Understanding the Search Behaviour of Greedy Best-First Search

Manuel Heusner, Thomas Keller, Malte Helmert
University of Basel
Basel, Switzerland
{manuel.heusner,tho.keller,malte.helmert}@unibas.ch

Abstract

A classical result in optimal search shows that A* with an admissible and consistent heuristic expands every state whose f-value is below the optimal solution cost and no state whose f-value is above the optimal solution cost. For satisficing search algorithms, a similarly clear understanding is currently lacking. We examine the search behaviour of greedy best-first search (GBFS) in order to make progress towards such an understanding.

We introduce the concept of *high-water mark benches*, which separate the search space into areas that are searched by a GBFS algorithm in sequence. High-water mark benches allow us to exactly determine the set of states that are not expanded under any GBFS tie-breaking strategy. For the remaining states, we show that some are expanded by all GBFS searches, while others are expanded only if certain conditions are met.

Introduction

Many classical algorithms for state-space search, such as greedy best-first search (Doran and Michie 1966), A* (Hart, Nilsson, and Raphael 1968), Weighted A* (Pohl 1970) and IDA* (Korf 1985), are representatives of a general family of uni-directional, expansion-based heuristic search algorithms. Such algorithms are largely agnostic to the state space to be searched, only requiring two pieces of information to be applicable in a given domain:

- a *generative model* of the state space, defined in terms of black-box functions producing the *initial state*, testing whether a given state is a *goal state*, and producing the *successor states* of a given state along with the costs of the outgoing transition towards each successor, and
- a black-box *heuristic function* which estimates the cost-to-go or distance-to-go from a given state.

An important characteristic of such search algorithms is whether they guarantee that the solutions they produce are optimal (i.e., have minimal total cost among all solutions).

Optimal search algorithms in this family have a fairly well-developed theory (e.g., Dechter and Pearl 1985). For example, we know that the A* algorithm (Hart, Nilsson, and Raphael 1968) is optimal when used with an *admissible* heuristic and that it never needs to reexpand states when using a *consistent* heuristic. Moreover, for admissible and consistent heuristics there is a well-known and easy to understand criterion that allows us to reason about the states expanded by A*:

Proposition 1. *Consider the A* algorithm used with an admissible and consistent heuristic h in a solvable state space. Let c^* be the optimal solution cost, and let s be a state. Then:*

- *A* will expand s if $f(s) < c^*$, and*
- *A* will not expand s if $f(s) > c^*$,*

where $f(s) = g(s) + h(s)$ and $g(s)$ is the shortest-path cost from the initial state to s.

While this criterion is not perfect – it does not predict whether or not states s with $f(s) = c^*$ are expanded – it goes a large way towards explaining the search behaviour of A*. Theoretical results of this kind are very useful to understand and explain cases where A* performs poorly (e.g., Helmert and Röger 2008). They can also shed light on how to improve the performance of A*-style search algorithms, for example by emphasizing the importance of *tie-breaking behaviour*, which greatly influences which states with an f-value that is equal to the optimal solution cost are expanded (Asai and Fukunaga 2017).

For *satisficing* (non-optimal) algorithms in the family, a comparably deep understanding is currently lacking. Many new algorithms for satisficing search have been proposed in recent years (e.g., Imai and Kishimoto 2011; Xie et al. 2014; Xie, Müller, and Holte 2014; Valenzano et al. 2014), yet our understanding of the behaviour of such algorithms is still quite limited.

For example, a recent study by Wilt and Ruml (2015) demonstrated that (and why) improving the accuracy of an admissible heuristic can be highly detrimental for greedy search, while being extremely beneficial for A*, an insight that clearly shows how "conventional wisdom" for optimal search algorithms fails to apply to the satisficing case.

In this paper, we attempt to reduce this gap in knowledge by developing similar results to Proposition 1 for the most basic and most commonly considered satisficing search algorithm, greedy best-first search (GBFS). Specifically, we consider the following questions:

- Which states is GBFS *guaranteed* to expand?
- Which states is GBFS guaranteed *not* to expand?
- Which states may GBFS *potentially* expand?

We will consider these questions theoretically, by studying the effect of state space topology on the expansions performed by GBFS. While we can only provide partial answers, we hope that our study can provide some further insight into the inner workings of GBFS as well as motivate the usefulness of asking questions of this kind in order to improve our understanding of satisficing search algorithms in general.

Background

State Space Topology We consider search algorithms that operate on a *state space* $\mathcal{S} = \langle s_I, S_\star, succ, cost\rangle$, where s_I is the initial state, $S_\star$ is the set of goal states, $succ$ is a successor function that maps each state to a finite (possible empty) set of successor states, and $cost(s, s')$ gives the cost of the transition from s to $s' \in succ(s)$. With S, we denote the set of states of $\mathcal{S}$, which for the purposes of this work can be defined as the smallest set satisfying $s_I \in S$, $S_\star \subseteq S$ and $succ(s) \subseteq S$ for all $s \in S$.

A sequence of pairwise distinct states $\rho = \langle s_0, \dots, s_n\rangle$ is a (cycle-free) *path* from s_0 to s_n in $\mathcal{S}$ if $s_i \in succ(s_{i-1})$ for $i = 1, \dots, n$. With $P(s, s')$, we denote the set of (cycle-free) paths from s to s'. A path $\langle s_0, \dots, s_n\rangle$ is an *s-plan* if $s_0 = s$ and $s_n \in S_\star$. With $P(s)$, we denote the set of s-plans for s. We say that state s' is *reachable* from s if there is a path from s to s'.

A *state space topology* $\mathcal{T} = \langle \mathcal{S}, h\rangle$ is a state space $\mathcal{S}$ combined with a *heuristic function* h, i.e., any function $h : S \to \mathbb{R}_0^+$. Typically, $h(s)$ estimates the cost of a cheapest s-plan. In this work, it suffices to regard the heuristic function as an arbitrary black-box function that assigns some non-negative real number to each state.[1]

Example 1. *As a running example for this paper, consider the search space topology $\langle\langle A, \{T, Z\}, succ, cost\rangle, h\rangle$ with unit cost function cost and where succ is given by the arcs and $h(s)$ by the shaded regions of state s in Figure 1.*

Greedy Best-First Search The input to GBFS is a state space topology, and the output is an s_I-plan if one exists and "unsolvable" otherwise. GBFS is driven by the assumption that states with lower heuristic values are on a cheaper path to the closest goal state. In each step, it expands a state that has the lowest heuristic value among all states that have been generated before but have not been expanded yet, until a goal state is generated. Because of its greediness, it provides no guarantee of optimality (or any other quality guarantee) of the computed s_I-plan. Due to heuristic inaccuracy, most challenging search problems contain (possibly prohibitively large) heuristic plateaus or local minima where the guidance provided by the heuristic fails. Then, GBFS often faces situations where it has to decide which state to expand next among a set of states with identical heuristic value.

