Sample Complexity of Learning Heuristic Functions for Greedy-Best-First and A* Search

Anonymous Author(s)
Affiliation
Address
email

Abstract

Greedy best-first search (GBFS) and A* search (A*) are popular algorithms for path-finding on large graphs. Both use so-called heuristic functions, which estimate how close a vertex is to the goal. While heuristic functions have been handcrafted using domain knowledge, recent studies demonstrate that learning heuristic functions from data is effective in many applications. Motivated by this emerging approach, we study the sample complexity of learning heuristic functions for GBFS and A*. We build on a recent framework called *data-driven algorithm design* and evaluate the *pseudo-dimension* of a class of utility functions that measure the performance of parameterized algorithms. Assuming that a vertex set of size n is fixed, we present $\mathrm{O}(n \lg n)$ and $\mathrm{O}(n^2 \lg n)$ upper bounds on the pseudo-dimensions for GBFS and A*, respectively, parameterized by heuristic function values. The upper bound for A* can be improved to $\mathrm{O}(n^2 \lg d)$ if every vertex has a degree of at most d and to $\mathrm{O}(n \lg n)$ if edge weights are integers bounded by $\mathrm{poly}(n)$. We also give $\Omega(n)$ lower bounds for GBFS and A*, which imply that our bounds for GBFS and A* under the integer-weight condition are tight up to a $\lg n$ factor. Finally, we discuss a case where the performance of A* is measured by the suboptimality and show that we can sometimes obtain a better guarantee by combining a parameter-dependent worst-case bound with a sample complexity bound.

1 Introduction

Given a graph with a start vertex s, a goal vertex t, and non-negative edge weights, we consider finding an s–t path with a small total weight. The Dijkstra algorithm [14] finds an optimal path by exploring all vertices that are as close to s as t. It, however, is sometimes impractical for large graphs since exploring all such vertices is too costly. Heuristic search algorithms are used to address such situations; among them, greedy best-first search (GBFS) [15] and A* search (A*) [21] are two popular algorithms. Both GBFS and A* use so-called heuristic functions, which estimate how close an input vertex is to t. GBFS/A* attempts to avoid redundant exploration by scoring vertices based on heuristic function values and iteratively expanding vertices with the smallest score. If well-suited heuristic functions are available, GBFS/A* can run much faster than the Dijkstra algorithm. Furthermore, if A* uses an *admissible* heuristic function, i.e., it never overestimates the shortest-path distance to t, it always finds an optimal path [21]. Traditionally, heuristic functions have been made based on domain knowledge; e.g., if graphs are road networks, the Euclidean distance gives an admissible heuristic.

When applying GBFS/A* to various real-world problems, a laborious process is to handcraft heuristic functions. Learning heuristic functions from data can be a promising approach to overcoming the obstacle due to the recent development of technologies for collecting graph data. Reseachers have demonstrated the effectiveness of this approach in robotics [9, 29, 25, 33], computational organic chemistry [11], and predestrian trajectory prediction [33]. With learned heuristic functions, however,

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

obtaining theoretical guarantees is difficult since we can hardly understand how the search can be guided by such heuristic functions. (A recent paper [1] studies learning of admissible heuristics for A*, but the optimality is confirmed only empirically.) Moreover, learned heuristic functions may be overfitting to problem instances at hand. That is, even if GBFS/A* with learned heuristic functions perform well over training instances, they may deliver poor future performance. In summary, the emerging line of work on search algorithms with learned heuristic functions is awaiting a theoretical foundation for guaranteeing their performance in a data-driven manner. Thus, a natural question is: *how many sampled instances are needed to learn heuristic functions with generalization guarantees on the performance of resulting GBFS/A*?*

1.1 Our contribution

We address the above question, assuming that path-finding instances defined on a fixed vertex set of size n are drawn i.i.d. from an unknown distribution. Our analysis is based on so-called *data-driven algorithm design* [19, 3], a PAC-learning framework for bounding the sample complexity of algorithm configuration. In the analysis, the most crucial step is to evaluate the *pseudo-dimension* of a class of utility functions that measure the performance of parameterized algorithms. We study the case where GBFS/A* is parameterized by heuristic function values and make the following contributions:

1. Section 3 gives $\mathrm{O}(n \lg n)$ and $\mathrm{O}(n^2 \lg n)$ upper bounds on the pseudo-dimensions for GBFS and A*, respectively. The bound for A* can be improved to $\mathrm{O}(n^2 \lg d)$ if every vertex has an at most d degree and to $\mathrm{O}(n \lg n)$ if edge weights are non-negative integers at most $\mathrm{poly}(n)$.
2. Section 4 presents $\Omega(n)$ lower bounds on the pseudo-dimensions for GBFS and A*. We prove this result by constructing $\Omega(n)$ instances with unweighted graphs. Thus, our bounds for GBFS and A* under the integer edge-weight condition are tight up to a $\lg n$ factor.
3. Section 5 studies a particular case of bounding the suboptimality of A*. We show that we can sometimes improve the guarantee obtained in Section 3 by using an alternative $\mathrm{O}(n \lg n)$ bound on the pseudo-dimension of a class of parameter-dependent worst-case bounds [31].

An important consequence of the above results is the tightness up to a $\lg n$ factor for GBFS and A* under the integer-weight assumption. Note that this assumption holds in various realistic situations. For example, the Internet network and state-space graphs of games are unweighted (unit-weight) graphs, and A* is often applied to path-finding instances on such graphs.

1.2 Related work

Data-driven algorithm design. Gupta and Roughgarden [19] proposed a PAC approach for bounding the sample complexity of algorithm configuration, which is called *data-driven algorithm design* and has been applied to a broad family of algorithms, including greedy, clustering, and sequence alignment algorithms. We refer the reader to a nice survey [3]. A recent line of work [4, 7, 8] has extensively studied the sample complexity of configuring integer-programming methods, e.g., branch-and-bound and branch-and-cut. In [7, 8], upper bounds on the pseudo-dimension for general tree search are presented, which are most closely related to our results. Our upper bounds, which are obtained by using specific properties of GBFS/A*, are better than the previous bounds for general tree search, as detailed in Appendix A. Balcan et al. [6] presented a general framework for evaluating the pseudo-dimension. Their idea is to suppose that performance measures form a class of functions of algorithm parameters, called *dual* functions, and characterize its complexity based on how they are piecewise structured. This idea plays a key role in the analysis of [7, 8], and our analysis of the upper bounds are also inspired by their idea. Its application to our setting, however, requires a close look at the behavior of GBFS/A*. Balcan et al. [5] showed that approximating dual functions with simpler ones is useful for improving sample complexity bounds, which is similar to our idea in Section 5. A difference is that while they construct simpler functions with a dynamic programming algorithm, we can use a known worst-case bound on the suboptimality of best-first search [31]. Lower bounds on the pseudo-dimension for graph-search algorithms have not been well studied.

Heuristic search with learning. Eden et al. [16] theoretically studied how the average-case running time of A* can be affected by the dimensions or bits of learned embeddings or labels of vertex features, based on which heuristic function values and computed. The sample complexity of learning heuristic functions, however, has not been studied.

2 Preliminaries

We present the background on learning theory and our problem setting. In what follows, we let $\mathbb{I}(\cdot)$ be a boolean function that returns 1 if its argument is true and 0 otherwise. We use $\mathcal{H} \subseteq \mathcal{R}^{\mathcal{Y}}$ to denote a class of functions that map $\mathcal{Y}$ to $\mathcal{R} \subseteq \mathbb{R}$. For any positive integer m, we let $[m] = \{1, \ldots, m\}$.

