Surrogate Search As a Way to Combat Harmful Effects of Cost Based Satisficing Search

Paper Number 127

William Cushing, J. Benton, Subbarao Kambhampati

Abstract

Recently, several researchers have found that cost-based satisficing search with A* often runs into problems. Although some "work arounds" have been proposed to ameliorate the problem, there has not been any concerted effort to pinpoint its origin. In this paper, we argue that the origins of this problem can be traced back to the fact that most planners that try to optimize cost also used cost-based evaluation functions (i.e., $f(n)$ is a cost estimate). We show that cost-based evaluation functions become ill-behaved whenever there is a wide variance in action costs; something that is all too common in planning domains. The general solution to this malady is what we call a *surrogate search*, where a surrogate evaluation function that doesn't directly track the cost objective, and is resistant to cost-variance is used. We will discuss some compelling choices for surrogate evaluation functions that are based on size rather than cost. Of particular practical interest is a cost-sensitive version of size-based evaluation function where the heuristic estimates the size of cheap paths, as it provides attractive quality vs. speed tradeoffs.

1 Introduction

Much of the scale-up, as well as the research focus, in the automated planning community in the recent years has been on satisficing planning. Unfortunately, there hasn't been a concomitant increase in our understanding of satisficing search. Too often, the "theory" of satisficing search defaults to doing (W)A* with inadmissible heuristics. While removing the requirement of admissible heuristics certainly relaxes the guarantee of optimality, there is no implied guarantee of efficiency. A combinatorial search can be seen to consist of two parts: a "discovery" part where the (optimal) solution is found and a "proof" part where the optimality of the solution is verified. While an optimizing search depends crucially on both these phases, a satisficing search is instead affected more directly by the discovery phase. Standard A* search conflates the discovery and proof phases together and terminates only when it picks the optimal path for expansion. By default, satisficing planners use the same search regime, but relax the admissibility requirement on the heuristics.[1] This may not cause too much of a problem in domains with uniform action costs, but when actions can have non-uniform costs, the optimal and second optimal solution can be arbitrarily far apart in depth. Consequently, (W)A* search with cost-based evaluation functions can be an arbitrarily bad strategy for satisficing search, as it waits until the solution is both discovered *and* proved to be (within some bound of) optimal.

To be more specific, consider a planning problem for which the cost-optimal and second-best solution to a problem exist on 10 and 1000 unspecified actions. *The optimal solution may be the larger one.* How long should it take just to find the 10 action plan? How long should it take to prove (or disprove) its optimality? In general (presuming PSPACE/EXPSPACE $\neq$ P):

1. Discovery should require time exponential in, at most, 10.
2. Proof should require time exponential in, at least, 1000.

That is, in principle, the only way to (domain-independently) prove that the 10 action plan is better or worse than the 1000 action one is to in fact go and discover the 1000 action plan. Thus, A* search with cost-based evaluation function will take time proportional to b^{1000} for either discovery or proof. Simple breadth-first search discovers a solution in time proportional to b^{10} (and proof in $O(b^{1000})$). The problem here is that the cost-based evaluation function can grow arbitrarily slowly with respect to the search depth.

Using both abstract and benchmark problems, we will demonstrate that this is a systematic weakness of any search that uses a cost-based *evaluation function* (*i.e.*, an f function that returns answers in cost units). In particular, we shall see that if ε is the smallest cost action (after all costs are normalized so the maximal cost action costs 1 unit), then the time taken to discover a depth d optimal solution will be $b^{\frac{d}{\varepsilon}}$. If all actions have same cost, then $\varepsilon \approx 1$, whereas if the actions have significant cost variance, then $\varepsilon \ll 1$. For a variety of reasons, most real-world planning domains do exhibit high cost variance, thus presenting an "*ε-cost trap*" that forces any cost-based satisficing search to dig its own ($\frac{1}{\varepsilon}$ deep) grave.

Consequently, we argue that satisficing search should reevaluate the decision to directly use cost-based evaluation functions given their suceptibility to ε-cost traps, even if they are interested in the cost measure of the resulting plan. But what exactly should take their place? We suggest using *surrogate evaluation functions* in lieu of the cost-based ones. There are two desiderata for such surrogate evaluation functions: (i) they should be immune to the ε-cost traps by in-

[1] In the extreme case, by using an infinite heuristic weight: "greedy best-first search".

creasing at a reasonable rate with respect to search depth and (ii) they should track the (true) cost objective well. We will consider two size-based branch-and-bound alternatives: the straightforward one which completely ignores costs and sticks to a purely size-based evaluation function, and a more subtle one that uses a cost-sensitive size-based evaluation function (specifically, the heuristic estimates the size of the cheapest cost path; see Section 2). We show that both of these outperform cost-based evaluation functions in the presence of ε-cost traps, with the second one providing better quality plans (for the same run time limits) than the first in our empirical studies.

While some of the problems with cost-based satisficing search have also been observed, in passing, by other researchers (*e.g.*, (Benton et al. 2010; Richter and Westphal 2010), and some work-arounds have been suggested, our main contribution is to bring to the fore its fundamental nature. We note that, though the focus of this paper is on cost-based search, the phenomenon is more general and applies to any objective functions that do not grow fast enough with the search depth (e.g. measured in terms of node expansion operations). Other examples of such ill-behaved objective functions include "makespan," which was used to illurate the problems of g-value plateaus in (Benton et al. 2010).

