A Problem with the Current Methodology for Comparing Search Algorithms and a Proposed Solution

Michael Barley[1], Natasha de Kriek[1], Santiago Franco[2], Angel Garcia-Olaya[3], Tim Hartill[1], Christopher Triggs[1], Henry Zwart[1], Vidal Alcázar[3], Patricia Riddle[1]

[1] University of Auckland, New Zealand,
[2] Royal Holloway University of London,
[3] Departamento de Informatica. Universidad Carlos III de Madrid, Spain,
m.barley@auckland.ac.nz, ndek000@aucklanduni.ac.nz, Santiago.FrancoAixela@rhul.ac.uk, agolaya@inf.uc3m.es, thar011@aucklanduni.ac.nz, cm.triggs@auckland.ac.nz, hzwa180@aucklanduni.ac.nz, valcazar@inf.uc3m.es, p.riddle@auckland.ac.nz

Abstract

This paper explores how incompletely described tie-break policies can invalidate the experimental results reported in papers on optimal bidirectional heuristic search (BiHS). Experiments usually use a single implementation of an algorithm with its specific tie-break policy. When the tie-breaks are insufficiently described, we show that the results can be irreproducible, vary dramatically under different implementations, and lead to misleading assessments of an algorithm's performance. To ensure reproducible and representative results, papers should either provide a description of the algorithm's implementation, i.e., the complete tie-break policy, or alternatively, give results as a summary statistic representative of all possible tie-break implementations. We developed a software tool for this purpose.

Introduction

In many papers, pseudocode describes optimal BiHS algorithms (Barker and Korf 2015; Barley et al. 2018; Kwa 1989; Pavlik, Sewell, and Jacobson 2021; Pohl 1969; Politowski and Pohl 1984; Sadhukhan 2013). The intention is to give sufficient detail such that algorithms are comparable by node expansion count. Implementation details that do not influence the number of expansions are omitted. One detail is the method for determining expansion order when nodes seem equally promising for exploration, i.e., are *tied*. This is called the *tie-break* policy. Most researchers acknowledge the possibly substantial impact of the tie-break policy on the number of nodes expanded (Bu and Korf 2022; Heusner, Keller, and Helmert 2018). They describe the portion believed to have the greatest impact. This is the *explicit* tie-break policy. Exactly how ties are broken is almost never completely described. This undescribed portion is the *implicit* tie-break policy. The explicit tie-break policy usually does not uniquely determine the node expansion order, whereas implicit tie-breaks do. That is the purpose of an implicit tie-break (Asai and Fukunaga 2016).

Unfortunately there is no guarantee that the implicit tie-break policy does not also have a substantial impact on expansions. This causes difficulty in interpreting the experimental results based on a single unknown implicit tie-break, as different implicit tie-breaks may cause the number of nodes expanded to vary substantially. Also, most analyses of these algorithms consider the lower bound (fewest expansions possible) and not the upper bound (most expansions possible). This paper's contributions are to:

1. Show that having implicit tie-break policies can essentially invalidate the experimental results reported.
2. Show that the smallest number of nodes expanded below C^*, does not guarantee the smallest total nodes expanded, where C^* is the optimal solution cost.
3. Demonstrate that in BiHS, implicit tie-breaks can affect whether an optimal solution is found below C^*, and thus have an effect on the number of expanded nodes.
4. Demonstrate that even when no optimal solution is found below C^*, expanding more nodes below C^* can reduce the search space available at C^*.
5. Present an algorithm, Multiple Episode Problem Solver (MEPS), that computes exact upper and lower bounds on the total number of expansions for a problem, showing it is possible to explore the distribution of the number of nodes that an algorithm expands using its explicit tie-break policy over all possible implicit tie-breaks. This is accomplished by exploring every expansion order of nodes consistent with the explicit tie-break.

The next section defines terms used in the paper, followed by a section describing our main claims. Then a section presents results from MEPS that support our claims. The last two sections present a discussion of results and conclude.

Background

In optimal unidirectional search, given (1) a weighted graph, $G = (N, E)$ where N is the set of states and E is the set of edges, (2) an initial state, s, (3) a goal state, t, and (4) a cost function, c, which maps edges to their non-negative costs, the problem is to find a minimal cost path in G between s and t. Edges are ordered pairs (n, m) where the edge goes from n to m. The cost of a path is the sum of the edge costs. The cost of the optimal solution is denoted by C^*. The least cost of any edge in E is denoted by ϵ.

In optimal heuristic search a heuristic, $h(n)$, estimates the minimal cost between a state n and the goal state. A *heuristic* is $admissible$ if it never overestimates the path cost between a state and the goal state. A heuristic search *algorithm* is $admissible$ if given an admissible heuristic, it is guaranteed to only return optimal solutions, i.e. minimal cost paths from the initial state to the goal state.

Hart, Nilsson, and Raphael (1968), wrote the first admissible optimal heuristic search algorithm, A^*. It uses an *open list* to store the nodes that are generated but not yet expanded. Expanding a node removes it from the open list and adds all adjacent nodes (children) to the open list. It stores expanded nodes in the *closed list*. A^* calculates $f(n)$, the estimated minimal cost of any path between the initial and the goal state going through the node n, as the sum of the cost, $g(n)$, to reach n from the initial state and the heuristic value $h(n)$. A^* uses the f-values of the open nodes to put them into equivalence classes and always expands nodes in the current lowest f-value equivalence class, C. The C^* level corresponds to the equivalence class with an f-value of C^*. Nodes expanded with f-values less than C^* are said to be expanded *below* the C^* level and nodes expanded with f-values equal to C^* are said to be expanded *at* the C^*level.

All nodes at the same level, i.e., with the same f-value, are considered *tied*. *Tie-break* is the selection of the next node to be expanded. U is the current lowest solution cost and C is the lowest f-value in the open list. A^* keeps expanding nodes with $f(n) \leq C$, updating U and C accordingly, until it eventually finds a solution with a cost $U \leq C$. If every node, n, where $f(n) < C^*$, has been expanded, this guarantees the optimality of A^*'s solution. Specifically in A^*, tie-break only affects the number of nodes expanded at the C^* level (Dechter and Pearl 1985; Russell and Norvig 2016). No unidirectional admissible algorithm, using the same information as A^*, can expand fewer nodes with f-values below C^*. This property is called *optimal efficiency*.

Unidirectional search can go forward from s towards t or backwards from t towards s. The path cost between s and n is $g_F(n)$, and between n and t is $g_B(n)$. Bidirectional heuristic search combines forward and backward heuristic search with two open lists (O_F and O_B) and two closed lists. Pohl (1969) wrote the first bidirectional heuristic search (BiHS) algorithm which combined A^* and bidirectional blind search (Dantzig 1963). Eckerle et al. (2017) gave sufficient conditions for guaranteeing admissibility for bidirectional heuristic algorithms. Chen et al. (2017) showed that no BiHS algorithm could be optimally efficient, only *nearly optimal* which means guaranteeing expanding no more than twice the minimum number of nodes below C^*.

