1F2-2

Tiebreaking Strategies for A* Search: How to Explore the Final Frontier (Extended Abstract)

Masataro Asai*1 Alex Fukunaga*1

Graduate School of Arts and Sciences, The University of Tokyo

Despite recent improvements in search techniques for cost-optimal classical planning, the exponential growth of the size of the search frontier in A* is unavoidable. We investigate tiebreaking strategies for A*, experimentally analyzing the performance of standard tiebreaking strategies that break ties according to the heuristic value of the nodes. We find that tiebreaking has a significant impact on search algorithm performance when there are zero-cost operators that induce large plateau regions in the search space. We develop a new framework for tiebreaking based on a depth metric which measures distance from the entrance to the plateau, and propose a new, randomized strategy which significantly outperforms standard strategies on domains with zero-cost actions.

This is an extended abstract of the paper accepted to the 30th AAAI Conference of Artificial Intelligence (peer reviewed, 25% accept ratio). For the full article, visit `http://guicho271828.github.io/publications/`.

1. Introduction

This paper investigates tiebreaking strategies for A^*, the 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 68]. In each iteration, A^* selects and *expands* a node n from the OPEN priority queue. n is the node which has the lowest f-cost in OPEN, where for node n, $f(n)$ is the sum of $g(n)$, the cost of the current path from the initial state to n, and $h(n)$, a heuristic estimate of the cost from n to a goal state. A^* returns an optimal solution when h is admissible, i.e., when $h \leq h^*$, where h^* is the optimal distance to the goal.

If f^* is the cost of the optimal solution, the *effective search space of* A^* is the set of nodes with $f(n) \leq f^*$, and much of the work in the search and planning literature has focused on reducing the size of this effective search space by developing more accurate, admissible heuristic functions.

In many problems, the size of the last layer of search (which explores the set of nodes with $f(n) = f^*$) accounts for a significant fraction of the effective search space of A^*. Figure 1 plots the number of states with $f(n) = f^*$ (y-axis) vs. the # of states with $f(n) \leq f^*$ 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 $f(n) = f^*$. For example, in the Openstacks domain, almost all states with $f(n) \leq f^*$ have cost f^* due to the large number of actions with cost 0. In such domains, the tiebreaking policy which decides which nodes to expand in the final frontier can have a significant impact on the performance of A^*.

In this paper, we investigate the *tiebreaking* strategy used by A^*, which is the policy for selecting which node to expand among nodes with the same f-cost. 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 tiebreaking requires more careful consideration, particularly for problems with large *plateaus* – regions of the search space with the same f and h values.

We first empirically evaluate standard tiebreaking strategies for A^*, and show that (1) a Last-In-First-Out (*lifo*) policy tends to be more efficient than a First-In-First-Out (*fifo*) policy, and (2) tiebreaking according to the heuristic value h, which frequently appears in the heuristic search literature, has little impact on the performance as long as a *lifo* policy is used. We show that there are significant performance differences among tiebreaking strategies when domains include zero-cost actions. While there are relatively few domains with zero-cost actions in the IPC benchmark set, we argue that zero-cost actions naturally occur in practical cost-minimization problems.

In order to solve such problems more efficiently, we propose tiebreaking methods based on a notion of *depth* within the plateau, corresponding to the number of steps a node is from the "entrance" to the plateau. We empirically show that: (1) a randomized, depth-based strategy significantly outperforms other tiebreaking strategies using the same heuristic function; (2) although depth is a component of a multi-level tiebreaking strategy, the depth is the principal factor in determining performance; and (3) depth-based tiebreaking is robust, in the sense that it does not rely on a particular action ordering in the domain definition. Note that all tiebreaking strategies in this paper maintain the optimality of the search algorithm because they only affect node expansion order among the nodes with the same f-cost.

2. Preliminaries and Definitions

We first define some notation and terminology used throughout the rest of the paper. A *tiebreaking strategy* selects from among nodes with the same f-value. Tiebreaking strategies are denoted as $[\text{criterion}_1, \text{criterion}_2, ..., \text{criterion}_k]$, which means: If there are multiple nodes with the same f-value, first, break ties using criterion_1. If there are still multiple nodes remaining, then break ties using criterion_2 and so on, until a single node is selected. The *first-level tiebreaking policy* of a strategy is criterion_1, the *second-level tiebreaking policy* is criterion_2, and so on.

