Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)

Best-Case and Worst-Case Behavior of Greedy Best-First Search

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

Abstract

We study the impact of tie-breaking on the behavior of greedy best-first search with a fixed state space and fixed heuristic. We prove that it is NP-complete to determine the number of states that need to be expanded by greedy best-first search in the best case or in the worst case. However, the best- and worst-case behavior can be computed in polynomial time for undirected state spaces. We perform computational experiments on benchmark tasks from the International Planning Competitions that compare the best and worst cases of greedy best-first search to FIFO, LIFO and random tie-breaking. The experiments demonstrate the importance of tie-breaking in greedy best-first search.

1 Introduction

While the theoretical properties of optimal state-space search algorithms like A* or IDA* have been quite extensively studied (e.g., Martelli, 1977; Pearl, 1984; Dechter and Pearl, 1985; Korf *et al.*, 2001; Helmert and Röger, 2008; Holte, 2010), satisficing search algorithms like greedy best-first search (GBFS, Doran and Michie, 1966) are considerably less well understood. The research goal of developing a theory of GBFS and related algorithms has recently received growing attention (e.g., Wilt and Ruml, 2014; 2015; 2016), but many basic questions remain unaddressed.

Anecdotal evidence suggest that *tie-breaking* has a significant impact on the performance of GBFS (e.g., Asai and Fukunaga, 2017), and indeed this is easy to see when considering an extreme scenario: with a constant heuristic, GBFS with *best-case* tie-breaking only expands the states along a shortest solution path, while it expands all states that can be reached without passing through a goal state with *worst-case* tie-breaking. However, practical applications of GBFS do not use constant heuristics, and understanding the impact of tie-breaking in the general case is much more challenging.

To understand the difficulty in analyzing the tie-breaking behavior of GBFS, it is instructive to contrast it with the much better understood behavior of A* search with an admissible and consistent heuristic. Consider such an A* search with an optimal solution cost of K. Let us call a state s an *early layer* state if $f(s) < K$ and a *goal layer* state if $f(s) = K$, s can be reached without passing through a goal state, and s itself is not a goal state.[1] The set of *possibly expanded* states then consists of all early layer states and all goal layer states. Under *worst-case tie-breaking*, A* expands exactly these states. The set of *necessarily expanded* states consists of all early layer states together with those goal-layer states that are part of every optimal solution. Under *best-case tie-breaking*, A* expands exactly the early layer states plus the minimum number of goal-layer states that occur in any optimal solution. The tie-breaking behavior of A* can be almost completely understood in terms of the possibly and necessarily expanded states, which can be easily computed.

For GBFS, the situation is significantly more complicated. We recently made a step towards a better understanding of GBFS by showing that every run of the algorithm can be partitioned into different episodes defined by so-called *high-water mark benches* [Heusner *et al.*, 2017] . Based on this insight, we fully characterized the possibly expanded states. We also discussed necessarily expanded states, although without providing a complete characterization. The goal of this paper is to deepen the understanding of the best- and worst-case behavior of GBFS with the following contributions:

1. We show that, given an explicit representation of a state space, it is NP-complete to compute lower or upper bounds on the number of states expanded by GBFS.
2. Based on the decomposition of GBFS runs into high-water mark benches, we describe (worst-case exponential-time) algorithms for determining the best-case and worst-case tie-breaking of GBFS.
3. We show that for *undirected* state spaces, best-case and worst-case tie-breaking of GBFS can be analyzed in polynomial time.
4. We provide an experimental analysis that demonstrates the gap between worst-case and best-case tie-breaking of GBFS on benchmarks from the International Planning Competitions. We also show how standard tie-breaking strategies compare experimentally to the best and worst case, and we explore the relationship between worst-case tie-breaking and the set of possibly expanded states.

[1] There is no universal agreement in the literature whether the final goal state considered by a search algorithm counts as expanded. As their successors are not generated, we exclude goal states here.

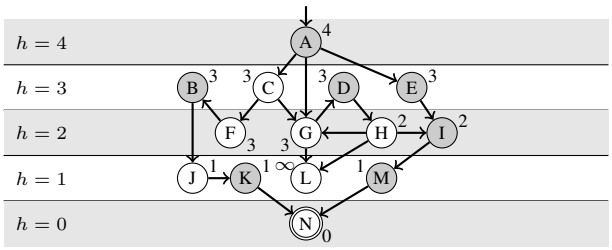

Figure 1: State space topology $\langle\{\mathrm{A},\ldots,\mathrm{N}\}, succ, \mathrm{A}, \{\mathrm{N}\}\rangle$ with *succ* as indicated by the arrows. Heuristic values are given by the levels, states are annotated with their high-water marks, and progress states are gray.

2 Background

State Space Topology We begin by introducing the formal background needed for this work.

Definition 1. *A* state space *is a 4-tuple* $\mathcal{S} = \langle S, succ, s_{init}, S_{goal}\rangle$*, where* S *is a finite set of* states*,* $succ : S \to 2^S$ *is the* successor function*,* $s_{init} \in S$ *is the* initial state*, and* $S_{goal} \subseteq S$ *is the set of* goal states*. If* $s' \in succ(s)$*, we say that* s' *is a* successor *of* s *and that* $s \to s'$ *is a (state)* transition*.*

A heuristic *for* $\mathcal{S}$ *is a function* $h : S \to \mathbb{R}$*.*

A state space topology *is a pair* $\langle\mathcal{S}, h\rangle$*, where* $\mathcal{S}$ *is a state space and* h *is a heuristic for* $\mathcal{S}$*.*

Figure 1 shows the state space topology that we use as an example throughout this paper.