2.1 Backgound on learning theory

The following *pseudo-dimension* [26] is a fundamental notion for quantifying the complexity of a class of real-valued functions.

Definition 1. Let $\mathcal{H} \subseteq \mathbb{R}^{\mathcal{Y}}$ be a class of functions that map some domain $\mathcal{Y}$ to $\mathbb{R}$. We say a set $\{y_1, \ldots, y_N\} \subseteq \mathcal{Y}$ is *shattered* by $\mathcal{H}$ if there exist target values, $t_1, \ldots, t_N \in \mathbb{R}$, such that

$$|\{\, (\mathbb{I}(h(y_1) \geq z_1), \ldots, \mathbb{I}(h(y_N) \geq z_N)) \mid h \in \mathcal{H} \,\}| = 2^N.$$

The *pseudo-dimension* of $\mathcal{H}$, denoted by $\mathrm{Pdim}(\mathcal{H})$, is the size of a largest set shattered by $\mathcal{H}$.

If $\mathcal{H}$ is a set of binary-valued functions that map $\mathcal{Y}$ to $\{0, 1\}$, the pseudo-dimension of $\mathcal{H}$ coincides with the so-called *VC-dimension* [32], which is denoted by $\mathrm{VCdim}(\mathcal{H})$.

The following proposition enables us to obtain sample complexity bounds by evaluating the pseudo-dimension (see, e.g., [2, Theorem 13.6] and [24, Theorem 11.8]).

Proposition 1. *Let $H > 0$, $\mathcal{H} \subseteq [0, H]^{\mathcal{Y}}$, and $\mathcal{D}$ be a distribution over $\mathcal{Y}$. For any $\delta \in (0, 1)$, with a probability of at least $1 - \delta$ over the i.i.d. draw of $\{y_1, \ldots, y_N\} \sim \mathcal{D}^N$, for all $h \in \mathcal{H}$, it holds that*

$$\left| \frac{1}{N} \sum_{i=1}^{N} h(y_i) - \mathop{\mathbb{E}}_{y \sim \mathcal{D}}[h(y)] \right| = \mathrm{O}\left(H \sqrt{\frac{\mathrm{Pdim}(\mathcal{H}) \lg \frac{N}{\mathrm{Pdim}(\mathcal{H})} + \lg \frac{1}{\delta}}{N}} \right).$$

In other words, for any $\epsilon > 0$, $N = \Omega\left(\frac{H^2}{\epsilon^2} \left(\mathrm{Pdim}(\mathcal{H}) \lg \frac{H}{\epsilon} + \lg \frac{1}{\delta} \right) \right)$ sampled instances are sufficient to ensure that with a probability of at least $1 - \delta$, for all $h \in \mathcal{H}$, the difference between the empirical average and the expectation over an unknown disrtibution $\mathcal{D}$ is at most ϵ.

2.2 Problem formulation

We describe path-finding instances, GBFS/A* algorithm, and performance measures considered in this paper.

Path-finding instances. We consider solving randomly generated path-finding instances repetitively. Let $x = (V, E, \{w_e\}_{e \in E}, s, t)$ be a path-finding instance, where (V, E) is a simple directed graph with n vertices, $\{w_e\}_{e \in E}$ is a set of non-negative edge weights (sometimes called costs), $s \in V$ is a start vertex, and $t \in V$ is a goal vertex. We let Π be a class of possible instances. Each instance $x \in \Pi$ is drawn from an unknown distribution $\mathcal{D}$ over Π. We impose the following assumption on Π.

Assumption 1. *For all $x \in \Pi$, the vertex set V and the goal node t are identical, and there always exists at least one directed path from $s \neq t$ to t, i.e., every instance $x \in \Pi$ is feasible.*

Fixing V is necessary for evaluating the pseudo-dimension in terms of $n = |V|$. Note that we can deal with the case where some instances in Π are defined on vertex subsets $V' \subseteq V$ by removing edges adjacent to $V \setminus V'$. The feasibility assumption is needed to ensure that GBFS/A* always returns a solution, and $s \neq t$ simply rules out the trivial case where the empty set is optimal. In Appendix B, we discuss how to extend our results to the case where t can change depending on instances.

Algorithm description. We sketch algorithmic procedures that are common to both GBFS and A* (see Algorithms 1 and 2 for details, respectively). Let $A_{\boldsymbol{\rho}}$ be a GBFS/A* algorithm, which is parameterized by heuristic function values $\boldsymbol{\rho} \in \mathbb{R}^n$. Given an instance $x \in \Pi$, $A_{\boldsymbol{\rho}}$ starts from s and iteratively builds a set of candidate paths. These paths are maintained by `OPEN` and `CLOSED` lists, together with pointers $\mathrm{p}(\cdot)$ to parent vertices. The `OPEN` list contains vertices to be explored, and the `CLOSED` list consists of vertices that have been explored. In each iteration, we select a vertex v from `OPEN`, expand v, and move v from `OPEN` to `CLOSED`.

Heuristic function values $\boldsymbol{\rho}$ are used when selecting vertices. For each $v \in V$, the corresponding entry in $\boldsymbol{\rho}$, denoted by ρ_v, represents an estimated shortest-path distance from v to t. (Although heuristic function values are usually denoted by $h(v)$, we here use ρ_v for convenience.) In each iteration, we select a vertex with the smallest *score*, which is defined based on $\boldsymbol{\rho}$ as detailed later. We impose the following assumption on the vertex selection step.

Assumption 2. *Define an arbitrary strict toral order on V; for example, we label elements in V by $v_1, \ldots, v_n$ and define a total order $v_1 < \cdots < v_n$. When selecting a vertex with the smallest score, we break ties, if any, in favor of the smallest vertex with respect to the total order.*

If we allow $A_{\boldsymbol{\rho}}$ to break ties arbitrarily, its behavior becomes too complex to obtain meaningful bounds on the pseudo-dimension. Assumption 2 is a natural rule to exclude such troublesome cases.

Performance measure. Let $A_{\boldsymbol{\rho}}$ be GBFS/A* with parameters $\boldsymbol{\rho} \in \mathbb{R}^n$. We measure performance of $A_{\boldsymbol{\rho}}$ on $x \in \Pi$ with a utility function u. We assume u to satisfy the following condition.

Assumption 3. *Let $H > 0$. A utility function u takes x and a series of all* `OPEN`*,* `CLOSED`*, and* $\mathtt{p}(\cdot)$ *generated during the execution of $A_{\boldsymbol{\rho}}$ on $x \in \Pi$ as input, and returns a scalar value in $[0, H]$.*

We sometimes use $A_{\boldsymbol{\rho}}$ to represent the series of `OPEN` and `CLOSED` lists and pointers generated by $A_{\boldsymbol{\rho}}$. Note that u meeting Assumption 3 can measure various kinds of performance. For example, since the pointers indicate an s–t path returned by $A_{\boldsymbol{\rho}}$, u can represent its cost. Moreover, since the series of `OPEN` and `CLOSED` lists maintain all search states, u can represent the time and space complexity of $A_{\boldsymbol{\rho}}$. We let $u_{\boldsymbol{\rho}} : \Pi \to [0, H]$ denote the utility function that returns the performance of $A_{\boldsymbol{\rho}}$ on any $x \in \Pi$, and define a class of such functions as $\mathcal{U} = \{\, u_{\boldsymbol{\rho}} : \Pi \to [0, H] \mid \boldsymbol{\rho} \in \mathbb{R}^n \,\}$. The upper bound, H, is necessary to obtain sample complexity bounds with Proposition 1. Setting such an upper bound is usual in practice. For example, if u measures the running time, H represents a time-out deadline.