The rest of the paper is organized as follows. In the next section, we present some preliminary notation to formally specify cost-based, size-based as well as cost-sensitive size-based search alternatives. Next, we present two abstract and fundamental search spaces, which demonstrate that *cost-based* evaluation functions are "always" needlessly prone to ε-cost traps (Section 3). Section 4 strengthens the intuitions behind this analysis by viewing best-first search as flooding topological surfaces set up by evaluation functions. We will argue that of all possible topological surfaces (*i.e.*, evaluation functions) to choose for search, cost-based is one of the worst. In Section 5 we propose a solution to this malady in terms of surrogate evaluation functions, and describe two candidate surrogates. In Section 6, we put all this analysis to empirical validation by experimenting with LAMA (Richter and Westphal 2010) and SapaReplan. The experiments do show that surrogate search based on size-based alternatives can out-perform direct cost-based search. Modern planners such as LAMA use a plethora of improvements beyond vanilla A* search, and in an extended version, we provide a deeper analysis on which extensions of LAMA seem to help it mask (but not fully overcome) the pernicious effects of cost-based evaluation functions.

2 Setup and Notation

We gear the problem set up to be in line with the prevalent view of state-space search in modern, state-of-the-art satisficing planners. First, we assume the current popular approach of reducing planning to graph search. That is, planners typically model the state-space in a causal direction, so the problem becomes one of extracting paths, meaning whole plans do not need to be stored in each search node. More important is that the structure of the graph is given *implicitly* by a procedure Γ, the child generator, with $\Gamma(v)$ returning the local subgraph leaving v; i.e., $\Gamma(v)$ computes the subgraph $(N^+[v], E(\{v\}, V-v)) = (\{u \mid (v,u) \in E\} + v, \{(v,u) \mid (v,u) \in E\})$ along with all associated labels, weights, and so forth. That is, our analysis depends on the assumption that *an implicit representation of the graph is the only computationally feasible representation*, a common requirement for analyzing the A* family of algorithms (Hart, Nilsson, and Raphael 1968; Dechter and Pearl 1985).

The search problem is to find a path from an initial state, i, to some goal state in $\mathcal{G}$. Let costs be represented as edge weights, where $c(e)$ is the cost of the edge e. Let $g^*_c(v)$ be the (optimal) cost-to-reach v (from i), and $h^*_c(v)$ be the (optimal) cost-to-go from v (to the goal). Then $f^*_c(v) := g^*_c(v) + h^*_c(v)$, the cost-through v, is the cost of the cheapest i-$\mathcal{G}$ path passing through v. For discussing smallest solutions, let $f^*_s(v)$ denote the smallest i-$\mathcal{G}$ path through v. We can also consider the size of the cheapest i-$\mathcal{G}$ path passing through v, which we call $\hat{f}^*_s(v)$.

We define a search node n as equivalent to a path represented as a linked list (of edges). In particular, we distinguish this from the state of n (its last vertex), $n\,.\,v$. We say $n\,.\,a$ (for action) is the last edge of the path and $n\,.\,p$ (for parent) is the subpath excluding $n\,.\,a$. Let $n' = na$ denote extending n by the edge a (so $a = (n\,.\,v, n'\,.\,v)$). The function $g_c(n)$ (g-cost) is just the recursive formulation of path cost: $g_c(n) := g_c(n\,.\,p) + c(n\,.\,a)$ ($g_c(n) := 0$ if n is the trivial path). So $g^*_c(v) \leq g_c(n)$ for all i-v paths n, with equality for at least one of them. Similarly let $g_s(n) := g_s(n\,.\,p) + 1$ (initialized at 0), so that $g_s(n)$ is an upper bound on the shortest path reaching the same state ($n\,.\,v$).

A goal state is a target vertex where a plan may stop and be a valid solution. We fix a computed predicate $\mathcal{G}(v)$ (a black-box), the *goal*, encoding the set of goal states. Let $h_c(v)$, the *heuristic*, be a procedure to estimate $h^*_c(v)$. (Sometimes h_c is considered a function of the search node, *i.e.*, the whole path, rather than just the last vertex.) The heuristic h_c is called *admissible* if it is a *guaranteed* lower bound. An inadmissible heuristic lacks the guarantee, but might anyways be coincidentally admissible. Let $h_s(v)$ estimate the remaining depth to the nearest goal, and let $\hat{h}_s(v)$ estimate the remaining depth to the cheapest reachable goal. Importantly, anything goes with such heuristics. Later we note that taking the size of a heuristic that optimizes on cost is an acceptable (and practical) interpretation of $\hat{h}_s(v)$.

We focus on two different definitions of f (the evaluation function). Since we study cost-based planning, we consider $f_c(n) := g_c(n) + h_c(n\,.\,v)$; this is the (standard, cost-based) *evaluation function* of A^*: cheapest-completion-first. We compare this to $f_s(n) := g_s(n) + h_s(n\,.\,v)$, the canonical size-based (or search distance) evaluation function, equivalent to f_c under uniform weights. Any combination of g_c and h_c is *cost-based*; any combination of g_s and h_s is *size-based* (*e.g.*, breadth-first search is size-based). The evaluation function $\hat{f}_s(n) := g_s(n) + \hat{h}_s(n\,.\,v)$ is also size-based, but cost-sensitive.

Pseudo-code for best-first branch-and-bound search of implicit graphs is shown below. It continues searching after a solution is encountered and uses the current best solution value to prune the search space (line 4). The search is performed on a graph implicitly represented by Γ, with the

assumption being that the explicit graph is so large that it is better to invoke expensive heuristics (inside of EVALUATE) during the search than it is to just compute the graph up front. The question considered by this paper is how to implement EVALUATE.