For simplicity, we do not consider infinite heuristic values, which can be derived from an arbitrary state space topology by considering states with $h(s) = \infty$ as non-existent. We also do not consider transition costs, as these are ignored by greedy best-first search. A *path* of length n from state s to state s' is a sequence of states $\pi = \langle s_0, \ldots, s_n\rangle$ where $s = s_0$, $s' = s_n$ and $s_i \in succ(s_{i-1})$ for all $1 \leq i \leq n$. We write $P(s)$ for the set of all paths from s to any goal state.

Greedy Best-First Search The objective in state-space search is to find a path from the initial state to any goal state. The *greedy best-first search* algorithm solves this problem by keeping track of an *open list* (a set of states ready to be expanded) and a *closed list* (a set of states that have already been expanded), beginning with an empty closed list and an open list including only the initial state. *Expanding* a state involves *generating* its successors and adding those that are not yet open or closed to the open list. Expanded states are moved from the open to the closed list.

The algorithm terminates with success when a goal state is generated and with failure when no further expansions are possible because the open list has run empty. The search behavior in the failure case is easy to analyze (all reachable states must be expanded), so we only consider the success case in the following.

A critical decision for the efficiency of GBFS is which state to expand next if there are multiple open states. GBFS always selects a state with minimum heuristic value among all candidates. If there are multiple minimizers, the algorithm is underspecified, and different choices of minimizers lead to different behaviors of GBFS. We refer to the process that selects among these possible behaviors as a *tie-breaking strategy*.

The execution of GBFS on a given state space topology where state s_i is expanded in the i-th iteration and where the algorithm terminates after n expansions is called a *run* $\langle s_1, \ldots, s_n\rangle$ of GBFS. In the trivial case where $s_{\text{init}} \in S_{\text{goal}}$, only the empty run is possible; otherwise, $s_1 = s_{\text{init}}$ and s_n always has a goal state as successor. We measure the performance of GBFS in terms of the number of state expansions, i.e., in the length of the run. This paper investigates the impact of tie-breaking on the performance of GBFS, with an emphasis on the *best case* (the shortest possible run for a given state space topology) and the *worst case* (the longest possible run). $\langle\mathrm{A}, \mathrm{G}, \mathrm{L}, \mathrm{E}, \mathrm{I}, \mathrm{M}\rangle$ and $\langle\mathrm{A}, \mathrm{G}, \mathrm{L}, \mathrm{C}, \mathrm{F}, \mathrm{D}, \mathrm{H}, \mathrm{I}, \mathrm{M}\rangle$ are best and worst case runs in our example state space topology with 6 and 9 expansions, respectively.

In the rest of this paper, we assume that the initial state has a larger heuristic value than all other states and that all goal states have lower heuristic values than all non-goal states. These assumptions help avoid special cases in some of the following definitions. They do not affect the behavior of GBFS: the initial state is always the first expanded state regardless of its heuristic value, and the heuristic values of goal states are never considered because we terminate as soon as a goal state is generated.

High-Water Marks and Benches We now briefly recapitulate the observations of our previous work [Heusner *et al.*, 2017] on the behavior of greedy best-first search in so far as they are relevant for this paper. We adapt the notations and terminology to suit the following discussions. The concept of *high-water marks* is central to the understanding of GBFS.

Definition 2. *Let* $\langle\mathcal{S}, h\rangle$ *be a state space topology with states* S*. The* high-water mark *of* $s \in S$ *is*

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

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

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

Intuitively, $hwm(s)$ is the largest heuristic value that needs to be considered to reach a goal state from s. Figure 1 shows the high-water mark values of states from our example state space topology. As A has the highest heuristic value of all paths from A to a goal state, $hwm(\mathrm{A}) = 4$, and $hwm(\mathrm{F}) = 3$ as all paths from F to N are via B and $h(\mathrm{B}) = 3$.

If S' is the set of open states at any time during a GBFS run, then at least one state s with $h(s) = hwm(S')$ and no state s with $h(s) > hwm(S')$ will be expanded during the rest of the run. Based on this observation, we showed that every GBFS run can be partitioned into *episodes*, where a new episode begins whenever a state is expanded whose high-water mark is lower than the high-water mark of all previously expanded states. We call the last state of every episode a *progress state*.

Definition 3. *A state* s *of a state space topology* $\langle\mathcal{S}, h\rangle$ *is a* progress state *iff* $hwm(s) > hwm(succ(s))$*.*

Progress states s always satisfy $\mathit{hwm}(s) = h(s)$. The high-water mark of the set of states expanded in an episode is always equal to $\mathit{hwm}(s)$, and the high-water mark of the set of states expanded after s has been expanded is always lower than $\mathit{hwm}(s)$.

Progress states are depicted in gray in our example state space topology in Figure 1. In the run $\langle \mathrm{A}, \mathrm{G}, \mathrm{L}, \mathrm{E}, \mathrm{I}, \mathrm{M} \rangle$, for instance, the episode that starts with the expansion of A lasts until the next progress state, E, is expanded. The high-water mark of the episode is $\mathit{hwm}(\{\mathrm{A}, \mathrm{G}, \mathrm{L}\}) = 3 = \mathit{hwm}(\mathrm{E})$ and the high-water mark of all states expanded after E is $\mathit{hwm}(\{\mathrm{I}, \mathrm{M}\}) = 2 < \mathit{hwm}(\mathrm{E})$.

The key insight into understanding the search behavior of GBFS is that progress states behave like *reset points* upon whose expansion the open list could be cleared without affecting the behavior of the algorithm. In the example just given, even though C is inserted into the open list upon expanding A, it will never be expanded by GBFS regardless of the tie-breaking strategy after the expansion of E.

