Planning Bias: Planning as a Source of Sampling Bias

Alison Paredes[1,2], Jeremy Frank[1], J. Benton[1], and Christian Muise[2]

[1]Intelligent Systems Division, NASA Ames Research Center
[2]Department of Computing, Queen's University

Abstract

Many data-driven planning methods are trained on data generated by planners. It is well known that many statistical learning methods are sensitive to sampling bias, and yet there has been little or no attention to planning as a sampling method and its role in introducing sampling bias into planner-generated training data. Recently, it has been demonstrated that A*, in the presence of problems with variable heuristic error, prefers some solutions over other equally cost-optimal solutions. But, as we discuss in this paper, mitigation may not be as simple as resolving arbitrary tie-breaking by sampling from ties uniformly at random. In this paper, we formalize an intuition of *planning bias*. We focus on problems which output a single solution. Diverse planning only complicates the problem by generalizing it to bias in the set of sets; we show how it is subject to bias in the single solution. We make some useful observations about deterministic algorithms in contrast to non-deterministic algorithms. We explain how information entropy may be a good way to measure planning bias, and discuss some issues in evaluating practical approaches to measurement. We address the intuition that uniform random tie-breaking should mitigate bias; and sketch a novel approach to constructing an appropriate random distribution for duplicate detection during forward search for unbiased A*. Finally, we suggest directions for future work.

Introduction

Though the topic is not frequently studied, the presence of biases in plans has profound implications for a number of applications of automated planners. For instance, planner-generated data may be used as input to a neural network to iteratively improve a heuristic, such as in (Arfaee, Zilles, and Holte 2010; Ernandes and Gori 2004). Model acquisition uses planner-generated examples for automatic model acquisition (Callanan et al. 2022; Mordoch et al. 2023) or for assisted model elicitation (Feblowitz et al. 2021; Sohrabi et al. 2016). It has also been hypothesized that plan validation may also benefit from using more planning data, for more robust validation via simulation (Benton et al. 2018).

Though its generally well-understood that planning models may include biases (as the famous George Box quote goes, "All models are wrong, some are useful"), a much less understood source of biases are those introduced by the actual process of plan synthesis.

Recently, we have demonstrated that A* prefers the solution with the poorest heuristic estimates along the solution path among other equally cost-optimal solutions, when heuristic error varies (Paredes 2023). We note, too, a consequence of this is that ODrM*, which uses A*, to generate examples of cooperative behavior (Sartoretti et al. 2019), may then bias learning toward some of the riskiest examples of cooperative behavior.

In this paper, we invite the planning community to look at planning as a source of sampling bias. Toward this end, we introduce the idea of *planning bias*. We believe that up until now data-driven planning and scheduling has assumed that planners are somehow a neutral source of examples. For some planning technology, that may indeed be probably the case. A brute force search may be a good sampling methodology for robust data-driven planning, assuming a planning model that we believe in. Faster methods, however, like A*, for example, are not necessarily unbiased sampling technology (Paredes 2023). We propose that robust data-driven planning and scheduling will require a much better understanding of *planning bias*. In the worst case, there may be an unavoidable trade-off between planning speed and planning bias. The topic has applications outside of planning such as in scheduling, constraint satisfaction, combinatorial design problems, and reinforcement learning (Trinh et al. 2024) and (Sartoretti et al. 2019). Below we attempt to formalize our intuition of planning bias, identify some major challenges, and present some initial results to stimulate discussion.

Preliminaries

We will use satisficing planning to develop some key ideas toward an intuition of *planning bias*.

Definition 1 (A Planning Task)**.** Let $M = \langle V, O \rangle$ be a set of variables and planning operators. Let I be an initial state consisting of a complete assignment to the variables. Let G be a set of goal states implicitly defined by an incomplete assignment to the variables. An operator is a function which transforms one state to another. Let $\Pi = \langle M, I, G \rangle$ be a planning problem instance. A solution s is a sequence of operators which transforms the initial state to a goal state.

A critical idea in the proceeding discussion is that of the solution space of the particular planning instance.

