Tie-Breaking in A* as Satisficing Search

Masataro Asai, Alex Fukunaga
Graduate School of Arts and Sciences
University of Tokyo

Abstract

Best-first search algorithms such as A* need to apply tie-breaking strategies in order to decide which node to expand when multiple search nodes have the same evaluation score. Recently, these tiebreaking strategies were shown to have significant impact on the performance of A* especially on domains with 0-cost actions, and a new method was proposed. In this paper, we propose a framework for interpreting A* search as a series of satisficing searches within plateaus consisting of nodes with the same f-cost. This new framework motivates a new class of tie-breaking strategy, a multi-heuristic tie-breaking strategy which embeds inadmissible, distance-to-go variations of various heuristics within an admissible search. This is shown to further improve the performance in combination with the depth metric proposed in the previous work.

1 Introduction

In this paper, we investigate *tie-breaking strategies* for cost-optimal A^*. A^* is a standard search algorithm for finding an optimal cost path from an initial state s to some goal state $g \in G$ in a search space represented as a graph (Hart, Nilsson, and Raphael 1968). It expands the nodes in best-first order of $f(n)$ up to f^*, where $f(n)$ is a lower bound of the cost of the shortest path that contains a node n and f^* is the cost of the optimal path. In many combinatorial search problems, the size of the last layer $f(n) = f^*$ of the search, called a *final plateau*, accounts for a significant fraction of the effective search space of A^*. Figure 1 (p.1) compares the number of states in this final plateau with $f(n) = f^*$ (y-axis) vs. $f(n) \leq f^*$ (x-axis) for 1104 problem instances from the International Planning Competition (IPC1998-2011). For many instances, a large fraction of the nodes in the effective search space have $f(n) = f^*$: The points are located very close to the diagonal line ($x = y$), indicating that almost all states with $f(n) \leq f^*$ have cost f^*.

Figure 2 depicts this phenomenon conceptually. On the left, we show one natural view of the search space that considers the space searched by A^* as a large number of closed nodes with $f < f^*$, surrounded by a thin layer of final plateau $f(n) = f^*$. This intuitive view accurately reflects the search spaces of some real-world problems such as 2D pathfinding on an explicit graph.

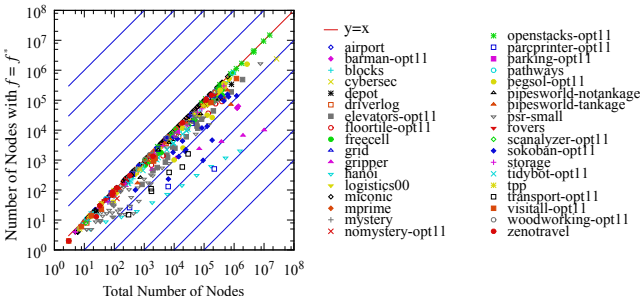

Figure 1: The number of nodes with $f = f^*$ (y-axis) compared to the total number of nodes in the search space (x-axis) with $f \leq f^*$ on 1104 IPC benchmark problems. This experiment uses a modified Fast Downward with LMcut which continues the search within the current f after any cost-optimal solution is found. This effectively generates all nodes with cost f^*.

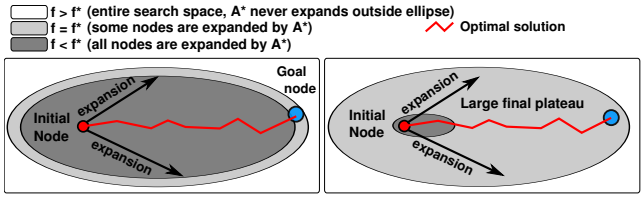

Figure 2: (Left) One possible class of search space which is dominated by the states with cost $f < f^*$. (Right) This paper focuses on another class of search space, where the plateau containing the cost-optimal goals ($f = f^*$) is large, and it even accounts for most of the search effort required by A^*.

However, for many other classes of combinatorial search problems, e.g., the IPC Planning Competition Benchmarks, the figure on the right is a more accurate depiction – here, the search space has a large plateau for $f = f^*$. Classical planning problems in the IPC benchmark set are clearly the instances of such combinatorial search problems.

For the majority of such IPC problem domains where the last layer ($f(n) = f^*$) accounts for a significant fraction of the effective search space, a *tie-breaking strategy*, which determines which node to expand among nodes with the same f-cost, can have a significant impact on the performance of A^*. It is widely believed that among nodes with the same f-cost, ties should be broken according to $h(n)$, i.e., nodes with smaller h-values should be expanded first. While this is a useful rule of thumb in many domains, it turns out that tie-breaking requires more careful consideration, particularly for problems where most or all of the nodes in the last layer have the same h-value.

In this paper, we provide an alternative view to the tie-breaking behavior of A*. More specifically, cost-optimal search using A^* can be considered as a series of satisficing searches on each plateau. This allows the problem of tie-breaking to be reduced to satisficing search within a plateau (Section 3), opening a wide variety of future work.

Based on this insight, we then investigate an admissible tie-breaking strategy which uses an inadmissible distance-to-go estimate, a heuristic function which treats every action to have the unit costs (Section 4), for tie-breaking. Although distance-to-go estimates are inadmissible, it does not compromise the admissibility of A^* as long as it is used only for tie-breaking.

[This paper presents work from Sections 7-8 from a recent journal paper (Asai and Fukunaga 2017). This work has not been previously presented in any conference or workshop.]

2 Preliminaries

We first define some notation and the terminology used throughout the rest of the paper. $h(n)$ denotes the estimate of the cost from the current node n to the nearest goal node. $g(n)$ is the current shortest path cost from the initial node to the current node. $f(n) = g(n) + h(n)$ is the estimate of the resulting cost of the path to a goal containing the current node. We omit the argument (n) unless necessary. h^*, g^* and f^* denotes the true optimal cost from n to a goal, from the start to n, or from the start to a goal through n, respectively.