Hence, we can simulate the search behavior by treating all episodes as separate searches on a restricted state space called *bench* with a progress state as the initial state and subsequent progress states as goal states.[2]

Definition 4. *Let $\langle \mathcal{S}, h \rangle$ be a state space topology with set of states S. Let $s \in S$ be a progress state.*

The bench level *of s is $\mathit{level}(s) = \mathit{hwm}(\mathit{succ}(s))$.*

The inner bench states *$\mathit{inner}(s)$ for s consist of all states $s'' \neq s$ that can be reached from s on paths on which all states $s' \neq s$ (including s'' itself) are non-progress states and satisfy $h(s') \leq \mathit{level}(s)$.*

The bench exit states *$\mathit{exit}(s)$ for s consist of all progress states s' with $h(s') = \mathit{level}(s)$ that are successors of s or of some inner bench state of s.*

The bench states *$\mathit{states}(s)$ for s are $\{s\} \cup \mathit{inner}(s) \cup \mathit{exit}(s)$.*

The bench *induced by s, denoted by $\mathcal{B}(s)$, is the state space with states $\mathit{states}(s)$, initial state s, and goal states $\mathit{exit}(s)$. The successor function is the successor function of $\mathcal{S}$ restricted to $\mathit{states}(s)$ without transitions to s and from bench exit states $\mathit{exit}(s)$.*

In our example state space topology, A induces the bench $\mathcal{B}(\mathrm{A})$ with $\mathit{inner}(\mathrm{A}) = \{\mathrm{C}, \mathrm{F}, \mathrm{G}, \mathrm{L}\}$ and $\mathit{exit}(\mathrm{A}) = \{\mathrm{B}, \mathrm{D}, \mathrm{E}\}$. Another bench is defined by progress state D with $\mathit{inner}(\mathrm{D}) = \{\mathrm{G}, \mathrm{H}, \mathrm{L}\}$ and $\mathit{exit}(\mathrm{D}) = \{\mathrm{I}\}$. Note that benches do not partition the state space but may overlap. In the example, states G and L are inner states both of $\mathcal{B}(\mathrm{A})$ and $\mathcal{B}(\mathrm{D})$.

The *bench transition system* is the meta state space where states are the benches that are generated by at least one GBFS run and where edges are such that two benches are connected if an exit state of one bench induces the other bench.

Definition 5. *Let $\mathcal{T} = \langle \mathcal{S}, h \rangle$ be a state space topology with initial state s_{init}. The* bench transition system *$\mathcal{B}(\mathcal{T})$ 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 the smallest sets that satisfy the following properties:*

1. *$\mathcal{B}(s_{\mathit{init}}) \in V$*
2. *If $\mathcal{B}(s) \in V$, $s' \in \mathit{exit}(s)$, and s' is a non-goal state, then $\mathcal{B}(s') \in V$ and $\langle \mathcal{B}(s), \mathcal{B}(s') \rangle \in E$.*

A bench may contain craters which consist of states that are guaranteed to be expanded after expansion of a state.

Definition 6. *Let $\langle \mathcal{S}, h \rangle$ be a state space topology with set of states S.*

The crater *$\mathit{crater}(s)$ of a progress state $s \in S$ is the set of all states $s'' \in S$ that can be reached from s on paths on which all states $s' \neq s$ (including s'') satisfy $h(s') < \mathit{hwm}(\mathit{succ}(s))$ and $\mathit{hwm}(s') \geq \mathit{hwm}(\mathit{succ}(s))$, and the crater of a non-progress state s is the set of all states $s'' \in S$ that can be reached from s on paths on which all states $s' \neq s$ (including s'') satisfy $h(s') < h(s)$ and $\mathit{hwm}(s') \geq \mathit{hwm}(s)$.*

The escape states *$\mathit{escape}(s)$ of a progress state s are all successor states s'' of states $s' \in \mathit{crater}(s)$ that satisfy $h(s'') = \mathit{hwm}(\mathit{succ}(s))$ and the escape state of a non-progress state s are all successor states s'' of states $s' \in \mathit{crater}(s)$ that satisfy $h(s'') = h(s)$.*

Progress state A induces crater $\mathit{crater}(\mathrm{A}) = \{\mathrm{G}, \mathrm{L}\}$ with $\mathit{escape}(\mathrm{A}) = \{\mathrm{D}\}$, and non-progress state C induces $\mathit{crater}(\mathrm{C}) = \{\mathrm{F}, \mathrm{G}, \mathrm{L}\}$ with $\mathit{escape}(\mathrm{C}) = \{\mathrm{B}, \mathrm{D}\}$. Note that G and L are part of both craters.

3 NP-Completeness Results

We now formally introduce the problems of analyzing the best and worst case of GBFS. As is usual in complexity theory, we study the related *decision problems* and show that these are already NP-complete. It follows that the more general problems of computing the number of states expanded in the best/worst case and computing the best/worst runs are NP-equivalent [Garey and Johnson, 1979].

Definition 7. *The* GBFSBESTCASE *problem is defined as follows: given a state space topology $\mathcal{T}$ and $K \in \mathbb{N}_0$, does there exist a GBFS run on $\mathcal{T}$ with at most K expanded states?*

Definition 8. *The* GBFSWORSTCASE *problem is defined as follows: given a state space topology $\mathcal{T}$ and $K \in \mathbb{N}_0$, does there exist a GBFS run on $\mathcal{T}$ with at least K expanded states?*

We now show that both problems are NP-complete. To be clear, we assume that the state spaces in the input are explicitly represented as directed graphs, so the hardness is *not* related to the state explosion problem that makes state-space search in implicitly defined state spaces hard.