Definition 2 (Solution Space of a Planning Task)**.** Let $S(\Pi)$

be the space of solutions to Π. A plan is a solution s if and only if it is ∈ S(Π). (To be clear, a sequence of operators which does not successfully transform the initial state into a goal state may never be ∈ S(Π)). Thus S(Π) contains all solutions s to Π and no invalid plans. This will be an important distinction for our purposes.

We can use this understanding of the solution space of a task to develop an idea of the probability of a solution, which we would like to be able to compare between algorithms.

Definition 3 (Probability Distribution Over the Solution Set)**.** Let $P_\Pi(s)$ be the probability of solution s to Π. Let $P_{A_i,\Pi}(s)$ be the probability that algorithm A_i generates solution s to Π. Let $P_{A_q i,\Pi}$ be the probability distribution of overall solutions s ∈ $S(\pi)$.

An algorithm that outputs any valid solution with equal probability may be described precisely as:

Definition 4 (Uniform Distribution Over the Solution Set)**.** Let $P_{A_0,\Pi}(s) = \frac{1}{|S(\Pi)|}$, for all s. Then $P_{A_0,\Pi}$ is a uniform distribution over a planning task's solution space.

We can now measure the distance $\Delta(P_{A_i,\Pi}, P_{A_j,\Pi}) \Rightarrow \mathbb{R}_{\geq 0}$ between the uniform distribution over solutions, and the distribution of solutions produced by planning algorithms. Different distance measures will give different answers, but this is not important for the development of our definition of bias in the next section.

A Formal Definition of Planning Bias

Our intuitive definition of algorithm bias is: if an algorithm does not produce valid solutions to a planning problem instance with uniform probability, it is biased. We formalize this as follows:

Definition 5 (Planning Bias)**.** Let A_p be a planning algorithm. $P_{A_0,\Pi}$ is a uniform distribution over a planning task's solution space (Definition 4). Then A_p is biased if $\Delta(P_{A_0,\Pi}, P_{A_p,\Pi}) > 0$.

Some useful results will help us more precisely describe our intuition of what planning bias looks like.

Theorem 1. Let A_i be a deterministic planning algorithm. Then A_i is biased if and only if $|S(\Pi)| > 1$.

Proof. A deterministic algorithm A_i returns the same solution s_i given the same inputs; equivalently, $P_{A_i,\Pi}(s_i) = 1$ for some s ∈ S(Π) and $P_{A_i,\Pi}(s_j) = 0$, where $s_i \neq s_j$. Therefore, $|S(\Pi)| > 1 \Rightarrow \Delta(P_{A_0,\Pi}, P_{A_i,\Pi}) > 0$. Otherwise, $\Delta(P_{A_0,\Pi}, P_{A_i,\Pi}) = 0$. □

Theorem 2. Let A_e be the (EXPTIME)[1] algorithm that enumerates all solutions in S(Π) and samples from among them uniform at random. Then A_e is unbiased.

Proof. Trivial. □

[1]The algorithm is EXPTIME and not EXPSPACE because we can count the number of solutions without saving them, decide which should be returned, then generate them again and count solutions, returning the proper one when it is generated.

Theorem 3. Let A_i be a deterministic algorithm and A_j be a non-deterministic algorithm. $P_{A_0,\Pi}$ is a uniform distribution over a planning task's solution space (Definition 4). Then $\Delta(P_{A_0,\Pi}, P_{A_i,\Pi}) \geq \Delta(P_{A_0,\Pi}, P_{A_j,\Pi})$.

Proof. Trivial because any non-deterministic algorithm must return at least one plan. □

Deterministic algorithms, however, must often make some arbitrary choice. In practice, we might resolve these with a random seed to force determinism. In some cases, we hope through some judicious use of tie-breakers we can gain efficiency (Asai and Fukunaga 2016; Corrêa, Pereira, and Ritt 2018; Heusner, Keller, and Helmert 2018; Ferber et al. 2022). Non-deterministic algorithms are a non-trivial class to consider and includes many superficially deterministic algorithms.