BEST-FIRST-SEARCH$(i, \mathcal{G}, \Gamma, h_c)$

```
1  initialize search
2  while open not empty
3      n = open.remove()
4      if BOUND-TEST(n, h_c) then continue
5      if GOAL-TEST(n, G) then continue
6      if DUPLICATE-TEST(n) then continue
7      s = n.v
8      star = Γ(s)                          // Expand s
9      for each edge a = (s, s') from s to a child s' in star
10         n' = na
11         f = EVALUATE(n')
12         open.add(n', f)
13 return best-known-plan                   // Optimality is proven.
```

With respect to normalizing costs, we can let $\varepsilon := \frac{\min_a c(a)}{\max_a c(a)}$, that is, ε is the least cost edge after normalizing costs by the maximum cost (to bring costs into the range $[0, 1]$). We use the symbol ε for this ratio as we anticipate actions with high cost variance in real world planning problems. For example: boarding versus flying (ZenoTravel), mode-switching versus machine operation (Job-Shop), and (unskilled) labor versus (precious) material cost.

3 ε-cost Trap: Two Canonical Cases

In this section we argue that the mere presence of ε-cost misleads cost-based search, and that this is not an accidental phenomenon, but a systemic weakness of the very concept of "cost-based evaluation functions + systematic search + combinatorial graphs". We base this analysis in two abstract search spaces, in order to demonstrate the fundamental nature of such traps. Though we look at traps from the perspective of a couple examples, another thorough analysis of models of search can be found in both the general search context (Pearl 1984) and the planning-specific context (Helmert and Röger 2008).

For our analysis, the first abstract space we consider is a simple, non-trivial, non-uniform cost, intractably large, search space: the search space of an enormous cycle with one expensive edge. The second abstract space we consider is a more natural model of search (in planning): a uniform branching tree. Traps in these spaces are just exponentially sized *and connected* sets of ε-cost edges: not the common result of, for example, a typical random model of search. We briefly consider why planning benchmarks naturally give rise to such structure.

3.1 Cycle Trap

In this section we consider the simplest abstract example of the ε-cost 'trap'. The notion is that applying increasingly powerful heuristics, domain analysis, learning techniques, ..., to one's search problem transforms it into a simpler 'effective graph' — the graph for which Dijkstra's algorithm (Dijkstra 1959) produces isomorphic behavior. For example, let c' be a new edge-cost function obtained by setting edge costs to the difference in f values of the edge's endpoints: Dijkstra's algorithm on c' is A* on f.[2] Similarly take Γ' to be the result of applying one's favorite incompleteness-inducing pruning rules to Γ (the child generator), say, helpful actions (Hoffmann and Nebel 2001); then Dijkstra's algorithm on Γ' is A* with helpful action pruning.

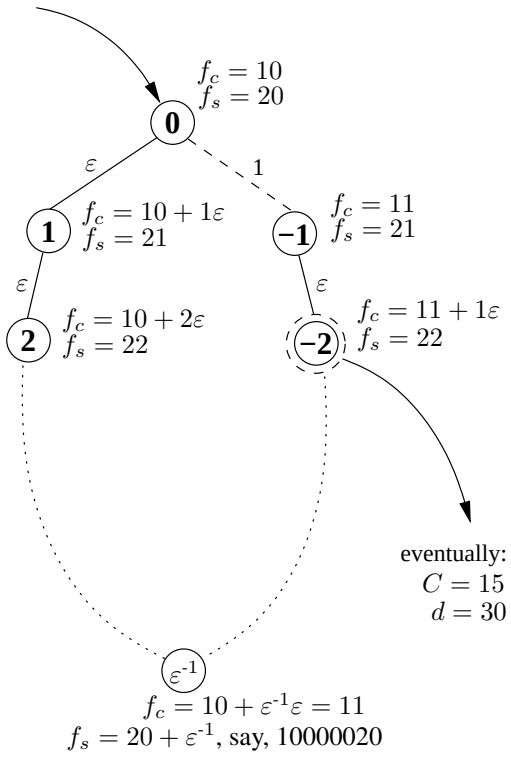

Figure 1: An ε-cost trap for cost-based search. The heuristic perceives all movement on the cycle to be irrelevant to achieving high quality plans. The state with label **-2** is one interesting way to leave the cycle, there may be (many) others. C denotes the cost of one such continuation from **-2**, and d its depth. Edge weights nominally denote changes in f_c: as given, locally, these are the same as changes in g_c. But increasing f_s by 1 at **-1** (and descendants) would, for example, model instead the special edge as having cost $\frac{1}{2}$ and being perceived as worst-possible in an undirected graph.

We presume the effective search graph remains very complex despite clever inference (or there is nothing to discuss). If there is a problem with search behavior in an exceedingly simple graph then we can suppose that no amount of domain analysis, learning, heuristics, and so forth, will incidentally address the problem: such inference must specifically address the issue of non-uniform costs. When none of the bells and whistles consider non-uniform costs to be a serious issue, the search permits wildly varying edge "costs" even in the effective search graph: $\varepsilon \approx \varepsilon' = \frac{\min_e c'(e)}{\max_e c'(e)}$. We demonstrate that this by itself is enough to produce very

[2] Systematic inconsistency of a heuristic translates to analyzing the behavior of Dijkstra's algorithm with many *negative* 'cost' edges, a typical reason to assume consistency in analysis.