[1]In this work, we ignore states that are recognized as unsolvable by the heuristic, i.e., with $h(s) = \infty$. For the purposes of state space topology, these can be treated as if they were not part of the state space at all.

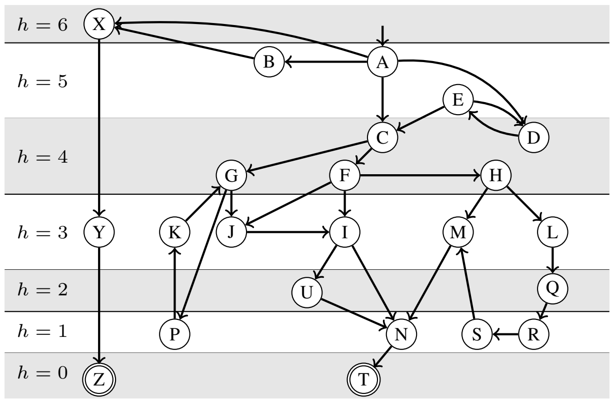

Figure 1: State space topology of our running example.

GBFS is actually not a well-defined algorithm but rather a *family* of algorithms that differ in a single parameter, the *tie-breaking strategy*. Formally, a GBFS tie-breaking strategy τ for a state space topology $\langle \mathcal{S}, h\rangle$ with states S maps all possible non-empty sets $S^k \subseteq \{s \in S \mid h(s) = k\}$ to a state $s \in S^k$. We will refer to GBFS coupled with a specific tie-breaking strategy as an *instance* of GBFS. In every iteration of GBFS where generated but unexpanded states (*open states*) still exist and no solution has yet been found, GBFS with tie-breaking strategy τ expands the state $\tau(S_{\min})$ where $S_{\min}$ is the set of all open states with minimal h-value. The tie-breaking strategy is the only parameter that sets different instances of GBFS apart. Given a state space topology and a tie-breaking strategy, GBFS performs a uniquely determined sequence of successive state expansions. The *search realization* of a GBFS search with tie-breaking strategy τ is a sequence of states $r^\tau = \langle s_1, \dots, s_n\rangle$, where $s_1 = s_I$, s_i is the i-th expanded state following τ, and s_n is either a goal state, or $\{s_1, \dots, s_n\}$ is the set of all reachable states in case no solution exists.

Example 2. *Let τ_1 and τ_2 be two tie-breaking strategies for our example topology from Figure 1. Then, $r^{\tau_1} = \langle A, C, D, G, P, J, K, I, N, T\rangle$ and $r^{\tau_2} = \langle A, C, F, I, N, T\rangle$ are the corresponding search realizations for τ_1 and τ_2.*

High-Water Marks and Apex

A first attempt to explain the behaviour of GBFS is due to Wilt and Ruml (2014), who base their analysis on the *high-water mark* of a state. As high-water marks are a central concept for this paper as well, we briefly recap their results.

Definition 1 (High water mark). *Let $\langle \mathcal{S}, h\rangle$ be a state space topology, and let $s \in S$ be a state. The* high-water mark *of s is defined as*

$$hw_h(s) := \begin{cases} \min_{\rho \in P(s)}(\max_{s' \in \rho} h(s')) & \text{if } P(s) \neq \emptyset \\ \infty & \text{otherwise.} \end{cases}$$

We define the high-water mark of a set of states $S' \subseteq S$ as

$$hw_h(S') := \min_{s \in S'} hw_h(s).$$

Intuitively, the high-water mark of a state s measures how high the heuristic values of expanded states must climb before a solution can be found in a search starting from s. Wilt and Ruml define high-water marks for individual states. We extend this definition to sets of states S' by selecting the minimum high-water mark of any of the states in S'. This reflects the intuition that if search begins with a set of candidate states S', then search will eventually follow the "path of least resistance" among the states in S'.

Example 3. *In our example topology, the set of K-plans is $P(K) = \{\langle K, G, J, I, N, T\rangle, \langle K, G, J, I, U, N, T\rangle\}$, and the set of Q-plans is $P(Q) = \{\langle Q, R, S, M, N, T\rangle\}$. Then, $hw_h(K) = 4$, $hw_h(Q) = 3$, and $hw_h(\{K, Q\}) = 3$.*

Wilt and Ruml use the high-water mark of the initial state to identify a set of states that is never expanded by GBFS, regardless of the used tie-breaking criterion.

Theorem 1 (due to Wilt and Ruml, 2014)**.** *Let $\mathcal{T} = \langle \mathcal{S}, h\rangle$ be a state space topology. For all tie-breaking strategies τ and all $s \in S$ with $h(s) > hw_h(s_I)$, it holds that $s \notin r^\tau$.*

Proof: Shown by Wilt and Ruml (2014). □

This result separates the states of a state space into two categories: states that are *certainly not* expanded by any GBFS instance on the one hand, and the remaining states, for which we *do not know* if they are expanded or not, on the other. It is our aim in the following to further refine the result to obtain a clear classification for a larger number of states.

Our first result in this direction is based on the following observation: as no state with a higher heuristic value than $hw_h(s_I)$ is expanded by GBFS under any tie-breaking strategy, no state that can be reached from the initial state only via a state with higher heuristic value than $hw_h(s_I)$ can be expanded by GBFS.

Example 4. *In our running example, $hw_h(A) = 5$, and hence the only state ruled out for expansion by the high-water mark criterion is X since $h(X) = 6 > hw_h(A)$.*

Consider state Y of the running example. Even though we have $h(Y) = 3 < hw_h(A)$, GBFS will not expand Y regardless of the used tie-breaking strategy as both paths from A to Y, $\langle A, X, Y\rangle$ and $\langle A, B, X, Y\rangle$, contain the state X with $hw_h(X) = 6$. As X is not expanded, Y is not expanded.

We formalize this insight by defining the *apex* of a state s, which is the lowest h threshold that must necessarily be reached before s can be expanded.

Definition 2. *Let $\langle \mathcal{S}, h\rangle$ be a state space topology, and let $s \in S$ be a state. The* apex *of s is defined as*

$$apex_h(s) := \begin{cases} \min\limits_{\rho \in P(s_I, s)} \max\limits_{s' \in \rho} h(s') & \text{if } P(s_I, s) \neq \emptyset \\ \infty & \text{otherwise.} \end{cases}$$

We can prove a similar result to Theorem 1 based on the apex of states.