A *plateau* is a set of nodes in OPEN with both the same f and

Contact: guicho2.71828@gmail.com

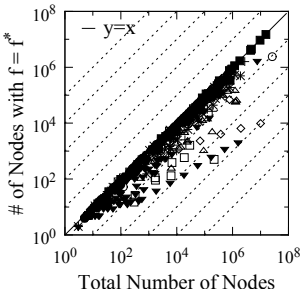

Figure 1: The # of nodes with $f = f^*$ (y-axis) compared to the total # of nodes in the search space (x-axis) with $f \leq f^*$ on 1104 IPC benchmark problems, using modified Fast Downward with LMcut which generates all nodes with cost f^*.

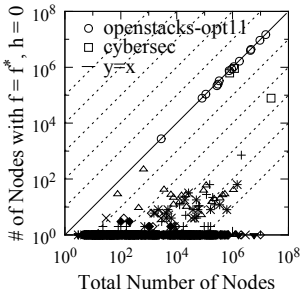

Figure 2: Similar to Figure 1; y-axis shows # nodes with $f = f^*, h = 0$, which forms the final plateau when h-based tiebreaking is enabled. Note that many Openstacks and Cybersec instances are near the $y = x$ line.

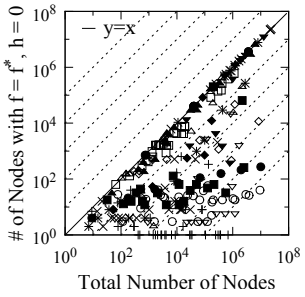

Figure 3: Similar to Figure 2, but for 620 instances from our *zerocost domains* (Sec. 4.), where zero-cost actions induce very large plateaus.

same h costs. A plateau whose nodes have f-cost f_p and h-cost h_p is denoted as *plateau* (f_p, h_p). An *entrance* to a *plateau* (f_p, h_p) is a node $n \in$ *plateau* (f_p, h_p), whose current parent is not a member of *plateau* (f_p, h_p). The *final plateau*, is the plateau containing the solution found by the search algorithm. In A^* using admissible heuristics, the final plateau is *plateau* $(f^*, 0)$.

3. Background: Tiebreaking Strategies in A^*

If multiple nodes with the same f-cost are possible, A^* must implement some tiebreaking policy (either explicitly or implicitly) which selects from among these nodes. The early literature on heuristic search seems to have been mostly agnostic regarding tiebreaking. The original A^* paper, as well as Nilsson's subsequent textbook states: "Select the open node n whose value f is smallest. Resolve ties arbitrarily, but always in favor of any [goal node]" [Hart 68, p.102 Step 2], [Nilsson 71, p.69]. Pearl's textbook on heuristic search specifies that best-first search should "break ties arbitrarily" ([Pearl 84], p.48, Step 3), and does not specifically mention tiebreaking for A^*. To the best of our knowledge, the first explicit mention of a tiebreaking policy that considers node generation order is by Korf in his analysis of IDA*: "If A^* employs the tiebreaking rule of 'most-recently generated', it must also expand the same nodes [as IDA*]", i.e., a *lifo* ordering.

In recent years, tiebreaking according to h-values has become "folklore" in the search community. [Hansen 07] state that "It is well-known that A^* achieves best performance when it breaks ties in favor of nodes with least h-cost" [Hansen 07]. [Holte 10] writes "A^* breaks ties in favor of larger g-values, as is most often done" [Holte 10, note that since $f = g + h$, preferring large g is equivalent to preferring smaller h]. In their detailed survey/tutorial on efficient A^* implementations, [Burns 12] ([Burns 12]) also break ties "preferring high g" (equivalent to low h). Thus, tiebreaking according to h-values appears to be ubiquitous in practice. To our knowledge, an in-depth, experimental analysis of tiebreaking strategies for A^* is lacking in the literature.

