An Improved Exponential-Time Approximation Algorithm for Fully-Alternating Games Against Nature

Andrew Drucker
Computer Science Dept.
University of Chicago
Chicago, IL, USA
Email: adrucker1@cs.uchicago.edu

***Abstract*—“Games against Nature” [1] are two-player games of perfect information, in which one player’s moves are made randomly (here, uniformly); the final payoff to the non-random player is given by some bounded-value function of the move history. Estimating the value of such games under optimal play, and computing near-optimal strategies, is an important goal in the study of decision-making under uncertainty, and has seen significant research in AI and allied areas [2], with only experimental evaluation of most algorithms’ performance. The problem’s PSPACE-completeness does not rule out nontrivial algorithms. Improved algorithms with theoretical guarantees are known in various cases where the payoff function F has special structure, and Littman, Majercik, and Pitassi [3] give a sampling-based improved algorithm for general F, for turn-orders which restrict the number of non-random player strategies.**

We study the case of general F for which the players strictly alternate with binary moves—for which the approach of [3] does not improve over brute force. We give a randomized algorithm to approximate the value of such games under optimal play, and to execute near-optimal strategies. Our algorithm, while exponential-time, achieves exponential savings over brute-force, and certifies a lower bound on the game value with exponentially small additive expected error. (On the downside, the constant of improvement in the base of the runtime exponent is tiny, and the algorithm uses exponential space.)

Our algorithm is recursive, and bootstraps a “base case” algorithm for fixed-size inputs. The method of recursive composition used, the specific base-case guarantees needed, and the steps to establish these guarantees are interesting and, we feel, likely to find uses beyond the present work.

***Keywords*-decision-making under uncertainty; games against Nature; interactive proofs; approximation algorithms**

I. Introduction

Games against Nature, as described by Papadimitriou [1], are a natural and general formulation of a ubiquitous problem: *decision-making under persistent uncertainty*, in scenarios where our knowledge of the world is continually *evolving* through observation. This process is modeled in [1] as a game in which a maximizing, “strategic” player takes turns with a randomly-playing player, Nature; the strategic player’s moves are described by strings $w^1, w^2, \ldots, w^k$ and Nature’s random moves are described by strings $r^1, \ldots, r^k$, which we will assume are chosen uniformly. These moves alternate, and each w^i is chosen with perfect knowledge of the preceding moves $w^1, r^1, \ldots, w^{i-1}, r^{i-1}$. The *payoff* to the strategic player is then determined by some function F as a numerical value, $F(w^1, r^1, \ldots, w^k, r^k) \in [0, 1]$. The function F is known to the strategic player.

In this work, we will focus on the case of “fully-alternating, binary” moves where each w_i and r_i is chosen from $\{0, 1\}$. However, these games are also of interest in more general setting where each w_i, r_i are bitstrings of some predetermined lengths, possibly depending on the specification of the game F and on the value i. This framework can represent a wide variety of practical scenarios for decision-making. It is thus natural to study the following computational problems: given as input a description of F,

1) Compute the (exact or approximate) expected payoff under *optimal* play, i.e. the game’s *value* to the maximizing player—given by $val(F) =$
$$\mathrm{Max}_{w^1}\, \mathbf{E}_{r^1} \ldots \mathrm{Max}_{w^k}\, \mathbf{E}_{r^k} \left[F(w^1, r^1, \ldots, w^k, r^k)\right] . \tag{1}$$
2) Implement an optimal or near-optimal *strategy* choosing next moves w^i in response to observed previous moves $w^1, r^1, \ldots, w^{i-1}, r^{i-1}$.

These problems are usually considered in cases where the description of F is *succinct*. Assume for now that F is specified by a description of a Boolean circuit C_F computing F, a circuit of size polynomially bounded in the total bitlength of $w^1, r^1, \ldots, w^k, r^k$.

A. Related work in AI and other areas

Many probabilistic reasoning and planning tasks are considered in AI, and in allied areas including decision and control theory, constraint programming, and operations research (see [2] for some interdisciplinary overview). One popular framework for this study is that of *Markov Decision Processes* [4], [5]—an intensively-studied model in which a strategic player’s and Nature’s moves affect a system state as described by a state automaton, and payoffs are delivered incrementally, determined by the current state. Various algorithmic results are known, for state spaces that are either small or in some way well-structured, see e.g., [6], [7], [8];

however, these models may fail to succinctly capture tasks in some application areas. Other existing work on more general games against Nature, e.g., in [3], [9], [10], [11], [12], often studies a fairly expressive class of functions F such as CNF formulas or more general Boolean formulas, giving mostly experimental assessments of algorithm performance; these works have adapted concepts and techniques from classical SAT solvers. (When the move-structure and payoff function of the game F are indicated by a Boolean formula quantified by "Exists" and "For-Random" quantifiers, the problem is known as *stochastic satisfiability* [1], [3].)

Decision-making under persistent uncertainty is also studied in a research area known as *competitive analysis of online algorithms* (see, e.g., [13]), in which a decision-maker views an input string x one bit (or one "item") at a time and must make decisions at multiple stages. The game being played is usually completely *known* in advance, rather than given as input; instead, the input x plays a role analogous to Nature's moves, but is no longer assumed to come from a known distribution. Instead researchers seek play-strategies for particular games or classes of games that minimize a *regret* measure, comparing the algorithm's performance with the best performance obtainable with foreknowledge of the input x. This can be a useful benchmark in applications, especially when a realistic distribution on Nature's behavior is not available. By contrast, in the current paper's framework we assume a simple distributional model for Nature's play, but regard the game's *description* as a variable input that must be inspected and reasoned about.

B. Complexity classification, algorithms, and reductions

Even in their approximate versions, problems 1-2 above for Eq. (1) are $PSPACE$-complete in the fully-alternating, binary case (where each w^i, r^i are a single bit). This follows from a beautiful line of work showing the power of *interactive proofs* for decision problems [14], [15], [16], [17], [18]. A public-coin interactive proof system can be considered a family of games against Nature—games designed to achieve epistemic goals through interaction. These proof systems are specified by a polynomial-time Verifier: an efficient algorithm that simultaneously generates Nature's random moves, serves as a referee for the game, and learns something by playing the game with an optimal strategic player.