A *sorting strategy* for a best first search algorithm tries to select a single node from the open list (OPEN). Each sorting strategy is denoted as a vector of several *sorting criteria*, such as [criterion$_1$, criterion$_2$, . . ., criterion$_k$], which means: First, select a set of nodes from OPEN using criterion$_1$. If there are still multiple nodes remaining in the set, then break ties using criterion$_2$ and so on, until a single node is selected. The *first-level sorting criterion* of a strategy is criterion$_1$, the *second-level sorting criterion* is criterion$_2$, and so on.

Using this notation, A^* without any tie-breaking can be denoted as $[f]$, and A^* which breaks ties according to h value is denoted as $[f, h]$. Similarly, GBFS is denoted as $[h]$. Unless stated otherwise, we assume the nodes are sorted in increasing order of the key value, and BFS always selects a node with the smallest key value.

A sorting strategy fails to select a single node when some nodes share the same sorting keys. In such cases, a search algorithm must select a node according to a *default* tie-breaking criterion, criterion$_k$, such as *fifo* (first-in-first-out), *lifo* (last-in-first-out) or *ro* (random ordering). For example, an A^* using h and *fifo* tie-breaking is denoted as $[f, h,$*fifo*$]$. By definition, default criteria are guaranteed to return a single node from a set of nodes. When the default criterion does not matter, we may use a wildcard $*$ as in $[f, h, *]$.

Given a search algorithm with a sorting strategy, a *plateau* (criterion . . .) is a set of nodes in OPEN whose elements share the same sort keys according to non-default sorting criteria and therefore are indistinguishable. In a case of A^* using tie-breaking with h (sorting strategy $[f, h, *]$), the plateaus are denoted as *plateau* (f, h), the set of nodes with the same f cost and the same h cost. We can also refer to a specific plateau with $f = f_p$ and $h = h_p$ by *plateau* (f_p, h_p).

Recently, Asai and Fukunaga proposed a Random Depth tiebreaking (2016) and its deterministic version (2017), resulting in significant performance improvements in a new set of benchmark domains called *Zerocost* domains [1].

Random Depth tiebreaking and its deterministic version diversify the search within each plateau using the depth metric $d(n)$, a distance from the current node n to the nearest ancestor that has the different f-value and the h-value. Each node in a h-plateau is stored into a bucket of the corresponding depth d, and the expansion happens on a node in a bucket that is selected at random, or in a round-robin manner (deterministic version). Such a configuration of A^* is denoted as $[f, h, \langle d \rangle, *]$.

Zerocost domains are the modified version of the standard IPC domains which characterizes the more practical cost-minimization problems where the most important actions directly related to resource usage incur the non-zero costs. We use this Zerocost domains for evaluation throughout the paper.

3 A^* as a Series of Satisficing Search

While A^* requires the first sorting criterion f to use an admissible heuristic in order to find an optimal solution, there are no requirements on the second or later sorting criterion. This means that the search within the same f plateau can be an arbitrary satisficing search without any cost minimization requirement (as opposed to the "satisficing" track setting in IPC which also seeks to minimize the plan cost with anytime algorithms). For example, if we ignore the first sorting criterion in the standard admissible strategy $[f, h,$*fifo*$]$, we have $[h,$*fifo*$]$, which is exactly the same configuration as a Greedy Best First Search (GBFS) using *fifo* default tie-breaking. This means that within a particular f-cost plateau, $[f, h,$*fifo*$]$ is performing a satisficing GBFS. As another example, the reason for the poor performance of $[f,$*fifo*$]$ is clearly that it is

[1] github.com/guicho271828/zerocost-opt

running [*fifo*], an uninformed satisficing breadth-first search in the plateau.

From this perspective, we can reinterpret A^* as in Algorithm 1: A^* expands the nodes in best-first order of f value. When the heuristic function is admissible, the f values of the nodes expanded by A^* never decreases during the search process. Thus, the entire process of A^* can be considered as a series of search episodes on each *plateau* (f). The search on each plateau terminates when the plateau is proven to contain no goal nodes (UNSAT), or when a goal is found (SAT). When the plateau is UNSAT, then the search continues to the plateau with the next smallest f value. Figure 3 also illustrates this framework.

Algorithm 1 Reinterpretation of A^* as iterations of satisficing search on plateaus

loop
  Search *plateau* (f) for any goal state, using satisficing search algorithm
  if *plateau* (f) contains some goal (Plateau is SAT) **then**
    return solution
  else
    Increase f

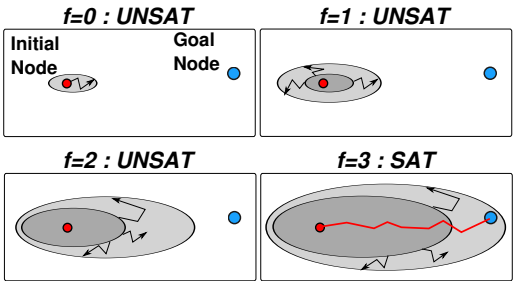

Figure 3: The concept of A^* as a sequence of satisficing searches.

This is somewhat similar to the standard approach to model-based planning using SAT/IP/CP solvers (Kautz and Selman 1992; van den Briel and Kambhampati 2005), based on an iterative strategy where a planning problem is converted to a corresponding constraint satisfaction problem with a finite horizon t (plan length / makespan). The search starts from the horizon 0 and tests if the problem is satisfiable. If not, then it increases the horizon, add constraints excluding solutions below t, and retests the same problem with additional constraints for a new horizon $t + 1$.

It is also reminiscent of the behavior of iterative deepening A^* (Korf 1985), which executes a series of satisficing searches with an f-cost limit which increases on each iteration. However, "A^*-as a sequence of satisficing search" differs from IDA* in that IDA*, in order to achieve linear memory usage, repeats previous work on each iteration. Instead of searching a particular plateau in each iteration, IDA* searches through the union of several plateaus.

The framework of "A^* as a series of satisficing searches" suggests that we can directly apply satisficing search techniques to optimal search using A^*, especially for each f-cost plateau search. In the following sections, we show that this framework (1) provides a better understanding of depth-diversification (Section 3.1) and (2) allows us to improve the performance of A^* on Zerocost domains (Section 4).

3.1 Depth Diversification and Satisficing Search

Within this framework, the implementation of depth diversification can be viewed as a variant of the Type-based diversification approach (Xie et al. 2014), specifically tailored for Zerocost domains.

