Domain-Dependent Heuristics and Tie-Breakers: Topics in Automated Planning

Augusto B. Corrêa, André G. Pereira, Marcus Ritt

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS) – Brasil

{abcorrea, agpereira, marcus.ritt}@inf.ufrgs.br

Abstract. *Automated planning is an important general problem solving technique in Artificial Intelligence. Given an initial state, a goal and a set of operators, we want to find a sequence of operators leading us to the goal. What makes planning interesting is that it can model different domains into planning tasks and solve them using a single method. In this work, we approach two different topics in planning. First, we study heuristics for the airport ground traffic problem and propose new heuristics that are better than any other known method. In the second part, we study tie-breakers for the A^* search algorithm. We propose a new tie-breaking method that is proved to be the best possible and also show that our methods solve more instances than previous methods in literature.*

1. Introduction

Automated planning is a general problem-solving technique in Artificial Intelligence (AI) where, given an initial state of a problem, we try to find a sequence of operators leading us to a goal. For example, in an airport ground traffic control problem we may want to find a sequence of movements that planes must perform to reach their goal position given a set of constraints in the airport. A vast number of domains can be modeled as planning tasks, such as logistics problems, transportation problems (e.g. Sokoban), puzzles (e.g. the $(n^2 - 1)$-puzzle), scheduling problems and even natural language generation.

The motivation of planning is to create one solver that performs well in any given problem, without having previous knowledge of the problem. This makes planning a cost-effective model for software engineering. One can build or select a planner; model any problem into a planning model and then solve it using the planner. If the problem changes, we just need to change the model of the problem, but not the solver. In another perspective, using planning to find good solutions for real-world problems can help to reduce time, workforce and (probably the most important) costs.

In this work, we are interested in a specific formalism of planning, called *optimal classical domain-independent planning*. In this formalism, a single agent with no prior knowledge of the problem executes a sequence of operators, with deterministic effects and positive cost for each operator, until it reaches the goal. However, this sequence of operators must be the most efficient one (i.e., the one with least cost). One could try to explore all possible sequence of operators, but this idea is unfeasible. Planning has the *state explosion* problem: the number of "states" is too large to be exhaustively checked. In fact, this version of planning is known to be PSPACE-complete [Helmert 2006b].

In the last decades, the main technique to try to avoid the state explosion problem is based on *heuristic search* [Bonet and Geffner 2001]. A *heuristic*, in the sense of this

work, is an function that estimates the cost from a state of the task to its goal. By performing an informed search (e.g. A* [Hart et al. 1968]), we can reduce the effort needed to find a solution for a planning task by reducing the state explosion problem.

1.1. Context of this Research

This work was produced in the Algorithms and Optimization research group from the Institute of Informatics of the Federal University of Rio Grande do Sul (UFRGS). This is the first work related to domain-independent planning from this group.

In previous research, [Gliesch and Ritt 2016] proposed heuristic search techniques to solve the Atomix puzzle. [Pereira et al. 2016] studied the class of moving-blocks problems and proved the PSPACE-completeness of many variants. They also proposed new methods to optimally solve these problems based on heuristic search [Pereira et al. 2015]. The doctoral dissertation resulted from this work was awarded the second place in Doctoral Dissertation Contest of the Brazilian Computer Society (2017). Our work extends this research line on planning. It started as an undergraduate project and it ended as a Bachelor thesis. It was funded by the CNPq and by UFRGS.

This project lead the Algorithms and Optimization Group to establish cooperation with the Artificial Intelligence Group of the University of Basel, Switzerland, a well known automated planning group. The author of this work is now a student and researcher in this university. Besides that, the first part of this work was published in the most important national conference for AI and the second part was published in an international conference and in an international workshop.

2. Background

A *planning task* is defined as a 4-tuple $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, S_* \rangle$, where $\mathcal{V} = \langle v_1, \ldots, v_n \rangle$ is a set of variables with finite-domain, $\mathcal{O}$ is a set of operators with preconditions and effects based on $\mathcal{V}$, s_0 is the initial state and S_* is a set of goal states. A *state* is an assignment of each variable in $\mathcal{V}$ to some value. An operator o is applicable to a state s if the preconditions of o are all true in s. When o is applied, its effects become true in s. Our goal is to find a sequence of operators from s_0 to some state in S_*.