This $PSPACE$-completeness result can be contrasted with the "bounded-alternation" case where the bitstrings w^i, r^i are individually large but $k = O(1)$. In this case the approximation problem is known to lie in AM [14]. However, while the fully-alternating case is "hardest" when viewed through the lens of complexity classes, such alternations may also *enable* algorithmic improvements. This phenomenon appears in the related setting of 2-player, zero-sum games with an *adversarial* opponent, where simple (randomized) pruning techniques give improved exponential-time algorithms for perfect play. These techniques yield exponential savings over brute force for binary fully-alternating games [19], [20] and form part of the bedrock of game-playing AI. Algorithmic improvements for more general move-bitlengths (or quantifier structure, in the framework of quantified Boolean formulas) against adversarial opponents were given by Santhanam and Williams [21].

The prospect of such improvements for games against Nature was highlighted by Littman, Majercik, and Pitassi [3]. The authors gave a sampling-based evaluation approach, called SampleEvalSSAT, and showed that, for games against Nature in which the number of distinct strategic-player policies is relatively small (due to the structure of the players' move-lengths), the algorithm improves over brute force [3, Sec. 2.3]. They also gave some general search algorithms (building on [22], [23]) with empirical performance studies, with followup work appearing in e.g. [10], [11].

The expressive power of interactive proofs has also been used in other ways to explore the "fine-grained complexity" of various algorithmic problems. For example, Williams [24] shows, among related results, that 3-round interactive proofs can yield exponential savings for evaluating quantified Boolean formulas (with runtime $2^{2n/3+o(n)}$, faster than the randomized algorithm of [19]), as well as exponential-time MA proof systems improving on best known runtimes for various NP problems, casting an interesting perspective on the study of their exponential-time complexity.

Chen et al. [26], building on work of Abboud and Rubinstein [27], use the Arthur-Merlin communication protocols for space-bounded computation of Aaronson and Wigderson [28], along with several additional reductions, to show conditional *hardness* results for several versions of the Longest Common Subsequence (LCS) problem for pairs of strings. This is a polynomial-time solvable problem whose complexity has seen intensive study (see, e.g., [29]). An important tool from [27], used also in [26], is an approximation-preserving reduction from a "Tropical Tensor Similarity" problem, involving alternating Max and $\mathbf{E}$ quantifiers, to LCS. This problem contains some specific structure tailored to a communication setting in the authors' work, but the core part of the reduction applies directly to games against Nature as in Eq. (1): If $F(r, w)$ is Boolean-valued on n total input bits then the reduction produces two output strings u, v each of length $N = 2^n$, each with symbols over an alphabet Σ of size $\Theta(N)$. The LCS of the pair is, in the fully-alternating case, of length $\sqrt{N} \cdot val(F)$.

The overall reduction is not sublinear-time, but can be observed to have efficient local-computability properties. Still, we are not aware of any algorithm for LCS which, in combination with the reduction of [27], would yield a nontrivial speedup for computing values of games against Nature. In particular these would need to have sublinear query complexity, and current fastest approximation algorithms for large alphabet-size ([30], see also [29]) incur linear runtime and queries while returning solutions which, in cases where the LCS is of size $\leq N^{1-\Omega(1)}$, may be a multiplicative $N^{-\Omega(1)}$ factor smaller than optimal in expectation.

A version of the reduction of [27] can also be given for the Longest Increasing Subsequence (LIS) problem, guided by the combination of [27] and a remark in [29, p. 1123] on a connection between LIS and LCS. On input a Boolean-valued game against Nature F as above, not all-zero, the output is a list of $N = 2^n$ integers, each of bitlength $O(n)$, whose LIS is, in the fully-alternating case, of length $val(F) \cdot \sqrt{N}$. See the full version for details. Now, some nontrivial query-efficient approximation algorithms for LIS are known—see [31], [29] and references—but their guarantees are not interesting when the LIS is as small as $\sqrt{N}$; the algorithm of [31] gives fast, nontrivial multiplicative approximations only for LIS size $N^{1-o(1)}$, and the algorithm of [29] on input with LIS of size $N^{1-\varepsilon}$ finds a subsequence of size $\Omega(N^{1-3\varepsilon})$. The portion of this latter algorithm designed to handle sufficiently-small LIS size ($< N^{19/20}$) employs a simple subsampling idea; translating to the setting of fully-alternating games against Nature, and taken flexibly, yields an algorithm nonadaptively querying a random p fraction of queries in expectation (for chosen value $p \in (0, 1)$) and achieving an expected certified lower bound of $p \cdot val(F)$ on the game value. Such an algorithm and its analysis may also be directly verified without the LIS reduction. In this work we will give an $N^{1-\Omega(1)}$-query algorithm for approximating Eq. (1), whose approximation quality is generally far better, achieving expected lower bound of $(1 - o(1)) \cdot val(F)$ provided $val(F) > N^{-o(1)} = 2^{-o(n)}$.

In complexity and cryptography, many interesting *transformations* of interactive proofs are known, and these can usually be applied to general games against Nature as well, changing the structure of games while holding their value nearly fixed. Some such transformations are superficially promising as algorithmic tools for the tasks 1-2 above, but there is typically a "catch" obstructing their direct use. For example, Leshkowitz [32] has shown that general interactive proofs using $r(n)$ bits of randomness can have their round complexity reduced to $O(r(n)/\log n)$. But considered as a transformation on games, the blowup in bitlength of individual moves by the strategic player is apparently too large to be of help.

C. Our result

In this work, we show that indeed, high alternation in games against Nature can be powerfully exploited. We focus on the case of *fully alternating, binary* games against Nature, in which the players strictly alternate making 0/1-valued moves $(w_1, r_1, w_2, r_2, \ldots, w_{n/2}, r_{n/2})$, and the payoff to the non-random player is given by a general, $[0, 1]$-valued function F, provided in black-box or white-box form. We show:

Theorem 1 (Main, informal): For some absolute constant $\delta > 0$, there is a randomized algorithm to approximate the value of fully-alternating binary games against Nature under optimal play, which makes $2^{(1-\delta)n}$ queries to the input function F, and produces a lower bound $\hat{v}$ on the game value v which satisfies $\mathbf{E}[v - \hat{v}] \leq \exp(-\Omega(n))$.

The algorithm explicitly witnesses its lower bounds $\hat{v}$ by exhibiting output values of F on particular sets of inputs that imply the lower bound. From these witnesses, a full description of a player strategy can be produced, or a next move given, also in time $2^{(1-\delta)n}$.

While we assume a specific move-structure on the games we study, we believe our algorithm and analysis contain powerful, flexible ideas that will have broader impact.

The boundedness of F is essential here; if its values can be huge, then approximating $val(F)$ to any reasonable additive error is as hard as needle-in-haystack search, which for (non-quantum) black-box algorithms admits no asymptotic speedup. Similarly, we cannot expect a speedup as above with *multiplicative* approximation guarantees (outputting an estimate $\hat{v}$ where $\hat{v} = \Theta(v)$ with high probability, say), even if F is bounded.