Generalization guarantees on performance. Given the above setting, we want to learn $\hat{\boldsymbol{\rho}}$ values that attain an optimal $\mathbb{E}_{x \sim \mathcal{D}}[u_{\hat{\boldsymbol{\rho}}}(x)]$ value, where available information consists of sampled instances $x_1, \ldots, x_N$ and $u_{\boldsymbol{\rho}}(x_1), \ldots, u_{\boldsymbol{\rho}}(x_N)$ values for any $\boldsymbol{\rho} \in \mathbb{R}^n$. To obtain generalization gaurantees on the performance of $A_{\hat{\boldsymbol{\rho}}}$, we bound $|\frac{1}{N}\sum_{i=1}^{N} u_{\boldsymbol{\rho}}(x_i) - \mathbb{E}_{x \sim \mathcal{D}}[u_{\boldsymbol{\rho}}(x)]|$ uniformly for all $\boldsymbol{\rho} \in \mathbb{R}^n$. Note that the uniform bound offers performance guarantees that are independent of learning procedures, e.g., manual or automated (without being uniform, learned $\hat{\boldsymbol{\rho}}$ may be overfitting sampled instances). As in Proposition 1, to bound the sample complexity of learning $\boldsymbol{\rho}$ values, we need to evaluate the pseudo-dimension of $\mathcal{U}$, denoted by $\mathrm{Pdim}(\mathcal{U})$, which is the main subject of this study.

Remarks on heuristic functions. While we allow heuristic function values $\boldsymbol{\rho}$ to be any point in $\mathbb{R}^n$, the range of heuristic functions may be restricted to some subspace of $\mathbb{R}^n$. Note that our upper bounds are applicable to such situations since restricting the space of possible $\boldsymbol{\rho}$ values does not increase $\mathrm{Pdim}(\mathcal{U})$. Meanwhile, such restriction may be useful for improving the upper bounds on $\mathrm{Pdim}(\mathcal{U})$; exploring this direction is left for future work. Also, our setting cannot deal with heuristic functions that take some instance-dependent features as input. To study such cases, we need more analysis that is specific to heuristic function models, which goes beyond the scope of this paper. Thus, we leave this for future work. Note that our setting still includes important heuristic function models on fixed vertex sets. For example, we can set $\boldsymbol{\rho}$ using learned distances to landmarks [18], or we can let $\boldsymbol{\rho}$ be distances measured on some metric space by learning metric embeddings of vertices [34].

3 Upper bounds on the pseudo-dimension

We present details of GBFS and A* and upper bounds on the pseudo-dimensions of $\mathcal{U}$. In this section, we suppose that vertices in V are labeled by $v_1, \ldots, v_n$ as in Assumption 2.

3.1 Greedy best-first search

Algorithm 1 shows the details of GBFS $A_{\boldsymbol{\rho}}$ with heuristic function values $\boldsymbol{\rho} \in \mathbb{R}^n$. When selecting vertices in Step 3, the scores are determined only by $\boldsymbol{\rho}$. This implies an obvious but important fact.

Lemma 1. *Let $\boldsymbol{\rho}, \boldsymbol{\rho}' \in \mathbb{R}^n$ be a pair of heuristic function values with an identical total order up to ties on their entries, i.e., $\mathbb{I}(\rho_{v_i} \leq \rho_{v_j}) = \mathbb{I}(\rho'_{v_i} \leq \rho'_{v_j})$ for all $i, j \in [n]$ such that $i < j$. Then, we have $u_{\boldsymbol{\rho}}(x) = u_{\boldsymbol{\rho}'}(x)$ for all $x \in \Pi$.*

Algorithm 1 GBFS with heuristic function values $\boldsymbol{\rho}$

1: $\texttt{OPEN} = \{s\}$, $\texttt{CLOSED} = \emptyset$, and $\mathrm{p}(s) = \mathrm{None}$.
2: **while** $\texttt{OPEN}$ is not empty **:**
3: $\quad v \leftarrow \mathrm{argmin}\{\, \rho_{v'} \mid v' \in \texttt{OPEN} \,\}$. ▷ Break ties as in Assumption 2.
4: $\quad$ **for** each child c of v **:**
5: $\quad\quad$ **if** $c = t$ **:**
6: $\quad\quad\quad$ **return** s–t path by tracing pointers $\mathrm{p}(\cdot)$, where $\mathrm{p}(t) = v$.
7: $\quad\quad$ **if** $c \notin \texttt{OPEN} \cup \texttt{CLOSED}$ **:**
8: $\quad\quad\quad$ $\mathrm{p}(c) \leftarrow v$ and $\texttt{OPEN} \leftarrow \texttt{OPEN} \cup \{c\}$.
9: $\quad$ Move v from $\texttt{OPEN}$ to $\texttt{CLOSED}$.

Algorithm 2 A* with heuristic function values $\boldsymbol{\rho}$

1: $\texttt{OPEN} = \{s\}$, $\texttt{CLOSED} = \emptyset$, $\mathrm{p}(s) = \mathrm{None}$, and $g_s = 0$.
2: **while** $\texttt{OPEN}$ is not empty **:**
3: $\quad v \leftarrow \mathrm{argmin}\{\, g_{v'} + \rho_{v'} \mid v' \in \texttt{OPEN} \,\}$. ▷ Break ties as in Assumption 2.
4: $\quad$ **if** $v = t$ **:**
5: $\quad\quad$ **return** s–t path by tracing pointers $\mathrm{p}(\cdot)$.
6: $\quad$ **for** each child c of v **:**
7: $\quad\quad$ $g_{\mathrm{new}} \leftarrow g_v + w_{(v,c)}$.
8: $\quad\quad$ **if** $c \notin \texttt{OPEN} \cup \texttt{CLOSED}$ **:**
9: $\quad\quad\quad$ $g_c \leftarrow g_{\mathrm{new}}$, $\mathrm{p}(c) \leftarrow v$, and $\texttt{OPEN} \leftarrow \texttt{OPEN} \cup \{c\}$.
10: $\quad\quad$ **else if** $c \in \texttt{OPEN}$ and $g_{\mathrm{new}} < g_c$ **:**
11: $\quad\quad\quad$ $g_c \leftarrow g_{\mathrm{new}}$ and $\mathrm{p}(c) \leftarrow v$.
12: $\quad\quad$ **else if** $c \in \texttt{CLOSED}$ and $g_{\mathrm{new}} < g_c$ **:** ▷ Steps 12–14 are for reopening.
13: $\quad\quad\quad$ $g_c \leftarrow g_{\mathrm{new}}$ and $\mathrm{p}(c) \leftarrow v$.
14: $\quad\quad\quad$ Move c from $\texttt{CLOSED}$ to $\texttt{OPEN}$.
15: $\quad$ Move v from $\texttt{OPEN}$ to $\texttt{CLOSED}$.