Theorem 2. *Let $\mathcal{T} = \langle \mathcal{S}, h\rangle$ be a state space topology. For all tie-breaking strategies τ and all $s \in S$ with $apex_h(s) > hw_h(s_I)$, it holds that $s \notin r^\tau$.*

Proof: Let s be a state with $apex_h(s) > hw_h(s_I)$. Then, with Definition 2, every path $\rho = \langle s_I, \ldots, s\rangle$ from s_I to s contains a state s' with $h(s') \geq apex_h(s) > hw_h(s_I)$. With Theorem 1, we know that s' is not expanded by a GBFS search. This means that it is impossible that the complete sequence of states ρ is expanded. Since this holds for *every* path leading to s, s is never expanded. □

It is easy to see that $apex_h(s) \geq h(s)$ for all states s: if s is unreachable (i.e., $P(s_I, s) = \emptyset$), this holds trivially because $apex_h(s) = \infty$, and otherwise it holds because every path ρ to s must include s, and hence $\max_{s' \in \rho} h(s') \geq h(s)$. Therefore, Theorem 2 is a proper generalization of Theorem 1.

Example 5. *In the running example, X is not expanded by any GBFS instance due to Theorem 1, and X, Y, and Z are not expanded by any GBFS instance due to Theorem 2.*

GBFS Progress and Benches

In A* searches with admissible heuristics, expanding a state with a higher f-value than any previously expanded state is a meaningful event because every time it happens, a new lower bound for the optimal solution cost has been proven. Consequently, such an occurrence is often interpreted as a measure of *progress* of the search.

In GBFS-style searches, somewhat analogously, an event that is often considered meaningful is when the search expands a state with a lower heuristic value than the minimal heuristic value of all previously expanded states. For example, the boosted dual-queue search algorithm by Richter and Helmert (2009) uses such an occurrence to further prioritize expansions based on preferred operators, and the GBFS-LE family of algorithms by Xie, Müller, and Holte (2014) uses it to determine when a GBFS search has become stalled.

Although it is intuitively appealing, reaching a new lowest h-value in GBFS does not necessarily translate into meaningful, quantifiable progress in finding a solution.

Example 6. *The heuristic and high-water mark values of all states in the search realization $r^{\tau_1} = \langle A, C, D, G, P, J, K, I, N, T\rangle$ of our running example are as follows (bold values indicate new all-time lows):*

s	A	C	D	G	P	J	K	I	N	T
$\mathbf{h(s)}$	**5**	**4**	4	4	**1**	3	3	3	1	**0**
$\mathbf{hw_h(s)}$	**5**	**4**	5	4	4	**3**	4	3	**1**	**0**

In the search realization r^{τ_1} of our running example, we can see that P is the first state with a heuristic value of 1 that is expanded. Even if we have not yet defined formally what it means to make progress in greedy-best first search, the expansion of P does certainly not bring search closer to finding a plan. In contrast, N is only the second state with a heuristic value of 1 that is expanded, but expanding N is a crucial step for finding a plan as all paths from A to a goal state that GBFS will consider (taking into account that it will not consider Z due to Theorem 2) pass through N.

However, it can also be seen that high-water marks offer a better way of describing when actual search progress has been made: the expansions of C, J, and N are certainly among the steps where intuitive progress is made (and all of these states are part of the s_I-plan that is eventually found with tie-breaking strategy τ_1), while expanding P does not bring the search closer to finding a plan.

Whenever a state is expanded with a lower high-water mark k than any state expanded before, this is a meaningful event because it means that after this point, no state with a heuristic value larger than k will be expanded. (Of course, the disadvantage of high-water mark values is that they are not known during the search, so they can only be used in hindsight or when adopting an omniscient view of the state space topology.)

To develop these ideas further, we now introduce *high-water mark benches* (or simply *benches*). Benches are the main structural element of a state space topology that we will use in the following to provide a more refined analysis of GBFS behaviour. Intuitively, GBFS expands states on a given bench until it finds a way to *exit* the bench towards a lower bench. Once a bench has been left, it is never entered again.

Definition 3. *Let $\langle \mathcal{S}, h\rangle$ be a state space topology with set of states S, and let $S' \subseteq S$. The* high-water mark bench *$\mathcal{B}_h(S')$ of S' is a 3-tuple $\langle I, B, E\rangle$ with $B \subseteq S$, $I \subseteq B$, and $E \subseteq B$. We define $\mathcal{B}_h(S')$ as follows: if S' contains a goal state, then $\mathcal{B}_h(S') = \langle \emptyset, \emptyset, \emptyset\rangle$.*

Otherwise, let all states $s \in S$ with $h(s) \leq hw_h(S')$, $hw_h(s) \geq hw_h(S')$, and $s \notin S_\star$ be the bench state candidate *set, and define the* bench states *B as the set of all states that can be reached from some state in S' on some path that only includes states from the candidate set; the set of bench* entry states *as $I = S' \cap B$; and the set of bench* exit states *as $E = \{s \in B \mid hw_h(succ(s)) < hw_h(S') \text{ or } succ(s) \cap S_\star \neq \emptyset\}$. We abbreviate the high-water mark of the bench states B of a bench $\mathcal{B}_h$ with $hw_h(\mathcal{B}_h)$, and write $s \in \mathcal{B}_h$ for $s \in B(\mathcal{B}_h)$.*

Example 7. *In our running example, consider the set of states that contains only the initial state A. We have $\mathcal{B}_h(\{A\}) = \langle \{A\}, \{A, B, D, E\}, \{A, E\}\rangle$. State X and all states that are reachable from A only via X are no candidates for $\mathcal{B}_h(\{A\})$ as $h(X) > hw_h(\{A\})$, and state C and all states that are reachable from A only via C are no candidates as $hw_h(C) < hw_h(\{A\})$.*

The key property of benches is that they structure a GBFS search into episodes: whenever a GBFS search expands an exit state s of a bench, *all further state expansions will be of descendants of s*. We call this observation the *bench exit property*. It implies that the behaviour of GBFS search would not change if, every time a bench exit state s is about to be expanded, all other states in the open list of the search are discarded. (Of course, to make this property practically exploitable, we would need to know during search which states are bench exit states, which requires global knowledge of the state space topology.)