In optimal bidirectional heuristic search, $h_F(n)$ estimates the minimal cost of any path from n to t and $h_B(n)$ estimates the minimal cost of any path from s to n. $f_F(n)$ is the estimated minimal cost of any path between s and t going through n. $f_B(n)$ is similarly defined. Unlike A^* which forms search levels based on f-values, state-of-the-art bidirectional heuristic algorithms (Alcázar, Riddle, and Barley 2020; Chen et al. 2017; Shperberg et al. 2021) form search levels from lower bound pairs $lb(x, y)$, with $x \in O_F$ and $y \in O_B$ (Eckerle et al. 2017). In recent years, many different forms of lower bounds have been invented (Alcázar, Riddle, and Barley 2020; Alcázar 2021; Siag et al. 2023a). They can be characterised as front-to-end (Pohl 1969), front-to-front (de Champeaux and Sint 1975; de Champeaux 1983), or somewhere in between (Alcázar 2021).

Algorithm 1: Generic BiHS Algorithm to Solve Problem P

```
1: function genericBiHS(P, h(), tieBreak()):
2: O_F, O_B, GLB, Episode, U ← initalize(P, h())
3: while O_F ≠ ∅ and O_B ≠ ∅ and U > GLB do
4:     EN ← nextFn(O_F, O_B, GLB)
5:     Pa ← tieBreak(EN)
6:     Ch ← expand(Pa)
7:     Episode ← [Pa|Episode]
8:     U, GLB, O_F, O_B ← update(Ch, Pa, O_F, O_B, h())
9: end while
10: return Episode
```

In this paper we only run a front-to-end BiHS (shown in Algorithm 1), because it is the most commonly used (Siag et al. 2023b) and therefore the best place to demonstrate the problem. The definition of lower bound that we use follows: $lb(x, y) = \max(f_F(x), f_B(y), g_F(x) + g_B(y) + \epsilon)$. The GLB is the lowest value of any open lower bound pair, since $lb(x, y)$ is symmetric. It is a lower bound for any solution to this problem. Lower bound pairs with lb-value $< C^*$ are called *must-expand pairs*, and one node of each pair must be expanded to guarantee optimality. Those pairs with lb-values $= C^*$ are *might-expand pairs*, as depending on the tie-break they might be expanded or not. Those with lb-values $> C^*$ are *cannot-expand pairs*, as a reasonable algorithm (Gilon, Felner, and Stern 2016) will never expand them. The two open lists form the two search frontiers and are the nodes from which the lower bound pairs are formed. The must-expand pairs form a bipartite graph of the two frontiers and the *minimum vertex cover (MVC)* is a lower bound on the number of expansions needed to guarantee the optimality of the solution (Chen et al. 2017). The lower bound pairs make equivalence classes like A*. All nodes in a lower bound pair whose value $\leq$ GLB are the current *expandable* nodes and you must use a tie-break policy to chose between them.

Algorithm 1 shows a generic BiHS search with input parameters problem P, and functions $h()$, and $tieBreak()$. Line 4 computes the set of expandable nodes, (EN), line 5 selects the parent node, Pa, to be expanded using explicit and implicit tie-breaks. As the code loops, it expands a sequence of nodes called an *expansion sequence* (Corrêa, Pereira, and Ritt 2018). An *episode* is an expansion sequence beginning with the first node expanded, i.e., the initial state or the goal state, and ending the first time a solution has been shown to be optimal. The algorithm returns the episode created using the $tieBreak$ policy.

The Importance of Implicit Tie-Breaking

The field's experience with A^* may have made us blasé about tie-breaking. With A^*, tie-break *cannot* affect the number of nodes expanded below C^* (Dechter and Pearl 1985), *cannot* affect the number of expandable nodes at C^*,

	Total Nodes Expanded (te)																									
te	4		5		6		7		8		9		10		11		12		13		14		15			
VC	|te|	Psum	|te|	Psum	|te|	Psum	|te|	Psum	|te|	Psum	|te|	Psum	|te|	Psum	|te|	Psum	|te|	Psum	|te|	Psum	|te|	Psum	|te|	Psum	WMean	PMean
2	4	0.01	31	0.01	156	0.01	657	0.01	2750	0.00	11213	0.00	43435	0.03	168952	0.05	622944	0.04	1859760	0.04	3825360	0.02	3991680	0.02	14.0 (1.02)	11.0 (2.91)
3	4	0.00	20	0.03	64	0.02	136	0.02	240	0.01	480	0.00	480	0.00											8.8 (1.24)	6.4 (1.32)
5					120	0.02	300	0.19	720	0.13	1440	0.06	1440	0.04											8.9 (1.05)	7.8 (1.03)
7									270	0.03	1350	0.02	6120	0.02	24840	0.01	86400	0.01	237600	0.00	453600	0.07	453600	0.05	13.9 (1.05)	12.2 (2.66)

Table 1: Vertex Cover (VC) for the 4-step 4-Pancake Problem (2, 4, 1, 3, 5), gLim tie-break, GAP heuristic at Degradation 1

and *cannot* affect whether an optimal solution is found while expanding nodes in an equivalence class below the C^* level. This is because, in A^*, expanding any open node has no effect on the f-value of any other open node. As we will see, this all changes when we use must-expand pairs in BiHS. Expanding a node in one open list can raise the lower bounds of nodes in the opposite open list. With must-expand pairs, tie-break *can* affect the number of expansions below C^*, *can* affect the number of expandable nodes at C^*, and *can* affect whether an optimal solution is found below C^*. This has ramifications for how experimental results should be reported when using implicit tie-breaks.

In BiHS papers, the tables and charts in the experiments' section normally include the total number of nodes that an algorithm expands when solving a problem and/or the number of nodes expanded below C^*. We suggest that the total number of expansions reported in experiments may misrepresent the performance of an algorithm on a single problem, when using implicit tie-break policies. But what about the number of expansions below C^*? How are we supposed to interpret the reported number of expansions below C^*? One possible interpretation assumes that fewer expansions below C^* leads to fewer total expansions. Some tie-break policies specifically try to expand as few nodes below C^* as possible (Alcázar, Riddle, and Barley 2020; Siag et al. 2023a). They do this to raise the GLB as quickly as possible to C^*. Optimal bidirectional search algorithms have to keep expanding nodes as long as the GLB remains below C^* and if the C^* level is reached faster then the search might stop sooner, i.e., expand fewer total nodes. However, as we see in Claims 3 and 4, this is not always true, i.e., expanding more nodes below C^* can lead to fewer nodes expanded overall.