Proof. For any $x \in \Pi$, if $\boldsymbol{\rho}$ and $\boldsymbol{\rho}'$ have an identical strict total order on their entries, vertices selected in Step 3 are the same in each iteration of $A_{\boldsymbol{\rho}}$ and $A_{\boldsymbol{\rho}'}$. Since this is the only step $\boldsymbol{\rho}$ and $\boldsymbol{\rho}'$ can affect, we have $A_{\boldsymbol{\rho}} = A_{\boldsymbol{\rho}'}$ for all $x \in \Pi$, hence $u_{\boldsymbol{\rho}}(x) = u_{\boldsymbol{\rho}'}(x)$. Moreover, this holds even if $\boldsymbol{\rho}$ and/or $\boldsymbol{\rho}'$ have ties on their entries because of Assumption 2. That is, the total order uniquely determines a vertex selected in Step 3 even in case of ties. Therefore, the statement holds. □

From Lemma 1, the behavior of GBFS is uniquely determined once a total order on $\{\rho_v\}_{v \in V}$ is fixed. Thus, for any $x \in \Pi$, the number of distinct $u_{\boldsymbol{\rho}}(x)$ values is at most $n!$, the number of permutations of $\{\rho_v\}_{v \in V}$. This fact enables us to obtain an $\mathrm{O}(n \lg n)$ upper bound on the pseudo-dimension of $\mathcal{U}$.

Theorem 1. *For GBFS $A_{\boldsymbol{\rho}}$ with parameters $\boldsymbol{\rho} \in \mathbb{R}^n$, it holds that* $\mathrm{Pdim}(\mathcal{U}) = \mathrm{O}(n \lg n)$.

Proof. Lemma 1 implies that we can parition $\mathbb{R}^n$ into $n!$ regions, $\mathcal{P}_1, \mathcal{P}_2, \ldots$, so that for every $\mathcal{P}_i$, any pair of $\boldsymbol{\rho}, \boldsymbol{\rho}' \in \mathcal{P}_i$ satisfies $u_{\boldsymbol{\rho}}(x) = u_{\boldsymbol{\rho}'}(x)$ for all $x \in \Pi$. Note that the construction of the regions, $\mathcal{P}_1, \mathcal{P}_2, \ldots$, does not depend on x. Thus, given any N instances $x_1, \ldots, x_N$, even if $\boldsymbol{\rho}$ moves over whole $\mathbb{R}^n$, the number of distinct tuples of form $(u_{\boldsymbol{\rho}}(x_1), \ldots, u_{\boldsymbol{\rho}}(x_N))$ is at most $n!$. To shatter N instances, $n! \geq 2^N$ must hold. Solving this for the largest N yields $\mathrm{Pdim}(\mathcal{U}) = \mathrm{O}(n \lg n)$. □

3.2 A* search

Algorithm 2 is the details of A*. As with GBFS, $\boldsymbol{\rho}$ only affects the vertex selection step (Step 3). However, unlike GBFS, the scores, $g_v + \rho_v$, involve not only $\boldsymbol{\rho}$ but also $\{g_v\}_{v \in V}$. Each g_v is called a g-cost and maintains a cost of some path from s to v. As in Algorithm 2, when v is expanded and a shorter path to c is found, whose cost is denoted by g_{new}, we update the g_c value. Thus, each g_v always gives an upper bound on the shortest-path distance from s to v. For each $v \in V$, there are at most $\sum_{k=0}^{n-2} k! \leq (n-1)!$ simple paths connecting s to v, and thus g_v can take at most $(n-1)!$ distinct values. We denote the set of those distinct values by $\mathcal{G}_v$, and define $\mathcal{G}_V = \{\, (v, g_v) \mid v \in V, g_v \in \mathcal{G}_v \,\}$ as the set of all pairs of a vertex and its possible g-cost. It holds that $|\mathcal{G}_V| \leq n \times (n-1)! = n!$.

Note that once $x \in \Pi$ is fixed, $\mathcal{G}_v$ for $v \in V$ and $\mathcal{G}_V$ are uniquely determined. To emphasize this fact, we sometimes use notation with references to x: $g_v(x)$, $\mathcal{G}_v(x)$, and $\mathcal{G}_V(x)$. As with the case of GBFS (Lemma 1), we can define a total order on the scores to determine the behavior of A* uniquely.

Lemma 2. *Fix any instance $x \in \Pi$. Let $\boldsymbol{\rho}, \boldsymbol{\rho}' \in \mathbb{R}^n$ be a pair of heuristic function values such that total orders on the sets of all possible scores, $\{\, g_v(x) + \rho_v \mid (v, g_v(x)) \in \mathcal{G}_V(x) \,\}$ and $\{\, g_v(x) + \rho'_v \mid (v, g_v(x)) \in \mathcal{G}_V(x) \,\}$, are identical up to ties. Then, it holds that $u_{\boldsymbol{\rho}}(x) = u_{\boldsymbol{\rho}'}(x)$.*

Proof. If the two sets of scores have an identical strict total order, we select the same vertex in Step 3 in each iteration of $A_{\boldsymbol{\rho}}$ and $A_{\boldsymbol{\rho}'}$. Thus, we have $A_{\boldsymbol{\rho}} = A_{\boldsymbol{\rho}'}$ for any fixed x, implying $u_{\boldsymbol{\rho}}(x) = u_{\boldsymbol{\rho}'}(x)$. We show that this holds even in the presence of ties by using Assumption 2. First, any two scores of the same vertices, $g_v(x) + \rho_v$ and $g'_v(x) + \rho_v$, never have ties since $\mathcal{G}_v$ consists of distinct g-costs. Next, if $g_{v_i}(x) + \rho_{v_i} = g_{v_j}(x) + \rho_{v_j}$ holds for some $i < j$, we always prefer v_i to v_j in Step 3 due to Assumption 2. Therefore, even in the presence of ties, we select a vertex in Step 3 as if the set of scores has a strict total order. Thus, if $\boldsymbol{\rho}$ and $\boldsymbol{\rho}'$ induce the same total order up to ties on the sets of possible scores, it holds that $u_{\boldsymbol{\rho}}(x) = u_{\boldsymbol{\rho}'}(x)$. □

By using Lemma 2, we can obtain an $\mathrm{O}(n^2 \lg n)$ upper bound on the pseudo-dimension of $\mathcal{U}$.

Theorem 2. *For A* $A_{\boldsymbol{\rho}}$ with parameters $\boldsymbol{\rho} \in \mathbb{R}^n$, it holds that $\mathrm{Pdim}(\mathcal{U}) = \mathrm{O}(n^2 \lg n)$.*

Proof. As with the proof of Theorem 1, we partition $\mathbb{R}^n$ into some regions so that in each region, the behavior of A* is unique. Unlike the case of GBFS, boundaries of such regions change over N instances. To deal with this situation, we use a geometric fact: for $m \geq n \geq 1$, m hyperplanes partition $\mathbb{R}^n$ into $\mathrm{O}((\mathrm{e}m)^n)$ regions.[1]