Of non-deterministic algorithms, we should ask under what conditions they perform more like A_e (Theorem 2). We propose there are several challenges to answering this question. Some of them we go into more detail about below. Challenges in practically computing bias without knowledge of the set of all solutions. How we might take advantage of existing ideas from information entropy. Such as addressing the intuition that all non-deterministic algorithms are unbiased at least in the long run.

What Planning Bias is Not That is not to say that planner-generated data cannot be biased due to biases in the model. A domain model that omits right turns will never output a solution containing right turns. The above discussion assumes a model we believe in. In other words, we are confident that the model is as accurate as it is ever going to get. Given that "all models are wrong but some are useful (George Box)," we assume a maximally useful model. Model elicitation is an important but different challenge.

It is tempting to think the proper approach to mitigating planning bias is to design the appropriate diversity metric. Diverse planning takes as input a planning task, a diversity metric, and an integer k, and returns a size-k set of solutions which conforms to the diversity metric. Diverse planning, however, only complicates the problem. We can easily generalize our definition of bias to diverse planning, where the solution to a diverse planning problem is not a single plan but a set of plans. The set of all solutions to the diverse planning problem is a set of sets. A diverse planning algorithm that will never output a particular set, a solution $s_{D,i}$, within the set of sets, all solutions to the diverse planning task, $S(\Pi_D)$, is biased.

Some Thoughts on Estimating Bias

The previous section provides some notation which helps us think about bias but does not provide much guidance on how to practically identify it. We may be able to assume that most deterministic algorithms are biased. Per Theorem 1, a deterministic algorithm is biased if $|S(\Pi)| > 1$. If we knew the size of the solution space then we could easily determine the existence of planning bias. But we do need to know exactly the size of the solution space. We may be able to approximate the size solution space from features of the domain.

For example, if a minimal cost solution is composed of more than one communitive actions then there must be at least one other minimal cost solution. Many more restrictions on the domain model may be necessary to ensure the size of the solution set is at most one. It is hard to imagine planning could be very difficult under these kinds of restrictions. Fortunately, it is not necessary that we know exactly the size of the solution space in the case of a deterministic algorithm. Unfortunately, we are more interested in non-deterministic algorithms.

Of non-deterministic algorithms it may be difficult to be sure of the existence of planning bias, as we have defined it. One way to identify if a non-deterministic algorithm is biased if we could analytically prove that algorithm A can never output some $s \in S(\Pi)$. If the probability of some solution via A is 0, then $\Delta(P_{A_0,\Pi}, P_{A,\Pi}) \neq 0$. In the absence of such a proof, experimental techniques may still be unsatisfying. We would need to find a solution that algorithm A will not produce. To experimentally show that some solution can never be output is undecidable.

Even if a planner can find all plans then we still cannot be confident it is not biased without additional work. Our definition of planning bias claims that any divergence from a uniform distribution over $S(\Pi)$ indicates bias, i.e. $\Delta(P_{A_i,\Pi}, P_{A_0,\Pi}) > 0$. A solution which is over-represented should indicate the existence of planning bias. Proving this empirically with any confidence will require appropriate tools.

Our intuition is to look to the entropy of $P_{A,\Pi}$. Recall that $P_{A_0,\Pi}$ represents the uniform distribution over $S(\Pi)$. Entropy is maximized when the probability distribution is uniform. Recall entropy is a measure of the uncertainty of the random variable. Let X be a discrete random variable with alphabet $\mathcal{X}$ and probability mass function $p(x) = Pr(X = x), x \in \mathcal{X}$.

$$H(X) = -\sum_{x \in \mathbf{X}} p(x) log(p(x)) \tag{1}$$

For our purposes, we claim that entropy would be maximized when A_i is unbiased, i.e. $\Delta(P_{A_0,\Pi}, P_{A_i,\Pi}) = 0$.

Definition 6 (Entropy of a planning algorithm)**.** Let S be a discrete random variable with alphabet $S(\Pi)$, the solution space of a particular planning task, and probability mass function $p(s) = Pr(S = s), s \in S(\Pi)$.

$$H(S) = -\sum_{s \in S(\Pi)} p(s) log(p(s)) \tag{2}$$

The theory of entropy provides some other useful properties for describing planning bias. Entropy is a convex function of the probability mass and monotonically decreases as the probability of an event increases, allowing us to compare whether an algorithm is more or less biased. When entropy equals zero, there is no uncertainty in the random variable; therefore, A_i is deterministic. Additionally, relative entropy between two random variables permits some level of comparison among algorithms. Relative entropy is zero if and only if two probability masses are equivalent, e.g. $P_{A_i,\Pi} = P_{A_j,\Pi}$.