To see that the bench exit property holds, consider a situation where bench exit state s on the bench $\mathcal{B}_h$ is about to be expanded. We can make the following observations:

- Because s is an exit state, it has a successor s' which is a goal state or a successor s' with $hw_h(s') < hw_h(\mathcal{B}_h)$. If it has a goal-state successor, search is about to terminate and the bench exit property obviously holds. Hence, in the following, let s' be a successor of s with $hw_h(s') < hw_h(\mathcal{B}_h)$.
- Let S' be the set of states that induces $\mathcal{B}_h$, i.e., $\mathcal{B}_h := \mathcal{B}_h(S')$. Because $s \in \mathcal{B}_h$, its high-water mark satisfies $hw_h(s) \geq hw_h(S')$ (one of the defining properties of candidate states). As that property holds for all bench states and as at least one state from S' is a bench state (otherwise, there cannot be a path that consists only of states in the candidate set), we have $hw_h(S') = hw_h(\mathcal{B}_h)$, and therefore also $hw_h(s) \geq hw_h(\mathcal{B}_h)$.
- By the definition of high-water marks, we must have $hw_h(s) \leq \max\{h(s), hw_h(s')\}$ because this maximum accounts for the high-water mark of all paths from s via s' to the goal, a subset of all paths from s to the goal. Combined with the inequality from the previous bullet point, we get $hw_h(\mathcal{B}_h) \leq \max\{h(s), hw_h(s')\}$, i.e., $hw_h(\mathcal{B}_h) \leq h(s)$ or $hw_h(\mathcal{B}_h) \leq hw_h(s')$. From the first bullet point, we can rule out the second option, and hence the first option must hold: $hw_h(\mathcal{B}_h) \leq h(s)$.
- Because s is on the bench, it also satisfies $h(s) \leq hw_h(\mathcal{B}_h)$ (another defining property of candidate states). Combining this with $hw_h(\mathcal{B}_h) \leq h(s)$, we get $h(s) = hw_h(\mathcal{B}_h)$.
- Let $\tilde{S}$ be the set of states that are currently candidates for expansion, i.e., the generated but not yet expanded states. Because s is about to be expanded and GBFS prefers low h-values, $h(s) = hw_h(\mathcal{B}_h)$ implies that *all* states $\tilde{s} \in \tilde{S}$ must satisfy $h(\tilde{s}) \geq hw_h(\mathcal{B}_h)$. Because the successor s' of s has a high-water mark strictly less than $hw_h(\mathcal{B}_h)$, for the rest of the search after expanding s, there will always be a candidate for expansion with heuristic value strictly less than $hw_h(\mathcal{B}_h)$ until a solution is found. Hence, after s, no state from $\tilde{S}$ will ever be expanded, concluding the argument.

In summary, once a GBFS reaches a bench, i.e., generates the *entry states* of the bench, it remains on the bench until it expands an *exit state* (which, in turn, creates entry states to a successor bench). Each time a GBFS search moves on to a new bench, it has made progress towards finding a goal, as subsequent benches have strictly decreasing high-water marks. This also means that the search never returns to a bench after it has left it.

To illustrate the underlying idea of benches, we define the *bench path* of a given search realization r^τ. It contains all benches that are "traversed" (i.e., entered and exited) by a GBFS search with tie-breaking strategy τ.

Definition 4. *Let $r^\tau = \langle s_1, \dots, s_n\rangle$ be a search realization of a GBFS search with tie-breaking strategy τ. The* bench path *of r^τ is the sequence of benches $\mathcal{B}_h(r^\tau) = \langle \mathcal{B}_h(\{s_1\}), \mathcal{B}_h(succ(s'_1)), \dots, \mathcal{B}_h(succ(s'_k))\rangle$, where s'_1 is the first state in r^τ that is also in $E(\mathcal{B}_h(\{s_1\}))$, and for all $1 < i \leq k$, s'_i is the first state in the subsequence $\langle s'_{i-1}, \dots, s_n\rangle$ of r^τ that is also in $E(\mathcal{B}_h(succ(s_{i-1})))$.*

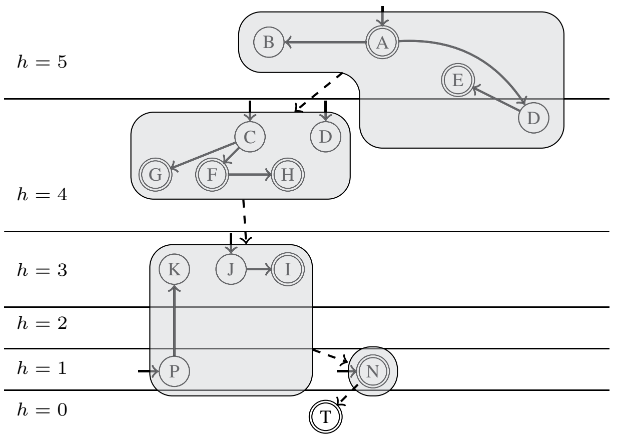

Figure 2: Bench path of the search realizations r^{τ_1} of Example 2. Arcs between states within a bench indicate expansion dependencies, and dashed arcs between benches show possible bench transitions. Bench entry states are annotated with arrows, and double lined circles indicate bench exit states.

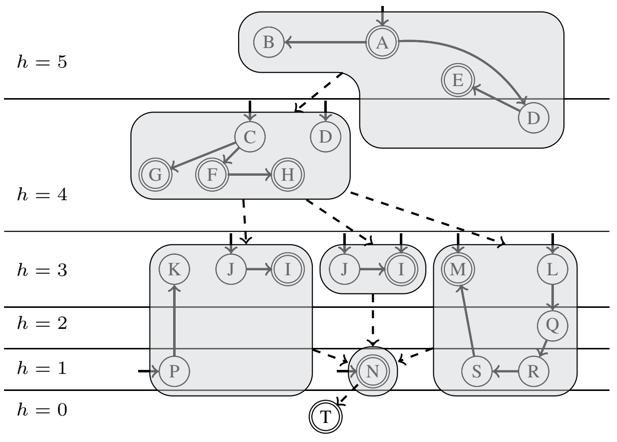

Figure 3: Bench transition system of our running example. Arcs between states within a bench indicate expansion dependencies, and dashed arcs between benches show possible bench transitions. Bench entry states are annotated with arrows, and double lined circles indicate bench exit states.

Example 8. *The bench path of the search realization r^{τ_1} of Example 2 is depicted in Figure 2. The initial bench $\mathcal{B}_h(\{A\})$ has already been discussed in Example 7.*

The next bench that is traversed by r^{τ_1} is $\mathcal{B}_h(succ(A)) = \mathcal{B}_h(\{B, C, D, X\}) = \langle\{C, D\}, \{C, D, F, G, H\}, \{F, G, H\}\rangle$. State E is not on this bench as $h(E) > hw_h(succ(A))$, while state D is part of the bench again, this time as an entry state.

The further benches are $\mathcal{B}_h(succ(G)) = \mathcal{B}_h(\{J, P\}) = \langle\{J, P\}, \{I, J, K, P\}, \{I\}\rangle$ and $\mathcal{B}_h(succ(I)) = \mathcal{B}_h(\{N, U\}) = \langle\{N\}, \{N\}, \{N\}\rangle$. State U is not on the latter bench as $h(U) > hw_h(succ(I))$. Finally, the goal state T is by definition not part of any bench and depicted only to indicate that a GBFS search terminates successfully when state N has been expanded.[2]

Hoffmann (2001; 2002) also introduces a type of bench – a so-called *heuristic bench* – to analyze state space topology. In contrast to his work, we do not restrict high-water mark benches to strongly connected components of the state space. Instead, a high-water mark bench is defined relative to a set of states (the entry states, which correspond to the successor states of an exit state of the previous bench), it can share states with other benches, and states on a single bench can be disconnected.