Fix a tuple of any N instances $(x_1, \ldots, x_N)$. We consider hyperplanes in $\mathbb{R}^n$ of form $g_{v_i}(x_k) + \rho_{v_i} = g_{v_j}(x_k) + \rho_{v_j}$ for all $k \in [N]$ and all pairs of $(v_i, g_{v_i}(x_k)), (v_j, g_{v_j}(x_k)) \in \mathcal{G}_V$ such that $i \neq j$. These hyperplanes partition $\mathbb{R}^n$ into some regions, $\mathcal{P}_1, \mathcal{P}_2, \ldots$, so that the following condition holds: for every $\mathcal{P}_i$, any $\boldsymbol{\rho}, \boldsymbol{\rho}' \in \mathcal{P}_i$ have the same total order on $\{\, g_v(x_k) + \rho_v \mid (v, g_v(x_k)) \in \mathcal{G}_V(x) \,\}$ and $\{\, g_v(x_k) + \rho'_v \mid (v, g_v(x_k)) \in \mathcal{G}_V(x_k) \,\}$ up to ties for all $k \in [N]$, which implies $u_{\boldsymbol{\rho}}(x_k) = u_{\boldsymbol{\rho}'}(x_k)$ for all $k \in [N]$ due to Lemma 2. That is, for every $k \in [N]$, if we see $u_{\boldsymbol{\rho}}(x_k)$ as a function of $\boldsymbol{\rho}$, it is piecewise constant where pieces are given by $\mathcal{P}_1, \mathcal{P}_2, \ldots$. Therefore, when $\boldsymbol{\rho}$ moves over whole $\mathbb{R}^n$, the number of distinct tuples of form $(u_{\boldsymbol{\rho}}(x_1), \ldots, u_{\boldsymbol{\rho}}(x_N))$ is at most the number of the pieces. Note that the pieces are generated by partitioning $\mathbb{R}^n$ with $\sum_{k \in [N]} \binom{|\mathcal{G}_V(x_k)|}{2} \leq N \binom{n!}{2}$ hyperplanes, which means there are at most $\mathrm{O}\left(\left(\mathrm{e}N\binom{n!}{2}\right)^n\right)$ pieces. To shatter N instances, $\mathrm{O}\left(\left(\mathrm{e}N\binom{n!}{2}\right)^n\right) \geq 2^N$ is necessary. Solving this for the largest N yields $\mathrm{Pdim}(\mathcal{U}) = \mathrm{O}(n^2 \lg n)$. □

Compared with GBFS, the additional n factor comes from the bound of $(n-1)!$ on $|\mathcal{G}_v|$. This bound may seem too pessimistic, but it is almost tight in some cases, as implied by the following example.

Example 1. Let (V, E) be a complete graph with edges labeled as $\{e_1, \ldots, e_{|E|}\}$. Set each edge weight w_{e_i} to 2^{i-1} for $i \in [|E|]$. Considering the binary representation of the edge weights, the costs of all simple s–v paths are mutually different for $v \in V$, which implies $|\mathcal{G}_v| = \sum_{k=0}^{n-2} k! \geq (n-2)!$.

This example suggests that improving the $\mathrm{O}(n^2 \lg n)$ bound is not straightforward. Under some realistic assumptions, however, we can improve it by deriving smaller upper bounds on $|\mathcal{G}_v|$.

First, if the maximum degree of vertices is always bounded, we can obtain the following bound.

Theorem 3. *Assume that the maximum out-degrees of directed graphs (V, E) of all instances in Π are upper bounded by d. Then, it holds that $\mathrm{Pdim}(\mathcal{U}) = \mathrm{O}(n^2 \lg d)$.*

Proof. Under the assumption on the maximum degree, there are at most $\sum_{k=0}^{n-2} d^k \leq (n-1)d^{n-2}$ simple s–v paths, which implies $|\mathcal{G}_v| \leq (n-1)d^{n-2}$ for every $v \in V$. Therefore, we have $|\mathcal{G}_V| \leq n \times (n-1)d^{n-2}$. Following the proof of Theorem 2, we can obtain an upper bound on $\mathrm{Pdim}(\mathcal{U})$ by solving $\mathrm{O}\left(N^n \binom{n(n-1)d^{n-2}}{2}^n\right) \geq 2^N$ for the largest N, which yields $\mathrm{Pdim}(\mathcal{U}) = \mathrm{O}(n^2 \lg d)$. □

[1] Even if some regions degenerate, from [20, Theorem 28.1.1] and [10, Proposition A2.1], the number of all d-dimensional regions for $d = 0, \ldots, n$ is $\sum_{d=0}^{n} \sum_{i=0}^{d} \binom{n-i}{d-i}\binom{m}{n-i} \leq 2(\mathrm{e}m)^n$. The fact has a close connection to Sauer's lemma [27] (see [17]). In this sense, our analysis is in a similar spirit to the general framework of [6].

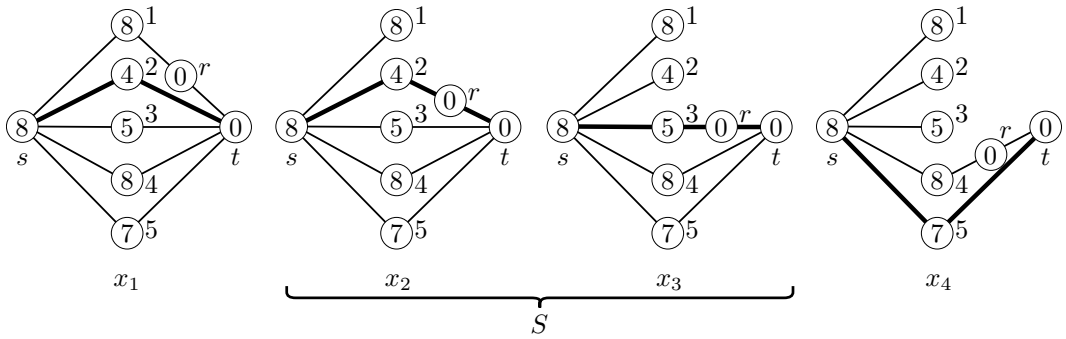

Figure 1: An illustration of the instances $x_1, \ldots, x_{n-4}$ for $n = 8$. Each vertex is labeled by s, r, t, or $i \in [n-3]$, as shown nearby the vertex circles. The values in vertex circles represent $\boldsymbol{\rho}$ that makes $A_{\boldsymbol{\rho}}$ return suboptimal paths to x_2 and x_3, i.e., $S = \{2, 3\}$. The thick edges indicate returned paths.

Second, if edge weights are non-negative integers bounded by ℓ, we can obtain the following bound.

Theorem 4. *Assume that edge weights $\{w_e\}_{e \in E}$ of all instances in Π are non-negative integers bounded by a constant ℓ from above. Then, it holds that* $\mathrm{Pdim}(\mathcal{U}) = \mathrm{O}(n \lg(n\ell))$.

Proof. Under the assumption, every g-cost g_v takes a non-negative integer value at most $n\ell$. Since $\mathcal{G}_v$ consists of distinct g-cost values, $|\mathcal{G}_v| \leq n\ell$ holds, hence $|\mathcal{G}_V| \leq n^2\ell$. Solving $\mathrm{O}\left(N^n \binom{n^2\ell}{2}^n\right) \geq 2^N$ for the largest N, we obtain $\mathrm{Pdim}(\mathcal{U}) = \mathrm{O}(n \lg(n\ell))$. □

Note that if $\ell = \mathrm{O}(\mathrm{poly}(n))$ holds, we have $\mathrm{Pdim}(\mathcal{U}) = \mathrm{O}(n \lg n)$.

On reopening. A* is usually allowed to reopen closed vertices as in Steps 12–14. This, however, causes $\Omega(2^n)$ iterations in general [23], albeit always finite [30]. A popular workaround is to simply remove Steps 12–14, and such A* without reopening has also been extensively studied [31, 28, 12, 13]. Note that our results are applicable to A* both with and without reopening.

4 Lower bounds on the pseudo-dimension

We present lower bounds on the pseudo-dimension for GBFS/A*. We prove the result by constructing $\Omega(n)$ shatterable instances with unweighted graphs. Therefore, the $\mathrm{O}(n \lg n)$ upper bounds for GBFS (Theorem 1) and A* under the edge-weight assumption (Theorem 4) are tight up to a $\lg n$ factor.