Although the standard practice of tiebreaking according to h might be sufficient in some domains, further levels of tiebreaking (explicit or implicit) are required if multiple nodes can have the same f and h values. While the survey of efficient A^* implementation techniques in [Burns 12] did not explicitly mention 2nd-level tiebreaking, their library code (https://github.com/eaburns/search) first breaks ties according to h, and then breaks remaining ties according to a *lifo* policy (most recently generated nodes first), i.e., a $[h, \textit{lifo}]$ strategy. Although not documented, their choice of a *lifo* 2nd-level tiebreaking policy appears to be a natural consequence of the fact it can be trivially, efficiently implemented in their two-level bucket (vector) implementation of OPEN. In contrast, the current implementation of the state-of-the-art A^* based planner Fast Downward [Helmert 06], as well as the work by [Röger 10] uses a $[h, \textit{fifo}]$ tiebreaking strategy. Although we could not find an explanation, this choice is most likely due to their use of alternating OPEN lists, in which case the *fifo* second-level policy serves to provide a limited form of fairness.

Evaluation of Standard Strategies

We evaluated tiebreaking strategies for domain-independent, classical planning. In our experiments, the planners are based on Fast Downward (revision 6251), and all experiments are run with a 5-minute, 2GB memory limit for the search binary (FD translation/preprocessing times are not included in the 5-minute limit). All experiments were conducted on Xeon E5410@2.33GHz CPUs. We used 1104 instances from 35 standard benchmark domains.

We first compared two commonly used tiebreaking strategies, $[h, \textit{fifo}]$, $[h, \textit{lifo}]$, which first break ties according to h, and then apply *fifo* or *lifo* second-level tiebreaking, respectively. Detailed results for the LMcut heuristic [Helmert 09], as well as summary results for the M&S heuristic [Helmert 14], are shown in Table 1 (leftmost 2 columns). Differences in coverage are observed in several domains, and $[h, \textit{lifo}]$ outperforms $[h, \textit{fifo}]$ in total. Figure 4 gives us a more fine-grained analysis by comparing the number of node evaluation (computations of LMcut) of the $[h, \textit{lifo}]$ and $[h, \textit{fifo}]$ strategies. It shows that the difference in the # of nodes evaluated can sometimes be larger than a factor of 10 (Openstacks, Cybersec domains).

4. Domains with Zero-Cost Actions

Openstacks is a cost minimization domain introduced in IPC-2006, where the objective is to minimize the number of stacks