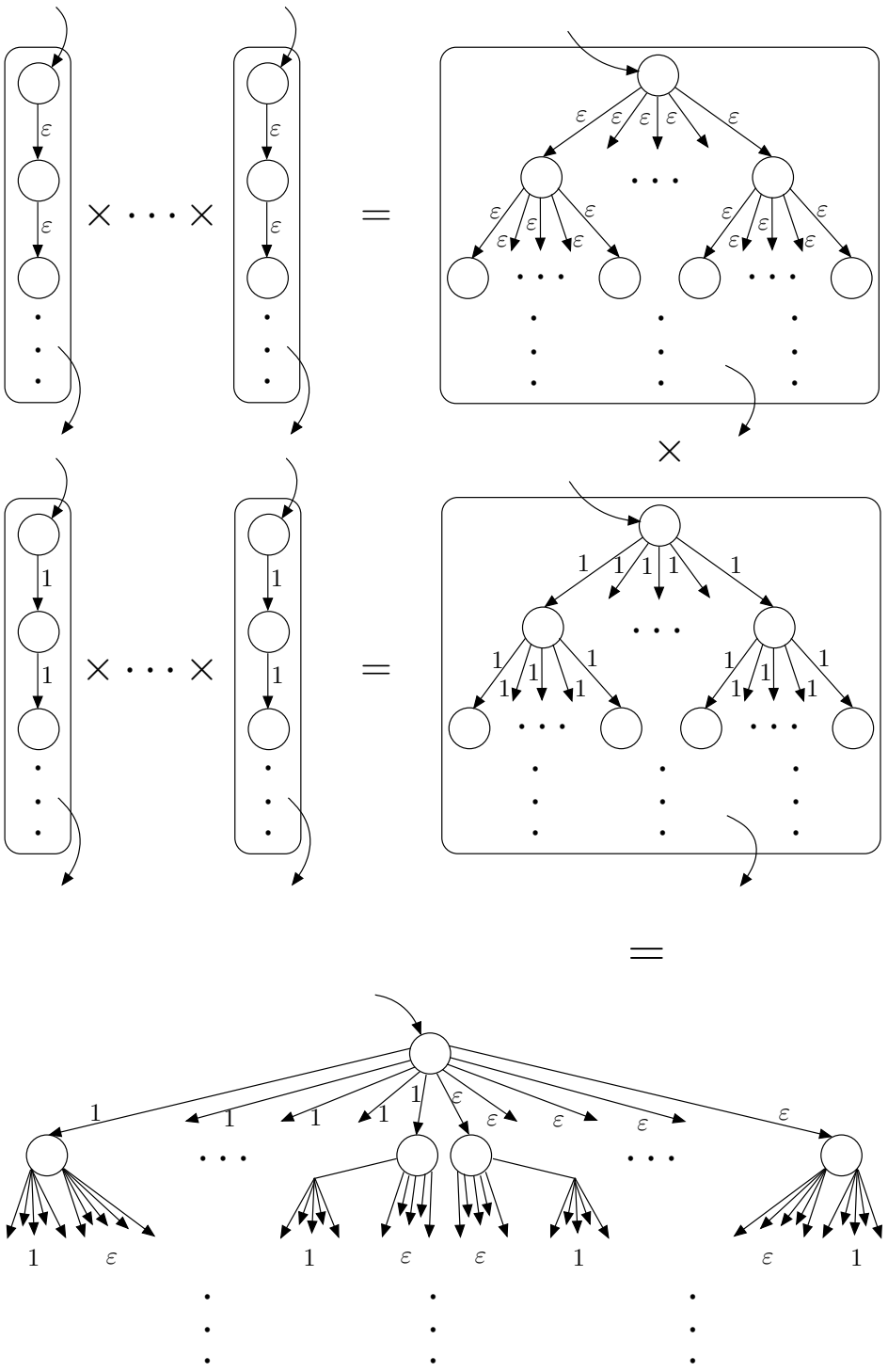

Figure 2: A trap for cost-based search involving exponentially large subtrees on ε-cost edges. We have two rather distinct kinds of physical objects that exist in the domain with primitive operators at rather distinct orders of magnitude; supposing uniformity and normalizing, one type involves ε-cost and the other involves cost 1. In this caes, there is a low-cost subspace, a high-cost subspace, and the full space, where each is a uniform branching tree. As trees are acyclic, it is probably best to think of these as search (rather than state) spaces. As depicted, planning for an individual object is trivial as there is no choice besides going forward. Other than that no significant amount of inference is being assumed, and in particular the effects of a heuristic are not depicted. For cost-based search to avoid "dying" due to memory or time computational resources, the heuristic would need to forecast every necessary cost 1 edge, so as to reduce its weight closer to 0. (Note that the aim of a heuristic is to drive all the weights to 0 along optimal/good paths, and to infinity for not-good/terrible/dead-end choices.) If any cut of the space across such edges (separating good solutions) is not foreseen, then backtracking into *all* of the low-cost subspaces so far encountered commences, to multiples of depth ε^{-1} — one such multiple for every unforeseen cost 1 cut.

troubling search behavior: ε-cost is a fundamental challenge to be overcome in planning.

There are several candidates for simple non-trivial state-spaces (*e.g.*, cliques), but clearly the cycle is fundamental (what kind of 'state-space' is acyclic?). So, the state-space we consider is the cycle, with associated exceedingly simple metric consisting of all uniform weights but for a single expensive edge. Its search space is certainly a simple non-trivial search space: the rooted tree on two leaves. So the single unforced decision to be made is in which direction to traverse the cycle: clockwise or counter-clockwise.

ε-cost Trap: Consider a counting problem of making some variable, x, encoded in k bits represent $2^k - 2 \equiv -2 \pmod{2^k}$, starting from 0, using only the operations of increment and decrement. There are 2 minimal solutions: incrementing $2^k - 2$ times, or decrementing twice. Set the cost of incrementing and decrementing to 1, except for transitioning between $x \equiv 0$ and $x \equiv -1$ costs, say, 2^{k-1} (in either direction). Then the 2 minimal solutions cost $2^k - 2$ and $2^{k-1} + 1$, or, normalized, $2(1 - \varepsilon)$ and $1 + \varepsilon$. Cost-based search loses: While both approaches prove optimality in exponential time ($O(2^k)$), size-based search discovered that optimal plan in constant time. See illustration of search over this example in Figure 1.