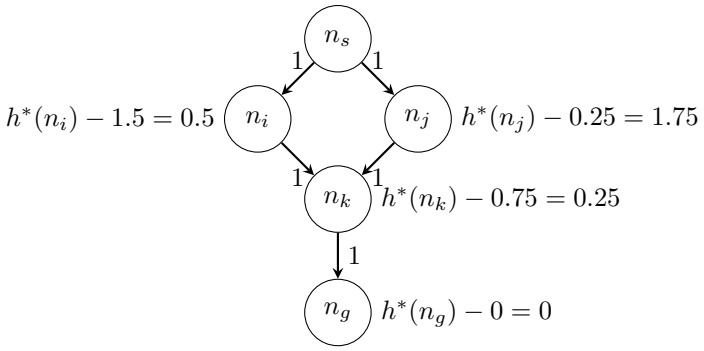


Figure 1: Adapted from (Paredes, 2023). The subgraph G_s of the state space under a admissible heuristic h. In this example, the true cost-to-go $h^*(n_i)$ and $h^*(n_j)$ are identical, however estimates $h(n_i) < h(n_j)$. The effect on solution recovery is that A* will always prefer the path through n_i. Randomized tie-breaking has no effect.

Practically, however, it can be difficult to efficiently compute a good estimate of entropy, when the true distribution is unknown. There are many different ways to estimate entropy. Some of the most well-known include the naive estimator.

Definition 7 (A naive estimator as described in (Valiant and Valiant 2017))**.** Let S be a multiset representing a sample of solutions of size n. A sample could be generated by any non-deterministic planning algorithm by running it n times. Let F represent the fingerprint of S. A fingerprint is a vector whose ith component, F_i, is the number of elements that occur exactly i times in S. Then the entropy of $P_{A_i,\Pi}$ can be estimated from a sample of solutions as follows:

$$H^{naive}(F) := -\sum_i n_i \frac{i}{n} |log \frac{i}{n}| \tag{3}$$

One of the limitations of the naive estimator is in the number of samples it requires to get a good estimate. Recent work has shown that entropy can be estimated without *a priori* assumptions about the target distribution in only a sublinear-size sample, $O(k/logk)$, where k is the number of distinct elements in the target distribution (Valiant and Valiant 2017).

Randomized Approaches to Existing Algorithms

We could replace an arbitrary choice in any deterministic algorithm with a random choice. But it may not be obvious what distribution to use in a randomization strategy that will ensure bias-free planning. Consider an example.

It was recently discovered that A* as originally defined with fifo duplicate detection in (Hart, Nilsson, and Raphael 1968) may some equally cost-optimal solutions over others under realistic conditions (Paredes 2023). We show below that we cannot just replace fifo duplicate detection with uniform random duplicate selection strategy; and we sketch an explanation of why.

In conventional A* (Hart, Nilsson, and Raphael 1968), search proceeds by expanding the node from open with the

minimum f value until a goal node is reached. When the heuristic is admissible, the path it finds is guaranteed to be an optimal cost path. When the heuristic is also consistent, a closed list can improve time-efficiency. In this case we can be sure during search that the first time search expands a state that it has found a minimal cost path to that state. First-in-first-out (fifo) duplicate detection preserves only the first minimal cost path encountered during search.

The problem with fifo duplicate selection is that the minimal cost path that is preserved for solution recovery is influenced by the character of heuristic error in the problem. The result is a bias toward paths with poorer heuristic estimates.

Consider the search space illustrated in Figure 1. There are two minimal length solutions from the n_s to n_g, passing through n_k. Let solution s_i be the path through n_k via n_i, and solution s_j be the path through n_k via n_j. In this example, although the true cost-to-go $h^*(n_i)$ and $h^*(n_j)$ are identical, estimates $h(n_i) \neq h(n_j)$. This could happen for many reasons. Consider the effect on solution recovery.