In the sections below we analyse problems in the Pancake Domain (Gates and Papadimitriou 1979). The pancake domain represents a stack of pancakes in some random order. The only action is to put the flipper between any two pancakes (or the bottom pancake and the table) and the action inverts all the pancakes above the flipper. Our pancake domain is unit cost and problems referred to by their initial state. The (4,2,3,1,5) problem represents 4 pancakes where the table is represented by 5 and pancake 1 is on the bottom of the stack on the table, pancake 3 is on pancake 1, and so forth. The goal state has the pancakes in ascending order on the table, (1,2,3,4,5). The heuristic is the symmetric GAP heuristic, which calculates the GAP (Helmert 2010) heuristic in both directions and takes the max. The paper shows degradations 0, 1 and 2. In the pancake domain, degradation 0 does not ignore any pancakes to calculate the distance with the GAP heuristic. In degradation 1 in the forward direction, the top pancake in the goal state is ignored and in the backward direction, the top pancake in the initial state is ignored. Degradation 2 is similarly defined.

We implemented a prototype, MEPS for investigating the impact of implicit tie-breaks upon performance, which generated all the data in this paper. MEPS takes a problem, a heuristic, an algorithm, and an explicit tie-break policy and finds every episode. Not all episodes are equally probable. This is not important if we expand the complete search tree, but is important if you sample the tree. Assuming there are N expandable nodes at vertex x then the probability of a specific node, n, being selected is $1/N$ and the probability of a specific episode, $E = < v_0, v_1, v_2, \ldots, v_m >$ with $m+1$ nodes, being expanded, $P(E) = 1/\prod_{i=0}^{m} |EN_i|$, where $|EN_i|$ is the number of expandable nodes at vertex v_i. For example, given the sequence of expanded nodes $< v_0, v_1 >$ where $|v_0| = 2$, $|v_1| = 3$, the probability of expanding this sequence is 1/6. The probabilistic mean, PMean, is:
$(\sum_{te=te_{min}}^{te_{max}} te * \mathrm{Psum}(te)) / (\sum_{te=te_{min}}^{te_{max}} \mathrm{Psum}(te))$.
te is the total nodes expanded. $E(te)$ is the set of all episodes that expand exactly te nodes. $\mathrm{Psum}(te)$ is the probability that exactly te nodes are expanded: $\sum_{e \in E(te)} P(e)$.
$|te|$ is the number of episodes that expanded exactly te nodes. te_{min} and te_{max} are the minimum and maximum total number of nodes expanded, respectively. The weighted mean (WMean) is calculated analogously, with the probabilities, $\mathrm{Psum}(te)$, replaced by the number of episodes with te nodes expanded, $|te|$.

Tables 1 and 2 describe all episodes for the 4-step 4-pancake problems (2, 4, 1, 3, 5) and (4, 2, 3, 1, 5) using degradation 1 and the explicit tie-breaks gLim (defined in Claim 4) and optimal-solution-first (OSF)[1] respectively. OSF is used by all modern optimal heuristic search algorithms. Each episode represents a unique implicit tie-break. The row heading VC=n means there were exactly n nodes expanded below C^*, e.g., for row VC=5, all episodes reported on that row had 5 nodes expanded below C^*. The second line of the column headings are the total nodes expanded, e.g., from 4 nodes to 15 nodes in Table 1. The third line alternates between the $|te|$, and the $\mathrm{Psum}(te)$ of te nodes being expanded. The last 2 columns in Tables 1 and 2 are WMean and PMean. The standard deviations are given in parenthesis after the means and (), means there is only 1 value and therefore 0 standard deviation.

For example, row VC=5 in Table 1 shows 120 episodes which expand 6 nodes, *te* = 6, 300 episodes that expand 7 nodes, 720 episodes that expand 8 nodes, etc., up to 1440

[1]This tie-break always picks an expandable node that guarantees an optimal solution has been found. This was the tie-break used in (Hart, Nilsson, and Raphael 1968).

	Total Nodes Expanded (te)																			
te	6		7		8		9		10		11		12		13		14			
	|te|	Psum	|te|	Psum	|te|	Psum	|te|	Psum	|te|	Psum	|te|	Psum	|te|	Psum	|te|	Psum	WMean	PMean
VC=5	266	0.08	532	0.02	532	0.00													7.2 (0.75)	6.3 (0.49)
VC=6	318	0.00	78	0.00	234	0.00	816	0.26	2592	0.02	5472	0.01	5760	0.01					10.9 (1.22)	9.2 (0.74)
VC=7			1308	0.46															7.0 ()	7.0 ()
VC=8					693	0.07													8.0 ()	8.0 ()
VC=9							1236	0.01	2880	0.01	11520	0.00	34560	0.00	69120	0.00	69120	0.04	13.0 (1.02)	12.5 (2.09)
VC=10									720	0.01									10 ()	10 ()
VC=11											576	0.00							11 ()	11 ()

Table 2: Vertex Cover (VC) for the 4-Step Pancake Problem (4, 2, 3, 1, 5) with OSF tie-break, GAP heuristic at Degradation 1

episodes that expand 10 nodes. te_{min} is 6 and te_{max} is 10. The mean number of nodes expanded when all episodes are considered equally probable is the *weighted* mean (WMean), e.g., $(6{*}120{+}7{*}300{+}8{*}720{+}9{*}1440{+}10{*}1440)/(120{+}300+720+1440+1440) = 35940/4020 \approx 8.9$ The *probabilistic* mean (PMean) is the mean number of nodes expanded when the episodes are sampled by their probability of being selected. It is the sum of the expansions' probability times its number of total expanded nodes divided by the sum of the probabilities in that row, e.g., for VC=5 in Table 1 it is $(6*.02+7*.19+8*.13+9*.06+10*.04)/(.02+.19+.13+.06+.04) = 3.42/.44 \approx 7.8$.

Shorter episodes are more likely to be found than longer episodes, therefore probabilistic means will usually be lower than weighted means. The probabilistic mean is closer to the results you will get if you sample the space by just running the BiHS algorithm and sample the implicit tie-breaks, while the WMean is closer to probability of implementing a single implict tiebreak. The WMean is equivalent to choosing an episode from the space without replacement, whereas the PMean is equivalent to choosing an episode from the space with replacement (the shorter episodes have a larger impact).

Claim 1: Implicit tie-break policies can essentially invalidate the experimental results reported

In A^*, implicit tie-breaks make no difference in the number of expansions below C^* but can make a tremendous difference at C^*. In BiHS algorithms, implicit tie-breaks can also make a difference in expansions below C^* and that, in turn, can make a difference in the number of expansions at C^*. The larger this difference, the more misleading the results.