Let $\mathcal{S} = \langle s_0, S_*, \mathcal{O}, \mathrm{cost} \rangle$ be a *state space* for a giving planning task, where s_0 is the initial state, S_* is a set of goal states and $\mathcal{O}$ is a set of operators. For a given state s there is a (possibly empty) subset of operators in $\mathcal{O}$ that can be applied to s to generate a set of *successor states* $\mathrm{succ}(s)$. Every operator $o \in \mathcal{O}$ has a cost $\mathrm{cost}(o) \in \mathbb{R}_0^+$ associated to the transition $s \to s'$. A sequence of distinct states denoted as $s_0 \to s_1 \to \cdots \to s_n$ is called a *path*, if for every pair of consecutive states $s \to s'$ we have $s' \in \mathrm{succ}(s)$. If $s_n \in S_*$ then the sequence is called a solution path (s-path). Among all possible solution paths, the one with the least cost is the *optimal path*.

A* is the most popular admissible best-first search algorithm [Hart et al. 1968]. It expands states s in order of increasing $f(s)$. Function $f(s)$ is the sum of the cost $g(s)$ of the current path from the initial state to state s, and the estimated cost $h(s)$ from s to a goal state. When using an admissible heuristic h, A* returns an optimal solution path of minimum cost $\mathcal{C}^*$, if there is one. A heuristic h^* is *perfect* if it returns the cost of an optimal path for all states. During the search, it is possible to have several states with the same f-value. Then, A* ordering states by $[f, \tau]$ with a *tie-breaking strategy* τ expands

a unique sequence of states $\langle s_0, s_1, \ldots, s_n \rangle$, called the *expansion sequence*. We assume that A* keeps a priority queue denoted as OPEN that sorts the states lexicographically in increasing order of $[f, \tau]$. To expand a state means to remove it from OPEN and to generate all its successors. Note that in this way goal states are only *processed*, i.e. removed from OPEN, but not expanded. If the expansion sequence of A* with a given tie-breaking strategy has the minimum number of states among all possible sequences we say that this strategy has *optimal expansion* – or simply that it is *optimal*. If the function f uses the perfect heuristic h^*, we denote it as $f^* = g + h^*$.

3. Contributions of This Work

This work approaches two different topics in classical planning. First, given the last advancements in planning using heuristic search, we want to solve the question: *are the state-of-the-art heuristics in planning better than domain-dependent heuristics?* To do so, we will investigate one of the hardest domains in planning, the airport ground traffic problem. Our studies in this topic were previously published as a paper in the BRACIS 2016 (Brazilian Conference on Intelligent Systems) [Corrêa et al. 2016]. Second, we address the fundamental problem of tie-breakers in heuristic search: *can we find a perfect tie-breaker for A*?* We study tie-breaking strategies for A* from a theoretical perspective and we present a novel method that guarantees optimal expansion. Our theoretical results are empirically tested and we obtain better performance on coverage (i.e., number of solved instances) than the state-of-the-art methods in literature. Our results were published in the IJCAI 2018 (International Joint Conference on Artificial Intelligence) [Corrêa et al. 2018a] and in the HSDIP Workshop – ICAPS 2018 (International Conference on Automated Planning and Scheduling) [Corrêa et al. 2018b].

3.1. Airport Ground Traffic Problem

One of the planning problems that arises in airports is the ground traffic control problem, which is to define the paths out-bound airplanes have to take from their parking position to the runway for take-off, or the paths in-bound airplanes have to take after landing until getting to a parking position. The main objective is to find a plan that achieves this goal most economically, i.e., with the least movement of the airplanes, respecting restrictions on the movement coming from concerns about security, for example the minimum distance between airplanes.

A version of this domain was introduced in the International Planning Competition (IPC) of 2004. In this version, the airport is modeled as a directed graph, with nodes representing segments of the airport and with specific designated parking and take-off positions. Given an initial and a goal position for each airplane, the goal is to find a sequence with the least number of movements to lead each airplane to its goal position.

The standard version has a fixed goal position for each airplane. However, in the real world, the parking positions of the in-bound planes are not necessarily fixed and they can sometimes change while the planes move through the airport. Hence, we propose a new version of this domain, where the in-bound planes may park in any parking segment. This version is more realistic but also harder, since in an instance with p park positions and a in-bound airplanes we would have up to $\binom{p}{a}$ possible goal states for these airplanes.