The *bench transition system* of a state space topology allows for a deeper structural analysis of GBFS behaviour.

Definition 5. *Let $\mathcal{T} = \langle\mathcal{S}, h\rangle$ be a state space topology. The* bench transition system *$\mathcal{B}_h(\mathcal{T})$ of $\mathcal{T}$ is a directed graph $\langle V, E\rangle$ whose vertices are benches. The vertex set V and directed edges E are inductively defined as follows:*

1. *$\mathcal{B}_h(\{s_I\}) \in V$*
2. *If $\mathcal{B}_h \in V$, $s \in E(\mathcal{B}_h)$ and $\mathcal{B}_h(succ(s)) \neq \langle\emptyset, \emptyset, \emptyset\rangle$, then $\mathcal{B}_h(succ(s)) \in V$ and $\langle\mathcal{B}_h, \mathcal{B}_h(succ(s))\rangle \in E$.*

[2]We assume that a goal check is performed upon state generation, which is adequate for a greedy search.

In words, the bench transition system can be constructed by starting from the bench defined by the initial state and then iteratively adding all further benches reached by expanding an exit state of a previously generated bench.

Example 9. *The bench transition system of our running example $\mathcal{T}$ is depicted in Figure 3. Starting from the initial bench $\mathcal{B}_h(\{A\})$, which is as described in Example 7, we iteratively select an exit state s and create $\mathcal{B}_h(succ(s))$. For the two exit states of the initial bench, A, and E, this leads to the same successor bench $\mathcal{B}_h(\{B, C, D, X\}) = \mathcal{B}_h(\{C, D\}) = \langle\{C, D\}, \{C, D, F, G, H\}, \{F, G, H\}\rangle$. This is also the case for the three predecessor benches of the bench that contains only state N.*

In our running example, the bench transition system is a directed acyclic graph. As we show next, this observation holds for all state space topology.

Theorem 3. *The bench transition system $\mathcal{B}_h(\mathcal{T})$ of a state space topology $\mathcal{T}$ is a directed acyclic graph.*

Proof: Let $\mathcal{B}_h(\mathcal{T}) = \langle V, E\rangle$ be a bench transition system of a state space topology $\mathcal{T}$, and let $\mathcal{B}_h$ and $\mathcal{B}'_h$ be two benches from V with $\langle\mathcal{B}_h, \mathcal{B}'_h\rangle \in E$. Due to Definition 5, there is a $s \in E(\mathcal{B}_h)$ such that $\mathcal{B}'_h = \mathcal{B}_h(succ(s))$. From Definition 3, we know that $hw_h(succ(s)) < hw_h(\mathcal{B}_h)$ (since $B(\mathcal{B}'_h)$ must be non-empty), and therefore $hw_h(\mathcal{B}_h) > hw_h(\mathcal{B}'_h)$. This shows that arcs can only lead from one bench to another with a lower high-water mark, and hence there cannot be a cycle in $\mathcal{B}_h(\mathcal{T})$. $\square$

Next, we observe that all states expanded by any instance of GBFS must be contained in a bench of the bench transition system.

Theorem 4. *Let $\mathcal{T} = \langle\mathcal{S}, h\rangle$ be a state space topology with bench transition system $\langle V, E\rangle$. Let $S_{\text{bench}} = \bigcup_{\langle I,B,E\rangle\in V} B$ be the set of all states included in some bench of the bench transition system. Then for all tie-breaking strategies τ and all states in r^τ, it holds that $s \in S_{\text{bench}}$.*

Proof: Clearly, all states expanded initially in $r^{\mathcal{T}}$ are part of the initial bench $\mathcal{B}_h(\{s_I\})$ until the first time an exit state of $\mathcal{B}_h(\{s_I\})$ is expanded. By the bench exit property (see discussion after Definition 3), every time a bench exit state s is expanded, the search then proceeds as if s were the only state on the open list, generating $S' = succ(s)$ and continuing the search from $\mathcal{B}_h(S')$, only expanding states from this bench until one of its exit states is reached. This process repeats until the search terminates. By Definition 5, the bench transition systems contains all benches that can be encountered in this process, and hence the search only expands states in S_{bench}. □

By contraposition, of course this means that all states that are not part of the bench transition system are never expanded by a GBFS search. In the following section, we will refine this result further to exactly characterize the states expanded by any instance of GBFS. In the meantime, the following result relates the preceding theorem to our earlier bounds based on high-water marks and apexes.

Theorem 5. *Let $\mathcal{T} = \langle \mathcal{S}, h\rangle$ be a state space topology with bench transition system $\langle V, E\rangle$. Let $S_{\text{bench}} = \bigcup_{\langle I,B,E\rangle \in V} B$ be the set of all states included in some bench of the bench transition system. Furthermore, let*

$$\bar{S}_{hw_h} := \{s \in S \mid h(s) > hw_h(s_I)\},$$
$$\bar{S}_{apex_h} := \{s \in S \mid apex_h(s) > hw_h(s_I)\}, \text{ and}$$
$$\bar{S}_{\text{bench}} := \{s \in S \mid s \notin S_{\text{bench}}\}.$$

Then $\bar{S}_{hw_h} \subseteq \bar{S}_{apex_h} \subseteq \bar{S}_{\text{bench}}$.

Proof: We have already discussed that Theorem 2 is a proper generalization of Theorem 1 and hence $\bar{S}_{hw_h} \subseteq \bar{S}_{apex_h}$. Furthermore, Example 5 shows that there are cases where $\bar{S}_{hw_h}$ is a proper subset of $\bar{S}_{apex_h}$.

We now compare $\bar{S}_{apex_h}$ and $\bar{S}_{\text{bench}}$. For unsolvable state spaces, all sets trivialize, so we assume that a solution exists. Unreachable states have an infinite apex value and are not included in any bench, so they belong to both $\bar{S}_{apex_h}$ and $\bar{S}_{\text{bench}}$.

Let s be a state in $\bar{S}_{apex_h}$. Then $apex_h(s) > hw_h(s_I)$ and, due to Definition 2, there is a state s' on each path from s_I to s with $h(s') \geq apex_h(s) > hw_h(s_I)$. Because the high-water marks of benches decrease as we follow arcs in the bench transition system, $hw_h(\mathcal{B}_h) \leq hw_h(s_I)$ for all benches $\mathcal{B}_h \in V$, and from the definition of benches, this implies $h(s) \leq hw_h(s_I)$ for all $s \in \mathcal{B}_h$, and hence $s \neq s'$. Thus no bench contains the state s, proving $\bar{S}_{apex_h} \subseteq \bar{S}_{\text{bench}}$.