Table 1 shows that depending on which implicit tie-break is implemented, anywhere from 4 to 15 total nodes can be expanded. Therefore, if the experimental results show only one tie-break implementation (i.e, only one value is produced) they can be misleading.

Claim 2: Smallest VC does not guarantee smallest total expansions

Table 1 shows that expanding the smallest VC does not guarantee expanding the fewest nodes overall. For pancake problem (2, 4, 1, 3, 5) using the gap heuristic, degradation 1, the gLim tie-break can expand 2, 3, 5, or 7 nodes below C^*. When the VC is smallest, VC=2, then the probabilistic mean is 11.0. The probabilistic means for VC=3 and 5 are both lower than for VC=2 at 6.4 and 7.8 respectively. This clearly shows that for some tie-breaks, expanding the smallest VC does not ensure expanding the fewest nodes overall.

The next two claims look at how tie-breaks affect the number of expansions below C^* and how that difference can affect the number of nodes expanded at C^*. Implicit tie-breaks affect how many nodes are expandable in two ways: (1) The implicit tie-break can enable an optimal solution to be found below C^* meaning no nodes are expanded at the C^* level. (2) Even when a solution cannot be found below C*, if an implicit tie-break can expand more nodes below C*, then some nodes at the C* level can be pushed into the cannot-expand list.

Claim 3: Implicit tie-breaks can affect whether an optimal solution is found below C^*

If an optimal solution is found below C^* then no nodes can be expanded at C^*. If the search space grows exponentially with respect to the GLB then more nodes may be expandable at C^* than the total number of nodes expanded below C^*. Thus if one episode finds an optimal solution below C^* while another finds it at C^* then the former can expand far fewer nodes overall than the latter. In BiHS, the tie-break can affect whether an optimal solution is found below C^*.

If an optimal solution is found below C^*, then this episode must appear in the table in a diagonal entry. In Table 2, diagonal table entries are underlined. In row VC=5, since $C^* = 5$ and there is no $te = 5$ column, there are no episodes where all nodes are expanded below C*. All the rows with a VC>5, have some episodes where all nodes are expanded below C^*. For example, VC rows 7, 8, 10 and 11 only have episodes where an optimal solution is found below C*. VC rows 6 and 9 show that the tie-break generating an episode affects whether an optimal solution is found below C^* or not. For example, if exactly 9 nodes are expanded below C^* then an optimal solution can be found below C^* in 1,236 episodes but in all other 187,200 episodes with 9 nodes expanded below C^*, the first optimal solution is found at C^*. This shows a problem where some implicit tie-breaks (episodes) find an optimal solution at C^* and others that find it below C^*, thus providing an existence proof for Claim 3.

Claim 4: Implicit tie-breaks can reduce the size of the search space available at C^*

Previously we saw that the implicit tie-break can affect whether an optimal solution is found below C^*, which can reduce the number of expansions. However, even when no solution is found below the C^* level, expanding more nodes below C^* can reduce the overall number of expansions. Looking at Table 1, there are no episodes where an optimal solution is found below C^*, i.e., there are no diagonal

TieBreak	Problems and Degradations					
	D0		D1		D2	
	$\|te\|$	PMean	$\|te\|$	PMean	$\|te\|$	PMean
alter	52	4.5 (0.67)	2804	**6.1** (1.64)	216	7 ()
gLim	4832	4.9 (1.00)	13533992	7.6 (2.22)	–	–
pohl	216	4.5 (0.68)	93708	6.2 (1.68)	1728	7 ()
uniFw	1476	5.8 (1.34)	10536696	8.6 (2.96)	–	–
uniBw	1476	5.8 (1.34)	2964540	10.7 (1.94)	–	–
nbs	128	4.7 (0.56)	3716	**6.1** (1.46)	1728	7 ()
lifo	16	**4.2** (0.38)	64	8.8 (2.20)	224	12.4 (1.67)
fifo	28	5.3 ()	100	6.2 (0.38)	132	**6.2** (0.40)
dvcbsFifo	56	5.3 ()	104	6.2 (0.26)	264	**6.2** (1.82)
eGBFHS	280	4.9 (0.60)	571	**6.1** (0.66)	336	**6.2** (0.42)

Table 3: Comparison of Probabilistic Means over all Completed 4-step 4 pancake problems.

entries. If only 2 nodes are expanded below C^*, anywhere from 4 to 15 total nodes can be expanded and the probabilistic mean expansions is 11. If 3 nodes are expanded below C^* then only 4 to 10 total nodes can be expanded and the probabilistic mean is only 6.4. This means that by expanding one more node below C^* the upper bound on expansions at the C^* level reduces from 15 to 10. This can happen because which nodes on a side can be expanded depend upon the nodes that are in the open list on the other side. So, expanding a forward node removes it from the forward open list which can make some backward nodes non-expandable. This is an existence proof for Claim 4 that by expanding more nodes below C^*, an implicit tie-break can reduce the number of nodes that are expandable at the C^* level.

Claim 5: MEPS - A Proposed Solution

Our claim is MEPS can find exact upper and lower bounds on the number of expansions for a problem, as well as providing mean numbers of expansions, standard deviations, and the distribution of how often different numbers of nodes could be expanded in solving the problem. This can be seen in Table 3 which shows the sum of episode counts, the average probabilistic means and standard deviations over all 3 of the 4-step 4-Pancake problems[2] at 3 degradation levels. The table shows the performance of 10 different tie-breaks.

The tie-breaks used are as follows. **alter** (called Alternating in (Nicholson 1966)), is satisfied if the episode only selects nodes from the opposite direction from the last direction in which a node was expanded. **gLim** (Barley et al. 2018) is satisfied if the episode only selects nodes that do not cause the frontiers to cross[3]. **pohl** (called Pohl's Cardinality Comparison principle in (Pohl 1969)) is satisfied if the episode only selects nodes from the direction with the smallest open list. **nbs** (Chen et al. 2017) is satisfied if when an lb-pair is chosen both nodes are expanded. **dvcbs** (Shperberg et al. 2019) is satisfied if all expanded nodes were chosen from the smallest set of open nodes with the smallest g values. **dvcbsFifo** is satisfied if the expanded nodes always satisfy **dvcbs** and the node's parent was the earliest expanded node of the set of nodes that satisfied **dvcbs**. **eGBFHS** is inspired by Alcazar's eager **GBFBH**$_e$ (Alcázar, Riddle, and Barley 2020), the only difference is that **eGBFHS** uses meet-in-the-middle as the split function where **GBFHS**$_e$ uses pohl. Our eager version, like **GBFHS**$_e$, raises the g limits as soon as the f limit is raised. **eGBFHS** is satisfied if both **gLim** is satisfied and the collision happens in the middle. Note that both **fifo** and **lifo** are standard implementations but consider the expansion number of the node's parent (so all children of the same parent have the same expansion number). This avoids the implicit tie-break of node generation ordering. **uniFw** and **uniBw** unidirectional forward and backwards search are each satisfied if all nodes are expanded in the forward or in the backward direction respectively.