Xie et al. proposed *type based buckets*, an implementation of the OPEN list which partitions the nodes into buckets according to some set of key values (*types*). They proposed several types such as $\langle 1 \rangle$, $\langle g \rangle$, $\langle h \rangle$ or $\langle g, h \rangle$. At each type-based expansion, a randomly selected node from a randomly selected single bucket is selected. For example, with type $\langle g, h \rangle$, a node with $g = 5$ and $h = 3$ is put into a bucket $\langle 5, 3 \rangle$. This mechanism diversifies the search so that it removes the cardinality bias in terms of the distance of a node from the initial state or the goal states. They then proposed Type-GBFS, which alternates the expansion between a normal GBFS $[h, \textit{fifo}]$ and the type-based expansion $[\langle g, h \rangle, \textit{ro}]$.

In our framework of A^* as a sequence of satisficing searches, depth diversification after h tie-breaking $([f, h, \langle d \rangle, *])$ can be viewed as the combination of (1) an implicit transformation of all 0-cost edges within a single *plateau* (f, h) to unit-cost edges, and (2) a pure type-based exploration within that plateau (unlike Type-GBFS, which alternates GBFS and type-based buckets).

The notion of *depth* counts the number of 0-cost actions, which does not change the f value and h value, on the path from the entrance to the current plateau, to the current node. Thus, depth-diversification treats the problem of finding an exit from a particular plateau as a unit-cost satisficing search problem – the depth is analogous to a g-value which is calculated with unit costs and is restricted to a particular plateau.

4 Tie-Breaking with Distance-to-Go Estimates

In the previous section, we proposed a framework which views cost-optimal A^* search as a series of satisficing searches on each f-cost plateau, and argued that the problem of tie-breaking can be reduced to a satisficing search. We showed that the depth diversification tie-breaking criterion, which is highly effective on Zerocost domains, is in fact a case where a previously studied technique for satisficing search (type-based exploration) turns out to be highly effective when applied to tie-breaking. In this section, we push this insight further and propose another approach to improving the search performance in plateaus produced by Zerocost domains – using inadmissible *distance-to-go* estimates (heuristics) as a tie-breaking criterion within an admissible A^* search.

Distance-to-go estimates are a class of heuristics which treat all actions as if they have unit cost. Even when 0-cost actions are present, these estimates can predict the number

of operations required to reach a goal. In general, the estimates are inadmissible (unless the estimates are guaranteed to underestimate the number of required actions and all actions in the original domain have unit cost). Previous work on distance-to-go-heuristics has focused on their use for satisficing planning.

A^*_ϵ (Pearl and Kim 1982) is one of the earliest algorithms that combines distance-to-go estimates with the cost estimates. It is a bounded-suboptimal search which expands nodes from the *focal* list, the set of nodes with $f(n) \leq w \cdot f_{min}$ where weight w serves as a suboptimality bound, similar to weighted A^*, and f_{min} is the minimum f value in the OPEN list. While f is based on an admissible heuristic function, the nodes in the focal list are expanded in increasing order of an inadmissible distance-to-go estimate $\hat{h}$. Since the search does not follow the best-first order according to f, it is not admissible, and is instead w-admissible. One exception is the case of $w = 1$ where the focal list is equivalent to the f plateau and the expansion order in the focal list corresponds to the tie-breaking on plateaus. In our notation, this algorithm can be written[2] as a BFS with the following sorting criteria:

$$[\lceil \frac{f}{w \cdot f_{min}} \rceil, \hat{h}, *]$$

This notation is derived from the fact that the focal list "blur"s f up to $w \cdot f_{min}$. For example, when $w = 2, f_{min} = 5$ and $f(n) = 5, 9, 11$, then $\lceil \frac{f}{w \cdot f_{min}} \rceil = 1, 1, 2$ respectively.

Continuing this line of work, Thayer and Ruml (2009; 2011) evaluated various distance-to-go configurations of Weighted A^*, Dynamically Weighted A^* (Pohl 1973) and A^*_ϵ, where some configurations use distance-to-go as part of tie-breaking. This work focused on bounded-suboptimal search rather than cost-optimal search. Cushing, Benton, and Kambhampati (2010) pointed out the danger of relying on cost estimates in a satisficing search by investigating "ε-cost traps" and other pitfalls caused by cost estimators for search guidance. Finally, the FD/LAMA2011 satisficing planner incorporates distance-to-go estimates in its iterated search framework (Richter, Westphal, and Helmert 2011). The first iteration of LAMA uses distance-to-go estimates combined with various satisficing search enhancements.

Benton et al. (2010) proposed an inadmissible technique for temporal planning where short actions are hidden behind long actions and do not increase makespan. Such actions cause "g-value plateaus", which are similar to the large plateaus caused by 0-cost actions in sequential planning. They implemented an inadmissible heuristic function combined with distance-to-go estimates as an extension of Temporal Fast Downward (Eyerich, Mattmüller, and Röger 2009).

4.1 Embedding Distance-to-Go Estimates in Admissible Search

Although previous work on distance-to-go estimates assume a satisficing context, we show that distance-to-go estimates can be useful for cost-optimal search. Since the admissibility of the sorting strategy and the optimality of the solution are not affected by the second or later levels of sorting criteria, it is possible to use an inadmissible distance-to-go estimate in these subsequent sorting criteria without sacrificing the optimality of the solution found. This means inadmissible heuristics can be used for tie-breaking.

Let h be an admissible heuristic function, and $\hat{h}$ be a distance-to-go variation of h, i.e., $\hat{h}$ uses essentially the same algorithm as h, except that while h uses the actual action costs for the problem domain, $\hat{h}$ replaces all action costs with 1. Since h is admissible, multi-heuristic sorting strategies such as $[g + h, h, \hat{h}]$ or $[g + h, \hat{h}]$ are admissible.

