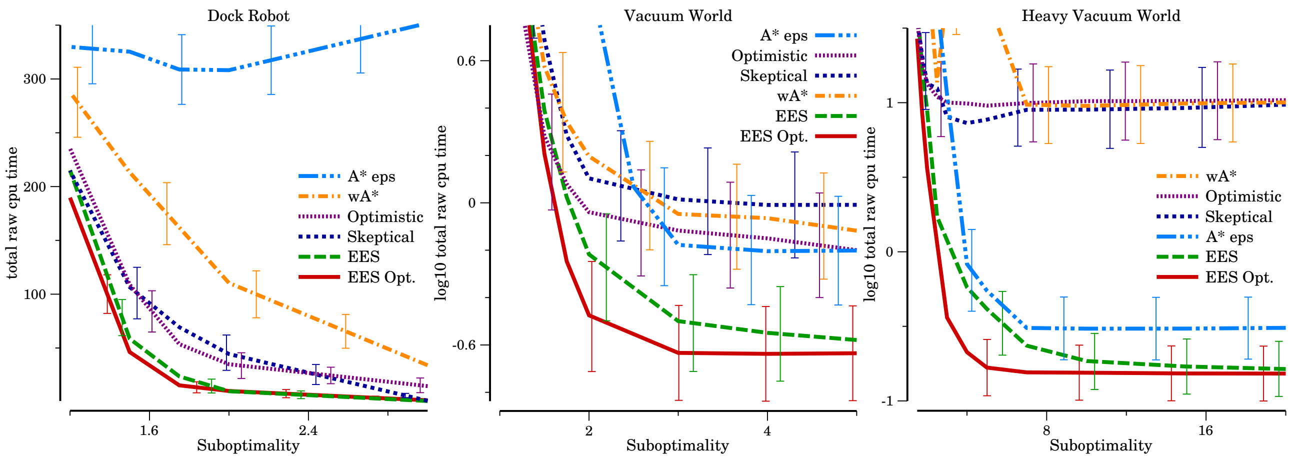

Figure 1: Mean CPU time required to find solutions within a given suboptimality bound.

ants solve the problems in fractions of a second instead of tens of seconds. A_ϵ^* makes a strong showing for high suboptimality bounds in this domain precisely because it can take advantage of the difference between solution cost and solution length. Of the algorithms that take advantage of distance estimates, it makes the weakest showing, as it begins to fail to find solutions for many problems at suboptimality bounds as large as 4, whereas the EES algorithms solve all instances down to a suboptimality bound of 1.5.

Dynamic Robot Navigation Following Likhachev *et al.* [2003], the goal is to find the fastest path from the initial state of the robot to some goal location and heading, taking momentum into account. We use worlds that are 200 by 200 cells in size. We scatter 25 lines, up to 70 cells in length, with random orientations across the domain and present results averaged over 100 instances. We precompute the shortest path from the goal to all states, ignoring dynamics. To compute h, we take the length of the shortest path from a node to a goal and divide it by the maximum velocity of the robot. For d, we report the number of actions along that path.

The leftmost panel of Figure 2 shows performance of the algorithms in a dynamic robot navigation domain. As in heavy vacuums and dock robots, there is a substantial difference between the number of actions in a plan and the cost of that plan, however here computing the heuristic is cheap. For tight bounds, EES is faster than A_ϵ^*, however for more generous bounds A_ϵ^* pulls ahead. When we evaluate these algorithms in terms of nodes expanded (omitted for space), we would see that their performance is similar. The better times of A_ϵ^* in this domain can be attributed to reduced overhead.

Sliding Tiles Puzzles We examined the 100 instances of the 15-puzzle presented by Korf [1985]. The center panel of Figure 2 shows the relative performance of the algorithms on the unit-cost sliding tiles puzzle. We use Manhattan distance plus linear conflicts for both h and d. This is exactly the wrong kind of domain for EES. Node generation and heuristic evaluation are incredibly fast, and there is no difference between the number of actions in a solution and the cost of that solution. Such a domain lays bare the overhead of EES and prevents it from taking advantage of its ability to distinguish between cost and length of solutions. We see that, in terms of time to solutions, EES is almost indistinguishable from weighted A* and skeptical search and worse, by about an order of magnitude, than optimistic search. If we examine these results in terms of nodes generated (omitted for space), we see that EES, skeptical, and optimistic search examine a similar number of nodes for many suboptimality bounds; the time difference is due to the differing overheads of the algorithms.