Theorem 1. GBFSBESTCASE *is NP-complete.*

Proof: For membership in NP, we guess a run for the given state space topology that satisfies the given bound and verify that it is a legal run. For hardness, we polynomially reduce from the NP-complete VERTEXCOVER problem: given a graph $\langle V, E \rangle$ and number M, does there exist a subset $C \subseteq V$ with $|C| \leq M$ such that $C \cap e \neq \emptyset$ for all $e \in E$?

Given a VERTEXCOVER instance with vertices V, edges $E = \{e_1, \ldots, e_n\}$ and bound M, we produce a GBFSBESTCASE instance with bound $K = 2n + 1 + M$ and a state space topology with initial state s_1 with heuristic value 3; *edge branch states* $s_2, \ldots, s_{n+1}$ with heuristic value 2; for

[2] The main differences of the following definitions to our previous work [Heusner *et al.*, 2017] are that we define benches as state spaces, that we consider the progress state which induces a bench as part of the bench and that we only consider *reduced* benches.

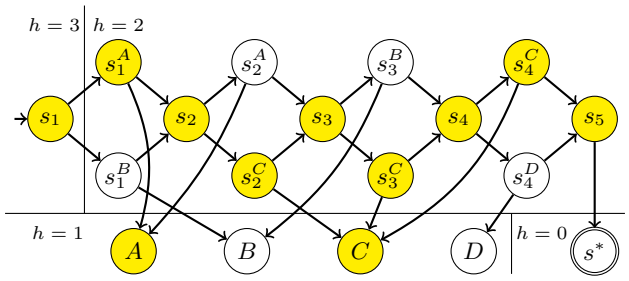

Figure 2: Illustration of the proof of Theorem 1, depicting a GBFS run that allows to compute the vertex cover of a graph $\langle\{A, B, C, D\}, \{\{A, B\}, \{A, C\}, \{B, C\}, \{C, D\}\}\rangle$.

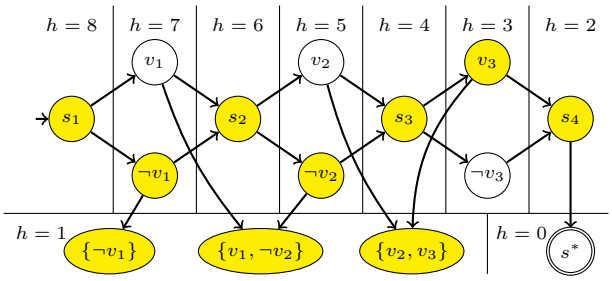

Figure 3: Illustration of the proof of Theorem 2, depicting a GBFS run that allows to determine the satisfiability of the formula $\neg v_1 \wedge (v_1 \vee \neg v_2) \wedge (v_2 \vee v_3)$.

every edge $e_i = \{u, v\}$, two *edge decision states* s_i^u and s_i^v with heuristic value 2; for every vertex $v \in V$, a *vertex state* v with heuristic value 1; a goal state s^* with heuristic value 0; for every edge e_i and vertex $v \in e_i$, transitions $s_i \to s_i^v$, $s_i^v \to s_{i+1}$ and $s_i^v \to v$; and a transition $s_{n+1} \to s^*$.

The reduction is illustrated in Figure 2. Every GBFS run must expand the initial state (1 expansion), all edge branch states (n expansions) and at least one edge decision state per edge (n expansions). If edge decision state e_i^u is expanded, vertex state u must also be expanded. It is easy to see that the expanded vertex states form a vertex cover, and that a run with at most $K = 2n + 1 + M$ expansions exists iff there exists a vertex cover of size at most M. □

We remark that the state space topology used in the reduction only consist of a single bench, and that the best case of GBFS is hence already hard to compute for a single bench. The difficulty of minimizing GBFS expansions stems from *overlapping crater states*: expanding certain states (the edge decision states in the reduction) forces the expansion of adjacent *crater states* that form a heuristic depression (the vertex states in the reduction). The cost incurred by crater state expansions cannot be quantified locally because craters can overlap and global optimization is hence required.

In contrast to the best case, the *worst case* of GBFS is easy to determine for a single bench: the worst case on a single bench is always to expand all non-exit states in some order, and then any exit state. (This is always possible.) Still, analyzing the worst case of GBFS is hard as we prove next.

Theorem 2. GBFSWORSTCASE *is NP-complete.*

Proof: For membership in NP, we guess a run for the given state space topology that satisfies the given bound and verify that it is a legal run. For hardness, we polynomially reduce from the NP-complete SAT problem: given a set of propositional variables V and a set of clauses C over V (represented as sets of literals), does there exist a truth assignment for V that satisfies all clauses?

Given a SAT instance with variables $V = \{v_1, \ldots, v_n\}$ and clauses C, we produce a GBFSWORSTCASE instance with bound $K = 2n + 1 + |C|$ and a state space topology with *variable branch states* $s_1, \ldots, s_{n+1}$ with heuristic values $h(s_i) = 2(n - i) + 4$ (s_1 is the initial state); for every variable v_i, two *literal states* v_i and $\neg v_i$ with heuristic value $2(n - i) + 3$; for every clause $c \in C$, a *clause state* c with heuristic value 1; a goal state s^* with heuristic value 0; for every variable v_i and literal $\ell \in \{v_i, \neg v_i\}$, transitions $s_i \to \ell$, $\ell \to s_{i+1}$ and $\ell \to c$ for all clauses c with $\ell \in c$; and a transition $s_{n+1} \to s^*$.

The reduction is illustrated in Figure 3. It is very similar to the one from the previous proof. The main difference is that we define the heuristic values in such a way that GBFS is prevented from expanding both literal states belonging to the same variable in one run. Truth assignments correspond to paths through the variable branch and literal states. From this, it is easy to see that satisfying assignments correspond to runs where all clause vertices are expanded, which is equivalent to saying that at least $K = 2n + 1 + |C|$ states are expanded. □

Again, the hardness stems from the difficulties caused by overlapping craters. The difference to the result above is that now the overlap is between craters from *different* benches, as determining the worst case for a single bench is easy.