Moreover, we can even use a multi-heuristic strategy which uses an inadmissible heuristic for tie-breaking which is unrelated to the primary, admissible heuristic h. For example, $[g + h^{\text{LMcut}}, \hat{h}^{\text{FF}}]$ is an admissible sorting strategy because the first sorting criterion $f = g + h^{\text{LMcut}}$ uses an admissible LMcut heuristic. Its second sorting criterion, the distance-to-go FF heuristic (Hoffmann and Nebel 2001), does not affect the admissibility of this entire sorting strategy.

A potential problem with sorting strategies which use multiple heuristics is the cost of computing additional heuristic estimates. For example, $[g + h^{\text{LMcut}}, \hat{h}^{\text{FF}}]$ requires more time to evaluate each node compared to a standard tie-breaking strategy such as $[g + h^{\text{LMcut}}, h^{\text{LMcut}}]$ because computing the $\hat{h}^{\text{FF}}$ heuristic incurs significant overhead per node while the results of h^{LMcut} can be reused by a caching mechanism. When the inadmissible heuristic for tie-breaking is $\hat{h}$, i.e. a distance-to-go (unit cost) variant of the primary, admissible heuristic h, it may be possible to reduce this overhead to some extent by implementing h and $\hat{h}$ so that they share some of the computation – this is a direction for future work.

Combining Distance-to-Go Estimates with Default Tie-Breaking and Depth Diversification Tie-breaking using distance-to-go estimates can still leave a set of nodes which are equivalent up to the distance-to-go criterion (multiple nodes can have the same f, h, and $\hat{h}$ values), so additional level(s) of tie-breaking are necessary in order to select a single node. By adding a standard default criterion such as *fifo*, *lifo*, *ro*, we obtain a sorting strategy that imposes a total order. For example, $[f^{\text{LMcut}}, \hat{h}^{\text{FF}}, \textit{fifo}]$ applies *fifo* after the distance-to-go estimate $\hat{h}^{\text{FF}}$.

Furthermore, it is possible to combine depth diversity based tie-breaking with distance-to-go estimates by applying the depth-diversity criterion after the distance-to-go estimate. For example, $[f^{\text{LMcut}}, \hat{h}^{\text{FF}}, \langle d \rangle, \textit{fifo}]$ applies depth diversification criterion after the $\hat{h}^{\text{FF}}$ distance-to-go estimate. As we shall see below, a sorting strategy which performs tie-breaking using both distance-to-go estimates and depth diversity results in the best performance overall.

[2] However, an actual implementation may differ due to dynamic updates to f_{min}.

4.2 Evaluation of Distance-to-Go Estimates as Tie-Breaking Criteria for Admissible Search

We tested various admissible sorting strategies on IPC domains and Zerocost domains. In all configurations, the first sorting criterion is the $f = g + h$ value where h is an admissible heuristic (either LMcut or M&S) using the actual action-cost based cost calculation. As the second (and third) criteria, we used $\hat{h}$, the distance-to-go version of the original heuristic function h, as well as a distance-to-go variation of FF heuristic ($\hat{h}^{\text{FF}}$). We also added configurations with the depth metric within *plateau* $\left(f, \hat{h}^{\text{FF}}\right)$. Detailed per-domain results are shown in Table 1.

Evaluation on Zerocost Domains In Zerocost domains, we see that $\hat{h}$ tie-breaking outperforms h tie-breaking for both LMcut (e.g. 256 $\rightarrow$ 295 with *fifo*) and M&S (e.g. 280 $\rightarrow$ 308 with *fifo*). Also, combining h and $\hat{h}$ can further improve performance when the heuristic is LMcut (e.g. 295 $\rightarrow$ 305 with *fifo*). The results of combining h and $\hat{h}$ were comparable to $\hat{h}$ when the main heuristic function h is M&S. Yet more surprisingly, using $\hat{h}^{\text{FF}}$ further improved the performance for both LMcut (e.g. $[f, h, \hat{h}, \textit{fifo}] : 305 \rightarrow [f, \hat{h}^{\text{FF}}, \textit{fifo}] : 337$) and M&S (e.g. $[f, h, \hat{h}, \textit{fifo}] : 307 \rightarrow [f, \hat{h}^{\text{FF}}, \textit{fifo}] : 336$). Thus, when the depth diversity criterion is not used, the best configurations are those which use $\hat{h}^{\text{FF}}$.

The reason for the good performance of $[f^{\text{LMcut}}, \hat{h}^{\text{FF}}, *]$ is not surprising: $\hat{h}^{\text{FF}}$ is by itself known to be a powerful inadmissible heuristic function for satisficing GBFS, and if we ignore the first sorting criterion, $[f^{\text{LMcut}}, \hat{h}^{\text{FF}}, *]$ is a GBFS with $[\hat{h}^{\text{FF}}, *]$.

Adding the depth diversity criterion further improves the performance of the $\hat{h}^{\text{FF}}$-based strategies, although the impact was small. The coverage increased in both $h = h^{\text{LMcut}}$ (*fifo*: 337 $\rightarrow$ 340, *lifo*: 340 $\rightarrow$ 342, *ro*: 341 $\rightarrow$ 344.3) and $h = h^{\text{M\&S}}$ (*fifo*: 336 $\rightarrow$ 337, *lifo*: 331 $\rightarrow$ 333). When the default tie-breaking was *ro* and the heuristic is M&S, $[f, \hat{h}^{\text{FF}}, \langle d \rangle, \textit{ro}]$ performed slightly worse than $[f, \hat{h}^{\text{FF}}, \textit{ro}]$, but the difference was very small (337.9 $\rightarrow$ 337.6) and $\langle d \rangle$ made the performance slightly more robust (smaller standard deviation: 2.1 $\rightarrow$ 1.3).

Evaluation on Standard IPC Domains For the standard IPC benchmark instances, the overhead due to the additional computation of $\hat{h}$ or $\hat{h}^{\text{FF}}$ tends to harm the overall performance. Therefore, the best configuration using LMcut was $[f, h, \langle d \rangle, \textit{lifo}]$ which uses depth and does not impose the cost of additional heuristics, and the best result using M&S was $[f, h, \textit{lifo}]$ which imposes no overhead including the depth.