	Coverages (# problems solved)				Coverage (# problems solved), 10 runs (mean±sd)					Wilcoxon p vs $[h, rd, ro]$		
Domain	$[h, fifo]$	$[h, lifo]$	$[fifo]$	$[lifo]$	$[h, fd, ro]$	$[h, ld, ro]$	$[h, rd, ro]$	$[rd, ro]$	$[h, ro]$	$[h, fd, ro]$	$[h, ld, ro]$	$[h, ro]$
LMcut IPC (1104)	558	565	442	556	556.6±0.7	570.3±2.1	**572.8±0.7**	558.8±2.1	559.8±1.0	**0.0**	**.01**	**0.0**
airport(50)	**27**	26	18	26	26.2±0.4	26.2±0.4	26.2±0.4	21.0±0.0	26.0±0.0	1.0	1.0	.17
cybersec(19)	2	3	0	3	2.0±0.0	8.5±2.0	**10.9±0.8**	7.4±0.7	4.4±1.0	**0.0**	**.01**	**0.0**
logistics00(28)	**20**	**20**	16	18	**20.0±0.0**	**20.0±0.0**	**20.0±0.0**	**20.0±0.0**	**20.0±0.0**	1.0	1.0	1.0
miconic(150)	**140**	**140**	68	**140**	**140.0±0.0**	**140.0±0.0**	**140.0±0.0**	135.5±1.2	**140.0±0.0**	1.0	1.0	1.0
openstacks-opt11(20)	11	**18**	11	**18**	11.0±0.0	**18.0±0.0**	**18.0±0.0**	**18.0±0.0**	11.6±0.5	**0.0**	1.0	**0.0**
pipesworld-notankage(50)	**15**	14	13	13	14.4±0.5	14.6±0.5	14.7±0.5	14.3±0.5	14.9±0.3	0.2	.68	0.3
scanalyzer-opt11(20)	**10**	**10**	4	**10**	**10.0±0.0**	**10.0±0.0**	**10.0±0.0**	9.0±0.0	**10.0±0.0**	1.0	1.0	1.0
woodworking-opt11(20)	10	10	6	9	10.0±0.0	10.0±0.0	10.0±0.0	**11.6±0.5**	10.0±0.0	1.0	1.0	1.0
LMcut Zerocost(620)	256	279	212	281	257.4±2.0	286.6±7.1	**294.2±2.3**	279.9±3.9	264.9±1.8	**0.0**	**.01**	**0.0**
airport-fuel(20)	**15**	13	7	**15**	14.7±1.0	14.0±0.6	14.6±0.5	10.5±0.7	14.4±0.7	.59	**.05**	.58
driverlog-fuel(20)	**8**	**8**	7	**8**	**8.0±0.0**	7.7±0.5	**8.0±0.0**	**8.0±0.0**	**8.0±0.0**	1.0	.08	1.0
elevators-up(20)	7	**13**	7	**13**	7.0±0.0	9.4±0.7	10.7±1.1	8.3±0.6	7.3±0.5	**0.0**	**.02**	**0.0**
freecell-move(20)	4	19	4	19	4.0±0.0	**19.7±0.5**	17.2±0.6	16.7±1.0	5.0±0.4	**0.0**	**0.0**	**0.0**
miconic-up(30)	16	17	10	17	15.7±0.5	19.4±0.7	**20.4±1.2**	**20.4±0.9**	17.0±0.4	**0.0**	**.03**	**0.0**
mprime-succumb(35)	15	14	12	14	16.3±0.5	18.9±4.0	**20.5±0.8**	18.1±1.6	17.9±0.5	**0.0**	.15	**0.0**
pipesnt-pushstart(20)	8	8	6	7	8.0±0.0	8.8±1.3	**9.8±0.4**	9.7±0.5	8.5±0.5	**0.0**	0.1	**0.0**
pipesworld-pushend(20)	3	4	2	4	3.0±0.0	4.2±1.0	4.9±0.5	**5.2±1.2**	3.9±0.3	**0.0**	.09	**0.0**
scanalyzer-analyze(20)	9	9	3	9	**9.8±0.9**	9.4±0.5	9.2±0.4	7.3±1.0	9.1±0.3	.07	.37	.58
tpp-fuel(30)	8	**11**	7	**11**	7.5±0.5	**11.0±0.0**	**11.0±0.0**	**11.0±0.0**	8.1±0.3	**0.0**	1.0	**0.0**
woodworking-cut(20)	5	7	2	7	5.0±0.0	6.9±0.3	**9.2±0.9**	7.7±0.6	7.1±0.3	**0.0**	**0.0**	**0.0**
LMcut Total(1724)	814	844	654	837	814.0±2.3	856.9±8.5	**867.0±2.1**	838.7±4.9	824.7±2.1	**0.0**	**.01**	**0.0**
M&S IPC (1104)	479	**488**	451	481	478.8±0.4	484.8±0.4	484.0±0.0	481.4±1.4	486.4±0.8	**.01**	**.02**	**.01**
M&S Zerocost (620)	276	290	226	283	274.0±0.9	293.4±2.1	**310.2±2.1**	303.2±1.7	288.0±1.7	**.01**	**.01**	**.01**
M&S Total(1724)	755	778	677	764	752.8±0.7	778.2±1.9	**794.2±2.1**	784.6±2.1	774.4±1.2	**.01**	**.01**	**.01**

Table 1: Coverage comparison (# of instances solved in 5min, 2GB), **bold**=best. Zerocost domains are named as [original name]-[name of nonzero action]. Due to space, we only show the domains whose maximum pairwise coverage difference $MaxDiff > 2$. (We used the means of 10 runs for the randomized strategies.) Domains with $MaxDiff \leq 2$ follows: (1) $MaxDiff = 0$ (same coverages by all configuration and all runs): barman-opt11, floortile-opt11, grid, gripper, hanoi, parking-opt11, pegsol-opt11, psr-small, rovers, sokoban-opt11, tpp, transport-opt11, grid-fuel, gripper-move, parking-movecc, psr-small-open, zenotravel-fuel. (2) $0 < MaxDiff \leq 1$: depot, driverlog, elevators-opt11, freecell, mystery, parcprinter-opt11, pathways, pipesworld-tankage, storage, tidybot-opt11, visitall-opt11, driverlog-fuel, floortile-ink, hiking-fuel, logistic00-fuel, nomystery-fuel, pathways-fuel, sokoban-pushgoal. (3) $1 < MaxDiff \leq 2$: blocks, nomystery-opt11, pipesworld-notankage, zenotravel, depot-fuel, rovers-fuel, storage-lift, tidybot-motion.