The techniques used by solvers in this domain usually are domain-independent. These techniques do not take into consideration particularities of the problem, and try to

Table 1. Results for the airport ground traffic problem. Best values are in bold.

Original (50)	**LM-cut**	**iPDB**	**Closest**	**Matching**
Coverage	28	28	**32**	–
Time (s)	603	3619	**61**	–
Initial h-value	**301.9**	299.8	**301.9**	–
Best f-value	301.9	299.8	**302.7**	–
Free Parking (50)				
Coverage	26	28	30	**33**
Time (s)	15083	15616	6478	**2158**
Initial h-value	260.4	248.9	260.4	**262.6**
Best f-value	260.4	248.9	260.7	**262.8**

solve it only by generic strategies. In this work, we compare how well simple domain-dependent heuristics perform when compared to state-of-the-art domain-independent heuristics in both variants of the problem mentioned above.

3.1.1. Proposed Domain-dependent Heuristics for the Airport Domain

Our main contribution to the airport ground traffic problem is the introduction of simple domain-dependent heuristics with better performance than any admissible heuristic in literature. We introduce two new heuristics for this domain: the closest goal heuristic and the matching heuristic. The former starts preprocessing the all-pair shortest path for each segment of the airport; then, in each state expanded by A*, it sums up the distance from each airplane to its closest goal segment possible. The pitfall of this method is that it can consider that two or more airplanes will park at a same parking position, which is not allowed. The latter method also uses the preprocessing scheme, but instead of simply summing up the distances, it creates a bipartite graph with airplanes in one part and the goal positions in the other; then, it computes the perfect matching of minimum cost in this graph. This method is slightly slower then the closest goal heuristic, but it guarantees that the heuristic considers that each parking position must be occupied for just one airplane.

Notice that, the problem mentioned above does not happen for out-bound planes. Since they are going to takeoff, if two planes go to a same runway position, it will not be permanently occupied. Also, in the original version with fixed parking positions the closest goal and the matching heuristic are the same: since there is only one possible goal position for each plane, both heuristics will have the same value.

3.1.2. Results

In this section we report computational experiments comparing domain-independent and domain-dependent heuristics on the airport domain with fixed and free goal positions. For the experiments we used the stable version 1.4 of the Fast Downward planner [Helmert 2006a]. We chose Fast Downward for our implementation to compare it

better to the domain-dependent heuristics. All experiments were run on a PC with an AMD FX-8150 processor running at 3.6 GHz and 32 GB of main memory. All domain-independent heuristics had their default parameters. We imposed a time limit of 30 minutes and a memory limit of 4 GB for each run. Both variants of the domain have 50 instances.

In our first experiment, we evaluated domain-independent heuristics on the airport domain. These heuristics solved between 18 and 28 instances. The two best heuristics in our tests are $h^{\text{LM-cut}}$ [Helmert and Domshlak 2009] and h^{iPDB} [Haslum et al. 2007]. While $h^{\text{LM-cut}}$ is based on *landmarks* of the planning task – i.e., a set of actions that must be performed by any plan to reach some goal state, the h^{iPDB} heuristic is based on *pattern databases* – i.e., it stores the cost of each state for relaxed versions of the problem and combines these patterns to estimate the cost of the state in the original problem.

In our second experiment, we compared $h^{\text{LM-cut}}$ and h^{iPDB} with the closest goal heuristic in the variant with fixed parking positions. The top part of Table 1 shows the results. "Coverage" indicates the number of solved instances. "Time (s)" represents the among of time in seconds that the method needed to solve all these instances. "Initial h-value" is the average of the h-value for the initial states of all instances. Since all heuristics are admissible, a higher initial h-value is always better. "Best f-value" is the average of the f-value for the last state expanded in each instance, it indicates how much progress the search did with a given heuristic, hence a higher f-value is also better. We can see that our new heuristic overperformed $h^{\text{LM-cut}}$ and h^{iPDB} in number of solved instances and time. It is also more informed since the beginning of the search when compared to h^{iPDB} and it makes more progress than the other two methods.

Our last experiment compared all the four heuristics in the version with free parking positions. The bottom part of Table 1 shows the results. Once again, our new methods presented better performance the state-of-the-art domain-independent heuristics. However, this time, the matching heuristic performed far better than any other method. Even though its more expensive to compute then the closest goal heuristic, the matching heuristic is more informed than any other method and it ends up performing better.