Delving into the detailed results, we observed the following: In Cybersec, distance-to-go variants (e.g. $[f^{\text{LMcut}}, \hat{h}^{\text{FF}}, \textit{lifo}]$:5) improve upon the standard strategy (e.g. $[f^{\text{LMcut}}, h^{\text{LMcut}}, \textit{lifo}]$:3), but does not improve upon depth (e.g. $[f, h, \langle d \rangle, \textit{lifo}]$: 12). When $h = h^{\text{M\&S}}$, all coverages are zero. Overheads by $\hat{h}^{\text{FF}}$ also slightly degrade the performance in Openstacks (e.g. $[f^{\text{LMcut}}, h^{\text{LMcut}}, \textit{lifo}]$:18, $[f^{\text{LMcut}}, \hat{h}^{\text{FF}}, \textit{lifo}]$:17, $[f^{\text{LMcut}}, h^{\text{LMcut}}, \langle d \rangle, \textit{lifo}]$: 18; Also, $[f^{\text{M\&S}}, h^{\text{M\&S}}, \textit{lifo}]$:19, $[f^{\text{M\&S}}, \hat{h}^{\text{FF}}, \textit{lifo}]$:18, $[f^{\text{M\&S}}, h^{\text{M\&S}}, \langle d \rangle, \textit{lifo}]$: 19). Thus, in these two domains, although there are some improvements in search efficiency due to the guidance by $\hat{h}^{\text{FF}}$ or $\hat{h}$, the runtime overhead of computing the distance-to-go heuristics outweighed the benefit.

In the domains with only positive cost actions (all IPC domains except Openstacks and Cybersec), $\hat{h}$ or $\hat{h}^{\text{FF}}$ only harm the overall performance due to the overhead. When the primary heuristics is LMcut, we do not observe a significant difference between single-heuristics strategies except for the fractional difference in the configurations using *ro*. When the primary heuristic is M&S, $[f^{\text{M\&S}}, h^{\text{M\&S}}, \textit{lifo}]$ performs slightly better than other default tie-breaking strategies; It also outperforms the depth-based variants.

4.3 Simple Dynamic Configuration for Overall Performance

In practice, the performance degradation when using multi-heuristic strategy in domains with only positive cost actions does not pose a problem. We can easily avoid the overhead incurred by the distance-to-go heuristics in those domains by applying the following simple policy: If there are any 0-cost actions, use a multi-heuristic strategy; Otherwise, use a single-heuristic strategy.

Since the impact of such a check on the total runtime is negligible, we can extrapolate the result of applying this rule based on the previously obtained results. Coverage results in Table 2 show the total coverage of Zerocost and IPC benchmark domains. The bottom two rows, labeled as *dynamic configuration*, are the extrapolated results when the switching policy is applied – this dynamic configuration achieves the highest overall coverage.

When the configuration rule is applied to standard IPC instances, the domains with 0-cost actions are Cybersec and Openstacks only. They are solved using a multi-heuristic strategy while other domains are solved in the best performing single-heuristic strategy. In Zerocost instances, all domains are solved using the multi-heuristic strategy.

We only tested this relatively simple dynamic configuration that switches between two strategies based on the presence of 0-cost operators. However, domain-specific solvers (as opposed to domain-independent solvers, which are the main focus of this paper) can benefit from fine-tuning the tiebreaking strategy so that it is most suited to the target domain.

5 Related Work

Previous work on escaping search space plateaus has focused on non-admissible search. DBFS (Imai and Kishimoto 2011) adds stochastic backtracking to Greedy Best First Search (GBFS) to avoid being misdirected by the heuristic function. Type based buckets (Xie et al. 2014) classify plateaus in GBFS according to the $[g, h]$ pair and distributes

$h =$ LMcut	$[f,h,$*ffifo*$]$	$[f,h,$*lifo*$]$	$[f,h,ro]$	$[f,h,\langle d\rangle,$*ffifo*$]$	$[f,h,\langle d\rangle,$*lifo*$]$	$[f,h,\langle d\rangle,ro]$	$[f,\hat{h},$*ffifo*$]$	$[f,\hat{h},$*lifo*$]$	$[f,\hat{h},ro]$	$[f,h,\hat{h},$*ffifo*$]$	$[f,h,\hat{h},$*lifo*$]$	$[f,h,\hat{h},ro]$	$[f,\hat{h}^{\text{FF}},$*ffifo*$]$	$[f,\hat{h}^{\text{FF}},$*lifo*$]$	$[f,\hat{h}^{\text{FF}},ro]$	$[f,\hat{h}^{\text{FF}},\langle d\rangle,$*ffifo*$]$	$[f,\hat{h}^{\text{FF}},\langle d\rangle,$*lifo*$]$	$[f,\hat{h}^{\text{FF}},\langle d\rangle,ro]$
Zerocost (620)	256	279	261.9	284	264	288.1	295	303	301.0	305	309	305.9	337	340	341	340	342	**344.3**
airport-fuel(20)	**15**	13	13.8	14	13	14	13	12	12.7	14	12	12.8	13	11	11.7	13	11	11.7
blocks-stack(20)	**17**	**17**	**17**	**17**	**17**	**17**	15	15	15.0	15	15	15	**17**	**17**	**17**	**17**	**17**	**17**
elevators-up(20)	7	13	7	7	9	9.1	**20**	**20**	19.9	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**
freecell-move(20)	4	**19**	4.9	17	10	16.4	12	14	13.3	12	14	13.2	17	18	17.9	17	18	18.3
miconic-up(30)	16	17	16.6	19	18	20.3	14	17	15.1	14	17	15.1	15	**21**	17.9	15	**21**	18
mprime-succumb(35)	15	14	17.1	22	14	20.1	19	16	19.1	20	16	20.1	**30**	23	28.3	**30**	27	29.3
mystery-feast(20)	7	5	7.7	6	5	7.2	7	6	6.9	6	5	5.9	**8**	**8**	**8**	**8**	**8**	**8**
parking-movecc(20)	0	0	0	0	0	0	13	14	14.3	13	15	14.4	**20**	**20**	**20**	**20**	**20**	**20**
pipesnt-pushstart(20)	8	8	8.4	8	8	**9.8**	7	8	7.7	8	8	7.8	9	9	9	9	9	9
pipesworld-pushend(20)	3	4	3.8	3	3	4.8	5	6	5.1	5	5	5	7	**8**	7.1	7	7	7.7
scanalyzer-analyze(20)	9	9	9.1	9	10	9.2	8	11	10.1	16	**18**	15.3	15	15	15	15	15	15
sokoban-pushgoal(20)	**18**	**18**	**18**	**18**	**18**	**18**	16	16	16.0	16	16	16	17	17	17	17	17	17
tidybot-motion(20)	**16**	**16**	**16**	**16**	**16**	**16**	14	14	14.0	14	14	14	15	**16**	**16**	**16**	**16**	15.9
tpp-fuel(30)	8	**11**	8	**11**	10	**11**	8	10	8.7	8	10	8.2	8	10	9.1	10	10	10
woodworking-cut(20)	5	7	7	8	5	8.2	**20**	**20**	**20.0**	**20**	**20**	**20**	19	**20**	**20**	19	**20**	**20**
IPC benchmark (1104)	558	565	558.9	571	**575**	571.4	534	534	534	536	535	534.7	564	562	563.7	563	560	561.9
airport(50)	**27**	26	25.7	**27**	26	25.7	24	25	23.9	24	24	23.8	25	24	24.8	25	24	24.6
cybersec(19)	2	3	3.9	8	**12**	10	5	3	5.9	6	4	5.4	6	6	5.9	6	5	5.6
openstacks-opt11(20)	11	**18**	11.7	**18**	**18**	**18**	10	10	10	10	10	9.9	17	17	17	17	17	17