Table 3 gives interesting insights into which tie-breaks do better or worse. For instance, for degradation 1, fifo, dvcbsFifo and eGBFHS have much smaller standard deviations showing there is less impact from the implict tiebreaks, whereas for degradation 2, alter, pohl and nbs, show no impact from implicit tiebreaks. Notice that some of the results for tie-breaks at degradation 2 didn't finish, shown as a "-" in the table. Since there are over a billion episodes in the space for those tiebreaks, we need a more practical solution. The next few sections describe different ways to use MEPS.

Exploring all Episodes Given a problem, P, a heuristic $h()$, and the explicit tie-break policy, $tieBreak()$, MEPS calls Algorithm 1 to generate every episode. This version backtracks after every episode is found, until all unique episodes are found. Obviously, this is very computationally expensive, so to reduce cost, much of the information needed to solve the problem is pre-computed and cached before MEPS starts running. For example, instead of MEPS generating the search graph, MEPS is given an explicit representation of the search graph. MEPS is implemented in Prolog to take advantage of its inbuilt backtracking capability.

Preprocessing: Generating the Data for MEPS We generate the explicit graph for MEPS from two A^* searches: one forward from the initial state, and one backward from the goal. We terminate A^* once all nodes with f-values $\leq C^*$ have been expanded. The visible search space is defined by the closed lists of the forward and backward A^* searches. It provides directional g-values, h-values, and successor relationships for the nodes, and allows us to compute lower-bound pairs. The per-node g/h-values, successor relationships, and pairwise lower-bounds are computed per-problem and serve as an explicit graph for MEPS.

Sampling The number of episodes, even for small problems, can be astronomical. Each episode involves a search for an optimal solution. Exhaustive search for episodes is infeasible for many problems. The number of episodes that satisfy strong tie-breaks can reduce the number of episodes. Sampling can also be used to reduce the number of episodes returned. The sampling version simply calls Algorithm 1 with a seed and terminates after an episode (a sample) is

[2]There are only 3 4-step 4-pancake problems. Their initial states are (2,4,1,3,5), (4,2,3,1,5) and (3,1,4,2,5).

[3]Specifically, let x be the max g-value of any expanded node on the other side, then the g-value of a selected node, n, cannot be such that $x + g(n) > C - \epsilon - 1$.

TieBreak	Probabilistic Means (PMeans) for SubSamples for problem (2, 4, 1, 3, 5) degradation 2										Whole	Episode
	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	Sample	$\|te\|$ at
	10,000	10,000	10,000	10,000	10,000	10,000	10,000	10,000	10,000	10,000	100,000	Term.
gLim	9.8 (3.67)	9.8 (3.66)	9.8 (3.64)	9.8 (3.64)	9.8 (3.64)	9.8 (3.67)	9.8 (3.68)	9.8 (3.68)	9.8 (3.65)	9.8 (3.65)	9.8 (3.65)	1.2b
uniFw	12.2 (2.96)	12.1 (2.97)	12.2 (2.99)	12.2 (2.96)	12.2 (2.98)	12.2 (2.98)	12.2 (2.99)	12.2 (2.97)	12.2 (2.98)	12.2 (2.95)	12.2 (2.97)	1.1b
uniBw	14.1 (2.46)	14.0 (2.46)	14.0 (2.42)	14.0 (2.43)	14.0 (2.45)	14.0 (2.46)	14.0 (2.44)	14.0 (2.45)	14.0 (2.44)	14.0 (2.47)	14.0 (2.45)	1.2b

Table 4: Comparison of Sampled PMeans over gLim, uniFw, uniBw for the 4-Step 4-Pancake Problem (2, 4, 1, 3, 5)

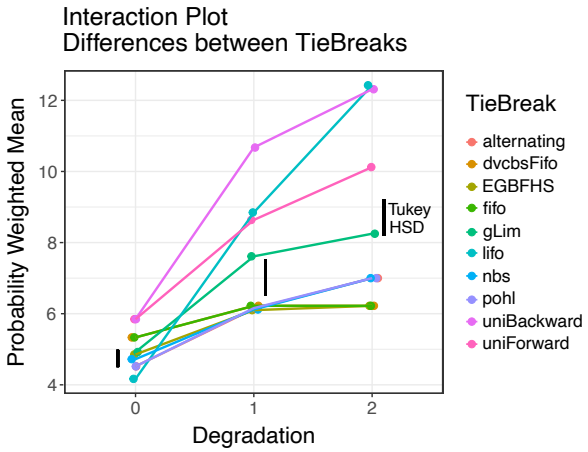

Figure 1: Tie-break Comparison Across Degradations over all 4-step 4 pancake problems

found. At the beginning of each new sample, we increment the seed of the random number generator. At each node selection point, we randomly chose a node from the set of expandable nodes, using the seed to sort the set. The episode is chosen randomly from the space of episodes that satisfy the explicit tie-break.

Sampling Results

Table 5 shows the same tiebreaks averaged over the same problems, where the episodes are sampled from the space instead of expanding the entire space. We sampled 100,000 episodes for all 3 4-step 4-pancake problems for all 10 tie-breaks at 3 degradations.

When we sample, rather than completely enumerate all possible outcomes, we need to consider whether our sample size is large enough. To investigate this question we considered the three tie-breaks where we could not enumerate the entire space. We only show these three tie-breaks because they were the larger ones and therefore more likely to be undersampled. The 100,000 episodes were split into 10 batches of 10,000 episodes without replacement. Table 4 gives for each tie-break the mean and standard deviation for the whole set of episodes and for each batch. We note that all the means for a given tie-break are very similar. The range, (max - min) across each tie-break, are 0.080, 0.080, and 0.110 for the three tie-breaks respectively. The standard deviations of each batch are also consistent, PMeans all have ranges of 0.05. The uncertainty of a sample mean is usually summarised by its standard error, the standard deviation divided by the square root of the sample size. In Table 4 the average standard error of the mean is 0.030. This shows that the uncertainty in the batch means falls in the second decimal place, the third or fourth significant digit. The standard errors of the means of the whole ensemble of size 100,000 are 0.0115, 0.0044, and 0.0077 for the three tie-breaks. The last column of the table shows the total episode count when the paper was submitted after more than 1,000 CPU hours each. Even though there are over a billion episodes, the 100,000 that were sampled still summarised the space accurately. Thus we see sampling is a practical future strategy.