3.2. Tie-Breaking Strategies for A*

A* expands all states with $f < C^*$ and additionally some states with $f = C^*$ (unless there is a path where all states $s \notin S*$ have $h(s) < h^*(s)$) [Dechter and Pearl 1985]. The set of states with $f = C^*$ is known as the *final f-layer*. There is always a tie-breaking strategy τ that expands, in addition to states with $f(s) < C^*$, only states on a shortest cost-optimal path in the final f-layer (i.e., states along the cost-optimal path with the least number of operators in the final f-layer). In this case, we say that tie-breaking strategy τ has *optimal expansion*, or simply is *optimal*. In practice, however, A* has to break ties among a large number of states in the final f-layer. Most of these states will not lead to a goal state, and their expansions do not add any novel information to the search. Ideally, A* should use a tie-breaking strategy that completely avoids any of these useless expansions.

In this section, we study tie-breaking strategies for A*. We analyze previously proposed tie-breaking strategies and prove that they are not always optimal. We also propose a new strategy which is guaranteed to have optimal expansion. In practical settings, our new strategies solve more instances than other methods in the literature. Our results

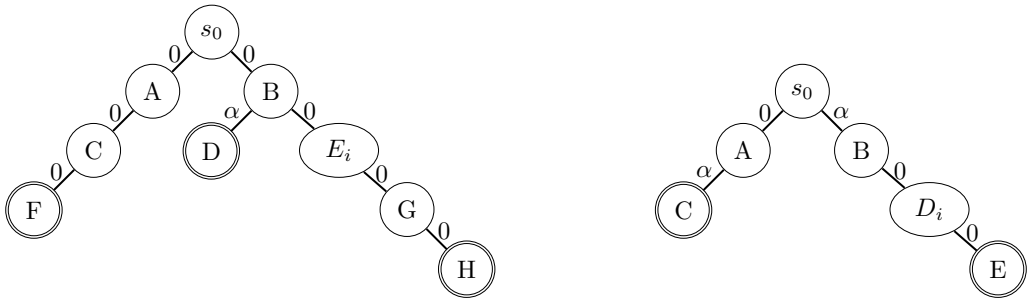

Figure 1. On the left: $[f^*, h^*]$ and $[f^*, \hat{h}^*]$ fail. On the right: $[f^*, h^*_\epsilon]$ fails.

show how to build an optimal tie-breaking strategy given h^* and our analysis improves the understanding of how to develop tie-breakers.

3.2.1. Theoretical Results

The heuristic search literature usually considers breaking ties by h to be a good approach. Thus, one would expect that breaking ties by the perfect heuristic h^* would be an optimal expansion strategy. In this setting, using *order* $[f^*, h^*]$ means that A* uses f^* as main evaluation function and h^* as tie-breaker, and any remaining ties are solved arbitrarily. However, using order $[f^*, h^*]$ is not optimal, as it may expand more states than another strategy. For example, the graph on left of Figure 1 shows an instance with only two paths to goal states using only zero-cost operators. The state s_0 is the initial state; doubly-circled states are goals and ellipses represent sequences with i transitions using only cost zero operators. In this situation, $[f^*, h^*]$ provides no information. To reach a goal from s_0, A* may expand three states using the left s-path ($s_0 \to A \to C \to F$), or an arbitrarily large set of states using the right s-path ($s_0 \to B \to \cdots \to G \to H$).

[Asai and Fukunaga 2017] proposed to use *distance-to-go* heuristics as tie-breakers. A distance-to-go heuristic, denoted $\hat{h}$, uses the same algorithm to compute h but replaces the cost of all operators by one. Thus $\hat{h}^*(s)$ is the minimum number of operator applications necessary to reach a goal state from s. In practice, A* using $[f^*, \hat{h}^*]$ improves coverage in zero-cost domains [Asai and Fukunaga 2017] (i.e., domains having operators with cost zero). But the order $[f^*, \hat{h}^*]$ also fails to guarantee optimal expansion, as the example on the left of Figure 1 shows. Let $\alpha > 0$. After expanding s_0, we have $\hat{h}^*(A) = 2$, because A can reach the closest goal F applying two operators, and $\hat{h}^*(B) = 1$, because B can reach its closest goal D applying only one operator. As a consequence, A* expands state B first. However, the s-path $s_0 \to B \to D$ is not optimal because the operator from B to the goal F has cost α. Thus, $[f^*, \hat{h}^*]$ expands four states ($\langle s_0, B, A, C\rangle$), and the optimal strategy only three ($\langle s_0, A, C\rangle$).