$h =$ M&S	$[f,h,$*ffifo*$]$	$[f,h,$*lifo*$]$	$[f,h,ro]$	$[f,h,\langle d\rangle,$*ffifo*$]$	$[f,h,\langle d\rangle,$*lifo*$]$	$[f,h,\langle d\rangle,ro]$	$[f,\hat{h},$*ffifo*$]$	$[f,\hat{h},$*lifo*$]$	$[f,\hat{h},ro]$	$[f,h,\hat{h},$*ffifo*$]$	$[f,h,\hat{h},$*lifo*$]$	$[f,h,\hat{h},ro]$	$[f,\hat{h}^{\text{FF}},$*ffifo*$]$	$[f,\hat{h}^{\text{FF}},$*lifo*$]$	$[f,\hat{h}^{\text{FF}},ro]$	$[f,\hat{h}^{\text{FF}},\langle d\rangle,$*ffifo*$]$	$[f,\hat{h}^{\text{FF}},\langle d\rangle,$*lifo*$]$	$[f,\hat{h}^{\text{FF}},\langle d\rangle,ro]$
Zerocost (620)	280	301	287.7	302	288	308.1	308	305	307.3	307	306	307.8	336	331	**337.9**	337	333	**337.6**
airport-fuel(20)	**5**	**5**	**5**	**5**	**5**	**5**	1	1	1	1	1	1	**5**	**5**	**5**	**5**	**5**	**5**
depot-fuel(22)	5	5	**6**	**6**	5	**6**	**6**	**6**	**6**	**6**	**6**	**6**	4	4	4	4	4	4
elevators-up(20)	8	14	8.6	9	13	11	19	19	19	19	19	19	**20**	**20**	**20**	**20**	**20**	**20**
floortile-ink(20)	8	8	8	7	7	6.9	8	8	8	8	8	8	**9**	8	8.8	**9**	8	8.8
freecell-move(20)	5	17	6.7	17	15	17.3	13	14	12.7	13	13	12.7	17	17	**17.4**	17	17	17.3
hiking-fuel(20)	**13**	**13**	12.8	**13**	12	12.1	**13**	**13**	12.1	**13**	**13**	12.1	11	11	11	11	11	11
miconic-up(30)	29	**30**	**30**	**30**	**30**	**30**	22	22	22	22	22	22.1	**30**	**30**	**30**	**30**	**30**	**30**
mprime-succumb(35)	21	19	19.6	25	15	23.4	21	17	20.4	21	17	20.4	**28**	23	27.4	**28**	25	27.7
mystery-feast(20)	4	4	5.9	4	4	**6**	5	5	5	5	5	5	3	3	3	3	3	3
parking-movecc(20)	0	0	0	0	0	0	2	2	2	2	2	2	10	10	**10.3**	10	10	**10.3**
pipesnt-pushstart(20)	3	3	3.4	**5**	3	**5**	1	2	1.9	1	2	1.8	**5**	**5**	**5**	**5**	**5**	**5**
pipesworld-pushend(20)	5	**9**	7.7	5	6	**9**	8	7	7.8	8	8	8	5	5	5.4	5	5	5.6
scanalyzer-analyze(20)	11	11	11	11	11	11	15	14	15	14	15	15	15	16	**15.4**	15	15	15.2
sokoban-pushgoal(20)	**19**	**19**	18	18	18	18	17	17	17	17	17	17	18	18	18.2	18	18	18
tpp-fuel(30)	9	10	9.6	**11**	10	**11**	9	10	9.4	9	10	9.8	10	**11**	10.9	**11**	**11**	10.9
woodworking-cut(20)	7	7	8	8	7	9	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**
IPC benchmark (1104)	491	**496**	489.4	487	487	485.6	477	475	470.4	476	475	470.9	458	457	457	457	457	456.8
airport(50)	**9**	**9**	**9**	**9**	**9**	**9**	7	7	7	7	7	7	**9**	**9**	**9**	**9**	**9**	**9**
blocks(35)	**22**	**22**	**22**	**22**	21	21.9	**22**	21	21	21	21	21	21	20	20.1	20	20	20
depot(22)	**6**	**6**	5	5	5	5	5	5	5	5	5	5	4	4	4	4	4	4
elevators-opt11(20)	**13**	**13**	**13**	12	12	12	**13**	**13**	12	**13**	**13**	12	10	10	10	10	10	10
freecell(80)	**17**	**17**	16	16	16	16	15	15	15	15	15	15	14	14	14	14	14	14
miconic(150)	73	73	**73.2**	73	73	72.2	72	72	72	72	72	72	69	69	69.2	69	69	69.2
mprime(35)	23	**24**	23.7	23	**24**	23.4	19	19	19.3	20	19	19.3	21	21	21.1	21	21	21.2
nomystery-opt11(20)	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	16	16	16	16	16	16
openstacks-opt11(20)	15	**19**	15.4	**19**	**19**	**19**	18	**19**	18	18	**19**	18	18	18	18	18	18	17.7
pegsol-opt11(20)	**19**	**19**	**19**	**19**	**19**	**19**	**19**	**19**	**19**	**19**	**19**	**19**	17	17	17	17	17	17
pipesworld-notankage(50)	**10**	**10**	9.9	**10**	9	9.8	6	5	5.7	6	5	5.9	9	9	8.7	9	9	8.8
pipesworld-tankage(50)	13	13	**13.2**	13	13	13	12	12	12	12	12	12	9	9	9	9	9	9
rovers(40)	**8**	**8**	**8**	**8**	**8**	7.1	**8**	**8**	6	7	**8**	6.1	6	6	6	6	6	6
scanalyzer-opt11(20)	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	9.9	**10**	**10**	9.8	7	7	6.8	7	7	6.8
sokoban-opt11(20)	**20**	**20**	**20**	19	19	19	18	18	18	18	18	18	19	19	19	19	19	19
zenotravel(20)	**12**	**12**	**12**	10	10	10.1	**12**	11	10.9	**12**	11	10.9	10	10	10	10	10	10