Theorem 5. *For GBFS/A* $A_{\boldsymbol{\rho}}$ with parameters $\boldsymbol{\rho} \in \mathbb{R}^n$, it holds that* $\mathrm{Pdim}(\mathcal{U}) = \Omega(n)$.

Proof sketch. We construct a series of $n-4$ instances, $x_1, \ldots, x_{n-4}$, that can be shattered by $\mathcal{U}$, where each $u_{\boldsymbol{\rho}}$ returns the length of an s–t path found by $A_{\boldsymbol{\rho}}$. We label vertices in V by s, r, t, or $i \in [n-3]$. See Figure 1 for an example with $n = 8$. We define $M = V \setminus \{s, r, t\}$. For each x_i ($i \in [n-4]$), we draw edges (s, v) for $v \in M$ and (v, t) for $v \in \{v' \in M \mid v' > i\}$, which constitute optimal s–t paths of length 2. In addition, for each x_i, we draw edges (i, r) and (r, t), where $s \to i \to r \to t$ is the only suboptimal path of length 3. Letting $t_i = 2.5$ for $i \in [n-4]$, we prove that $\mathcal{U}$ can shatter those $n-4$ instances, i.e., $A_{\boldsymbol{\rho}}$ can return suboptimal solutions to any subset of $\{x_1 \ldots, x_{n-4}\}$ by appropriately setting $\boldsymbol{\rho}$.

Let $S \subseteq [n-4]$ indicate a subset of instances, to which we will make $A_{\boldsymbol{\rho}}$ return suboptimal solutions. We show that for any S, we can set $\boldsymbol{\rho}$ so that $A_{\boldsymbol{\rho}}$ returns $s \to i \to r \to t$ to x_i if and only if $i \in S$. We refer to the vertex labeled by $n-3$ as m, which we use to ensure that every instance has an optimal path $s \to m \to t$. We set $\boldsymbol{\rho}$ as follows: $\rho_s = n$ (or an arbitrary value), $\rho_r = \rho_t = 0$, $\rho_i = i + 2$ if $i \in S \cup \{m\}$, and $\rho_i = n$ (or a sufficiently large value) if $i \in [n-4] \setminus S$. If $A_{\boldsymbol{\rho}}$ with this $\boldsymbol{\rho}$ is applied to x_i, it iteratively selects vertices in $S \cup \{m\}$ in increasing order of their labels until a vertex that has a child is selected. Once a vertex with a child is expanded, it ends up returning $s \to i \to r \to t$ if $i \in S$ and $s \to v \to t$ for some $v > i$ if $i \notin S$. We detail this in the full proof presented in the supplementary. (As we will there, both GBFS and A* return the same s–t paths.) □

5 Toward better guarantees on the suboptimality of A*

Given the results in Sections 3 and 4, a major open problem is to close the $\tilde{\mathrm{O}}(n)$ gap[2] between the $\mathrm{O}(n^2 \lg n)$ upper bound and the $\Omega(n)$ lower bound for A* in general cases. This problem seems very complicated, as we will discuss in Section 6. Instead, we here study a particular case where we want to bound the expected suboptimality of A*, which is an important performance measure since learned heuristic values are not always admissible. We show that a general bound obtained from Theorem 2 can be sometimes improved by using a $\boldsymbol{\rho}$-dependent worst-case bound [31].

For any $x \in \Pi$, let $\mathrm{Opt}(x)$ and $\mathrm{Cost}_{\boldsymbol{\rho}}(x)$ be the costs of an optimal solution and an s–t path returned by $A_{\boldsymbol{\rho}}$, respectively, and let $u_{\boldsymbol{\rho}}(x) = \mathrm{Cost}_{\boldsymbol{\rho}}(x) - \mathrm{Opt}(x)$ be the suboptimality. From Theorem 2 and Proposition 1, we can obtain the following high-probability bound on the expected suboptimality:

$$\mathop{\mathbb{E}}_{x \sim \mathcal{D}}[\mathrm{Cost}_{\boldsymbol{\rho}}(x) - \mathrm{Opt}(x)] \leq \frac{1}{N} \sum_{i=1}^{N} (\mathrm{Cost}_{\boldsymbol{\rho}}(x) - \mathrm{Opt}(x)) + \tilde{\mathrm{O}}\left(H \sqrt{\frac{n^2 + \lg \frac{1}{\delta}}{N}} \right). \tag{1}$$

That is, the expected suboptimality can be bounded from above by the empirical suboptimality over the N training instances (an empirical term) plus an $\tilde{\mathrm{O}}(H\sqrt{n^2/N})$ term (a complexity term). While this bound is useful when $N \gg n^2$, we may not have enough training instances in practice. In such cases, the complexity term becomes dominant and prevents us from obtaining meaningful guarantees. In what follows, we present an alternative bound of the form "an empirical term + a complexity term" that can strike a better balance between the two terms when N is not large enough relative to n^2.

To this end, we use the notion of *consistency*. We say $\boldsymbol{\rho}$ is *consistent* if $\rho_v \leq \rho_c + w_{(v,c)}$ holds for all $(v, c) \in E$. If $\boldsymbol{\rho}$ is consistent, A* without reopening returns an optimal solution. Valenzano et al. [31, Theorem 4.6] revealed that for any instance $x \in \Pi$, the suboptimality of A* can be bounded by the inconsistency accumulated along an optimal path (excluding the first edge containing s) as follows:[3]

$$\mathrm{Cost}_{\boldsymbol{\rho}}(x) - \mathrm{Opt}(x) \leq \Delta_{\boldsymbol{\rho}}(x) \coloneqq \sum_{(v,c) \in S^*(x), v \neq s} \max\{\rho_v - \rho_c - w_{(v,c)}, 0\}, \tag{2}$$

where $S^*(x) \subseteq E$ is an optimal solution to x (if there are multiple optimal solutions, we break ties by using the lexicographical order induced from the total order defined in Assumption 2). We call $\Delta_{\boldsymbol{\rho}}(x)$ the inconsistency (of $\boldsymbol{\rho}$ on $S^*(x)$).

Given N instances $x_1, \ldots, x_N$, we can compute the empirical inconsistency, $\frac{1}{N}\sum_{i=1}^{N} \Delta_{\boldsymbol{\rho}}(x_i)$, at the cost of solving the N instances, which we will use as an empirical term. To define the corresponding complexity term, we regard $\Delta_{\boldsymbol{\rho}}(\cdot) : \Pi \to [0, \hat{H}]$ as an inconsistency function parameterized by $\boldsymbol{\rho}$, where we will discuss how large $\hat{H} > 0$ can be later, and we let $\hat{\mathcal{U}} = \{\, \Delta_{\boldsymbol{\rho}} : \Pi \to [0, \hat{H}] \mid \boldsymbol{\rho} \in \mathbb{R}^n \,\}$. The following theorem says that the class $\hat{\mathcal{U}}$ of inconsistency functions has a smaller pseudo-dimension than the class $\mathcal{U}$ of general utility functions.

Theorem 6. *For the class $\hat{\mathcal{U}}$ of inconsistency functions, it holds that* $\mathrm{Pdim}(\hat{\mathcal{U}}) = \mathrm{O}(n \lg n)$.