The right panel of Figure 2 shows the performance of the algorithms on inverse tiles problems, where the cost of moving a tile is the inverse of its face value, $\frac{1}{face}$. This separates the cost and length of a solution without altering other properties of the domain, such as its connectivity and branching factor. This simple change to the cost function makes these problems remarkably difficult to solve, and reveals a hitherto unappreciated brittleness in previously proposed algorithms. We use a weighted Manhattan distance for h and the unit Manhattan distance for d. Explicit estimation search and skeptical search are the only algorithms shown for this domain, as all of the other algorithms fail to solve at least half of the problems within a 600 second timeout across all suboptimality bounds shown in the plot. This failure can be attributed in part to their inability to correct the admissible heuristic for the problem into a stronger inadmissible heuristic during search, as this is the fundamental difference between skeptical search, which can solve the problem, and optimistic search, which cannot. The ability to solve instances is not entirely due to reliance on $d(n)$, as A_ϵ^* is unable to solve many of the instances in this domain. EES is significantly faster than skeptical search in this domain, about 5 to 6 times as fast for moderate suboptimality bounds and a little more than a full order of magnitude for tight bounds, because it can incorporate distance-to-go estimates directly for guidance. While we show data out to a suboptimality bound of 50, this trend holds out to at least 100,000.

Summary In our benchmark domains, we saw that explicit estimation search was consistently faster than other approaches for bounded suboptimal search in domains where actions had varying costs. Although A_ϵ^* was faster for some bounds in one domain, its behavior is so brittle that it is of little practical use. EES' advantage increased in those do-

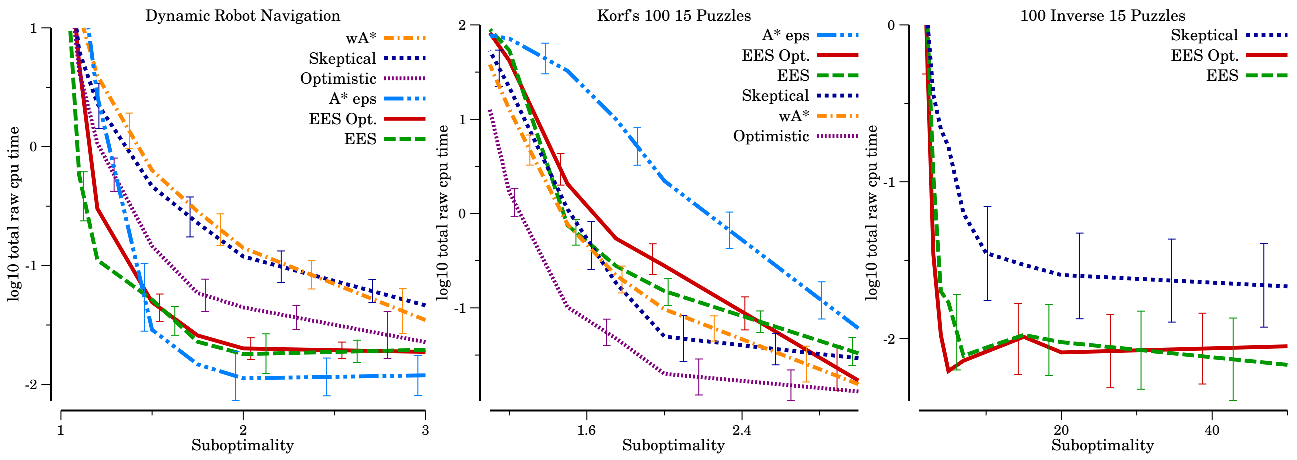

Figure 2: Mean CPU time required to find solutions within a given suboptimality bound.

mains where either child generation or heuristic computation were relatively expensive. Here, a superior search order can overcome search overhead. For domains with cheap heuristics and unit cost actions, EES was merely competitive with previously proposed algorithms in terms of solving time.