Performance Comparison: All Goals. The goal $x \equiv -2$ is chosen to best illustrate the trap. Consider the discovery problem for other goals. With the goal in the interval $2^k \cdot [0, \frac{1}{2}]$ cost-based search is twice as fast. With the goal in the interval $2^k \cdot [\frac{1}{2}, \frac{2}{3}]$ the performance gap narrows to break-even. For the last interval, $2^k \cdot [\frac{2}{3}, 1\rangle$, the size-based approach takes the lead — *by an enormous margin*. There is one additional region of interest. Taking the goal in the interval $2^k \cdot [\frac{2}{3}, \frac{3}{4}]$ there is a trade-off: size-based search finds a solution before cost-based search, but cost-based search finds the optimal solution first. Concerning time till optimality is proven, the cost-based approach is monotonically faster (of course). Specifically, the cost-based approach is faster by a factor of 2 for goals in the region $2^k \cdot [0, \frac{1}{2}]$, not faster for goals in the region $2^k \cdot [\frac{3}{4}, 1\rangle$, and by a factor of $(\frac{1}{2} + 2\alpha)^{-1}$ (bounded by 1 and 2) for goals of the form $x \equiv 2^k(\frac{1}{2} + \alpha)$, with $0 < \alpha < \frac{1}{4}$.

Performance Comparison: Feasible Goals. Considering all goals is inappropriate in the satisficing context; to illustrate, consider $k = 1000$ as an example of a large value of k). Fractions of exponentials are still exponentials — even the most patient reader working out this example will have forcibly terminated either search *long* before receiving any useful output. Except if the goal is of the form $x \equiv 0 \pm f(k)$ for some sub-exponential $f(k)$. Both approaches discover (and prove) the optimal solution in the positive case in time $O(f(k))$ (with size-based performing twice as much work). In the negative case, only the size-based approach manages to discover a solution (the optimal one, in time $O(f(k))$) before being killed. Moreover, while it will fail to produce a proof of such before termination, based on our understanding of the domain, we can show it to be posthumously correct. ($2^k - f(k) > 2^k \cdot \frac{3}{4}$ for any sub-exponential $f(k)$ with large enough k.)

How Good is Almost Perfect Search Control? Keep in mind that the representation of the space as a simple k bit counter is *not* available. In particular what 'increment' actually stands for is an inference-motivated choice of a single operator out of a large number of executable and promising operators at each state — in the language of Markov Decision Processes, we are allowing inference to be so close to perfect that the optimal policy is known at all but 1 state. Only one decision remains ... but no methods cleverer than search remain. Still the graph is intractably large. Cost-based search only explores in one direction: left, say. In the satisficing context such behavior is entirely inappropriate. What is appropriate? We want the search to explore the space so that a solution that exists only one step to the right can still be found, even if it is not optimal.

3.2 Branching Trap

In the counter problem the trap is not even *combinatorial*; the search problem consists of a single decision at the root, and the trap is just an exponentially deep path. For example, appending a large enough Towers of Hanoi problem to a planning benchmark, setting its actions at ε-cost, will hurt cost-based search — even given the perfect heuristic for the puzzle! Besides Hanoi, though, exponentially deep paths are not typical of planning benchmarks. So in this section we demonstrate that exponentially large subtrees on ε-cost edges are also traps.

Consider $x > 1$ high cost actions and $y > 1$ low cost actions in a uniform branching tree model of search space. The model is appropriate up to the point where duplicate state checking becomes significant. See the example illustrated in Figure 2. Observe that in the object-specific subspaces (the paths), a single edge ends up being multiplied into such a cut of the global space. Suppose the solution of interest costs C, in normalized units, so the solution lies at depth C or greater. Then cost-based search faces a grave situation: $O((x + y^{\frac{1}{\varepsilon}})^C)$ possibilities will be explored before considering all potential solutions of cost C.

A size-based search only ever considers at most $O((x + y)^d) = O(b^d)$ possibilities before consideration of all potential solutions of size d. But the more interesting question is how long it takes to find solutions of fixed cost rather than fixed depth. Note that $\frac{C}{\varepsilon} \geq d \geq C$. Assuming the high cost actions are relevant, that is, some number of them are needed by solutions, we have that solutions are not actually hidden as deep as $\frac{C}{\varepsilon}$. To help see this, suppose that solutions tend to be a mix of high and low cost actions in equal proportion. Then the depth of those solutions with cost C is $d = 2\,\frac{C}{1+\varepsilon}$ (*i.e.*, $\frac{d}{2} \cdot 1 + \frac{d}{2} \cdot \varepsilon = C$). At such depths the size-based approach is the clear winner: $O((x+y)^{\frac{2C}{1+\varepsilon}}) \ll O((x+y^{\frac{1}{\varepsilon}})^C)$ (normally).

Consider the case where $x = y = \frac{b}{2}$, then:

$$\begin{aligned}\text{size effort}/\text{cost effort} &\approx \\ b^{\frac{2C}{1+\varepsilon}} / \left(x + y^{\frac{1}{\varepsilon}}\right)^C &< b^{\frac{2C}{1+\varepsilon}} / y^{\frac{C}{\varepsilon}}, \\ &< 2^{\frac{C}{\varepsilon}} / b^{\frac{C}{\varepsilon}\frac{1-\varepsilon}{1+\varepsilon}}, \\ &< \left(\frac{2}{b^{\frac{1-\varepsilon}{1+\varepsilon}}}\right)^{\frac{C}{\varepsilon}},\end{aligned}$$

and, provided $\varepsilon < \frac{1-\log_b 2}{1+\log_b 2}$ (for $b = 4$, $\varepsilon < \frac{1}{3}$), the last is always less than 1 and, for that matter, goes quickly to 0 as C increases and/or b increases and/or ε decreases.