Table 1: Coverage results with LMcut (top) and M&S (bottom) for computing f, and various tie-breaking strategies, on 620 Zerocost instances and 1104 IPC instances. We only show the domains when the difference between the maximum and the minimum coverage exceeds 2, and highlight the best results.

	LMcut	M&S
$[f, h, \textit{lifo}]$	844	797
$[f, h, \langle d \rangle, \textit{fifo}]$	855	789
$[f, h, \langle d \rangle, \textit{lifo}]$	839	775
$[f, h, \langle d \rangle, \textit{ro}]$	859.5	793.7
Multi-heuristic strategies		
$[f, \hat{h}^{\text{FF}}, \langle d \rangle, \textit{fifo}]$	903	794
$[f, \hat{h}^{\text{FF}}, \langle d \rangle, \textit{lifo}]$	902	790
$[f, \hat{h}^{\text{FF}}, \langle d \rangle, \textit{ro}]$	906.2	794.4
Dynamic Configuration		
If a problem contains zerocost actions:		
$[f, \hat{h}^{\text{FF}}, \langle d \rangle, \textit{ro}]$; Else $[f, h, \langle d \rangle, \textit{lifo}]$	**911.9**	
If a problem contains zerocost actions:		
$[f, \hat{h}^{\text{FF}}, \langle d \rangle, \textit{ro}]$; Else $[f, h, \textit{lifo}]$		**832.3**

Table 2: Summary Results: Coverage comparison, the total of IPC domains and Zerocost domains (the number of instances solved in 5min, 4GB) between several sorting strategies, plus a dynamic configuration strategy. $[f, h, \textit{fifo}], [f, h, \textit{ro}], [f, \hat{h}, *], [f, h, \hat{h}, *], [f, \hat{h}^{\text{FF}}, *]$ are not shown because they achieve smaller coverage.

the effort.[3] Marvin (Coles and Smith 2007) learns plateau-escaping macros from the Enhanced Hill Climbing phase of the FF planner (Hoffmann and Nebel 2001). Hoffmann gives a detailed analysis of the structure of the search spaces of satisficing planning (2005; 2011).

Benton et al. (2010) proposes inadmissible technique for temporal planning where short actions are hidden behind long actions and do not increase makespan. Wilt and Ruml (2011) also analyzes inadmissible distance-to-go estimates. This differs from our work on cost-optimal search because admissible and inadmissible search differ significantly in how non-final plateaus (plateaus with $f < f^*$) are treated: Inadmissible search can skip or escape plateaus whenever possible, while admissible search cannot, unless it is the plateau with $f = f^*$ where the goals can immediately be found.

In their work on combining multiple inadmissible heuristics in a planner, Röger and Helmert (2010) considered a tie-breaking approach which works as follows: When combining two heuristics h_1 and h_2, h_1 is used as the primary criterion, and h_2 is used to break ties among nodes with the same h_1 — $[h_1, h_2, \textit{fifo}]$. This did not perform well in their work on satisficing planning compared to the approaches based on alternation queues and Pareto-optimal queue selection. Since their focus is on how to combine multiple heuristics, this tie-breaking-based approach is just one instance of various implementations of OPEN lists. In contrast, this paper provides a focused, in-depth investigation of various tie-breaking strategies, and shows how tie-breaking enables the efficient search on the plateau created by the earlier levels of sorting criteria.

[3]The relationship between Type-GBFS and our work is discussed in detail in Section 3.1.

6 Conclusions and Future Work

We introduced a new interpretation of cost-optimal A^* search as a series of satisficing searches among f-cost plateaus of an increasing order of f. This perspective led to a novel approach for effective tie-breaking in Zerocost domains, the use of inadmissible distance-to-go estimates as part of a multi-heuristics tie-breaking strategy. Combination of depth diversification and distance-to-go estimates results in the best overall performance. Although there is an additional cost to compute multiple heuristic values, the overhead can be eliminated by a simple case-based configuration which only uses multiple heuristics when 0-cost actions are present in the problem instance.

Our reformulation of A^* as a sequence of satisficing searches points to an interesting direction for future work. Although we evaluated only one relatively simple, satisficing configuration ($\hat{h}^{\text{FF}}$) in the experiments, many techniques which have previously been developed for satisficing planning can be applied to enhance tie-breaking (plateau-search) in cost-optimal search, including lazy evaluation (Richter and Westphal 2010), alternating/Pareto open list (Röger and Helmert 2010), helpful actions (preferred operators) (Hoffmann and Nebel 2001), random walk local search (Nakhost and Müller 2009), macro operators (Botea et al. 2005; Chrpa, Vallati, and McCluskey 2015), factored planning (Amir and Engelhardt 2003; Brafman and Domshlak 2006; Asai and Fukunaga 2015) and exploration-based search enhancements (Valenzano et al. 2014; Xie et al. 2014; Valenzano and Xie 2016).