A downside to our algorithm is that the value $\delta > 0$ we obtain is tiny. We make no effort here to optimize it, and do not opine whether a "respectable" value can be achieved by something close to the current approach. Also, the algorithm given uses exponential space. Reducing space usage is a natural goal for future work.

For the fully-alternating case we study, the SampleEvalSSAT algorithm of [3] yields no significant improvement over brute force, let alone one as strong as Theorem 1. We explain this point in detail in the full version (with observations similar to ones in [3]).

Theorem 1 can be compared to a simple (and polynomial-space) recursive algorithm of Snir [19], for which best constants in the recurrence-based analysis were obtained by Saks and Wigderson [20]. This algorithm computes the value of fully alternating, n-move, 2-player games against an adversarial opponent (and with $\{0, 1\}$-valued payoff function F) using $O(2^{\alpha n})$ queries, with $\alpha = \log_2((1 + \sqrt{33})/4) \approx .753$. The constant α here was shown to be *optimal* for black-box randomized algorithms (by [20] for "Las Vegas" zero-error algorithms, and by Santha [33] for bounded-error ones).

In contrast, essentially the only black-box lower bound we know for our problem is a $\Theta(2^{n/2})$ bound for distinguishing values $[v = 0]$ from $[v = 1]$; such a lower bound can be proved easily by considering the problem of distinguishing the all-zero F from one of the functions $F_z(w, r) = \mathbf{1}[w = z]$, ranging over all possible $z \in \{0, 1\}^{n/2}$. This lower-bound example, in which the random variables are irrelevant, basically relies again on the difficulty of black-box needle-in-haystack search.

Similarly, in the white-box setting, and allowing reasonably expressive classes of functions F, we cannot expect an algorithm of runtime $2^{(.5-\varepsilon)n}$ unless the Strong Exponential Time Hypothesis [25] fails; but this also leaves us quite uncertain of the true exponential complexity of the problem. We note that our algorithm, like that of [19], has little direct relation to known improved exponential-time algorithms for, say, Satisfiability of CNF formulas, notably those of [34], [35], [36], which cleverly exploit specific white-box structure in the input. (In the cited works, bounded clause-width is exploited. Unlike our problem, NP Satisfiability problems have no useful "black-box structure" per se for classical algorithms to exploit, since the corresponding unstructured query problem—computing the OR of N bits—requires $\Omega(N)$ queries for classical randomized algorithms.)

Thus, we still do not know whether fully-alternating binary games against Nature are *easier, or harder,* to approximately evaluate than their adversarial counterparts![1] In view of Theorem 1, however, we at least know that a speedup qualitatively similar to that of [19], [20] is possible for the games we study. We believe this is a significant finding for the study of decision-making under certainty.

D. Our techniques

Our proof is fairly long and in this extended abstract we can only outline our techniques; see the full version for details.

For binary, fully alternating games against Nature lasting n moves, the task of computing the value of a game specified by its payoff function F can be viewed equivalently as computing the value of a particular *real arithmetic formula*, alternating between layers of "Max" and "Average" gates. (Each gate is of fanin two, with the formula a full binary tree of depth $n = 2k$.) The output Max gate corresponds to the initial move by the strategic, maximizing player; its two input Average gates correspond to the next, random move by Nature, and so on. This alternating "Max/Average" formula, which we denote MA_k, is given input x equal to the *payoff function F* (with appropriate indexing), and outputs the value of the game associated with F.[2] In the body of our work, and in what follows, we will adopt this formula-evaluation perspective. MA_k is a *monotone* real formula, with inputs assumed to come from $[0, 1]$, so any subset of revealed input values naturally certifies a lower bound on the output value, by direct evaluation of the formula (replacing unseen values with 0). We use this basic idea throughout, and in particular, will use it to determine the final estimate output by our algorithm, which will therefore be a lower bound on the true value.

1) All-queries algorithms and recursive composition: To describe our algorithm for estimating values $\mathsf{MA}_k(x)$ with queries to x, we begin at a high level. While our goal is an algorithm making *few* queries, almost all of our work focuses on the design and study of *"all-queries"* algorithms, which probe every input-coordinate in some order, and whose partial views of x certify progressively better lower bounds on the "true" value $\mathsf{MA}_k(x)$. Our final goal is to design such algorithms which reach "good" lower bounds so quickly that we can simply halt the algorithm early. However, to bootstrap effectively toward this goal, we will actually need approximation guarantees that are more informative about the entire course of the algorithm's execution. Thus, we consider all-queries algorithms with associated *epoch markers* dividing up their execution, and we wish to establish approximation guarantees at the end of each epoch. For now, one may think of these guarantees as bounds, for all possible inputs x, on the expected gap $\mathbf{E}[v - \hat{v}^j]$ between the true value $v = \mathsf{MA}_k(x)$, and the lower bound $\hat{v}^j \leq v$ certified by the partial view after the j^{th} epoch. (This idea will need to modified later.) We note too that the epoch markers will not be evenly spaced, but will be determined by j and k.

For a given, fixed k, an all-queries algorithm $A_k(x)$ for $\mathsf{MA}_k(x)$ can naturally serve as a *subroutine* in the evaluation of $\mathsf{MA}_{k'}$-formulas on larger inputs x', for $k' > k$, by an algorithm $A'_{k'}$ which applies A to each of the *subformulas* of height $2k$ and aggregates the results in some way. (To get strong results from the recursive use of this idea, we will want $k' = k + O(1)$.)

Now let $x \in [0, 1]^{4^{k'}}$ denote the input to $A'_{k'}$ and let the height-$2k$ subtrees be indexed by $i \in [N] = [4^{\ell}]$, where $\ell = k' - k$. Let $x^i \in [0, 1]^{4^{\ell}}$ be the part of x input to the i^{th} such subtree.

The all-queries property of $A_k(x^i)$ helps $A'_{k'}(x)$ to "squeeze all the juice" from each x^i, and to avoid accumulating losses at higher levels of a recursive evaluation scheme. But queries are expensive and making all queries to one x^i before making any to another is risky, since a given x^i may not contribute much (or at all) to the value of $\mathsf{MA}_{k'}(x)$. Thus, our $A'_{k'}(x)$ will hedge its bets and instead maintain *parallel simulations* of each $A_k(x^i)$. Its goal will be to intelligently *allocate* successive queries between different subroutine calls based on what it has seen so far. (These parallel simulations, multiplied recursively, will cause our final algorithm to use exponential space, which is a drawback to the approach.)