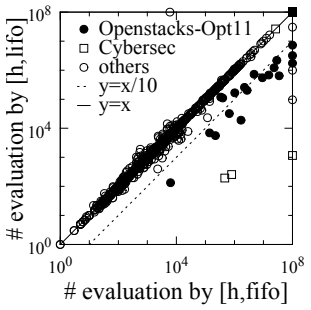

Figure 4: # of evaluations of standard *fifo* vs *lifo* second-level tiebreaking, with first-level h tiebreaking. *lifo* evaluates less than $1/10$ of the nodes evaluated by *fifo* in Cybersec and Openstacks.

used. There are many zero-cost actions (i.e., actions that don't increase the number of stacks), and they prevent the standard heuristics from producing informative guidance.

Although domains with zero-cost actions are not common in the current set of benchmarks, we argue that such domains are of an important class of models for cost-minimization problems, i.e., assigning zero costs make sense from a practical, modeling perspective. For example, consider the driverlog domain, where the task is to move packages between locations using trucks. The IPC version of this domain assigns unit costs to all actions. Thus, cost-optimal planning on this domain seeks to minimize the number of steps in the plan. However, another natural objective function would be the one which minimizes the amount of fuel spent by driving the trucks, assigning cost 0 to all actions except drive-truck.

Similarly, for many practical applications, a natural objective is to optimize the usage of one key consumable resource, e.g., fuel/energy minimization. In fact, two of the IPC domains, Openstacks and Cybersec, which were shown difficult for standard tiebreaking methods in the previous section, both contain many zero-cost actions, and both are based on industrial applications: Openstacks models production planning [Fink 99] and Cybersec models Behavioral Adversary Modeling System [Boddy 05, minimizing decryption, data transfer, etc.].

Therefore, in this paper, we modified various domains into cost minimization domains with many zero-cost actions. Specifically, the domain is modified so that all action schemas are assigned cost 0 except for 1 action schema which consumes some key resource. The last word in the names of these domains indicate the action which is assigned non-zero cost, e.g., elevator-up is a modified elevator domain where the up action is assigned non-zero cost, and all other actions have 0 cost. Most of the transportation-type domains are modified to optimize energy usage (Logistics-fuel, elevator-up etc.), and assembly-type domains are modified to minimize resource usage (Woodworking-cut minimizes wood usage, etc.). We did not include domains with only a single action schema and standard domains which already had many zero-cost actions (these are already in the results for standard IPC domains). We refer to these 28 new domains as *zerocost domains*.

Figure 3 plots the size of the final plateau of the zerocost domain instances. As expected, many of these zerocost domains have large plateaus. Thus, in these cost-minimization problems, the search strategy within plateaus, i.e., tiebreaking, becomes very important.

5. Depth-Based Tiebreaking

In order to solve zerocost problems, the planner needs to perform an efficient knowledge-free search within a large, final plateau. One useful notion which can be used to both understand and control the search in this situation is the *depth* of a node, which represents the number of steps (edges in the search space graph) from the entrance of the plateau. Given a node n, if its current parent *parent* (n) is from the other plateau, i.e., *parent* (n) has a different f-value, or different h-value when the first tiebreaking is present, then *depth* $(n) = 0$. Nodes with *depth* $(n) = 0$ correspond to the *entrance* of the plateau. If n and *parent* (n) are in the same plateau i.e. share the same f and h, *depth* (n) is defined as *depth* $($*parent* $(n)) + 1$. Based on this simple notion of depth, we propose three *depth-based tiebreaking* strategies, where the nodes are inserted into buckets associated with depths, and upon expansion, the buckets are chosen according to some policy. "First depth" (*fd*), "last depth" (*ld*), and "random depth" (*rd*) choose a bucket with the smallest depth, the largest depth, and a depth randomly selected at each expansion, respectively.

The effectiveness of each of these depth-based policies depends on the problem instance. Within the plateau region, all nodes have the same f and h values, and the goals can be near or far from the entrance. In the former case, the search should be focused around the entrance favoring the smaller depths (*fd*), and the behavior in the plateau should be much like breadth-first. In the latter case, the planner should greedily explore the various area of the plateau by preferring largest depth (*ld*), much like in depth-first. It may also be possible for a goal to be at an intermediate depth, in which case *fd* could take too much time to reach that depth, and *ld* may greedily pass and miss that depth. By an adversary argument, *rd*, which selects a random depth and has no depth bias would seem to be the safest policy.