By using (2), Proposition 1, and Theorem 6, we can obtain the following high-probability bound on the expected suboptimality, whose complexity term has a better dependence on n than that of (1):

$$\mathop{\mathbb{E}}_{x \sim \mathcal{D}}[\mathrm{Cost}_{\boldsymbol{\rho}}(x) - \mathrm{Opt}(x)] \leq \mathop{\mathbb{E}}_{x \sim \mathcal{D}}[\Delta_{\boldsymbol{\rho}}(x)] \leq \frac{1}{N} \sum_{i=1}^{N} \Delta_{\boldsymbol{\rho}}(x_i) + \tilde{\mathrm{O}}\left(\hat{H} \sqrt{\frac{n + \lg \frac{1}{\delta}}{N}} \right).$$

This bound is uniform for all $\boldsymbol{\rho} \in \mathbb{R}^n$, as with other bounds discussed so far. Thus, the bound holds even if we choose $\boldsymbol{\rho}$ to minimize the empirical inconsistency. Note that the empirical inconsistency is convex in $\boldsymbol{\rho}$ since $\Delta_{\boldsymbol{\rho}}(x_i)$ consists of a maximum of a linear function of $\boldsymbol{\rho}$ and zero, hence easier to minimize than the raw empirical suboptimality in practice (and suitable for a recent online-convex-optimization framework [22]).

Before proving Theorem 6, we present a typical example to show that the inconsistency is not too large relative to the suboptimality.

[2] We use $\tilde{\mathrm{O}}$ and $\tilde{\Omega}$ to hide logarithmic factors of n and N for simplicity.

[3] The original theorem in [31] is applicable only to the case where A* does not reopen vertices and $\rho_t = 0$. These restrictions are unnecessary as detailed in Appendix C, and thus our result holds regardless of reopening.

Example 2. Suppose that every edge weight w_e is bounded to $[0, \ell]$, which ensures that the suboptimality $u_{\boldsymbol{\rho}}$ is at most $H = \ell(n-1)$ for any $\boldsymbol{\rho} \in \mathbb{R}^n$. For simplicity, we consider the following natural way to compute $\boldsymbol{\rho}$ values: compute an estimate $\hat{w}_e \in [0, \ell]$ of w_e for each $e \in E$ and let ρ_v be the cost of a shortest v–t path with respect to $\{\hat{w}_e\}_{e \in E}$. Then, $\boldsymbol{\rho}$ enjoys the consistency with respect to $\{\hat{w}_e\}_{e \in E}$, i.e., $\rho_v \leq \rho_c + \hat{w}_{(v,c)}$ for every $(v, c) \in E$. Therefore, it holds that

$$\Delta_{\boldsymbol{\rho}}(x) \leq \sum_{(v,c) \in S^*(x), v \neq s} \max\{\rho_v - \rho_c - w_{(v,c)}, 0\} \leq \sum_{(v,c) \in S^*(x), v \neq s} \left|\hat{w}_{(v,c)} - w_{(v,c)}\right|.$$

Hence $\Delta_{\boldsymbol{\rho}}$ is at most $\hat{H} = \ell(n-2)$, implying that $\Delta_{\boldsymbol{\rho}}$ does not largely exceed the suboptimality. If empirically accurate estimates $\hat{w}_e$ for $e \in S^*(x)$ are available, the inconsistency becomes small.

We prove Theorem 6 by using the analysis framework by Balcan et al. [6]. Roughly speaking, if we fix $x \in \Pi$ and see $\Delta_{\boldsymbol{\rho}}(x)$ as a function of $\boldsymbol{\rho}$, it exhibits a piecewise linear structure. By using this structure and [6, Theorem 3.3], we obtain Theorem 6. We below present a proof sketch due to the space limitation. Please see the supplementary for the complete proof.

Proof sketch. Fixing any instance $x \in \Pi$, we define the so-called *dual class* of $\hat{\mathcal{U}}$ as $\hat{\mathcal{U}}^* \subseteq [0, \hat{H}]^{\mathbb{R}^n}$, where each $\Delta_x^* \in \hat{\mathcal{U}}^*$ takes $\boldsymbol{\rho} \in \mathbb{R}^n$ as input and returns $\Delta_{\boldsymbol{\rho}}(x) \in [0, \hat{H}]$. Once x is fixed, $S^*(x)$ is unique due to the tie-breaking. Thus, $\Delta_x^*(\boldsymbol{\rho}) = \sum_{(v,c) \in S^*(x), v \neq s} \max\{\rho_v - \rho_c - w_{(v,c)}, 0\}$ is uniquely defined as a piecewise linear function of $\boldsymbol{\rho}$, where pieces are specified by $|E| = \mathrm{O}(n^2)$ halfspace boundary functions: $b^{(v,c)}(\boldsymbol{\rho}) = \mathbb{I}\big(\rho_v - \rho_c - w_{(v,c)} > 0\big)$ for all edges $(v, c) \in E$. Let $\mathcal{F}$ and $\mathcal{B}$ be the classes of linear and halfspace functions of $\boldsymbol{\rho}$, respectively. From [6, Theorem 3.3],

$$\mathrm{Pdim}(\hat{\mathcal{U}}) = \mathrm{O}((\mathrm{Pdim}(\mathcal{F}^*) + \mathrm{VCdim}(\mathcal{B}^*)) \lg(\mathrm{Pdim}(\mathcal{F}^*) + \mathrm{VCdim}(\mathcal{B}^*)) + \mathrm{VCdim}(\mathcal{B}^*) \lg K)$$

holds, where $\mathcal{F}^*$ and $\mathcal{B}^*$ are the dual classes of $\mathcal{F}$ and $\mathcal{B}$, respectively, and K is the number of boundary functions required to specify all the pieces. As mentioned above, $K = \mathrm{O}(n^2)$ holds. Furthermore, we can regard $\mathcal{F}^*$ and $\mathcal{B}^*$ as classes of linear and halfspace functions on $\mathbb{R}^{n+1}$, respectively, hence $\mathrm{Pdim}(\mathcal{F}^*) = \mathrm{VCdim}(\mathcal{B}^*) = n+1$. Thus, we obtain $\mathrm{Pdim}(\hat{\mathcal{U}}) = \mathrm{O}(n \lg n)$. $\square$

6 Conclusion and discussion

We have studied the sample complexity of learning heuristic functions for GBFS and A* on graphs with a fixed vertex set of size n. The crucial step is to evaluate the pseudo-dimension of the class of utility functions. For GBFS and A*, we have proved that the pseudo-dimensions are upper bounded by $\mathrm{O}(n \lg n)$ and $\mathrm{O}(n^2 \lg n)$, respectively. As for A*, we have shown that the bound can be improved to $\mathrm{O}(n^2 \lg d)$ if every vertex has a degree of at most d and to $\mathrm{O}(n \lg n)$ if edge weights are bounded integers. We have also presented the $\Omega(n)$ lower bounds for GBFS and A*, implying that our bounds for GBFS and A* under the integer-weight condition are tight up to a $\lg n$ factor. Finally, we have discussed bounds on the suboptimality of A* and obtained a guarantee with a better complexity term by evaluating the pseudo-dimension of the class of inconsistency functions.