4 Algorithms

Despite the discouraging results of the previous section, we now present algorithms for the computation of the best- and worst-case tie-breaking behavior of GBFS. A straightforward approach searches in the space of open and closed list configurations and starts with a search node whose open list only contains the initial state and whose closed list is empty. Search nodes are expanded by creating successors for each state in the open list of the search node with minimal heuristic value, and successor nodes are equal to their predecessors except that the expanded state is moved to the closed list and all successor states of the expanded state that are not yet open or closed are added to the open list.

Applying this algorithm to our example state space topology leads to a search node whose open list contains the states C and E and whose closed list conisists of A, G and L. Expanding this search node creates the two successors $\langle\{\mathrm{E}, \mathrm{F}\}, \{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{L}\}\rangle$ and $\langle\{\mathrm{C}, \mathrm{I}\}, \{\mathrm{A}, \mathrm{E}, \mathrm{G}, \mathrm{L}\}\rangle$. The straightforward algorithm keeps track of the whole open and closed list in each search node and creates a search node for each possible open and closed list configuration. We present two algorithms in the following that operate on a more compact data structure and omit search inside of craters (for GBFSBESTCASE) and benches (for GBFSWORSTCASE) by exploiting previous insights on the search behavior of GBFS.

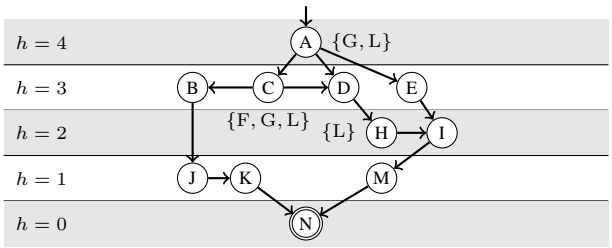

Figure 4: Surface graph of our example state space topology from Figure 1. States are annotated with their crater unless it is empty.

Best Case Analysis The algorithm that determines the minimum number of states expanded by GBFS for a given state space topology operates on the *surface graph*, the search space that contains only states s with $h(s) = \mathit{hwm}(s)$ that are expanded by at least one instance of GBFS. To derive the surface graph, we

1. compute $\mathit{hwm}(s_{\text{init}})$ by running an instance of GBFS with an arbitrary tie-breaking strategy;
2. expand all states s with $h(s) \leq \mathit{hwm}(s_{\text{init}})$ that can be reached from s_{init} without passing through a goal state;
3. determine $\mathit{hwm}(s)$ for all expanded states s by backpropagation from goal states;
4. reduce the state space by removing all states and adjacent transitions that are not included in a bench of the bench transitions system;
5. remove all remaining states s and adjacent transitions with $h(s) \neq \mathit{hwm}(s)$; and
6. for each remaining state s, add a transition from s to each $s' \in \mathit{escape}(s)$.

Steps 1 and 2 compute an overapproximation of the set of possibly expanded states, and steps 3 and 4 refine the approximation to the exact set (the fact that finite high-water marks larger than $\mathit{hwm}(s_{\text{init}})$ are considered infinite by our procedure is irrelevant for the computation of the set of possibly expanded states). The final steps 5 and 6 reduces this further to the surface graph $\langle \overline{S}, \overline{E} \rangle$ by removing all states with different heuristic and high-water mark values and by adding transitions between states that lead into and out of each crater.

Our algorithm exploits the fact that we are not interested in the behavior of any tie-breaking strategy, but in the *best-case* behavior of GBFS: first, it only considers successors nodes that are implied by a transition, as a best-case run must be such that subsequent states in the surface graph are connected in the surface graph (this is not the case for other tie-breaking strategies: e.g., it is possible to expand state E after state F in our example even though they are not connected). And second, rather than keeping track of all states in the closed list it restricts to the set of states that can be encountered in the best-case more than once, i.e., visited states that are part of at least one crater. By picking an appropriate cost function that ensures that states that have to be expanded are reflected in the cost of the expansion and by tracking visited crater states, we can turn the problem into a shortest path problem that can be solved with *uniform cost search* on the surface graph. Search nodes are hence pairs $\langle s, D \rangle$, where $s \in \overline{S}$ is the last expanded state, and D is the set of crater states that has been expanded on the way to s.

A search node $\langle s, D \rangle$ is expanded by creating a successor $\langle s', D \cup \mathit{crater}(s') \rangle$ for each $(s, s') \in \overline{E}$, and the cost of the transition from $\langle s, D \rangle$ to $\langle s', D' \rangle$ is $\mathit{cost}(\langle s, D \rangle, \langle s', D' \rangle) = 1 + |D' \setminus D|$; 1 for the expansion of s' and $|D' \setminus D|$ for the expansion of previously unexpanded crater states. The initial search node is $\langle s_{\text{init}}, \mathit{crater}(s_{\text{init}}) \rangle$ and each $\langle s, D \rangle$ with $\mathit{succ}(s) \cap S_{\text{goal}} \neq \emptyset$ is a goal node. The number of states that are expanded by GBFS under best-case tie-breaking corresponds to the sum of cost of the shortest path and $1 + |\mathit{crater}(s_{\text{init}})|$ for the expansion of the initial state and its crater states.

The surface graph of our example state space topology is depicted in Figure 4. It contains exactly the states s with $h(s) = \mathit{hwm}(s)$. All crater states $\mathit{crater}(s)$, which are annotated with s in Figure 4, are removed from the surface graph. This also leads to the removal of the transition from H to G even though G is not in a crater on $\mathcal{B}(\mathrm{D})$. However, the transition from H to G cannot be part of a best-case run of GBFS because there cannot be a path to a goal from G on $\mathcal{B}(\mathrm{D})$ (since it is part of another crater on a higher bench, it only has escape states on the higher bench).