References

Amir, E., and Engelhardt, B. 2003. Factored Planning. In *Proc. of International Joint Conference on Artificial Intelligence (IJCAI)*.

Asai, M., and Fukunaga, A. 2015. Solving Large-Scale Planning Problems by Decomposition and Macro Generation. In *Proc. of the International Conference on Automated Planning and Scheduling(ICAPS)*.

Asai, M., and Fukunaga, A. 2016. Tiebreaking Strategies for Classical Planning Using A^* Search. In *Proc. of AAAI Conference on Artificial Intelligence*.

Asai, M., and Fukunaga, A. 2017. Tie-Breaking Strategies for Cost-Optimal Best First Search. *J. Artif. Intell. Res.(JAIR)* 58:67–121.

Benton, J.; Talamadupula, K.; Eyerich, P.; Mattmüller, R.; and Kambhampati, S. 2010. G-Value Plateaus: A Challenge for Planning. In *Proc. of the International Conference on Automated Planning and Scheduling(ICAPS)*.

Botea, A.; Enzenberger, M.; Müller, M.; and Schaeffer, J. 2005. Macro-FF: Improving AI Planning with Automatically Learned Macro-Operators. *J. Artif. Intell. Res.(JAIR)* 24:581–621.

Brafman, R. I., and Domshlak, C. 2006. Factored Planning: How, When, and When Not. In *Proc. of AAAI Conference on Artificial Intelligence*.

Chrpa, L.; Vallati, M.; and McCluskey, T. L. 2015. On the Online Generation of Effective Macro-Operators. In *Proc. of International Joint Conference on Artificial Intelligence (IJCAI)*.

Coles, A., and Smith, A. 2007. Marvin: A Heuristic Search Planner with Online Macro-Action Learning. *J. Artif. Intell. Res.(JAIR)* 28:119–156.

Cushing, W.; Benton, J.; and Kambhampati, S. 2010. Cost Based Search Considered Harmful. In *Proc. of Annual Symposium on Combinatorial Search*.

Eyerich, P.; Mattmüller, R.; and Röger, G. 2009. Using the Context-Enhanced Additive Heuristic for Temporal and Numeric Planning. In *Proc. of the International Conference on Automated Planning and Scheduling(ICAPS)*.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *Systems Science and Cybernetics, IEEE Transactions on* 4(2):100–107.

Hoffmann, J., and Nebel, B. 2001. The FF Planning System: Fast Plan Generation through Heuristic Search. *J. Artif. Intell. Res.(JAIR)* 14:253–302.

Hoffmann, J. 2005. Where 'Ignoring Delete Lists' Works: Local Search Topology in Planning Benchmarks. *J. Artif. Intell. Res.(JAIR)* 24:685–758.

Hoffmann, J. 2011. Analyzing Search Topology Without Running Any Search: On the Connection Between Causal Graphs and h^+. *J. Artif. Intell. Res.(JAIR)* 41(2):155–229.

Imai, T., and Kishimoto, A. 2011. A Novel Technique for Avoiding Plateaus of Greedy Best-First Search in Satisficing Planning. In *Proc. of AAAI Conference on Artificial Intelligence*.

Kautz, H. A., and Selman, B. 1992. Planning as Satisfiability. In *Proc. of European Conference on Artificial Intelligence*, volume 92, 359–363.

Korf, R. E. 1985. Depth-First Iterative-Deepening: An Optimal Admissible Tree Search. *Artificial Intelligence* 27(1):97–109.

Nakhost, H., and Müller, M. 2009. Monte-Carlo Exploration for Deterministic Planning. In *Proc. of International Joint Conference on Artificial Intelligence (IJCAI)*.

Pearl, J., and Kim, J. H. 1982. Studies in Semi-Admissible Heuristics. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* (4):392–399.

Pohl, I. 1973. The Avoidance of (Relative) Catastrophe, Heuristic Competence, Genuine Dynamic Weighting and Computational Issues in Heuristic Problem Solving. In *Proc. of International Joint Conference on Artificial Intelligence (IJCAI)*.

Richter, S., and Westphal, M. 2010. The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks. *J. Artif. Intell. Res.(JAIR)* 39(1):127–177.

Richter, S.; Westphal, M.; and Helmert, M. 2011. LAMA 2008 and 2011. In *Proc. of the International Planning Competition*, 117–124.

Röger, G., and Helmert, M. 2010. The More, the Merrier: Combining Heuristic Estimators for Satisficing Planning. In *Proc. of the International Conference on Automated Planning and Scheduling(ICAPS)*.

Thayer, J. T., and Ruml, W. 2009. Using Distance Estimates in Heuristic Search. In *Proc. of the International Conference on Automated Planning and Scheduling(ICAPS)*.

Thayer, J. T., and Ruml, W. 2011. Bounded Suboptimal Search: A Direct Approach using Inadmissible Estimates. In *Proc. of International Joint Conference on Artificial Intelligence (IJCAI)*.

Valenzano, R. A., and Xie, F. 2016. On the Completeness of BestFirst Search Variants that Use Random Exploration. In *Proc. of AAAI Conference on Artificial Intelligence*.

Valenzano, R. A.; Schaeffer, J.; Sturtevant, N.; and Xie, F. 2014. A Comparison of Knowledge-Based GBFS Enhancements and Knowledge-Free Exploration. In *Proc. of the International Conference on Automated Planning and Scheduling(ICAPS)*.

van den Briel, M., and Kambhampati, S. 2005. Optiplan: Unifying IP-based and Graph-based Planning. *J. Artif. Intell. Res.(JAIR)* 24:919–931.

Wilt, C. M., and Ruml, W. 2011. Cost-Based Heuristic Search is Sensitive to the Ratio of Operator Costs. In *Proc. of Annual Symposium on Combinatorial Search*.

Xie, F.; Müller, M.; Holte, R. C.; and Imai, T. 2014. Type-Based Exploration with Multiple Search Queues for Satisficing Planning. In *Proc. of AAAI Conference on Artificial Intelligence*.