Table 4 shows that MEPS gives additional information for unidirectional algorithms. For uniFW the VC=8 and the range is 9 to 17. So more information can now be derived to analyse unidirectional search algorithms as well.

Notice also that even though some of the tie-breaks are oversampled. For instance from Table 3 we see there are only 52 episodes combined over all the 4-step 4 pancake problems for the alter tie-break at degradation 0. We sampled 100,000 episodes for each of the 3 problems and the probabilistic mean (4.5) is exactly the same, so oversampling is not an issue.

Tie Break	Problems and Degradations					
	D0		D1		D2	
	Compl.	Sampled	Compl.	Sampled	Compl.	Sampled
alter	4.5	4.5 (0.65)	**6.1**	6.1 (1.08)	7.0	7.0 ()
gLim	4.9	4.9 (0.92)	7.6	7.6 (2.20)	–	9.4 (3.24)
pohl	4.5	4.5 (0.65)	6.2	6.2 (1.05)	7.0	7.0 ()
uniFw	5.8	5.9 (1.17)	8.6	8.6 (2.15)	–	11.5 (2.18)
uniBw	5.8	5.9 (1.17)	10.7	10.7 (1.73)	–	13.3 (2.01)
nbs	4.7	4.7 (0.52)	**6.1**	6.1 (0.96)	7.0	7.0 ()
lifo	**4.2**	4.2 (0.35)	8.8	8.9 (2.32)	12.4	12.4 (1.64)
fifo	5.3	5.3 ()	6.2	6.2 (0.38)	**6.2**	6.2 (0.38)
dvcbsFifo	5.3	5.3 ()	6.2	6.2 (0.38)	**6.2**	6.2 (0.38)
eGBFHS	4.9	4.9 (0.54)	**6.1**	6.1 (0.57)	**6.2**	6.2 (0.38)

Table 5: Comparison of Complete (Compl.) to Sampled PMeans over 4-step 4 pancake problems

Statistical Comparisons of Tie-breaks

To draw firm conclusions as to which tie-breaks are better than others it is necessary to carry out a statistical signifi-

TB \ DG	0	1	2	Ave.	Group
	PMean	PMean	PMean	PMean	
eGBFHS	4.9	6.1	6.2	5.7	A
alter	4.5	6.1	7.0	5.9	A
pohl	4.5	6.2	7.0	5.9	A
dvcbsFifo	5.3	6.2	6.2	5.9	A
fifo	5.3	6.2	6.2	5.9	A
nbs	4.7	6.1	7.0	5.9	A
gLim	4.9	7.6	8.2	6.9	B
uniFw	5.8	8.6	10.1	8.2	C
lifo	4.2	8.8	12.4	8.5	C
uniBw	5.8	10.7	12.3	9.6	D

Table 6: Variance of PMeans on all 4-step 4-pancake probs

cance test as to whether there is evidence of differences between tie-breaks, and then some multiple comparisons procedure to determine which pairs of tie-breaks are significantly different. We carry out a two-way analysis of variance comparing level of degradation and tie-break regarding our three problems as blocks. Note that we wish to compare a two-way array of means, Tie-break x Degradation. There is no two-way analog of the Friedman rank test (Conover 1999), which is widely used to compare one-way arrays of means. The residuals from the fitted analysis of variance model were examined graphically to confirm that the underling assumptions of analysis of variance, constancy of variance and at least approximate normality of the distribution of residuals, were satisfied. All of the fitted means; between levels of degradation, between tie-breaks, and the degradation x tie-break interaction were highly significant, (p-value < 0.001 respectively). The significant interaction provided strong evidence that the differences between tie-breaks was not consistent between the different levels of degradation. The results are displayed graphically in Figure 1, which shows the Tukey Honest Significant Difference, HSD, (Welham et al. 2014) to compare tie-break PMeans at each level of degradation. In Table 6 the tie-breaks are ordered by their PMean across all 3 degradations. Note that lifo has the lowest PMean at degradation 0, 4.2 but then does badly at degradation 1 and 2, ranking second to worst. The final column in Table 6 compares tie-breaks averaged over the three levels of degradation and groups the tie-break methods into four non-overlapping groups. All the PMeans of all the methods within a group do not differ significantly (p-value < 0.05) as indicated by the Tukey HSD. In Tables 3 and 5 we put boldface on the best PMeans at each degradation, specifying which algorithm did better, but notice that the variance analysis shows that any difference is just minor variance and these tie-breaks are actually statistically the same. Tie-breaks in different groups are statistically different and you can claim one is better than the other.

Additional Results on Larger Problems

Earlier we looked at a few problems. In this section we look at how common the behaviour is over a larger number of problems. We report problem/tiebreak/degradation combinations that completed the entire search space for 4-pancake and sliding tile problems. We also report additonal 5-step 5-pancake problems that were sampled 100,000 times. The *impact* of a tie-break measures how misleading the reported results can be.[4]

Claim 1: The largest range of total expansions for the bidirectional tie-breaks in 4-pancake and 5-pancake problems was for lifo at 42 with a max value of 50 giving an impact of 84%. alter had a range of 13 and an impact of 76%. nbs, fifo, and pohl had a range of 11 with impacts of 73%, 50% and 69% respectively. gLim had a range of 10 with an impact of 77%. dvcbsFifo had a range of 8 with an impact of 42%. eGBFHS had a range of 6 with an impact of 60%. For the sampled 5-pancake problems this increased to a range of 89 and an impact of 89% for gLim. lifo had a range 65 with an impact of 65%. pohl had a range of 35 and an impact of 83%. alter had a range of 34 with an impact of 83%. nbs had a range of 25 and an impact of 78%. eGBFHS had a range of 13 and an impact of 59%. DvcbsFifo and fifo had a range of 11 and an impact of 50%. Implicit tie-breaks frequently give misleading results.

Claim 2: Over 30% of the 1582 problem / tie-break / degradation combinations which had more than one VC, found a lower PMean at a VC level larger than the minimum VC. This occurred more often in the sliding tiles problems, 35%, but also in the pancake problems, 24%. In both domains, it occurred in all bidirectional tie-breaks. In the 5-step 5-pancake problems which we sampled, this occurred in over 55% of the combinations. Smallest VCs frequently do not give the fewest overall expansions.