Since epochs are a central unit of analysis for our query

[1]There is at least one sense in which playing against a random opponent is "easier"—namely, any strategy deployed against a random opponent has value at least as large as when played against an optimal adversarial opponent. But we are concerned with *taking full advantage* of the opponent's random strategy, in games whose payoff against an optimal opponent might be very poor. Also, the improved algorithm of [19] can be applied to $\{0, 1\}$-payoff games against Nature to determine whether their value is 1 or less than 1, but it does not approximate games against Natures' values or yield near-optimal play in cases when the game's value is less than 1.

[2]This follows from Eq. (1); we have $val(F) = \text{Max}\{\text{Avg}(val(F_{0,0}, F_{0,1}) \;,\; val(F_{1,0}, F_{1,1})\}$, where $F_{b,b'}$ is the restricted game in which $(w_1, r_1) = (b, b')$, and this expands further to give the full arithmetic formula.

algorithms, the outer algorithm $A'_{k'}(x)$ will as its basic action always choose to *advance one sub-input x^i's subroutine $A_k(x^i)$, by one entire epoch*. Thus, the epochs we define also structure our recursive scheme itself. This scheme allows the outer algorithm to prioritize more "promising"-looking inputs x^i—as we will do, by a subtle and context-sensitive criterion. However, this prioritization is done conservatively and does not lead to runaway disparities in the number of queries allocated to the different x^i. In fact, the outer algorithm proceeds in "phases" where, after the end of each j^{th} phase, every $A_k(x^i)$ has advanced by j epochs—equalizing the query counts between each x^i. Moreover, in every phase, only a *single* adaptive decision is made by the outer algorithm about which subroutine to advance next! It is immediately after this adaptively-chosen query block that the outer algorithm hopes to enjoy a nontrivial approximation guarantee for its certified lower bound on $\mathsf{MA}_{k'}(x)$. Thus it is this moment, midway through the j^{th} *phase* of $A'_{k'}(x)$, which we select to mark the end of its j^{th} *epoch*. After a final end-marker is added, our scheme endows the execution of $A'_{k'}(x)$ with $q+1$ epochs, if each $A_k(x^i)$ has q epochs.

The question then becomes how to ensure good approximation guarantees at the new epoch-markers for $A'_{k'}(x)$. As the j^{th} marker for $A'_{k'}$ is placed within a transition between the $(j-1)^{st}$ and j^{th} epochs for the subroutines $A_k(x^i)$, it is natural to imagine that the new guarantee will be obtained as some kind of *weighted average* between the guarantees for those two epochs on the subroutines. And such a guarantee can certainly be attained, even non-adaptively. For example, if $A'_{k'}(x)$ simply randomly selects some .75 fraction of the values i to advance the execution of $A_k(x^i)$ through its j^{th} epoch (and we end the j^{th} epoch of $A'_{k'}(x)$ at this point), then it is not hard to show that $A'_{k'}$ will obey an expected approximation-error bound of $\leq .25\varepsilon_{j-1} + .75\varepsilon_j$, where ε_j is an expected-error bound assumed to hold for A_k after j epochs.

The strength of this new error bound for $A'_{k'}$ is, unfortunately, counterbalanced by the query cost of advancing a .75 fraction of the subroutines $A_k(x^i)$ by one epoch, and the recursive composition of this scheme does *not* lead to good algorithms. But a tantalizing prospect appears: if we could achieve the same error bound as above, while advancing only a .74 fraction of the subroutines, then this approach would indeed achieve our main goal! (To show this, we relate the recurrences defining the epoch markers and error bounds to *tail probabilities* of Bernoulli sums, and use Chernoff bounds to identify an epoch after which the error bound and the fraction of queries made are both small.) And this is essentially how we proceed—with two caveats. First, the actual averaging constants involved are different, and the useful numerical gap is far smaller than .01. The main cause of this is that our outer algorithm $A'_{k'}$ for $\mathsf{MA}_{k'}$ will need to work with a very large value $\ell = k' - k$ (but still $\ell = O(1)$) to make a single, smart adaptive choice, diluting its strength.

Second, to make the above plan succeed, we will actually need to replace error bounds $\mathbf{E}[v - \hat{v}^j] \leq \varepsilon_j$ associated with epoch-markers, with *exponential-moment bounds* of the form

$$\mathbf{E}[\exp(s(v - \hat{v}^j))] \leq e^{s\varepsilon_j} \ ,$$

for some $s > 0$. In fact we will need such bounds with particular s-values that are specific to j, and sufficiently large compared to the gap $\varepsilon_j - \varepsilon_{j-1}$ (the product of the two should exceed a large constant). These types of bounds imply powerful well-behavedness of the error when s is sufficiently large; and despite their form, these bounds are tractable to work with due to a basic independence property maintained between our subroutines.

Furthermore, the "averaging behavior" of the formula MA_ℓ will help the execution of $A'_{k'}(x)$ to obey exponential-moment bounds at its epoch-markers with s-values significantly *larger* than those holding for the subroutines $A_k(x^i)$. This boost is important because the minimum values among the error gaps $\varepsilon_j - \varepsilon_{j-1}$ also become *smaller* at higher levels of our recursive scheme.

In our whole approach it appears unavoidable to contend with such error gaps that are arbitrarily small compared to 4^ℓ (the number of inputs to our fixed-sized reference formula MA_ℓ)—even if our final goal was a more modest additive .1-approximation to the value of games against Nature. Thus it is somewhat surprising that our approach works at all, and in hindsight, the use of exponential-moment bounds with s-values large compared to the gaps is well-motivated as a countermeasure to this challenge.

There is an inherent tension in our use of exponential-moment bounds: while such bounds are stronger and can be more useful when s is large, it is only when s is somewhat *small* that these bounds can be effectively "aggregated" over different outcomes of a random variable to yield new and useful bounds.[3] We use two methods to cope with this. First, we use Jensen's inequality to convert our bounds from higher to lower s-values at appropriate points in our analysis.

Second, we use care in our timing of the adaptive choice of query block made by the outer algorithm $A'_{k'}(x)$. We do so after advancing not a .75 fraction, but a $1 - \Theta(N^{-1/2})$ fraction, of the subroutines $A_k(x^i)$ by one epoch, where $N = 4^\ell$. Actually, it seems merely convenient in several respects that this fraction be large (intuitively, it means we have learned a great deal about the j^{th}-epoch increments on the executions of $A_k(x^i)$ for the various i, which helps us make a good adaptive choice). But it is apparently critical that the fraction not be *too* close to 1, because we cannot allow the minimum separation $\varepsilon_j - \varepsilon_{j-1}$ to decay too quickly with the recursion depth. This is because we are only able to grow the value s in our exponential-moment bounds by

[3]A bit more concretely: a weighted average such as $pe^{sa} + (1-p)e^{sb}$, with $a > b$, is sufficiently close to $e^{s(pa+(1-p)b)}$ for our purposes, provided $s(a-b)$ is somewhat small.