Generalizing from this example, the size-based approach is faster at finding solutions of any given cost, as long as (1) high-cost actions constitute at least some constant fraction of the solutions considered (high-cost actions are relevant), (2) the ratio between high-cost and low-cost is sufficiently large, (3) the effective search graph (post inference) is reasonably well modeled by an infinite uniform branching tree (*i.e.*, huge enough to render duplicate checking negligible, or at least not especially favorable to cost-based search), and most importantly, (4) the cost function in the effective search graph *still* demonstrates a sufficiently large ratio between high-cost and low-cost edges (no inference has attempted to compensate).

4 Search Effort as Flooding Topological Surfaces of Evaluation Functions

We view evaluation functions (f) as topological surfaces over search nodes, so that generated nodes are visited in, roughly, order of f-altitude. With non-monotone evaluation functions, the set of nodes visited before a given node is all those contained within some basin of the appropriate depth — picture water flowing from the initial state: if there are dams then such a flood could temporarily visit high altitude nodes before low altitude nodes. (With very inconsistent heuristics — large heuristic weights — the metaphor loses explanatory power, as there is nowhere to go but downhill.)

All reasonable choices of search topology will eventually lead to identifying and proving the optimal solution (*e.g.*, assume finiteness, or divergence of f along infinite paths). Some will produce a whole slew of suboptimal solutions along the way, eventually reaching a point where one begins to wonder if the most recently reported solution is optimal. Others report nothing until finishing. The former are *interruptible* (Zilberstein 1998), which is one way to more formally define satisficing.[3] Admissible cost-based topology is the least interruptible choice: the only reported solution is also the last path considered. Define the cost-optimal footprint as the set of plans considered. Gaining interruptibility is a matter of raising the altitude of large portions of the cost-optimal footprint in exchange for lowering the altitude of a smaller set of non-footprint search nodes — allowing suboptimal solutions to be considered. Note that interruptibility comes at the expense of total work.

[3] Another way that Zilberstein suggests is to specify a *contract*; the 2008 and 2011 planning competitions have such a format (Helmert, Do, and Refanidis 2008).

So, along the lines of confirming the intuition that interruptibility is a reasonable notion of satisficing: the cost-optimal approach is a poor choice for satisficing solutions. Said another way, proving optimality is about increasing the lower bound on true value, while solution discovery is about decreasing the upper bound on true value. It seems appropriate to assume that the fastest way to decrease the upper bound is more or less the opposite of the fastest way to increase the lower bound — with the notable exception of the very last computation one will ever do for the problem: making the two bounds meet (proving optimality).

Intuitively the search should not be completely blind to costs: just defensive about possible ε-traps. For size-based topology, with respect to any cost-based variant, the 'large' set is the set of *longer* yet *cheaper* plans, while the 'small' set is the *shorter* yet *costlier* plans. In general one expects there to be many more longer plans than shorter plans in combinatorial problems, so that the increase in total work is small, relative to the work that had to be done eventually (exhaust the many long, cheap, plans). The additional work is considering exactly plans that are costlier than necessary (potentially suboptimal solutions). The idea of the trade-off is good, but even the best version of a purely size-based topology will not be the best trade-off possible.

Not finding the cheapest path first comes with the price of re-expansion, so the satisficing intent comes hand in hand with re-expansion of states. Indeed, duplicate detection and re-expansion are, in practice, important issues. Besides the obvious kind of re-expansion that IDA* (Korf 1985) performs between iterations, it is also true that it considers paths which A* never would (even subsequent to arming IDA* with a transposition table) — it is not really true that one can reorder consideration of paths however one pleases. In particular at least some kind of breadth-first bias is appropriate, so as to avoid finding woefully suboptimal plans to states early on, triggering giant cascades of re-expansion later on.

5 Handling ε-Cost Traps with Surrogate Search

We argue that an effective way to overcome the harmful effects of ε-cost traps is to replace the ill-behaved evaluation function that tracks the objective directly with a surrogate evaluation function that is well-behaved. Developing effective surrogate evaluation functions however involves navigating the tradeoff between defending against ε-cost traps and focusing search on the objective. We will consider two specific alternatives:

Pure Size-Based Evaluation Function: Here we replace cost-based evaluation function with a pure size-based one. That is, we search with f value being the size (number of actions) in the solution, even though we are interested in optimizing the cost of the solution. In particular, the heuristic will estimate the size of the shortest length path from the current state. By using size-based evaluation function, we effectively force ε to be 1. Since the search is not cost-focused, the first solution it finds may not necessarily have high quality. We handle this by employing a branch-and-bound search.

Cost-Sensitive Size-Based Evaluation Function: This is

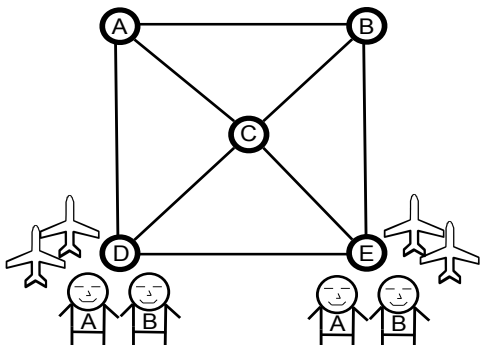

Figure 3: Rendezvous problems. Diagonal edges cost 7,000, exterior edges cost 10,000. Board/Debark cost 1.

an improvement of the pure-size based evaluation function that improves its quality focus while still retaining ε being 1. Specifically, the evaluation function here is still measured in terms of size or number of actions, but the heuristic aims to estimate the size of the cheapest *cost* path from the current state. By tracking the cheapest cost path, the evaluation function becomes more quality focused, and by measuring that path in terms of the number of actions (rather than the cumulative cost), it avoids the ε-cost trap. Once again, the quality of the solution can be improved with a branch-and-bound search regime. To elaborate further, the cost-sensitive size-based evaluation functions can use the standard relaxed-plan heuristics that select the relaxed plan itself in terms of the cost (Do and Kambhampati 2002). However, rather than take the the cost of the relaxed plan, we take the *size* of the relaxed plan, computing $\hat{h}_s$, and use $\hat{f}_s = g_s + \hat{h}_s$ for the evaluation function (see Section 2).