Claim 3: Over 16% of the 4116 bidirectional problem/tie-break/degradation combinations at a C^* of 4 or 5 could find a solution below C^* AND find a solution at C^* (due to variance in the implicit tie-break). This did not occur with a $C^* < 4$. This occurred in both domains and increased in frequency at the higher degradations. It occurred in 15% of the $C^* = 4$ and 18% of the $C^* = 5$. In both domains, it was restricted to only certain bidirectional tie-breaks: alter, nbs, pohl. It occurred in 34% to 49% of the cases in these tie-breaks. In the 5-step 5-pancake problems which we sampled, this occurred in 17% of the combinations and 39% to 40% of the cases in the 3 affected tie-breaks. So, in 3 of the 10 tie-breaks, an implicit tie-break frequently affects whether an optimal solution was found below C^*.

Claim 4: Over 14% of the 1582 problem / tie-break / degradation combinations, which had more than 1 VC, had a lower te_{max} at a higher VC level than at a lower VC. This occurred in both domains, but only when the C^* value was 3 or higher and occurred in over 26% of the problems with a C^* of 5. In both domains, only dvcbsFifo did not show this behaviour. In the 5-step 5-pancake problems which we sampled, this occurred in over 56% of the combinations. So, frequently expanding more nodes below C^* can lead to fewer nodes being expanded over all.

Discussion

This paper brings out the following issues that need further exploration: (1) When is it better to expand more nodes below C^*; and (2) How do we tell if the tie-breaks are sufficiently described to make the experiments repeatable and reliably evaluate algorithms' performances. These are discussed in turn, after we briefly review the related research.

Related Research

Not much research has been done on tie-break policies in unidirectional search (e.g., A^*). When A^* (Hart, Nilsson, and Raphael 1968) was introduced, the authors proved that tie-break had no impact on the number of expansions done below C^*. The only tie-break policy mentioned was that ties could be broken arbitrarily but always in favor of

[4]The complete results, problems, and code are given in the repository: https://github.com/pjriddle/bidirectional-search.

goal nodes (OSF). This tie-break policy meant that for non-pathological instances, adding other tie-policies would have no effect on the total number of expansions (Dechter and Pearl 1985). Unfortunately, many interesting problem instances are pathological and (Bu and Korf 2022) state "Tie-breaking among nodes of equal f-value significantly affects the set of expanded nodes whose f-value equals C^*."

Until recently (Asai and Fukunaga 2016, 2017; Corrêa, Pereira, and Ritt 2018; Corrêa 2018; Heusner, Keller, and Helmert 2018; Rathfux et al. 2019) there has been little research into the impact of tie-breaking policies on node expansions in unidirectional search. However, tie-breaking has been an essential part of bidirectional algorithms from the beginning. Nicholson (1966) proposed the alternating tie-break policy for blind bidirectional search. Pohl (1969) proposed Pohl's Cardinality principle. These were the main tie-break policies used until Eckerle et al. (2017) proved sufficient conditions for the admissibility of BiHS algorithms which provided the basis for new tie-breaking policies (Eckerle et al. 2017; Chen et al. 2017; Barley et al. 2018; Alcázar, Barley, and Riddle 2019; Alcázar, Riddle, and Barley 2020; Pavlik, Sewell, and Jacobson 2021; Hu and Speck 2022; Siag et al. 2023a).

Breaking ties in favor of nodes with higher g-values (Hansen and Zhou 2007), suggests that the g-value component of f-values are known costs, while the h-value component is only an estimate and more likely to be inaccurate. Ties can be broken not only by h, cost-to-go, but also by d, distance-to-go (Thayer and Ruml 2009). Using d for tie-breaks can reduce the size of the search tree.

In BiHS, tie-breaks can focus on another choice, which frontier to expand next. The first idea was that the number of expansions can be reduced by finding the right balance between expanding the forwards and backwards frontiers (Nicholson 1966; Pohl 1969; Chen et al. 2017; Barley et al. 2018; Alcázar, Barley, and Riddle 2019; Alcázar, Riddle, and Barley 2020; Pavlik, Sewell, and Jacobson 2021; Hu and Speck 2022; Siag et al. 2023a). Expanding fewer nodes below C^* also affects how quickly the GLB is raised to the C^* level. Unfortunately, as we have seen in this paper, sometimes delaying raising GLB to the C^* level, by expanding more nodes below C^*, is a better strategy.

When to Expand More Nodes below C^*

We have shown that tie-break policies can affect the number of expansions in two ways. (1) It can affect whether the optimal solution is found below C^* and consequently not expand any nodes at the C^* level; and (2) A tie-break policy, by expanding more nodes below C^*, can push more nodes beyond the C^* level and expand fewer nodes overall. Are these two behaviors connected and if so by what? The answer seems to be they are both connected to the strength of the heuristic. The weaker the heuristic, the more often an optimal solution is found below C^* and the stronger the heuristic the more often expanding more nodes below C^* could raise the lower bounds of more nodes on the opposite side beyond C^*, to reduce the overall number of expansions.

However, it is obvious that expanding more nodes below C^* can also increase the overall number of expansions. So, we also need to identify the conditions when this is true. One such condition is whether an optimal solution has already been found, in which case expanding the fewest nodes below C^* to raise the GLB to C^* is best. Another such condition could be when expanding a node below C^* does not push any nodes on the other side beyond C^*. Further analysis must be done to determine the best strategy.

How to tell if tie-breaks are adequately described?

While we have argued that implicit tie-break policies can make the experiments irreproducible and the results misleading, this is not always the case. In Table 3 using the Gap, heuristic degradation 2, and using **alter**, **pohl**, or **nbs** explicit tie-breaks, the PMean expansions would be 7.00 with a standard deviation of 0.00. This means regardless of which implicit tie-breaks were used, the answer would have been the same. Likewise for degradation 0 using either **fifo** or **dvcbs-Fifo** explicit tie-breaks. In these cases, a single implemented implicit tie-break would have been enough to get reliable results. But these are exceptions, in 25 of the 30 PMean table entries, using a single implemented implicit tie-break would have given mis-leading results. The problem is, how do you tell whether the explicit tie-breaks are sufficient to give reliable results? MEPS can answer this. The episodes will probably need to be randomly sampled, but only sampled enough until either a different result is gotten or the same result has been gotten sufficiently often that there is confidence that continued sampling is unlikely to yield different results.

Conclusions and Future Research

This paper explores whether it is possible for implicit tie-break policies to essentially invalidate the experimental results reported in papers by making them irreproducible and enabling the results to give misleading comparisons between algorithms by giving misleading assessments of algorithms' performances. The results on larger problems gives evidence that these issues are wide-spread and substantial. As shown above, sometimes the description is specific enough to uniquely report the number of expansions regardless of what implicit tie-break policies are used. We wish we were able to say this about all experiments reported in papers. We fear this is frequently not the case. Usually the experimental results will depend upon which implicit tie-break policy is used. This means that the results can be irreproducible and misleading. MEPS, could in principle, determine whether the algorithm, heuristic, and explicit tie-break policy were described in sufficient detail to uniquely specify the results. Also, if the description is not sufficient then it could compute the distribution of results and the reported results could be seen in the light of how probable those results were for a randomly implemented implicit tie-break policy.