(at most) about a $\sqrt{N}$ factor per layer of recursion—which is roughly because MA_ℓ is only $N^{-.5}$-Lipschitz. (This is best seen by an alternative description of MA_ℓ that will be mentioned in the next section.)

We believe our use of exponential-moment bounds is not just a technical detail, but actually helps to illuminate the behavior of our algorithm in generating progressively more-concentrated estimates at higher stages of recursion, and to explain how such concentration helps it succeed. It may also provide useful guidance for future work in related contexts. (While exponential-moment bounds are widely used to prove concentration properties for the analysis of randomized algorithms, and of algorithms on random inputs [37], we are not aware of previous applications with strong similarity to our work.)

2) Steepeners: We now look more closely into the behavior of the "outer" algorithm $A'_{k'}(x)$ above and how it makes its crucial adaptive choice in each j^{th} epoch of its operation. Recall that $A'_{k'}(x)$ maintains parallel simulations of $A_k(x^i)$, an all-queries algorithm for MA_k, on each sub-input x^i to a height-$2k$ subtree. The queries of $A_k(x^i)$ during its first $(j-1)$ epochs together yield a certified lower bound on $\mathsf{MA}_k(x^i)$; call this value $X_i \in [0,1]$, and let $Y_i \geq X_i$ be the corresponding lower bound after j epochs. Our outer algorithm uses fresh randomness in each simulation, so the pair (X_i, Y_i) is independent of all other such pairs. Moreover, letting $z_i = \mathsf{MA}_k(x^i) \geq Y_i$, each of $(z_i - X_i)$ and $(z_i - Y_i)$ obey an exponential-moment bound by assumption.

We call triples (X, Y, z) of vectors of random variables obeying the above properties *ensembles*. To guide $A'_{k'}$, we design a special type of all-queries algorithm for $\mathsf{MA}_\ell = \mathsf{MA}_{k'-k}$ (for our large, but fixed $\ell = O(1)$) that enjoys a performance guarantee for *all* ensembles for which the exponent s is large enough (relative to the moment-bound gap corresponding to the quantity $\varepsilon_j - \varepsilon_{j-1}$, as described earlier). This algorithm is given the values X at the outset as "baseline advice", and queries the N coordinates of Y one at a time, making a single adaptive query toward the end of its operation (as its t^{th} query for $t = N - \Theta(\sqrt{N})$, in fact). Its goal is to maximize the certified lower bound $\hat{v}$ for $\mathsf{MA}_\ell(Y)$ implied by the "baseline" values X, superimposed by the first t queries to Y. We call an algorithm achieving a nontrivial guarantee here (expressed as an exponential-moment bound on $\mathsf{MA}_\ell(z) - \hat{v}$) a *steepener*—it "steepens" the rate of ascent toward the final value $\mathsf{MA}_\ell(Y)$ across its first t queries, compared to a naïve approach.

The value $\mathsf{MA}_\ell(z)$ is nicely characterized as $\mathsf{MA}_\ell(z) = \max_T \operatorname{Avg}_{i \in T}(z_i)$, taken over a natural family of subsets T of inputs which we call *M-trees*; these consist of $\sqrt{N}$ coordinates each, and in the games-against-Nature view, correspond to strategies.[4] In the design and analysis of our steepener algorithm, a central concern is whether our adaptive t^{th} query to Y is made inside T^*, an *optimal* M-tree for the input z (this subset is not known to the algorithm). After making $t-1$ queries to Y in a semi-random way, the t^{th} query is chosen from one of two candidates $\mathbf{i}, \mathbf{j}$. These are chosen in such a way that at most one can come from T^*. The algorithm inspects the surroundings of the $\mathbf{i}$ coordinate in the partial view of Y so far over the baseline X. It essentially looks for signs marking this coordinate as "special"—which can be very roughly interpreted as "likely to come from T^*"—in which case $\mathbf{i}$ is selected; otherwise $\mathbf{j}$ is chosen by default.[5]

Before discussing our specific decision rule, we give a brief upshot of its analysis. First, in the event where neither $\mathbf{i}$ nor $\mathbf{j}$ is in T^* (as occurs most often), we are actually "lucky" in that we have made sufficiently many queries to T^* already to get satisfactory bounds regardless of the adaptive decision's outcome. If instead $\mathbf{j} \in T^*$ (which we call the "$\mathbf{j}$-critical case"), we essentially show that most possible values for $\mathbf{i}$ will be rejected, causing $Y_\mathbf{j}$ to receive the adaptive query as desired. In the trickier "$\mathbf{i}$-critical case" ($\mathbf{i} \in T^*$), we show something weaker but still sufficient: in a "substantial bulk" of cases (under an appropriate exponential measure), it holds that *either* $Y_\mathbf{i}$ receives the adaptive query, *or* an appreciable portion of the gap $z_\mathbf{i} - X_\mathbf{i}$ is not "felt" as a contribution to the gap $\mathsf{MA}_\ell(z) - \hat{v}$, so that a failure to query $Y_\mathbf{i}$ would not be as harmful to the estimate $\hat{v}$ as it might naïvely appear. (We mention that our actual disjunctive analysis here in the $\mathbf{i}$-critical case seems to rely, for its needed quantitative strength, on the use of exponential-moment bounds.)

As for the decision rule, we adaptively query $Y_\mathbf{i}$ if at least one of three "special" selection-conditions hold for the path $P_\mathbf{i}$ from the output gate to input $\mathbf{i}$, considered on the "hybrid" input u which superimposes the partial view of Y after $t-1$ queries upon the baseline X. The path $P_\mathbf{i}$ is called "dominant" for u at a Max gate g, if it passes through g to the larger-valued of its two input/child Avg gates. As a first selection-condition, if $P_\mathbf{i}$ is dominant on more than, say, a .51 fraction of its Max gates, then (as the input size N is large) this is a fairly strong sign that $\mathbf{i}$ is "special" and worthy of the adaptive t^{th} query.

To motivate a further selection-condition, we consider cases in which the previous one fails to apply. If $\mathbf{i} \in T^*$ but $P_\mathbf{i}$ does not have the ".51-dominant" property on u, let us consider the derived input $\tilde{u}$ in which $u_\mathbf{i} = X_\mathbf{i}$ is replaced with z_i. If $\mathbf{i}$ is not part of every optimal M-tree for input $\tilde{u}$, then a decrement of coordinate $\mathbf{i}$ from $z_\mathbf{i}$ down to $X_\mathbf{i}$ does not reduce the MA_ℓ-value, and in this case we can be content not to query $Y_\mathbf{i}$ for our adaptive t^{th} query.