Tiebreaking within Depth Buckets

Since there can be multiple nodes within the same depth bucket, a further tiebreaking criterion may be necessary to break ties among them. We could, for example, apply *lifo* or *fifo* policies at this level – note that $[h, fd, fifo]$ and $[h, ld, lifo]$ are equivalent to $[h, fifo]$ and $[h, lifo]$, respectively.

However we use a Random Order (*ro*) policy, which randomly selects an element from the depth bucket selected by the depth-based tiebreaking. This is because the effectiveness of the tiebreaking behavior within a bucket can be affected by accidental biases, e.g., names/orders of action schema in the PDDL domain definition [Vallati 15]. Thus, we avoid bias at this level of tiebreaking by using *ro* and assess its expected/average performance.

5.1 Evaluating Depth-Based Tiebreaking

Due to space, we could not fully describe the evaluation of our tiebreaking strategies in Table 1. For the full article, visit `http://guicho271828.github.io/publications/`.

6. Conclusion

In this paper, we evaluated standard tiebreaking strategies for A^*. We showed that contrary to conventional wisdom, tiebreaking based on the heuristic value is not necessary to achieve good performance, and proposed a new framework for defining tiebreaking policies based on *depth*. We showed that a depth-based, randomized strategy $[h, rd, ro]$, which uses the heuristic value, but explicitly avoids depth and ordering biases present in previous methods, significantly outperforms previous strategies on domains with zero-cost actions, including practical application domains with resource optimization objectives in the IPC benchmarks. The proposed approach is highly effective on domains where zero-cost actions create large plateau regions where all nodes have the same f and h costs and the heuristic function provides no useful guidance.

References

[Boddy 05] Boddy, M. S., Gohde, J., Haigh, T., and Harp, S. A.: Course of Action Generation for Cyber Security Using Classical Planning., in *Proc. ICAPS*, pp. 12–21 (2005)

[Burns 12] Burns, E. A., Hatem, M., Leighton, M. J., and Ruml, W.: Implementing Fast Heuristic Search Code., in *Proc. Symposium on Combinatorial Search* (2012)

[Fink 99] Fink, A. and Voss, S.: Applications of Modern Heuristic Search Methods to Pattern Sequencing Problems, *Computers & Operations Research*, Vol. 26, No. 1, pp. 17–34 (1999)

[Hansen 07] Hansen, E. A. and Zhou, R.: Anytime Heuristic Search., *J. Artif. Intell. Res.(JAIR)*, Vol. 28, pp. 267–297 (2007)

[Hart 68] Hart, P. E., Nilsson, N. J., and Raphael, B.: A Formal Basis for the Heuristic Determination of Minimum Cost Paths, *IEEE Transactions on Systems Science and Cybernetics*, Vol. 4, No. 2, pp. 100–107 (1968)

[Helmert 06] Helmert, M.: The Fast Downward Planning System., *J. Artif. Intell. Res.(JAIR)*, Vol. 26, pp. 191–246 (2006)

[Helmert 09] Helmert, M. and Domshlak, C.: Landmarks, Critical Paths and Abstractions: What's the Difference Anyway?, in *Proc. ICAPS* (2009)

[Helmert 14] Helmert, M., Haslum, P., Hoffmann, J., and Nissim, R.: Merge-and-Shrink Abstraction: A Method for Generating Lower Bounds in Factored State Spaces, *J. ACM*, Vol. 61, No. 3, pp. 16:1–16:63 (2014)

[Holte 10] Holte, R. C.: Common Misconceptions Concerning Heuristic Search., in *Proc. Symposium on Combinatorial Search* (2010)

[Nilsson 71] Nilsson, N.: *Problem Solving Methods in Artificial Intelligence*, McGraw-Hill (1971)

[Pearl 84] Pearl, J.: *Heuristics: Intelligent Search Strategies for Computer Problem Solving*, Addison-Wesley Pub. Co., Inc., Reading, MA (1984)

[Röger 10] Röger, G. and Helmert, M.: The More, the Merrier: Combining Heuristic Estimators for Satisficing Planning, in *Proc. ICAPS*, pp. 246–249 (2010)

[Vallati 15] Vallati, M., Hutter, F., Chrpa, L., and McCluskey, T. L.: On the Effective Configuration of Planning Domain Models, in *Proc. IJCAI* (2015)