As mentioned in Section 5, an open problem is to close the gap between the upper and lower bounds regarding A* for general cases. This, however, does not seem straightforward. We here discuss the reasons for the difficulty. As regards the upper bound, the bottleneck is the bound of $(n-2)!$ on $|\mathcal{G}_v|$, but this cannot be improved in general, as shown in Example 1. Taking this into account, the direct use of Sauer's lemma would not yield better upper bounds. Thus, we need to use some special structures of the hyperplanes (e.g., each has only two variables), which would require more complicated analysis. As for the lower bound, the construction of the $\Omega(n)$ instances in Theorem 5 relies on the fact that $\boldsymbol{\rho}$ has an n degree of freedom. In addition, Theorem 4 implies that we need to consider instances with non-integer edge weights (or exponentially large integer weights in n) to obtain a lower bound of $\tilde{\Omega}(n^2)$. Considering the above, we would have to use involved techniques for constructing a set of $\tilde{\Omega}(n^2)$ shatterable instances. Another interesting future direction is to improve upper bounds on the pseudo-dimension by restricting heuristic functions to some classes, as mentioned in Section 2. We finally discuss limitations of our work. As mentioned in Section 2, we require every instance to be defined on (subsets of) a fixed vertex set. Also, our work does not cover the case where heuristic function values can change depending on instance-dependent features. Studying how to overcome these limitations would also constitute interesting future work.

References

[1] F. Agostinelli, S. Mc Aleer, A. Shmakov, R. Fox, M. Valtorta, B. Srivastava, and P. Baldi. Obtaining approximately admissible heuristic functions through deep reinforcement learning and A* search. In *Bridging the Gap Between AI Planning and Reinforcement Learning Workshop at ICAPS 2021*, 2021.

[2] M. Anthony and P. L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999.

[3] M.-F. Balcan. Data-driven algorithm design. In *Beyond the Worst-Case Analysis of Algorithms*, pages 626–645. Cambridge University Press, 2021.

[4] M.-F. Balcan, T. Dick, T. Sandholm, and E. Vitercik. Learning to branch. In *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*, volume 80, pages 344–353. PMLR, 2018.

[5] M.-F. Balcan, T. Sandholm, and E. Vitercik. Refined bounds for algorithm configuration: The knife-edge of dual class approximability. In *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)*, volume 119, pages 580–590. PMLR, 2020.

[6] M.-F. Balcan, D. DeBlasio, T. Dick, C. Kingsford, T. Sandholm, and E. Vitercik. How much data is sufficient to learn high-performing algorithms? Generalization guarantees for data-driven algorithm design. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC 2021)*, pages 919–932. ACM, 2021.

[7] M.-F. Balcan, S. Prasad, and T. Sandholm. Sample complexity of tree search configuration: Cutting planes and beyond. In *Advances in Neural Information Processing Systems 34 (NeurIPS 2021)*, pages 4015–4027. Curran Associates, Inc., 2021.

[8] M.-F. Balcan, S. Prasad, T. Sandholm, and E. Vitercik. Improved learning bounds for branch-and-cut. *arXiv:2111.11207*, 2021.

[9] M. Bhardwaj, S. Choudhury, and S. Scherer. Learning heuristic search via imitation. In *Proceedings of the 1st Annual Conference on Robot Learning (CoRL 2017)*, volume 78, pages 271–280. PMLR, 2017.

[10] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Learnability and the Vapnik–Chervonenkis dimension. *J. ACM*, 36(4):929–965, 1989.

[11] B. Chen, C. Li, H. Dai, and L. Song. Retro*: Learning retrosynthetic planning with neural guided A* search. In *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)*, volume 119, pages 1608–1616. PMLR, 2020.

[12] J. Chen and N. R. Sturtevant. Conditions for avoiding node re-expansions in bounded suboptimal search. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI 2019)*, pages 1220–1226. IJCAI Organization, 2019.

[13] J. Chen and N. R. Sturtevant. Necessary and sufficient conditions for avoiding reopenings in best first suboptimal search with general bounding functions. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI 2021)*, volume 35, pages 3688–3696. AAAI Press, 2021.

[14] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numer. Math.*, 1(1):269–271, 1959.

[15] J. E. Doran, D. Michie, and D. G. Kendall. Experiments with the graph traverser program. *Proc. R. Soc. Lond. A Math. Phys. Sci.*, 294(1437):235–259, 1966.

[16] T. Eden, P. Indyk, and H. Xu. Embeddings and labeling schemes for A*. In *Proceedings of the 13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*, volume 215, pages 62:1–62:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

[17] B. Gärtner and E. Welzl. Vapnik–Chervonenkis dimension and (pseudo-)hyperplane arrangements. *Discrete Comput. Geom.*, 12(4):399–432, 1994.

[18] A. V. Goldberg and C. Harrelson. Computing the shortest path: A* search meets graph theory. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2005)*, pages 156–165. SIAM, 2005.

[19] R. Gupta and T. Roughgarden. A PAC approach to application-specific algorithm selection. *SIAM J. Comput.*, pages 123–134, 2017.

[20] D. Halperin and M. Sharir. Arrangements. In *Handbook of Discrete and Computational Geometry*, pages 723–762. CRC press, third edition, 2017.

[21] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Man Cybern. Syst.*, 4(2):100–107, 1968.

[22] M. Khodak, M.-F. Balcan, A. Talwalkar, and S. Vassilvitskii. Learning predictions for algorithms with predictions. *arXiv:2202.09312*, 2022.

[23] A. Martelli. On the complexity of admissible search algorithms. *Artif. Intell.*, 8(1):1–13, 1977.

[24] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. MIT Press, second edition, 2018.

[25] M. Pándy, R. Ying, G. Corso, P. Velickovic, J. Leskovec, and P. Liò. Learning graph search heuristics. In *Physical Reasoning and Inductive Biases for the Real World Workshop at NeurIPS 2021*, 2021.

[26] D. Pollard. *Convergence of Stochastic Processes*. Springer, first edition, 1984.

[27] N. Sauer. On the density of families of sets. *J. Combin. Theory Ser. A*, 13(1):145–147, 1972.

[28] V. Sepetnitsky, A. Felner, and R. Stern. Repair policies for not reopening nodes in different search settings. In *Proceedings of the 9th International Symposium on Combinatorial Search (SoCS 2016)*, volume 7, pages 81–88. AAAI Press, 2016.

[29] T. Takahashi, H. Sun, D. Tian, and Y. Wang. Learning heuristic functions for mobile robot path planning using deep neural networks. In *Proceedings of the 29th International Conference on Automated Planning and Scheduling (ICAPS 2019)*, volume 29, pages 764–772. AAAI Press, 2019.

[30] R. Valenzano and F. Xie. On the completeness of best-first search variants that use random exploration. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI 2016)*, volume 30. AAAI Press, 2016.

[31] R. Valenzano, N. Sturtevant, and J. Schaeffer. Worst-case solution quality analysis when not re-expanding nodes in best-first search. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI 2014)*, volume 28, pages 885–892. AAAI Press, 2014.

[32] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory Probab. Appl.*, 16(2):264–280, 1971.

[33] R. Yonetani, T. Taniai, M. Barekatain, M. Nishimura, and A. Kanezaki. Path planning using neural A* search. In *Proceedings of the 38th International Conference on Machine Learning (ICML 2021)*, volume 139, pages 12029–12039. PMLR, 2021.

[34] J. You, R. Ying, and J. Leskovec. Position-aware graph neural networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML 2019)*, volume 97, pages 7134–7143. PMLR, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Section 1.1 for details of our main claims.
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] This is a theoretical paper and no negative societal impacts are expected.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 2.2.

(b) Did you include complete proofs of all theoretical results? [Yes] Some complete proofs are shown in the supplementary due to the page limitation.

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [N/A]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]