In the next section, we will see that the effect of ε-cost traps on cost-based evaluation functions is so enormous that even the pure size-based surrogate does better. The cost-sensitive size-based evaluation function improves further on this.

6 Empirical Evaluation

In this section we demonstrate existence of the problematic planner behavior in a realistic setting: running LAMA-2008 on problems in the travel domain (simplified ZenoTravel, zoom and fuel removed), as well as two other IPC domains. Analysis of LAMA is complicated by many factors, so we also test the behavior of SapaReplan on simpler instances (but in all of ZenoTravel). The first set of problems concern a rendezvous at the center city in the location graph depicted in Figure 3; the optimal plan arranges a rendezvous at the center city. The second set of problems is to swap the positions of passengers located at the endpoints of a chain of cities.

6.1 LAMA

In this section we demonstrate the performance problem wrought by ε-cost in a state-of-the-art (2008) planner — LAMA (Richter and Westphal 2010), the leader of the cost-sensitive (satisficing) track of IPC'08 (Helmert, Do, and Refanidis 2008). With a completely trivial recompilation (set a flag) one can make it ignore the given cost function, effectively searching by f_s. With slightly more work one can do better and have it use $\hat{f}_s$ as its evaluation function, *i.e.*, have the heuristic estimate $\hat{d}$ and the search be size-based, but still compute costs correctly for branch-and-bound. Call this latter modification LAMA-size. Ultimately, the observation is that LAMA-size outperforms LAMA — an astonishing feat for such a trivial change in implementation.

Domain	LAMA	LAMA-size
Rendezvous	70.8%	83.0%
Elevators	79.2%	93.6%
Woodworking	76.6%	64.1%

Table 1: IPC metric on LAMA variants.

LAMA[4] defies analysis in a number of ways: *landmarks*, *preferred operators*, *dynamic evaluation functions*, *multiple open lists*, and *delayed evaluation*, all of which effect potential search plateaus in complex ways. Nonetheless, it is essentially a cost-based approach.

Results.[5] With more than about 8 total passengers, LAMA is unable to complete any search stage except the first (the greedy search). For the same problems, LAMA-size finds the same first plan (the heuristic values differ, but not the structure), but is then subsequently able to complete further stages of search. In so doing it sees marked improvement in cost; on the larger problems this is due only to finding better variants on the greedy plan. Other domains are included for broader perspective, woodworking in particular was chosen as a likely counter-example, as all the actions concern just one type of physical object and the costs are not wildly different. For the same reasons we would expect LAMA to out-perform LAMA-size in some cost-enhanced version of Blocksworld.

6.2 SapaReplan

We also consider the behavior of SapaReplan on the simpler set of problems.[6] This planner is much less sophisticated in terms of its search than LAMA, in the sense that it does not use dual queues or lazy evaluation. The problem is just to swap the locations of passengers located on either side of a chain of cities. A plane starts on each side, but there is no actual advantage to using more than one (for optimizing either of size or cost): the second plane exists to confuse the planner. Observe that smallest and cheapest plans are the same. So in some sense the concepts have become only superficially different; but this is just what makes the problem interesting, as despite this similarity, still the behavior of search is strongly affected by the nature of the evaluation function. We test the performance of $\hat{f}_s$ and f_c, as well as a hybrid evaluation function similar to $\hat{f}_s + f_c$ (with costs normalized). We also test hybridizing via tie-breaking conditions, which ought to have little effect given the rest of the search framework.

[4]Options: 'fFlLi'.

[5]New best plans for Elevators were found (largely by LAMA-size). The baseline planner's score is 71.8% against the better reference plans.

[6]Except that these problems are run on all of ZenoTravel.

	2 Cities		3 Cities	
Mode	Score	Rank	Score	Rank
Hybrid	88.8%	1	43.1%	2
Size	83.4%	2	43.7%	1
Size, tie-break on cost	82.1%	3	43.1%	2
Cost, tie-break on size	77.8%	4	33.3%	3
Cost	77.8%	4	33.3%	3

Table 2: IPC metric on SapaReplan variants in ZenoTravel.

Results.[7] The size-based evaluation functions find better cost plans faster (within the deadline) than cost-based evaluation functions. The hybrid evaluation function also does relatively well, but not as well as could be hoped. Tie-breaking has little effect, sometimes negative.

We note that Richter and Westphal (2010) also report that replacing cost-based evaluation function with a pure size-based one improves performance over LAMA in multiple other domains. Our version of LAMA-size uses a cost-sensitive size-based search ($\hat{h}_s$), and our results, in the domains we investigated, seem to show bigger improvements over the size-based variation on LAMA obtained by *completely* ignoring costs (h_s, *i.e.*, setting the compilation flag). Also observe that one need not accept a tradeoff: calculating $\log_{10} \varepsilon^{-1} \leq 2$ (3? 1.5?) and choosing between LAMA and LAMA-size appropriately would be an easy way to improve performance simultaneously in ZenoTravel (4 orders of magnitude) and Woodworking (< 2 orders of magnitude).