We introduce a novel type of tie-breaking strategy called *cost adaptation*. The tie-breaking strategy using the perfect heuristic h^* guides the search along a cost-optimal path but fails to identify the cost-optimal path with the least number of operators, whereas the tie-breaking strategy using the distance-to-go heuristic $\hat{h}^*$ guides the search along a path with fewest operators to the goal but fails to estimate the total cost of the path. Combining both estimates improves the search performance. The cost-adapted heuristic h_c uses the

same algorithm to compute h on a state space $\mathcal{S}$, but adds a constant c to each operator cost. In the special case with $c = 1$, we denote h_c as h_{+1}.

To analyze the behavior of h^*_c, we consider different magnitudes of c. First, consider $c = \epsilon$ where ϵ is a small constant such that $\epsilon \ll \min_{o \in \mathcal{O}}\{\text{cost}(o) \mid \text{cost}(o) > 0\}$. The effect of making ϵ very small is that even for the longest path with l operators, the product $l\epsilon$ is still smaller than the smallest difference between a cost-optimal and a non-cost-optimal solution. Using the order $[f^*, h^*_\epsilon]$ on the left example of Figure 1, it produces the optimal expansion $\langle s_0, A, C\rangle$. Yet, $[f^*, h^*_\epsilon]$ can also fail. The right graph of Figure 1 shows an example where A* with $[f^*, h^*_\epsilon]$ expands three states and the optimal expansion only two. In the case where $i = 1$, after expanding s_0 A* can expand A and B, where $h^*_\epsilon(A) = \alpha + \epsilon$ while $h^*_\epsilon(B) = 2\epsilon + |D_i|\epsilon$. Thus, B is chosen for expansion, followed by the sequence of states D_i, leading to goal state E. A* expands the path $s_0 \to B \to \cdots \to E$ instead of the shortest cost-optimal path $s_0 \to A \to C$.

An approach to solve the example of the right side of Figure 1 is to use $c = M$, where $M \gg \max_{o \in \mathcal{O}}(\text{cost}(o))$. In this instance, breaking ties by h^*_M produces the optimal expansion. Now, $h^*_M(A) = \alpha + M$ and $h^*_M(B) = 2M + |D_i|M$. Since $M \gg \alpha$, A* expands A instead of B, and terminates at the goal state C, leading the search to the optimal expansion sequence $\langle s_0, A\rangle$. But the strategy h^*_M also fails to achieve the optimal expansion in the left example of Figure 1, where we have $h^*_M(A) = 2M$ and $h^*_M(B) = \alpha + M$. Since $M \gg \alpha$, we have $h^*_M(A) > h^*_M(B)$ causing the search to expand B, leading to the same problem of $[f^*, \hat{h}^*]$.

3.2.2. A Strategy with Optimal Expansion

Combining our novel method h_c with the g-value of the state, A* guarantees optimal expansion. The following theorem presents a single tie-breaker that achieves optimal expansion for admissible and consistent heuristic functions h.

Theorem 1. *For an admissible and consistent heuristic h, A* with order $[f, \tau]$ and tie-breaker $\tau = g + h^*_\epsilon$ has optimal expansion.*

Proof. If there is no solution A* will always expand all reachable states and thus has optimal expansion. Otherwise, since h is admissible and consistent, A* will process states by non-decreasing f-values, ending with $f = \mathcal{C}^*$ at some goal state. We will show that A* with tie-breaker τ expands the least number of states in the final f-layer, from which the claim follows, since states with $f < \mathcal{C}^*$ must be expanded by all searches which find an optimal solution.

Consider the moment when for the first time the state of least f-value in OPEN has $f = \mathcal{C}^*$. From now on, all processed states have $f = \mathcal{C}^* = g + h^*$ and therefore are processed in τ-order. For a state s on a cost-optimal path to a goal we have $h^*_\epsilon(s) \leq h^*(s) + \epsilon\bar{d}$, where $\bar{d}$ is an upper bound on the distance from s to some goal, since a non-cost-optimal path from s to some goal costs at least $h^*(s) + \Delta$ for some $\Delta > 0$, and thus $h^*_\epsilon(s) \leq h^*(s) + \epsilon\bar{d} < h^* + \Delta$, by choice of ϵ.[1] Thus, for the state s of least τ-value we have $\tau(s) = g(s) + h^*_\epsilon(s) = g(s) + h^*(s) + \epsilon d^*(s)$ where $d^*(s)$ is the shortest distance from s to

[1] For integer costs, we can choose $\epsilon < 1/\bar{d}$.