Finally, our running example shows that there are cases where $\bar{S}_{apex_h} \neq \bar{S}_{\text{bench}}$: from Figure 3, we can see that state U is not part of any bench, and hence $U \in \bar{S}_{\text{bench}}$, but $apex_h(U) = 5 = hw_h(s_I)$ and therefore $U \notin \bar{S}_{apex_h}$. □

Structural Analysis of Benches

Up to here, we have only discussed how a state space topology induces a set of benches that are connected in a way that allows to split a GBFS search to phases. From here on, we are also interested in the *inner* structure of benches, which defines the sub-search space that is induced by a bench. We have already discussed the bench exit property, which states that a GBFS search never returns to a bench once it has left it, i.e., it only expands states from the current bench and from benches it reaches in the future, but not from benches it has traversed in the past.

The sub-search space of a bench therefore consists only of states from that bench, and all GBFS instances expand states from that bench until an exit state is expanded. GBFS keeps track of all states that have been generated (i.e., all successors of states that have been expanded) but not expanded, and selects one of the states with minimal heuristic value according to its tie-breaking strategy. A GBFS bench sub-search hence

- starts with all entry states of the bench as candidates for expansion;
- selects among the candidates a state with minimal heuristic value according to its tie-breaking strategy;
- adds all successors of the selected state to the set of candidates;
- ends as soon as the first exit state is expanded.

A GBFS bench sub-search ends with the *first* expanded exit state because it is "forced" to proceed to the next bench as the exit state must have a successor with a lower heuristic value than all remaining candidates. The fact that a GBFS algorithm cannot choose to stay on a bench once an exit state is reached has consequences for the set of states that is potentially expanded: all states on a bench that are reachable from the set of entry states only on paths via an exit state can never be expanded, and therefore do not need to be considered in the bench transition system.

Definition 6. *Let $\mathcal{T} = \langle \mathcal{S}, h\rangle$ be a state space topology, and let $S' \subseteq S$. The* reduced bench *$\mathcal{B}^*_h(S') = \langle I^*, B^*, E^*\rangle$ of S' is defined via the same set of candidate states as the bench of S' (see Definition 3). Then, the set of* bench states *B^* is defined as the set of all states that can be reached from some state in S' on some path that only includes* non-exit states *from the candidate set, while I^* and E^* are defined in the same way as I and E in Definition 3 but relative to B^* rather than B.*

Reduced bench paths $\mathcal{B}^*_h(r^{\mathcal{T}})$ and *reduced bench transition systems* $\mathcal{B}^*_h(\mathcal{T})$ are defined analogously to the original definitions for reduced benches.

Example 10. *Consider the bench transition system of our running example in Figure 3. As A is both entry and exit state, all GBFS searches have to continue search on the lower bench $\mathcal{B}_h(succ(\{A\}))$ as there is a successor that has both a lower heuristic value and high-water mark than the current high-water mark of the current bench. This excludes states B, D, and, in turn, E on bench $\mathcal{B}_h(\{A\})$ from being expanded by any GBFS instance.*

However, note that state D, which has a lower heuristic value than B and E, is also part of the successor bench $\mathcal{B}_h(succ(A))$, and can hence still be selected for expansion in the sub-search on the next bench. The path to state E, however, is forever blocked, as E is only part of $\mathcal{B}_h(\{A\})$.

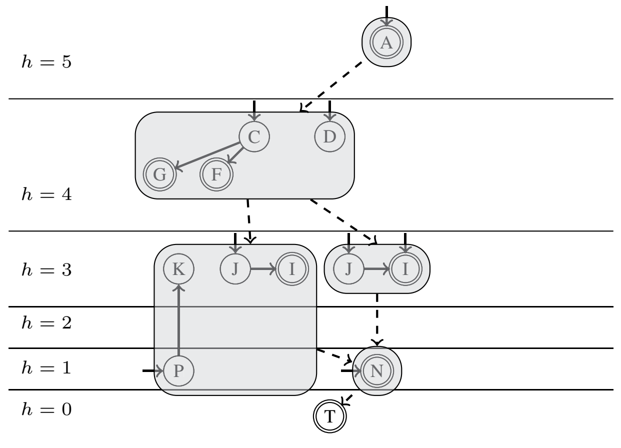

Figure 4: Reduced bench transition system of our running example. Arcs between states within a bench indicate expansion dependencies, and dashed arcs between benches show possible bench transitions. Bench entry states are annotated with arrows, and double lined circles indicate bench exit states.

A similar situation arises for state H on bench $\mathcal{B}_h(succ(A))$: the only path to H is via exit state F, and every GBFS search that expands the exit state F has to proceed to bench $\mathcal{B}_h(succ(F))$. Therefore, H can never be expanded (on bench $\mathcal{B}_h(succ(A))$) by any GBFS search. This not only excludes state H from the search, but the whole bench $\mathcal{B}_h(succ(H))$ will never be expanded by a GBFS search, regardless of the tie-breaking strategy.

The resulting reduced bench transition system of our running example is depicted in Figure 4.

A generalization of Theorem 5 with the set of states that are included in some bench of the reduced bench transition system would be straightforward (as that set is clearly a subset of the set of states that are included in some bench of the bench transition system). However, an even stronger result (and one of our main results) is that a reduced bench transition system contains *exactly* the states that are expanded by at least one possible GBFS search. It therefore provides the answer to the second and third question we raised in the introduction: the reduced bench transition system separates the set of states into one part that each GBFS search is guaranteed not to expand (those states that are not on any reduced bench), and one part that each GBFS may potentially expand (those states that are on at least one reduced bench).

Theorem 6. *Let $\mathcal{T} = \langle \mathcal{S}, h \rangle$ be a state space topology with set of states S and reduced bench transition system $\langle V^*, E^* \rangle$. For each state $s \in S$, it holds that $s \in \mathcal{B}_h$ for some $\mathcal{B}_h \in V^*$ iff there is a GBFS search realization $r^\mathcal{T}$ that expands s.*

Proof: From the discussion at the start of this section, it is obvious that if a state is expanded by some realization, then it is included in the reduced bench transition system: the only states excluded by moving from the bench transition system to the reduced bench transition systems are ones that cannot possibly be expanded.

It remains to show that for each state s of each bench of the reduced bench transition system, it is possible to find a GBFS realization that expands s. Due to the inductive definition of the reduced bench transition system as benches reached from the initial bench and due to the way that GBFS realizations fall into episodes that perform expansions within a given bench, for this it is sufficient to show that for every reduced bench $\mathcal{B}_h^*$, a GBFS search that begins with the entry states in $I(\mathcal{B}_h^*)$ as candidates for expansion can potentially expand any given state in $B(\mathcal{B}_h^*)$ before exiting the bench by expanding a state from $E(\mathcal{B}_h^*)$.