Computing GBFSBESTCASE on the surface graph rather than a search in the space of open and closed list configurations results in smaller search nodes where we only have to keep track of the last expanded state and all expanded crater states, and it results in a smaller number of search nodes since we do not perform search in craters (the expansion of crater states is encoded in the cost and transition functions) and non-crater states with infinite high-water mark which can never be part of a best-case tie-breaking strategy.

Worst Case Analysis The introduction of the surface graph allows us to perform uniform cost search to determine the best-case behavior of GBFS. For the worst-case, we can simplify the search space even further since transitions within benches are irrelevant: for each bench, all non-exit states and a single exit state are expanded in the worst-case (it is only a single exit state as an algorithm has to move onto the next bench once an exit state is expanded). To put this insight into an algorithm, we reduce the surface graph even further and remove all transitions within one bench and add a transition from the progress state that defines a bench to all exit states of the bench. The resulting graph is equivalent to the bench transition system (with the exception that nodes are progress states rather than benches induced by the progress states), which we have shown to be a directed acyclic graph in previous work [Heusner *et al.*, 2017].

We derive a similar search structure from this graph as in the best-case. The only difference is the successor generation, which creates a search node $\langle s', D \cup \mathit{inner}(s') \rangle$ for each (s, s') in the bench transition system when $\langle s, D \rangle$ is expanded. In this search space, we perform a *longest path search*, which can be computed in time polynomial in the size of an directed acyclic graph [Lawler, 1976] as the *hwm*-levels induce a suitable topological ordering.

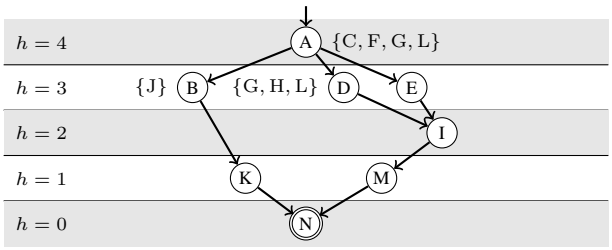

Figure 5: Bench transition system of our example state space topology from Figure 1. Progress states are annotated with the set of inner bench states of the induced bench unless it is empty.

5 Tractability Results

In this section, we discuss properties of state space topologies that can be determined in polynomial time and that allow a polynomial computation of GBFSBESTCASE and GBFSWORSTCASE despite the complexity of the general case.

Definition 9. *Let $\mathcal{T}$ be a state space topology with set of states S and bench transition system $\mathcal{B}(\mathcal{T}) = \langle V, E \rangle$.*

$\mathcal{T}$ is undirected *iff $s' \in succ(s) \Leftrightarrow s \in succ(s')$ for all states $s, s' \in S$.*

$\mathcal{T}$ is overlap-free with respect to benches *iff $inner(s) \cap inner(s') = \emptyset$ for all benches $\mathcal{B}(s), \mathcal{B}(s') \in V$ with $level(s) \neq level(s')$.*

$\mathcal{T}$ is overlap-free with respect to craters *iff $crater(s) \cap crater(s') = \emptyset$ for all pairwise distinct $s, s' \in states(p)$ and all benches $\mathcal{B}(p) \in V$ with $h(s) = h(s') = hwm(s) = hwm(s') = level(p)$.*

Let us now investigate what these properties mean for the complexity of GBFSWORSTCASE and GBFSBESTCASE.

Theorem 3. GBFSWORSTCASE *can be decided in polynomial time and space for a given state space topology $\mathcal{T}$ and $K \in \mathbb{N}_0$ if $\mathcal{T}$ is overlap-free with respect to benches.*

Proof: We know that benches along each path in the bench transition system have monotonously decreasing and therefore distinct *hwm*-levels [Heusner *et al.*, 2017]. Moreover, the complexity of GBFSWORSTCASE results from states that occur on benches of different *hwm*-levels along a bench path.

As each pair of benches in a run is overlap-free, we can determine the worst-case behavior of GBFS by running our general algorithm of the previous section *directly on the bench transition system* with cost function $cost(s, s') = 1 + |inner(s')|$ in time linear in the number of exit states from the bench transition system.[3] □

Theorem 4. GBFSWORSTCASE *can be decided in polynomial time and space for a given state space topology $\mathcal{T}$ and $K \in \mathbb{N}_0$ if $\mathcal{T}$ is undirected.*

Proof: We show that a state can only be shared between two consecutive benches of the bench transition system.

Let us assume there is a state s and three consecutive benches defined by progress states p, p', and p'' with $level(p) > level(p') > level(p'')$ such that $s \in inner(p)$ and $s \in inner(p'')$. As s is in $inner(p)$, we know that $hwm(s) \geq level(p)$ (1). As s is in $inner(p'')$ and as $\mathcal{S}$ is undirected, there must be a path from s over p'' to a goal bench which only includes states s' with $h(s') \leq level(p'')$, and hence $h(p'') \geq hwm(s)$ (2). Finally, as p'' is in $exit(p')$, we know that $level(p') = h(p'')$ and hence $level(p) > h(p'')$ due to $level(p) > level(p')$ (3).

With this, we have $hwm(s) \overset{(1)}{\geq} level(p) \overset{(3)}{>} h(p'') \overset{(2)}{\geq} hwm(s)$, which contradicts the assumption.