[4]The above was basically observed in [3], [1], and pointed out to me by Rahul Santhanam [38].

[5]There is some notional resemblance between our approach here and an improved randomized, zero-error query algorithm of Magniez et al. [39] for the (exact) Recursive Majority-of-3 problem, in that their algorithm forms "predictions" for the values of certain nodes and uses these to drive decisions. The details appear very different, however.

On the other hand, if $\mathbf{i}$ *is* in each optimal M-tree for $\tilde{u}$, then $P_{\mathbf{i}}$ is dominant at each Max gate on $\tilde{u}$. To study this situation, we will define the "decisiveness" of a Max gate on a given input, as a certain "height-normalized" multiple of the gap between its two input values. (To build intuition: if a path P_i is dominant at each Max gate, then incrementing input i by θ will increase the decisiveness of each such gate by θ.) Then the aforementioned decrement of coordinate $\mathbf{i}$ can be seen to reduce the MA_ℓ-value only proportionally to ($N^{-.5}$ times) the *minimum* decisiveness of $P_{\mathbf{i}}$ on $\tilde{u}$; at the same time, the total decrement must be proportional to the *near-median* decisiveness along the same path, in order to destroy this path's dominance on a .49 fraction of these gates. If the gap between these two decisiveness values is large, then enough of the coordinate decrement is not felt as loss in MA_ℓ value, and again we are content not to query $Y_{\mathbf{i}}$ for our adaptive t^{th} query.

Thus we are led to worry about the case where the minimum and near-median decisiveness along $P_{\mathbf{i}}$ on $\tilde{u}$ are near-equal. But overall, we need only worry if this happens for *sufficiently many* possible values of $\mathbf{i}$. Now strictly speaking, we need to clarify relative to which partial conditioning this holds, and exercise care since $\tilde{u}$ is defined relative to $\mathbf{i}$. Such concerns make the $\mathbf{i}$-critical case subtle, but we pass over details here and just suggest the main idea. We recenter our analysis on $\hat{u}$, which is $\tilde{u}$ with the $\mathbf{i}^{th}$ coordinate $z_{\mathbf{i}}$ replaced by $Y_{\mathbf{i}}$, and now consider $\mathbf{i}$ as undetermined (but we condition on $\mathbf{i} \in T^*$; in fact it could be almost any index in T^*). Focusing on our "worrisome" case sketched above, consider an outcome to the vectors (X, Y), determining $\hat{u}$, for which:

1) Most paths P_i with $i \in T^*$ are dominant for $\hat{u}$ at all Max gates; and,
2) Most such paths also have minimum decisiveness (on $\hat{u}$) nearly equal to their near-median decisiveness.

By an analysis of random walks from the root/output gate, and constrained to end at an input i with $i \in T^*$, we conclude that most such P_i also have many *"pendant"* Max gates (adjacent to P_i, as a sibling to one of its Max gates) whose decisiveness is also nearly equal to the minimum decisiveness on P_i itself.

Consider one such P_i for $\hat{u}$, and now imagine $\mathbf{i} = i$ is selected and Y_i is decremented down to X_i, yielding the input u seen by our steepener algorithm before the adaptive t^{th} query. If this decrement is significantly larger than the minimum decisiveness d_{min} for P_i on $\hat{u}$, then an appreciable portion of this decrement is not felt as loss in MA_ℓ-value, and our algorithm can be content not to adaptively query $\mathbf{i}$. But if the decrement is very nearly equal to P_i's minimum decisiveness, then one can show that the quantity $S_i^-(u)$, measuring the *sum* of decisiveness values of all Max gates on which P_i is *non*-dominant, is small compared to d_{min}, and thus also small compared to the decisiveness (for u) of many Max gates *pendant* to P_i—whose decisiveness values are, moreover, all nearly equal.

It is this type of observable property for $P_{\mathbf{i}}$ that we identify as our next "special" one, and use to define our second selection-condition for $P_{\mathbf{i}}$ on u (to make the adaptive t^{th} query to $\mathbf{i}$). Crucially, we also show (by another, similar analysis of random walks on trees) that this selection-condition is unlikely to be met by $\mathbf{i}$ in the $\mathbf{j}$-critical case (where we should query $\mathbf{j} \in T^*$ instead), a case in which $\mathbf{i}$ is essentially uniform over $\approx N/2$ possible values.

A third selection-condition for querying $Y_{\mathbf{i}}$ concerns a relatively short initial segment of $P_{\mathbf{i}}$ on u starting from the root. The third condition is less central, but facilitates our random-walk-based analysis of the other two conditions. This completes our sketch description and motivation for the steepener algorithm we provide. We have glossed over some significant aspects of its analysis, but we believe the above represents the most important core ideas of our approach.

II. Key definitions

In what follows we will lay out some important definitions and suggest how they are used.

Definition 1: Let U be a nonnegative random variable, and $s > 0$. For $\beta \in \mathbf{R}$ we say that U is (s, β)-*small* if

$$\mathbf{E}[\exp(sU)] \ \le \ e^{s\beta} \ .$$

Proposition 1: If $0 < s' < s$ and U is (s, β)-small, then U is (s', β)-small.

We consider particular triples of related inputs (X, Y, z) to MA_ℓ:

Definition 2 (Ensembles): Fix $N = 4^\ell$ and values $0 \le \beta < \alpha \le 1$, as well as a value $s > 0$, which may be $+\infty$. An (α, β, s)-*ensemble* over N input variables is a tuple (X, Y, z), where:

1) $z \in [0,1]^N$ is a fixed vector;
2) (X, Y) are a pair of vector-valued random variables, each with support in $[0,1]^N$, and obeying $X \le Y \le z$ coordinate-wise;
3) each coordinate tuple (X_i, Y_i) takes finitely many possible values, and is independent of $\{(X_j, Y_j)\}_{j \neq i}$, although X_i need not be independent of Y_i;
4) for each such tuple, $(z_i - X_i)$ is (s, α)-small and $(z_i - Y_i)$ is (s, β)-small. That is (for the case where $s < \infty$),

$$\mathbf{E}[\exp(s(z_i - X_i))] \ \le \ e^{s\alpha} \ , \ \mathbf{E}[\exp(s(z_i - Y_i))] \ \le \ e^{s\beta} \ .$$

Item 4 constrains X_i and Y_i to be "typically close" to z_i, which in our application is regarded as a "true" value for the i^{th} coordinate.