All states on $\mathcal{B}_h^*$ are reachable from the entry states without passing through an exit state. Moreover, the search on the bench only ends when an exit state has been expanded. Hence, the only situation in which *no* tie-breaking strategy might expand a state $s' \in \mathcal{B}_h^*$ is if the GBFS search order *forces* the expansion of an exit state s'' before s' because s'' has a low heuristic value. However, this cannot happen because exit states always have the *highest* h-values among all states on the bench: all states on a bench satisfy $h(s) \leq hw_h(\mathcal{B}_h^*)$ by definition, and exit states s'' satisfy $h(s'') = hw_h(\mathcal{B}_h^*)$ as shown in the discussion of the bench exit property (after Definition 3). Hence, we can devise a tie-breaking strategy that first expands all non-exit states and then any desired exit state. □

Theorem 6 indicates that there are no further states that can be excluded from the set of states that are potentially expanded. We therefore turn our attention to a finer categorization of the states that are potentially expanded. In particular, we determine criteria for states that are expanded by *every* GBFS instance, regardless of the used tie-breaking strategy, and derive conditions under which some states must be expanded.

Bottleneck States The first concept for this categorization are *bottleneck states*. A state is a bottleneck state of a bench if all paths from any entry state of the bench to any exit state of the bench are via that state. Such a state cannot be avoided by a GBFS search that reaches the bench.

Definition 7. *Let $\mathcal{T} = \langle \mathcal{S}, h \rangle$ be a state space topology with reduced bench transition system $\langle V^*, E^* \rangle$, and let $\mathcal{B}_h \in V^*$ be a bench. A state s is a* bottleneck state *of bench $\mathcal{B}_h$ iff all paths from all $s' \in I(\mathcal{B}_h)$ to all $s'' \in E(\mathcal{B}_h)$ are via s, i.e., of the form $\langle s', \dots, s, \dots, s'' \rangle$.*

Example 11. *In our running example, there are several bottleneck states: the initial state A is a bottleneck state of its bench (unless $s_I \in S_\star$, this is trivially always true for an initial state), and so are C (since there is no path to an exit state from D), I (on both benches that contain I), and N.*

The existence of an isolated bottleneck state here and there does not affect the search behaviour of GBFS significantly. However, there are cases where bottleneck states have additional properties that make the fact that they have to be expanded important.

Crater States A phenomenon of GBFS that has drawn plenty of interest in the planning community (e.g., Hoffmann 2005; Nakhost and Müller 2009; Lipovetzky and Geffner 2011; Xie, Müller, and Holte 2014) is what is often called an uninformed heuristic region, local minimum, or heuristic plateau. Even though all of these differ slightly in definition, they have in common that they are causing serious problems for GBFS search as they represent (potentially large) parts of the search space where the heuristic does not offer guidance or, in the worst case, even misguides the search. While benches already refine what is typically understood as a heuristic plateau, our framework also allows us to provide a more accurate definition of local minima in the form of *craters*.

Definition 8. *Let $\mathcal{T} = \langle \mathcal{S}, h \rangle$ be a state space topology with reduced bench transition system $\langle V^*, E^* \rangle$, and let $\mathcal{B}_h \in V^*$ be a bench. We call a state $s \in \mathcal{B}_h \setminus E(\mathcal{B}_h)$ a* crater entry state *iff $h(s) = hw_h(\mathcal{B}_h)$ and there is a state $s' \in succ(s)$ with $h(s') < hw_h(\mathcal{B}_h)$.*

The crater *of a crater entry state s is the largest set of states $\mathcal{C}_h(s)$ that is such that $s' \in \mathcal{C}_h(s)$ if $h(s') < hw_h(\mathcal{B}_h)$ and if there is a path $\langle s, s_1, \ldots, s_n, s' \rangle$ such that $s_1, \ldots, s_n \in \mathcal{C}_h(s)$.*

Craters are interesting for our analysis as each GBFS search is such that *all* states from a crater $\mathcal{C}_h(s)$ *must* be expanded once the crater entry state s has been expanded, as all states in the crater have in common that their heuristic value is lower than their (and the current benches') high-water mark. This means that they are prioritized in search, but all paths to a goal state have to "get out" of the crater “back to” the top of the bench (due to the larger high-water mark) before a path to a goal state can be found.

Example 12. *Consider the bench transition system of our running example in Figure 3. It contains a crater that is part of the bench $\mathcal{B}_h := \mathcal{B}_h(succ(H))$. Even though $\mathcal{B}_h$ is not part of the* reduced *bench transition system (which we used to define craters) in Figure 4, this kind of crater could occur in a reduced bench transition system as well.*

The crater entry state is L, as L has a successor (Q) with $h(Q) < hw_h(\mathcal{B}_h)$. Following Definition 8, we get $\mathcal{C}_h(L) = \{Q, R, S\}$. All GBFS searches that decide to expand L have to expand Q, R, and S before they may expand a state outside of the crater (in this case, only M is left on the bench). Apparently, a search with a tie-breaking strategy that prefers M over L avoids the expansion of the whole crater. However, if L were a bottleneck state, each GBFS search that reaches the bench would have to expand the crater entry state and then also all states from the crater.

In our running example, there is another state that resembles the concept of a crater, but isn’t one according to our definition of crater states: on the bench $\mathcal{B}_h(succ(G))$, each GBFS algorithm that searches $\mathcal{B}_h(succ(G))$ has to expand state P first. As this kind of crater has no crater entry state on its bench and cannot be avoided by any GBFS search that enters its bench, we call it a *bench crater*.

Definition 9. *Let $\mathcal{T} = \langle \mathcal{S}, h \rangle$ be a state space topology with reduced bench transition system $\langle V^*, E^* \rangle$, and let $\mathcal{B}_h \in V^*$ be a bench. Let $\mathcal{C}^I_h(\mathcal{B}_h) := \{s \in I(\mathcal{B}_h) \mid h(s) < hw_h(\mathcal{B}_h)\}$ be the set of* bench crater entry states. *The* bench crater *of a bench $\mathcal{B}_h$ is the largest set of states $\mathcal{C}_h(\mathcal{B}_h)$ that is such that $s' \in \mathcal{C}_h(\mathcal{B}_h)$ if $s' \in \mathcal{C}^I_h(\mathcal{B}_h)$, or if $h(s') < hw_h(\mathcal{B}_h)$ and there is a path $\langle s, s_1, \ldots, s_n, s' \rangle$ such that $s \in \mathcal{C}^I_h(\mathcal{B}_h)$ and $s_1, \ldots, s_n \in \mathcal{C}_h$.*

Note that crater entry states and bench crater entry states differ in the fact that the latter are part of the crater while the former are not. This is because each crater entry state on bench $\mathcal{B}_h$ has a heuristic value that is equal to $hw_h(\mathcal{B}_h)$, while bench crater entry states have a heuristic value that is lower than $hw_h(\mathcal{B}_h)$. Crater entry states are interesting as they describe a “point of no return” – once expanded, each GBFS search is forced to expand the whole crater.