While optimistic search is faster than either version of EES for one domain, unit cost tiles, note that it cannot solve a simple real-valued variant, inverse tiles, reliably. Similarly, while A^*_ϵ outperforms all algorithms for part of the dynamic robot navigation problems, it performs poorly, sometimes failing catastrophically, in the other domains. EES' overhead affords it an amount of robustness lacking in the other algorithms.

Neither variant of EES dominates the other in our evaluation. The effectiveness of EES$_{opt}$ depends on the accuracy of $\widehat{h}$. If $\widehat{h}$ is relatively accurate, it is more likely that the incumbent solution found by EES$_{opt}$ will be within the desired suboptimality bound, and EES$_{opt}$ should outperform EES. When $\widehat{h}$ is inaccurate, it is very likely that the incumbent solution will be outside of the bound, and EES should be faster.

5 Conclusions

We introduced a new bounded suboptimal heuristic search algorithm, Explicit Estimation Search (EES), that takes advantage of inadmissible heuristics and distance estimates to quickly find solutions within a given suboptimality bound. It does this by directly optimizing the objective of bounded suboptimal search. While it is widely appreciated that inadmissible heuristics are useful for guiding search, it is equally important to realize that we can use these inadmissible heuristics to guide search without sacrificing bounds on solution quality. EES is significantly faster than previous approaches, several orders of magnitude in some cases. It achieves this in large part by explicitly estimating the cost of completing a partial solution in addition to computing a lower bound when deciding which node to expand next. EES is robust, but works best in domains with actions of varying cost.

6 Acknowledgments

We gratefully acknowledge support from NSF (grant IIS-0812141) and the DARPA CSSG program (grant N10AP20029).

References

[Ghallab and Allard, 1983] M. Ghallab and D.G. Allard. A$_\epsilon$: An efficient near admissible heuristic search algorithm. In *Proceedings of IJCAI-83*, 1983.

[Ghallab *et al.*, 2004] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning: Theory and Practice*. Morgan Kaufmann, San Francisco, CA, 2004.

[Hart *et al.*, 1968] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions of Systems Science and Cybernetics*, SSC-4(2):100–107, July 1968.

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

[Likhachev *et al.*, 2003] Maxim Likhachev, Geoff Gordon, and Sebastian Thrun. ARA*: Anytime A* with provable bounds on sub-optimality. In *Proceedings of NIPS-03*, 2003.

[Pearl and Kim, 1982] Judea Pearl and Jin H. Kim. Studies in semi-admissible heuristics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-4(4):391–399, July 1982.

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

[Pohl, 1973] Ira Pohl. The avoidance of (relative) catastrophe, heuristic competence, genuine dynamic weighting and computation issues in heuristic problem solving. In *Proceedings of IJCAI-73*, pages 12–17, 1973.

[Russell and Norvig, 2010] Stuart Russell and Peteer Norvig. *Artificial Intelligence: A Modern Approach*. Third edition, 2010.

[Thayer and Ruml, 2008] Jordan T. Thayer and Wheeler Ruml. Faster than weighted A*: An optimistic approach to bounded suboptimal search. In *Proceedings of ICAPS-08*, 2008.

[Thayer and Ruml, 2009] Jordan T. Thayer and Wheeler Ruml. Using distance estimates in heuristic search. In *Proceedings of ICAPS-09*, 2009.

[Thayer and Ruml, 2011] Jordan T. Thayer and Wheeler Ruml. Learning inadmissible heuristics during search. In *Proceedings of ICAPS-11*, 2011.

[Thayer *et al.*, 2009] Jordan T. Thayer, Wheeler Ruml, and Jeff Kreis. Using distance estimates in heuristic search: A re-evaluation. In *Symposium on Combinatorial Search*, 2009.