Although states n_i and n_j both fall along minimal cost paths to n_g, A* will always prefer the path through n_i for solution recovery. Expansion will proceed as follows: $< n_s, n_i, n_k, n_j, n_g >$. Because A* will expand every node on open with f $<$ f*, it will expand n_j and generate n_k a second time. But n_k will be discarded this time. A minimal cost path to n_k has already been found. And the solution s_j will never be recoverable.

One might think that this situation could easily be resolved with a random duplicate selection. The challenge here is in what distribution to use and how to compute it. Consider the naive approach: During duplicate detection, choose between the first node to close and the newly generated duplicate uniformly at random. More formally, let the incumbent be the first node closed. During duplicate detection, choose between the incumbent and the newly generated node with probability 0.5 and 0.5. Unfortunately, this approach selects the most recently generated duplicate.

The sequence of decisions, to keep the incumbent node or replace it with the newly discovered duplicate, forms a Markov chain M. Each node in M represents a decision to keep the incumbent or replace it with the newly discovered duplicate. A directed edge from node A and B in M with label p denotes the probability of transition from A to B, the probability that the the incumbent survives or the candidate replaces it. The probability that the first node put on closed surviving to termination is the joint probability of surviving all previous decision points.

Our first intuition is to adjust the weight of the edges between decision points in proportion to the number of duplicates discovered so far. For example, when the first duplicate is discovered, the number of duplicates discovered so far is 2, therefore the probability that the incumbent survives or the candidate survives are weighted equally 0.5 and 0.5. When the second duplicate is discovered, the number of duplicates discovered so far is 3, and the probability that the incumbent survives is weighted 0.66 and 0.33.

But the this does not quite work because all duplicates will not be discovered before termination. Each decision point effectively prunes the subtree rooted at the losing duplicate search node. This saves time in conventional A*. These efficiency gains comes the expense of bias-free search. It may take some careful bookkeeping to compute the number of duplicates for each decision point for the appropriate distribution to preserve the theoretical efficiency gains of conventional duplicate detection for A*.

Randomization even with the appropriate probability distribution may not be able to mitigate bias everywhere. Distance-guided heuristics for example in may be intrinsically biased toward shorter plans, their efficiency gains coming from an implicit restriction on the planning problem, making them more difficult to correct.

Discussion

Initial results in Paredes (2023) suggest that planning bias could be a bigger problem in generalizing methods using planner-generated data to real-world problems than we had realized. We have tried to more precisely to define planning bias in order to make some important distinctions from existing work in diverse planning and to explain our intuition that there may be a useful relationship between planning bias and information entropy for practically computing planning bias. Further, we have revisited tie-breaking from this novel perspective. We believe that there is much work to be done to identify and mitigate planning bias for robust data-driven planning and scheduling, and actively solicit contributions and insights from the broader planning community to address.

Directions for Future Work

Though we have considered the theory of bias in planning algorithms, we also wish to examine how to measure bias in practice. To do this, we can generate a sample set of plans from a "time-limited planner" and determine whether the measure of entropy in the sample set of plans matches the true entropy found in all possible plans. We can assume we will forbid finding the same plan twice in such a scenario. From this setup, we would like to investigate how increasing the amount of time given to the planner (allowing us to find more plans) changes entropies for varying domains. We'd further like to use this empirical study to find how entropy estimate and bias estimates compare. We hope to show whether entropy works as a surrogate measure for bias in practice.

For instance, for both top-k and top-quality planning problems, the practicality of finding timely optimal solutions becomes a central concern. We expect to investigate a the landscape of optimization algorithms further, especially those tackling problems in higher complexity classes (e.g., EXPTIME), to determine whether they exhibit bias both theoretically and empirically. For such complex problems, practitioners often use approximation algorithms and we hope to also examine how approximation approaches may express their (likely) biases.