Since dependencies can only arise between consecutive benches, we can determine the worst-case behavior of GBFS by running our general algorithm of the previous section *directly on the bench transition system* with cost function $cost(s, s') = 1 + |inner(s') \setminus inner(s)|$. □

	best case				worst case			
	of	**u**	**o**	**total**	**of**	**u**	**o**	**total**
instances	406	31	327	764	471	10	283	764
covered	406	31	242	679	466	10	263	739

Table 1: Number of undirected (u), overlap-free (of), other (o) and total number of instances where the search space and the best- and worst-case tie-breaking have been computed.

Theorem 5. GBFSBESTCASE *can be decided in polynomial time and space for a given state space topology $\mathcal{T}$ and $K \in \mathbb{N}_0$ if $\mathcal{T}$ is overlap-free with respect to craters and benches.*

Proof: The proof is analogously to the proof of Theorem 3 applied to craters instead of benches and with cost function $cost(s, s') = 1 + |crater(s')|$. □

Theorem 6. GBFSBESTCASE *can be decided in polynomial time and space for a given state space topology $\mathcal{T}$ and $K \in \mathbb{N}_0$ if $\mathcal{T}$ is undirected.*

Proof: Since $s \in escape(s')$ iff $s' \in escape(s)$ in undirected state spaces, the surface graph of an undirected state space is undirected as well. Moreover, $(s, s') \in \overline{E}$ of the surface graph iff $crater(s) \cap crater(s') \neq \emptyset$. Therefore, dependencies arise only between two neighboring states and we can run the general algorithm directly on the surface graph with cost function $cost(s, s') = 1 + |crater(s') \setminus crater(s)|$. □

6 Experimental Results

We implemented all presented algorithms in the Fast Downward planning system [Helmert, 2006] and performed experiments on the benchmark sets of the 1998–2014 International Planning Competitions (IPC) with the h^{FF} heuristic [Hoffmann and Nebel, 2001] and unit cost. We restricted the benchmark sets to tasks where GBFS search with h^{FF} and FIFO tie-breaking found a plan within 30 minutes and 3.5 GB memory, which has been possible for 2519 instances.

For each instance, we computed the reduced state space with step 1 - 4 of surface graph computation, and determined the relevant properties of the state space topology, i.e., if it is overlap-free (with respect to benches and craters) or not overlap-free but undirected. This was possible for 764 instances from 78 domains within a time limit of 30 minutes

[3] Our result also holds if there is no pair of benches on a path in the bench transition system that shares a state. However, since it is not possible to compute this property in polynomial time for a given state space topology, we present the less general result here.

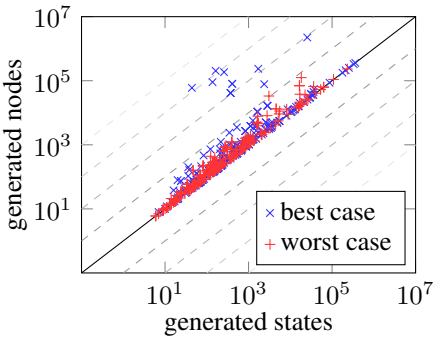

Figure 6: Number of states versus number of search nodes generated by the best case and worst case algorithms for each search space that is neither undirected nor overlap-free.

and a memory limit of 3.5 GB. These instances form the benchmark set that we used for the further experiments. The surface graphs and the bench transition systems were generated on the fly for the computation of best or worst case in order to save memory.

6.1 Evaluation of Tractability

An overview on the number of instances for which we were able to compute the minimal and/or the maximal number of states that are expanded by GBFS is given in Table 1. In 665 instances, both values have been computed, and only the best or worst case in an additional 14 and 74, respectively.

Even though the percentage of instances for which we could perform the tie-breaking analysis is high in general, one can see that the success rate is much higher for the polynomial special cases: we have results for the best case for all overlap-free or undirected instances, while there are only 5 instances for which we could not come up with an upper bound on the number of GBFS expansions.

The NP-hardness of the general problem can be observed for the numbers on the remaining instances, where the possibly exponential blow-up seems to be more present when computing the best case (not successfull in 85 instances) than when computing the worst case (not successfull in just 20 instances). To support this hypothesis, Figure 6 plots the number of generated states in the surface graph (best case) or bench transition system (worst case) in comparison to the number of generates search nodes for each instance that is neither undirected nor overlap-free. For the worst case, the blow-up is almost always signifcantly below an order of magnitude, while it is usually close to one and up to more than three orders of magnitude for the best case.

6.2 Evaluation of Tie-Breaking Strategies

In our second experiment, we compare the performance of the well-known tie-breaking strategies FIFO, LIFO and random (averaged over 10 runs) with the best and worst case. One of our main results is that the influence of tie-breaking is indeed highly significant in practice - Figure 7 shows that GBFS with worst-case tie-breaking has to expand approximately 10 times as many states as GBFS with best-case tie-breaking in state space topologies without craters (left plot), and as much as 100 as many in the presence of craters (center plot).

The three considered tie-breaking strategies are not far from the best case in crater-free state space topologies. This result matches the insights of Hoffmann [2005], whose heuristic benches relate to high-water mark benches in crater-free instances. In state space topologies with craters, the picture is different: the number of expanded states of the specific tie-breaking strategies is about 10 times the number of the best case. Since the differences between FIFO, LIFO and random are additionally negligible, we assume that no good GBFS tie-breaking strategy has been found so far.

6.3 Worst Case and Possibly Expanded States

In our final experiment, we are interested in the relationship between the number of possibly expanded states and the number of states expanded under worst-case tie-breaking, which is shown in Figure 7 for state spaces with or without craters. In the presence of craters, both numbers are similar, i.e., almost all states that are expanded by some tie-breaking strategy are expanded in the worst case. We assume this is because most states can be reached on many different ways.