Definition 3: If $w \in \{[0,1] \cup \{*\}\}^N$ where $N = 4^k$, and $x \in [0,1]^N$ satisfies $x_i \le w_i$ whenever $w_i \neq *$, then we let $[w \searrow x]$ be w with all entries $w_i = *$ replaced with the corresponding x_i. We let

$$LB_{\mathsf{MA}_k}(w; x) \ := \ \mathsf{MA}_k([w \searrow x]) \ ,$$

and note that this is the largest possible lower bound on $\mathsf{MA}_k(y)$ valid for all $y \in [0,1]^N$ satisfying $y \geq x$ and agreeing with w on the non-$*$ entries of w. We also let $LB_{\mathsf{MA}_k}(w) := LB_{\mathsf{MA}_k}(w; 0^N)$.

Next we describe our key type of "base case" algorithm (using baseline advice) for approximating MA_ℓ with useful guarantees; we call such an algorithm a "steepener".

Definition 4 (Query algorithms with baseline advice): Fix $N = 4^\ell$. A *query algorithm with baseline advice* is a possibly-randomized algorithm $A = A^x(y)$ that is given unlimited access to a "baseline advice" vector $x \in [0,1]^N$ and query access to an unknown $y \in [0,1]^N$, with the promise $x \leq y$.

A is called an *all-queries algorithm* (with baseline advice) if it always queries every coordinate of y, for all such pairs $x \leq y$.

Definition 5 (Steepeners): Assume $N = 4^\ell \geq 10^{21}$, and let $\mathbf{e} := 2^{30}\sqrt{N} + 1 < N/2$. Consider an $\underline{\text{all-queries}}$ algorithm $A = A^x(y)$ with baseline advice for a $\underline{\text{fixed}}$ input size $N = 4^\ell$. Assume that $A^x(y)$, given advice string $x \leq y$, makes $(N - \mathbf{e})$ queries to y, yielding a partial view $y^* \in \{[0,1] \cup \{*\}\}^N$, before making the remaining $\mathbf{e}$ queries.

Suppose $\alpha > \beta \geq 0$ and $s, s' \in (0, +\infty)$, and $c \in (0,1)$.

We say that A as above is a *c-steepener, with respect to the 4-tuple* (α, β, s, s') if, for every (α, β, s)-ensemble (X, Y, z), when $(x,y) \sim (X,Y)$ we have the s'-exponential-moment bound

$$\mathbf{E}[\ \exp(s'(\mathsf{MA}_\ell(z) - LB_{\mathsf{MA}_\ell}(y^*; X))\ ]\ \leq \exp\left(s'\left[\frac{\mathbf{e} - c}{N} \cdot \alpha + \frac{N - \mathbf{e} + c}{N} \cdot \beta\right]\right)\ .$$

Here the expectation is over the randomness in (X, Y) and any random choices by A.

Thus the algorithm makes all N queries to Y (which is useful in our application), but its steepener property only concerns the partial view after $N - \mathbf{e}$ queries.

To gain familiarity with the definition, note that a c-steepener for 4-tuple $(\alpha, \beta, s_1, s'_1)$ is automatically one for $(\alpha, \beta, s_2, s'_2)$ if $s_2 \geq s_1$ and $s'_2 \leq s'_1$. This follows directly from Proposition 1—the set of ensembles for which the algorithm must succeed only shrinks, *and* the definition of "success" for a given ensemble becomes more lenient.

Definition 6: We say that all-queries algorithm $A = A^x(y)$ with baseline advice, taking fixed input size $N = 4^\ell \geq 10^{21}$, is an *admissible steepener* if, for some fixed $c \in (0,1)$, it is a c-steepener for all 4-tuples in the family

$$\{\ (\alpha, \beta, s, s')\ :\ s\Delta\ =\ 1000\ ,\quad s'\ =\ 10^{-4}\sqrt{N} \cdot s\ \}$$

where for each such tuple we use $\Delta := \alpha - \beta$.

Note the large *exponent boost*, $s' \gg s$, which will be a key source of progress. In the full version we show that an admissible steepener does exist, for $N = 4^{10^{10}}$, and that its guarantee suffices to achieve our main goals. It is applied recursively in conjunction with a form of composition of query algorithms described below.

First, the following definitions help us study the incremental progress of query algorithms in computing certified lower bounds for the Max-Average function MA_k.

Definition 7: Fix $N = 4^k$, and let $A = A(y)$ be a randomized query algorithm on a length-N input $y \in [0,1]^N$.

- For $t \in [0, N]$ and input y, let $u^t \in \{[0,1] \cup \{*\}\}^N$ be the (random) string describing the partial view of y gained by A after t queries. (If A makes only $t' < t$ queries, then we take $u^t := u^{t'}$).
- Again for fixed t, y as above, we define the *accrued value after t steps* (with respect to MA_k), a random variable denoted Val_t, as the certified lower-bound $LB_{\mathsf{MA}_k}(u^t)$. We also define the *defect after t steps* by $\mathrm{Def}_t := \mathsf{MA}_k(y) - \mathrm{Val}_t$. Note that these form a non-increasing sequence, with $\mathrm{Def}_0 = \mathsf{MA}_k(y)$.

Definition 8 (Epochs and defect bounds): Let A be as in the previous definition.

- If $\mathcal{T} = (t_0, \ldots, t_m)$ with $0 = t_0 < t_1 < \ldots < t_m = N$ are a collection of values (for some $m \geq 1$), we refer to the values $t_0, \ldots, t_m$ as *epoch-markers*, and for $j \in [m]$ we regard the $(t_{j-1} + 1)^{th}$ through $(t_j)^{th}$ queries made by the algorithm as belonging to the j^{th} *epoch* with respect to $\mathcal{T}$. (Thus the $m+1$ markers define m epochs.)
- Suppose $s_0, \ldots, s_m > 0$ are given, and
$$\mathcal{P} = \{(t_0, s_0, \varepsilon_0),\ \ldots,\ (t_m, s_m, \varepsilon_m)\}$$
are a collection of triples $(t_j, s_j, \varepsilon_j)$ with $\mathcal{T} = (t_0, \ldots, t_m)$ a valid set of epoch-markers, and that each $\varepsilon_j \in [0,1]$. We say that A *obeys the defect bounds* $\mathcal{P}$ (with respect to MA_k) if for each $j \in [0, m]$ and $\underline{\text{all}}$ inputs $y \in [0,1]^N$, the random variable Def_{t_j} (defined with respect to y, over the randomness in the execution of $A(y)$) is (s_j, ε_j)-small as in Def. 1.