Finally, while LAMA-size outperforms LAMA, our theory of ε-cost traps suggests that cost-based search should fail even more spectacularly. In the extended version of this paper, we take a much closer look at the travel domain and present a detailed study of which extensions of LAMA help it temporarily mask the pernicious effects of cost-based search. Our conclusion is that both LAMA and SapaReplan manage to find solutions to problems in the travel domain despite the use of a cost-based evaluation function by using various tricks to induce a limited amount of *depth-first behavior* in an A*-framework. This has the potential effect of delaying exploration of the ε-cost plateaus slightly, past the discovery of a solution, but still each planner is ultimately trapped by such plateaus before being able to find really good solutions. In other words, such tricks are mostly serving to mask the problems of cost-based search (and ε-cost), as they merely delay failure by just enough that one can imagine that the planner is now effective (because it returns a solution where before it returned none). Using a size-based evaluation function more directly addresses the existence of cost plateaus, and not surprisingly leads to improvement over the equivalent cost-based approach — even with LAMA.

[7]The results differ markedly between the 2 and 3 city sets of problems because the sub-optimal relaxed plan extraction in the 2-cities problems coincidentally produces an essentially perfect heuristic in many of them. One should infer that the solutions found in the 2-cities problems are sharply bimodal in quality and that the meaning of the average is then significantly different than in the 3-cities problems.

7 Related Work

Others have noted issues with using cost-based search. For instance, Thayer and Ruml (2010) have considered the issues of blending size and cost in the design of evaluation functions, and earlier work in evaluation function design beyond just simplistic heuristic weighting as done by Pohl (1973). Dechter and Pearl (1985) give a highly technical account of the properties of generalized best-first search strategies, focusing on issues of computational optimality, but, mostly from the perspective of search constrained to proving optimality in the path metric. Additionally, Richter and Westphal (2010) have a thorough empirical analysis of cost issues in standard planning benchmarks in their planner LAMA. Subsequently LAMA was modified to include a surrogate, size-based greedy best-first search search to find a first solution before beginning the same search approach as its previous version (Richter, Westphal, and Helmert 2011).

As already mentioned in Section 1, our work is also related to the phenomenon of g-value plateaus (c.f. (Benton et al. 2010)), in that ε-cost traps can occur whenever the objective function is "ill-behaved" in that it doesn't increase at a reasonable rate w.r.t. search depth. Indeed, the solutions advocated for g-value plateaus can be interpreted in terms of surrogate evaluation functions. As an example, Benton, et al. (2011) recently suggested a way of defending against ε-traps by using surrogate evaluation functions along side an evaluation function known to suffer from g-value plateaus (i.e., makespan).

8 Conclusion

Several researchers have noted the troublesome behavior of cost-based search in many planning benchmarks, suggesting several *ad hoc* work arounds for it. In this paper, we tried to provide a general explanation for the malady. Specifically, we argued that the origins of this problem can be traced back to the fact that most planners that try to optimize cost also use cost-based evaluation functions (i.e., $f(n)$ is a cost estimate). We showed that cost-based evaluation functions become ill-behaved whenever there is a wide variance in action costs; something that is all too common in planning domains. The general solution to this malady is what we call a *surrogate search*, where a surrogate evaluation function that doesn't directly track the cost objective, and is resistant to cost-variance, is used. We discussed some compelling choices for surrogate evaluation functions that are based on size rather than cost, and showed that they are immune to the difficulties faced by cost-based search. Of particular practical interest is a cost-sensitive version of size-based evaluation function where the heuristic estimates the size of cheap paths, as it provides attractive quality vs. speed tradeoffs. Our empirical evaluations with a state-of-the-art planner demonstrate the effectiveness of surrogate search. A worthwhile direction for future work is finding other surrogate evaluation functions that strike an even better balance between resistance to ε-cost traps, and focus on the objective.

References

Benton, J.; Talamadupula, K.; Eyerich, P.; Mattmüller, R.; and Kambhampati, S. 2010. G-value plateaus: A challenge for planning. In *ICAPS*. AAAI Press.

Benton, J.; Eyerich, P.; and Kambhampati, S. 2011. Enhancing search for satisficing temporal planning with objective-driven decisions. In *Workshop on Heuristics for Domain-Independent Planning*. AAAI Press.

Dechter, R., and Pearl, J. 1985. Generalized best-first search strategies and the optimality of A*. *ACM* 32(3):505–536.

Dijkstra, E. W. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik* 1:269–271.

Do, M. B., and Kambhampati, S. 2002. Planning graph-based heuristics for cost-sensitive temporal planning. In *AIPS*, 3–12.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics* 4:100–107.

Helmert, M., and Röger, G. 2008. How good is almost perfect. In *AAAI*, 944–949.

Helmert, M.; Do, M.; and Refanidis, I. 2008. The deterministic track of the international planning competition. `http://ipc.informatik.uni-freiburg.de/`.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *JAIR* 14:253–302.

Korf, R. E. 1985. Depth-first iterative-deepening: An optimal admissible tree search. *AIJ* 27:97–109.

Pearl, J. 1984. *Heuristics: intelligent search strategies for computer problem solving*. Addison-Wesley.

Pohl, I. 1973. The avoidance of (relative) catastrophe, heuristic competence, genuine dynamic weighting and computational issues in heuristic problem solving. In *IJCAI*.

Richter, S., and Westphal, M. 2010. The LAMA planner: Guiding cost-based anytime planning with landmarks. *JAIR* 39:127–177.

Richter, S.; Westphal, M.; and Helmert, M. 2011. Lama 2008 and 2011. In *International Planning Competition 2011*, 51–54. AAAI Press.

Thayer, J., and Ruml, W. 2010. Finding acceptable solutions faster using inadmissible information. In *Proceedings of the International Symposium on Combinatorial Search*.

Zilberstein, S. 1998. Satisficing and bounded optimality. In *AAAI Spring Symposium on Satisficing Models*.