We have shown that implicit tie-break policies can affect the total nodes expanded below C^* and dramatically affect the nodes that are expandable at the C^* level. There are limitations to the current study. We intend to do larger problems and explore more domains. We currently use only one definition of lower-bound pairs, there are many other definitions that we want to try. Heuristics also seem to interact with tie-break policies, this is another avenue for exploration.

References

Alcázar, V. 2021. The Consistent Case in Bidirectional Search and a Bucket-to-Bucket Algorithm as a Middle Ground between Front-to-End and Front-to-Front. In *ICAPS Proceedings*, volume 31, 7–15.

Alcázar, V.; Barley, M.; and Riddle, P. 2019. A Theoretical Comparison of the Bounds of MM, NBS, and GBFHS. In *SoCS Proceedings*, volume 10, 160–161.

Alcázar, V.; Riddle, P. J.; and Barley, M. 2020. A Unifying View on Individual Bounds and Heuristic Inaccuracies in Bidirectional Search. In *AAAI Proceedings*, volume 34.

Asai, M.; and Fukunaga, A. 2016. Tiebreaking Strategies for A* Search: How to Explore the Final Frontier. In *AAAI Proceedings*.

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

Barker, J. K.; and Korf, R. E. 2015. Limitations of Front-To-End Bidirectional Heuristic Search. In *AAAI Proceedings*.

Barley, M.; Riddle, P.; López, C. L.; Dobson, S.; and Pohl, I. 2018. GBFHS: A Generalized Breadth-First Heuristic Search Algorithm. In *SoCS Proceedings*.

Bu, Z.; and Korf, R. E. 2022. A*+ BFHS: A hybrid heuristic search algorithm. In *AAAI Proceedings*, volume 36.

Chen, J.; Holte, R. C.; Zilles, S.; and Sturtevant, N. R. 2017. Front-to-End Bidirectional Heuristic Search with Near-Optimal Node Expansions. In *IJCAI Proceedings*.

Conover, W. J. 1999. *Practical nonparametric statistics*. John Wiley & Sons.

Corrêa, A. B. 2018. *Domain dependent heuristics and tie breakers: topics in automated planning*. Universidade Federal do Rio Grande do Sul.

Corrêa, A. B.; Pereira, A. G.; and Ritt, M. 2018. Analyzing Tie-Breaking Strategies for the A* Algorithm. In *IJCAI Proceedings*, 4715–4721.

Dantzig, G. 1963. *Linear Programming And Extensions*. Princeton University Press.

de Champeaux, D. 1983. Bidirectional Heuristic Search Again. *Journal of the ACM*, 30(1): 22 – 32.

de Champeaux, D.; and Sint, L. 1975. An improved bidirectional heuristic search algorithm. In *IJCAI Proceedings*.

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

Eckerle, J.; Chen, J.; Zilles, S.; and Holte, R. C. 2017. Sufficient Conditions for Node Expansion in Bidirectional Heuristic Search. In *ICAPS Proceedings*, volume 27.

Gates, W. H.; and Papadimitriou, C. H. 1979. Bounds for sorting by prefix reversal. *Discrete mathematics*, 27(1).

Gilon, D.; Felner, A.; and Stern, R. 2016. Dynamic potential search—a new bounded suboptimal search. In *SoCS Proceedings*.

Hansen, E.; and Zhou, R. 2007. Anytime heuristic search. *Journal of Artificial Intelligence Research*, 28(1): 267–297.

Hart, P.; Nilsson, N.; 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.

Helmert, M. 2010. Landmark heuristics for the pancake problem. In *SoCS Proceedings*.

Heusner, M.; Keller, T.; and Helmert, M. 2018. Best-case and worst-case behavior of greedy best-first search. In *IJCAI Proceedings*.

Hu, K.; and Speck, D. 2022. On Bidirectional Heuristic Search in Classical Planning: An Analysis of BAE. In *SoCS Proceedings*, volume 15, 91–99.

Kwa, J. 1989. BS*: An Admissible Bidirectional Staged Heuristic Search Algorithm. *Artificial Intelligence*, 38.

Nicholson, T. 1966. Finding the shortest route between two points in a network. *The Computer Journal*, 9(3): 275–280.

Pavlik, J. A.; Sewell, E. C.; and Jacobson, S. H. 2021. Two new bidirectional search algorithms. *Computational Optimization and Applications*, 1–33.

Pohl, I. 1969. *Bi-directional and heuristic search in path problems*. Ph.D. thesis, Stanford University.

Politowski, G.; and Pohl, I. 1984. D-node retargeting in bidirectional heuristic search. In *AAAI Proceedings*, 274–277.

Rathfux, T.; Kaindl, H.; Hoch, R.; and Lukasch, F. 2019. Efficiently Finding Optimal Solutions to Easy Problems in Design Space Exploration: A* Tie-breaking. In *Proceedings of International Conference on Software Technologies*.

Russell, S. J.; and Norvig, P. 2016. *Artificial intelligence: a modern approach*. England; Pearson Education Limited,.

Sadhukhan, S. K. 2013. Bidirectional heuristic search based on error estimate. *Computer Society of India (CSI) Journal of Computing*, 2(1-2): S1.

Shperberg, A.; Felner, A.; Sturtevant, N.; Shimony, S.; and Hayoun, A. 2019. Enriching Non-parametric Bidirectional Search Algorithms. In *AAAI Proceedings*.

Shperberg, S. S.; Felner, A.; Sturtevant, N. R.; Shimony, E.; and Hayoun, A. 2021. Bidirectional heuristic search: expanding nodes by a lower bound. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, 4775–4779.

Siag, L.; Shperberg, S.; Felner, A.; and Sturtevant, N. 2023a. Front-to-End Bidirectional Heuristic Search with Consistent Heuristics: Enumerating and Evaluating Algorithms and Bounds. In *IJCAI Proceedings*, 5631–5638.

Siag, L.; Shperberg, S.; Felner, A.; and Sturtevant, N. R. 2023b. Comparing Front-to-Front and Front-to-End Heuristics in Bidirectional Search. In *SoCS Proceedings*, volume 16, 158–162.

Thayer, J.; and Ruml, W. 2009. Using distance estimates in heuristic search. In *ICAPS Proceedings*, volume 19.

Welham, S. J.; Gezan, S. A.; Clark, S. J.; and Mead, A. 2014. *Statistical methods in biology: design and analysis of experiments and regression*. CRC press.