Bottleneck Benches Finally, there are not only states on benches that are expanded by all GBFS searches that enter a bench, but also benches that are reached independently of the used tie-breaking strategy.

Definition 10. *Let $\mathcal{T} = \langle \mathcal{S}, h \rangle$ be a state space topology with reduced bench transition system $\langle V^*, E^* \rangle$. A bench $\mathcal{B}_h \in V^*$ is a* bottleneck bench *if each bench path in $\langle V^*, E^* \rangle$ is via $\mathcal{B}_h$.*

Example 13. *Our running example contains several bottleneck benches, namely $\mathcal{B}_h(\{A\})$, $\mathcal{B}_h(\{C, D\})$, and $\mathcal{B}_h(\{N\})$. We can therefore observe that:*

- *states A, C, and N are expanded by all GBFS searches;*
- *if bench $\mathcal{B}_h(succ(G))$ is searched, state P is expanded (due to being part of a bench crater), and J and I are expanded (both are bottleneck states);*

Our insights on bottleneck states, craters, and bottleneck benches allow us to conclude that a state s on bench $\mathcal{B}_h$ is expanded by a GBFS search with tie-breaking strategy τ if

1. i) s is a bottleneck state and ii) $\mathcal{B}_h$ is a bottleneck bench or $\mathcal{B}_h$ is searched under τ;
2. i) s is part of a bench crater and ii) $\mathcal{B}_h$ is a bottleneck bench or $\mathcal{B}_h$ is searched under τ;
3. i) s is part of a crater $\mathcal{C}_h(s')$ and ii) $\mathcal{B}_h$ is a bottleneck bench or $\mathcal{B}_h$ is searched under τ and iii) s' is a bottleneck state or s' is expanded under τ

Discussion and Conclusion

We started our GBFS search analysis by extending the results of Wilt and Ruml (2014), who introduce the *high-water mark* of a state, to the *apex* of a state, which allows us to more accurately separate the state space of a search problem into a part that is guaranteed not to be expanded and a part for which we have no further information.

We introduced the concept of high-water mark benches, and showed that they can be used to describe an even more accurate separation of the state space. Refining this concept further by excluding states from benches that are reachable only via exit states allows us to *exactly* partition the state space into states that are guaranteed not to be expanded by any GBFS tie-breaking strategy and ones that are expanded by at least one GBFS tie-breaking strategy. Moreover, we describe two properties of states – bottlenecks and craters –

that allow us to determine conditions under which states are guaranteed to be expanded.

First and foremost, our paper provides a theoretical study of GBFS search behaviour. However, we believe that this analysis also suggests promising possibilities for practical applications. One such opportunity is to use the notions of benches, bottleneck states and craters to *learn* useful properties of a heuristic in small state spaces in a given domain of search problems, and then use this information to improve the heuristic for other (possibly larger) state spaces in the same domain. An analysis of craters of popular heuristic functions in search domains of interest might also suggest ways in which such heuristics could be improved in general.

Another possible avenue is to use the concepts of our analysis to explicitly design state spaces with varying structural properties, which could then help develop a better understanding of the advantages and (possible disadvantages) of alternatives to GBFS that have recently been proposed.

In current work, we are aiming to use our framework for a best- and worst-case analysis of GBFS tie-breaking strategies by performing a meta-search in the reduced bench transition system. Information on benches, bottlenecks and craters speeds up this otherwise intractable meta-search significantly and allows us to quantify precisely how important tie-breaking in GBFS is and how close currently used tie-breaking strategies are to the best- or worst-case behaviour.

Acknowledgments

This work was supported by the European Research Council as part of the project “State Space Exploration: Principles, Algorithms and Applications”. We thank the anonymous reviewers for their constructive and helpful comments.

References

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

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

Doran, J. E., and Michie, D. 1966. Experiments with the graph traverser program. *Proceedings of the Royal Society A* 294:235–259.

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.

Helmert, M., and Röger, G. 2008. How good is almost perfect? In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI 2008)*, 944–949. AAAI Press.

Hoffmann, J. 2001. Local search topology in planning benchmarks: An empirical analysis. In Nebel, B., ed., *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI 2001)*, 453–458. Morgan Kaufmann.

Hoffmann, J. 2002. Local search topology in planning benchmarks: A theoretical analysis. In Ghallab, M.; Hertzberg, J.; and Traverso, P., eds., *Proceedings of the Sixth International Conference on Artificial Intelligence Planning and Scheduling (AIPS 2002)*, 92–100. AAAI Press.

Hoffmann, J. 2005. Where ‘ignoring delete lists’ works: Local search topology in planning benchmarks. *Journal of Artificial Intelligence Research* 24:685–758.

Imai, T., and Kishimoto, A. 2011. A novel technique for avoiding plateaus of greedy best-first search in satisficing planning. In Burgard, W., and Roth, D., eds., *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI 2011)*, 985–991. AAAI Press.

Korf, R. E. 1985. Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence* 27(1):97–109.

Lipovetzky, N., and Geffner, H. 2011. Searching for plans with carefully designed probes. In Bacchus, F.; Domshlak, C.; Edelkamp, S.; and Helmert, M., eds., *Proceedings of the Twenty-First International Conference on Automated Planning and Scheduling (ICAPS 2011)*, 154–161. AAAI Press.

Nakhost, H., and Müller, M. 2009. Monte-Carlo exploration for deterministic planning. In Boutilier, C., ed., *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*, 1766–1771. AAAI Press.

Pohl, I. 1970. Heuristic search viewed as path finding in a graph. *Artificial Intelligence* 1:193–204.

Richter, S., and Helmert, M. 2009. Preferred operators and deferred evaluation in satisficing planning. In Gerevini, A.; Howe, A.; Cesta, A.; and Refanidis, I., eds., *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS 2009)*, 273–280. AAAI Press.

Valenzano, R.; Sturtevant, N. R.; Schaeffer, J.; and Xie, F. 2014. A comparison of knowledge-based GBFS enhancements and knowledge-free exploration. In *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling (ICAPS 2014)*, 375–379. AAAI Press.

Wilt, C., and Ruml, W. 2014. Speedy versus greedy search. In Edelkamp, S., and Barták, R., eds., *Proceedings of the Seventh Annual Symposium on Combinatorial Search (SoCS 2014)*, 184–192. AAAI Press.

Wilt, C., and Ruml, W. 2015. Building a heuristic for greedy search. In Lelis, L., and Stern, R., eds., *Proceedings of the Eighth Annual Symposium on Combinatorial Search (SoCS 2015)*, 131–139. AAAI Press.

Xie, F.; Müller, M.; Holte, R. C.; and Imai, T. 2014. Type-based exploration with multiple search queues for satisficing planning. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI 2014)*. AAAI Press.

Xie, F.; Müller, M.; and Holte, R. C. 2014. Adding local exploration to greedy best-first search in satisficing planning. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI 2014)*. AAAI Press.