In crater-free state space topologies, this is not the case, and the number of states expanded by the worst-case tie-breaking strategy is often less than 1% of the possibly expanded states. This is very different from the result in A* and is one explanation why GBFS is often a good choice in satisficing planning.

7 Conclusion

We analyzed the problem of computing the number of states expanded by GBFS under worst-case and best-case tie-breaking and showed that the computation is NP-complete for the general case, but polynomial in undirected or overlap-free state space topologies. We presented worst-case exponential time algorithms for arbitrary state spaces and polynomial variants for the tractable subsets.

Our experimental analysis reveals that the gap between best- and worst-case tie-breaking is large and that there is plenty of room for improvement over standard tie-breaking strategies in state space topologies with craters.

We plan to extend our work in multiple directions: there is room for improvement in the computation of the graphs required for our analysis, which is a promising direction since the current implementation works for only approximately one third of the instances that can be solved by GBFS; we plan to use our insights to design tie-breaking strategies for GBFS that are closer to the best case even in the presence of craters; and we work on a comparison of different heuristics, where numbers on best- and worst-case tie-breaking strategies allow to compare the quality of heuristics for GBFS similar to the dominance of higher admissible heuristic estimates in A*.

Acknowledgments

This work was supported by the European Research Council as part of the project “State Space Exploration: Principles, Algorithms and Applications”.

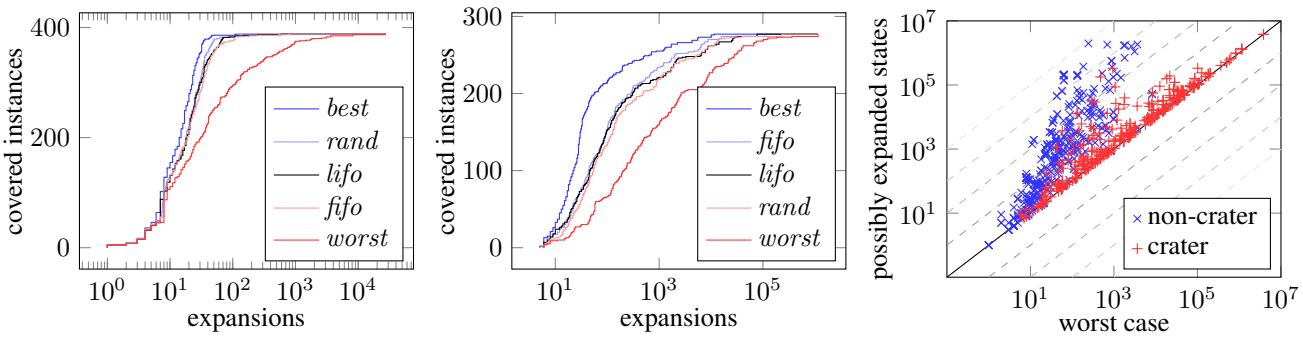

Figure 7: Left and middle: number of instances where GBFS expands at most a given number of states for different tie-breaking strategies. (Legend is sorted by decreasing performance of tie-breaking strategies) Right: comparison of the number of possibly expanded states and expanded states under worst case tie-breaking.

References

[Asai and Fukunaga, 2017] Masataro Asai and Alex Fukunaga. Exploration among and within plateaus in greedy best-first search. In *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling (ICAPS 2017)*, pages 11–19. AAAI Press, 2017.

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

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

[Garey and Johnson, 1979] Michael R. Garey and David S. Johnson. *Computers and Intractability — A Guide to the Theory of NP-Completeness*. Freeman, 1979.

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

[Helmert, 2006] Malte Helmert. The Fast Downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.

[Heusner *et al.*, 2017] Manuel Heusner, Thomas Keller, and Malte Helmert. Understanding the search behaviour of greedy best-first search. In Alex Fukunaga and Akihiro Kishimoto, editors, *Proceedings of the 10th Annual Symposium on Combinatorial Search (SoCS 2017)*, pages 47–55. AAAI Press, 2017.

[Hoffmann and Nebel, 2001] Jörg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

[Hoffmann, 2005] Jörg Hoffmann. Where 'ignoring delete lists' works: Local search topology in planning benchmarks. *Journal of Artificial Intelligence Research*, 24:685–758, 2005.

[Holte, 2010] Robert C. Holte. Common misconceptions concerning heuristic search. In Ariel Felner and Nathan Sturtevant, editors, *Proceedings of the Third Annual Symposium on Combinatorial Search (SoCS 2010)*, pages 46–51. AAAI Press, 2010.

[Korf *et al.*, 2001] Richard E. Korf, Michael Reid, and Stefan Edelkamp. Time complexity of iterative-deepening A*. *Artificial Intelligence*, 129:199–218, 2001.

[Lawler, 1976] Eugene L. Lawler. *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart and Winston, 1976.

[Martelli, 1977] Alberto Martelli. On the complexity of admissible search algorithms. *Artificial Intelligence*, 8:1–13, 1977.

[Pearl, 1984] Judea Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, 1984.

[Wilt and Ruml, 2014] Christopher Wilt and Wheeler Ruml. Speedy versus greedy search. In Stefan Edelkamp and Roman Barták, editors, *Proceedings of the Seventh Annual Symposium on Combinatorial Search (SoCS 2014)*, pages 184–192. AAAI Press, 2014.

[Wilt and Ruml, 2015] Christopher Wilt and Wheeler Ruml. Building a heuristic for greedy search. In Levi Lelis and Roni Stern, editors, *Proceedings of the Eighth Annual Symposium on Combinatorial Search (SoCS 2015)*, pages 131–139. AAAI Press, 2015.

[Wilt and Ruml, 2016] Christopher Wilt and Wheeler Ruml. Effective heuristics for suboptimal best-first search. *Journal of Artificial Intelligence Research*, 57:273–306, 2016.