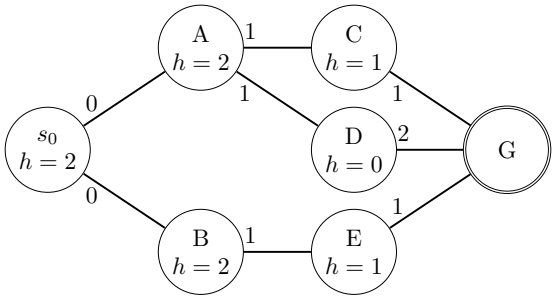

Figure 2. Example of an instance where order $[f, g + h^*_\epsilon]$ fails if h is inconsistent.

a goal on some cost-optimal path, and since $g + h^*$ is constant for all states with $f = C^*$, they are processed in d^*-order. Since each state of distance d^* has at least one successor of shortest distance $d^* - 1$ on a cost-optimal path, the distance to the goal decreases in each iteration, and A* expands exactly $d^* - 1$ states before processing a goal state. As d^* is the shortest distance on a cost-optimal path, optimal expansion follows. □

As a simple consequence of Theorem 1 we have that for A* with the perfect heuristic function h^*, tie-breaker $\tau = g + h^*_\epsilon$ has optimal expansion. Notice that optimal expansion does not imply that A* finds a shortest cost-optimal solution, since the shortest path is guaranteed only for the final f-layer.

Figure 2 illustrates an instance where order $[f, g + h^*_\epsilon]$ fails if h is inconsistent. Heuristic values are shown inside each state. To achieve optimal expansion the algorithm should expand paths $s_0 \to A \to C \to G$ or $s_0 \to B \to E \to G$. However, whenever we expand state A, we *must* expand state D as well. Due to the inconsistency of the heuristic function h, we have $f(D) < f(A)$ and $\arg\min_{s \in \mathsf{OPEN}} f(s) = D$, hence this successor must be expanded before any other successor of A. Since our tie-breaking strategy $g + h^*_\epsilon$ cannot guarantee to favor the expansion of B over the expansion of A, it does not guarantee optimal expansion if h is inconsistent.

3.2.3. Results

In order to verify the importance of our theoretical results, we experimentally compared the coverage of different tie-breaking strategies using $f = g + h^{\text{LM-cut}}$ to guide the search on the complete set of 1104 instances from the IPC and the 620 instances from zero-cost domains. We had limits of 4 GB and 5 min for each run, following [Asai and Fukunaga 2017] metrics. We used the Fast-Downward planner once again.

Table 2 shows the results. We compared our main cost adaptation methods against the standard methods in the literature and the current best deterministic tie-breaker from [Asai and Fukunaga 2017]. Looking at the group of tie-breakers using $h^{\text{LM-cut}}$ we find that that all methods using cost adaption perform better than the standard tie-breaker h. The second group using h^{FF} in the tie-breaker dominates the strategies using $h^{\text{LM-cut}}$ only. The h^{FF} heuristic [Hoffmann and Nebel 2001] is inadmissible, but it approximates h^* better than $h^{\text{LM-cut}}$ and, as we use it only as tie-breaker, it will not influence the optimality. The overall best method was h^{FF}_{+1}. It solved five instances more on IPC domains than $\hat{h}^{\text{FF}}$, the

Table 2. Comparison of the number of solved instances in IPC and zero-cost.

Method	**IPC** (1104)	**Zero-cost** (620)
$[f, h^{\text{LM-cut}}]$	525	237
$[f, \hat{h}^{\text{LM-cut}}]$	531	301
$[f, h^{\text{LM-cut}}_{+1}]$	530	299
$[f, h^{\text{LM-cut}}_{\epsilon}]$	532	301
$[f, g + h^{\text{LM-cut}}_{\epsilon}]$	524	300
$[f, h^{\text{FF}}]$	548	251
$[f, \hat{h}^{\text{FF}}]$	557	338
$[f, h^{\text{FF}}_{+1}]$	**562**	**352**
$[f, h^{\text{FF}}_{\epsilon}]$	559	351
$[f, g + h^{\text{FF}}_{\epsilon}]$	553	346
$[f, \hat{h}^{\text{FF}}, \langle d \rangle$, LIFO$]$	530	328

best tie-breaker from the literature. The best known tie-breaker for zero-cost instances is $[f, \hat{h}^{\text{FF}}, \langle d \rangle$, LIFO] [Asai and Fukunaga 2017]. Here, h^{FF}_{+1} solved 24 instances more.