References

Arfaee, S. J.; Zilles, S.; and Holte, R. C. 2010. Bootstrap Learning of Heuristic Functions. In Felner, A.; and Sturte-

vant, N. R., eds., *Proceedings of the Third Annual Symposium on Combinatorial Search, SOCS 2010, Stone Mountain, Atlanta, Georgia, USA, July 8-10, 2010*, 52–60. AAAI Press.

Asai, M.; and Fukunaga, A. S. 2016. Tiebreaking Strategies for A* Search: How to Explore the Final Frontier. In Schuurmans, D.; and Wellman, M. P., eds., *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 673–679.

Benton, J.; Smith, D. E.; Kaneshige, J.; Keely, L.; and Stucky, T. 2018. CHAP-E: A Plan Execution Assistant for Pilots. In de Weerdt, M.; Koenig, S.; Röger, G.; and Spaan, M. T. J., eds., *Proceedings of the Twenty-Eighth International Conference on Automated Planning and Scheduling, ICAPS 2018, Delft, The Netherlands, June 24-29, 2018*, 303–311. AAAI Press.

Callanan, E.; Venezia, R. D.; Armstrong, V.; Paredes, A.; Chakrborti, T.; and Muise, C. 2022. MACQ: A Holistic View of Model Acquisition Techniques. In *Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)*.

Corrêa, A. B.; Pereira, A. G.; and Ritt, M. 2018. Analyzing Tie-Breaking Strategies for the A* Algorithm. In Lang, J., ed., *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, 4715–4721. ijcai.org.

Ernandes, M.; and Gori, M. 2004. Likely-Admissible and Sub-Symbolic Heuristics. In de Mántaras, R. L.; and Saitta, L., eds., *Proceedings of the 16th Eureopean Conference on Artificial Intelligence, ECAI'2004, including Prestigious Applicants of Intelligent Systems, PAIS 2004, Valencia, Spain, August 22-27, 2004*, 613–617. IOS Press.

Feblowitz, M.; Hassanzadeh, O.; Katz, M.; Sohrabi, S.; Srinivas, K.; and Udrea, O. 2021. IBM Scenario Planning Advisor: A Neuro-Symbolic ERM Solution. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, 16032–16034. AAAI Press.

Ferber, P.; Cohen, L.; Seipp, J.; and Keller, T. 2022. Learning and Exploiting Progress States in Greedy Best-First Search. In Raedt, L. D., ed., *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, 4740–4746. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2): 100–107.

Heusner, M.; Keller, T.; and Helmert, M. 2018. Best-Case and Worst-Case Behavior of Greedy Best-First Search. In Lang, J., ed., *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, 1463–1470. ijcai.org.

Mordoch, A.; Stern, R.; Scala, E.; and Juba, B. 2023. Safe Learning of PDDL Domains with Conditional Effects. In *ICAPS'23 Workshop on Reliable Data-Driven Planning and Scheduling (RDDPS), Prague, Czech Republic, July 10, 2023*.

Paredes, A. 2023. Structural Bias in Heuristic Search (Student Abstract). In Barták, R.; Ruml, W.; and Salzman, O., eds., *Proceedings of the Sixteenth International Symposium on Combinatorial Search, SOCS 2023, Prague, Czech Republic, July 14-16, 2023*, 196–197. AAAI Press.

Sartoretti, G.; Kerr, J.; Shi, Y.; Wagner, G.; Kumar, T. K. S.; Koenig, S.; and Choset, H. 2019. PRIMAL: Pathfinding via Reinforcement and Imitation Multi-Agent Learning. *IEEE Robotics Autom. Lett.*, 4(3): 2378–2385.

Sohrabi, S.; Udrea, O.; Riabov, A. V.; and Hassanzadeh, O. 2016. Interactive Planning-Based Hypothesis Generation with LTS++. In Kambhampati, S., ed., *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, 4268–4269. IJCAI/AAAI Press.

Trinh, T. H.; Wu, Y.; Le, Q. V.; He, H.; and Luong, T. 2024. Solving olympiad geometry without human demonstrations. *Nat.*, 625(7995): 476–482.

Valiant, G.; and Valiant, P. 2017. Estimating the Unseen: Improved Estimators for Entropy and Other Properties. *J. ACM*, 64(6): 37:1–37:41.