4. Conclusion

Publications: This work resulted in three publications:

- Corrêa, A. B., Pereira, A. G., and Ritt, M. (2016). Improved airport ground traffic control with domain-dependent heuristics. In Brazilian Conference on Intelligent Systems (BRACIS).
- Corrêa, A. B., Pereira, A. G., and Ritt, M. (2018). Analyzing Tie-Breaking Strategies for the A* Algorithm. International Joint Conference on Artificial Intelligence (IJCAI-ECAI 2018).
- Corrêa, A. B., Pereira, A. G., and Ritt, M. (2018). Analyzing Tie-Breaking Strategies for the A* Algorithm. Workshop on Heuristics and Search for Domain-independent Planning – International Conference on Automated Planning and Scheduling (HSDIP Workshop – ICAPS).

Impact: Our work was divided in two distinct topics: the airport ground traffic control problem and analysis of tie-breaking strategies for A*. In the former part, our novel domain-dependent heuristics improved the performance of planners over the studied domain. Our results showed that heuristics which are successful in other transport domains also present good performance in an IPC domain. The latter part showed how to build A* with optimal expansion. For the first time since A* was proven the admissible best best-first search algorithm, we know which setting one must use to have an optimal A*. Furthermore, our empirical analysis demonstrated that our new techniques performed better than the state-of-the-art methods in literature and that, in fact, tie-breakers are more important to heuristic search as previously imagined.

Acknowledgment

This work was supported by FAPERGS as part of the project $17/2551 - 0000867 - 7$.

References

Asai, M. and Fukunaga, A. (2017). Tie-breaking strategies for cost-optimal best first search. *Journal of Artificial Intelligence Research*, 58:67–121.

Bonet, B. and Geffner, H. (2001). Planning as heuristic search. *Artificial Intelligence*, 129(1-2):5–33.

Corrêa, A. B., Pereira, A. G., and Ritt, M. (2016). Improved airport ground traffic control with domain-dependent heuristics. In *Brazilian Conference on Intelligent Systems*, pages 73–78.

Corrêa, A. B., Pereira, A. G., and Ritt, M. (2018a). Analyzing tie-breaking strategies for the A* algorithm. In *International Joint Conference on Artificial Intelligence*.

Corrêa, A. B., Pereira, A. G., and Ritt, M. (2018b). Analyzing tie-breaking strategies for the A* algorithm. In *Workshop on Heuristics and Search for Domain-independent Planning – International Conference on Automated Planning and Scheduling*.

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

Gliesch, A. and Ritt, M. (2016). Solving atomix with pattern databases. In *Brazilian Conference on Intelligent Systems*, pages 61–66.

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(2):100–107.

Haslum, P., Botea, A., Helmert, M., Bonet, B., and Koenig, S. (2007). Domain-independent construction of pattern database heuristics for cost-optimal planning. In *AAAI Conference on Artificial Intelligene*, pages 1007–1012.

Helmert, M. (2006a). The Fast Downward Planning System. *Journal of Artificial Intelligence*, 26:191–246.

Helmert, M. (2006b). New complexity results for classical planning benchmarks. In *International Conference on Automated Planning and Scheduling*, pages 52–62.

Helmert, M. and Domshlak, C. (2009). Landmarks, critical paths and abstractions: What's the difference anyway? In *International Conference on Automated Planning and Scheduling*, pages 162–169.

Hoffmann, J. and Nebel, B. (2001). The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302.

Pereira, A. G., Ritt, M., and Buriol, L. S. (2015). Optimal sokoban solving using pattern databases with specific domain knowledge. *Artificial Intelligence*, 227:52–70.

Pereira, A. G., Ritt, M., and Buriol, L. S. (2016). Pull and PushPull are PSPACE-complete. *Theoretical Computer Science*, 628:50–61.