It will be important for our work to allow the values s_j, ε_j in such a sequence to vary with j.

Definition 9 (Composed algorithms): The form of recursive composition we will use applies within the following setting. Let us fix:

- Values $N_1 = 4^\ell,\ N_2 = 4^k$, and $m \in [1, N_2]$;
- An all-queries algorithm $A_1 = A_1^x(y)$ with baseline advice x and input $y \geq x$, both vectors in $[0,1]^{N_1}$;
- An all-queries algorithm $A_2(w)$ making queries to $w \in [0,1]^{N_2}$ (both A_1 and A_2 may be randomized);
- A collection $\mathcal{T} = (t_0, \ldots, t_{m-1}, t_m)$, satisfying
$$0\ =\ t_0\ <\ t_1\ <\ \ldots\ < t_m\ =\ N_2\ ,$$
of epoch-markers for the N_2-query algorithm A_2.

We then define the *composed query algorithm* $A' = \mathrm{comp}(A_1, A_2; \mathcal{T})$, an all-queries algorithm working on input

$z \in [0,1]^{N_1 \cdot N_2}$, as follows. We first describe the high-level framework, then fully specify the algorithm.

- First, we regard z as having the indexing $z = (z^1, \ldots, z^{N_1})$, where each $z^i \in [0,1]^{N_2}$.
- A' maintains $\underline{\text{parallel simulations}}$ of the executions of $A_2(z^i)$, for each $i \in [N_1]$. Each such simulation uses an independent source of randomness. Letting $U^t \in \{[0,1] \cup *\}^{N_1 \cdot N_2}$ denote the overall partial view of A' after t queries ($t < N_1 N_2$), the algorithm chooses a *next-query target group* index $j = j(t) \in [N_1]$ as a function of t and U^t.
- The simulation of $A_2(z^j)$ is then $\underline{\text{advanced}}$ by a single query; we also describe this as $\underline{\text{allocating}}$ a query to $A_2(z^j)$. (Our rule will only choose the index j if z^j has not yet been fully queried in the simulation.)
- This process continues for $N_1 N_2$ steps, after which all of z has been queried.

It remains to give the selection rule for the next-query target group based upon (t, U^t). The query indices $t \in [N_1 N_2]$ are divided into m *Phases*,[6] where Phase $r \in [1, m]$ consists of the queries whose index t lies in $[t_{r-1}N_1 + 1, t_r N_1]$.

Phase r in turn consists of N_1 *sub-Phases*, each consisting of $(t_r - t_{r-1})$ queries. Phase r is described below.

Phase r of algorithm $A' = \mathrm{comp}(A_1, A_2; \mathcal{T})$:

//Precondition: each of the simulations of $A_2(z^1), \ldots, A_2(z^{N_1})$ have been allocated $\underline{\text{exactly}}$ t_{r-1} queries so far, yielding some partial view $v^i \in \{[0,1] \cup \{*\}\}^{N_2}$ of z^i for each $i \in [N_1]$.

1) For each $i \in [N_1]$, let $x_i := LB_{\mathsf{MA}_k}(v^i)$;
   // $x_i \in [0,1]$ is computable from the algorithm's partial view at the outset of Phase r
2) Simulate the complete execution of query algorithm $A_1^x(y)$, taking baseline advice x as defined above, and with values $y \in [0,1]^{N_1}$ computed upon request in the course of the simulation, as follows:
   - If the simulation of $A_1^x(y)$ chooses to query y_i for position $i \in [N_1]$, advance the simulation of $A_2(z^i)$ by $(t_r - t_{r-1})$ queries, yielding a new partial view w^i of z^i; and let $y_i := LB_{\mathsf{MA}_k}(w^i)$.

     //Each such block of $(t_r - t_{r-1})$ queries forms a sub-Phase, of which there are N_1; in each such sub-Phase, the selected index $i \in [N]$ is chosen as the next-query target group $(t_r - t_{r-1})$ times in succession; after Step 2, each z^i has received t_r queries, extending the Precondition.

In brief, what we show for this definition is that, if A_1 is a steepener, and A_2 an all-queries algorithm obeying a given sequence of defect bounds associated with its epoch-markers, then their composition will obey a new (and in a sense "improved") derived sequence of defect bounds associated with carefully chosen new epoch markers. We then apply composition repeatedly using a fixed, admissible steepener A_1.

[6]While we will define epochs to analyze our recursively composed algorithms, the chosen epoch-markers will *not* coincide with the endpoints of these "Phases"—hence the use of a distinct term here.

This completes our survey of some of the key definitions used in the full version.

Acknowledgment

This work owes a great debt to Rahul Santhanam. Rahul provided generous support with a postdoc at U. Edinburgh (2014-15); made sustained joint efforts with me on the problem studied in the present work, and discussed extensions and variant problems [38]; has provided many, many helpful conversations; and has offered material and moral support throughout this research. I acknowledge support from his ERC Consolidator Grant (AL Unif) and from the University of Edinburgh, and travel support from Oxford University.

I also thank Laci Babai for helpful discussions about the exposition of games against Nature and interactive proofs.

References

[1] C. H. Papadimitriou, "Games against nature," *J. Comput. Syst. Sci.*, vol. 31, no. 2, pp. 288–301, 1985. [Online]. Available: https://doi.org/10.1016/0022-0000(85)90045-5

[2] B. Hnich, R. Rossi, S. A. Tarim, and S. Prestwich, *A Survey on CP-AI-OR Hybrids for Decision Making Under Uncertainty*. Springer New York, 2011, pp. 227–270.

[3] M. L. Littman, S. M. Majercik, and T. Pitassi, "Stochastic boolean satisfiability," *Journal of Automated Reasoning*, vol. 27, no. 3, pp. 251–296, Oct 2001.

[4] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 1994.

[5] C. Boutilier, T. L. Dean, and S. Hanks, "Decision-theoretic planning: Structural assumptions and computational leverage," *J. Artif. Intell. Res.*, vol. 11, pp. 1–94, 1999.

[6] C. H. Papadimitriou and J. N. Tsitsiklis, "The complexity of Markov decision processes," *Mathematics of Operations Research*, vol. 12, no. 3, pp. 441–450, 1987.

[7] C. Boutilier, R. Dearden, and M. Goldszmidt, "Stochastic dynamic programming with factored representations," *Artif. Intell.*, vol. 121, no. 1-2, pp. 49–107, 2000.

[8] C. Guestrin, D. Koller, R. Parr, and S. Venkataraman, "Efficient solution algorithms for factored MDPs," *J. Artif. Intell. Res.*, vol. 19, pp. 399–468, 2003.

[9] T. Walsh, "Stochastic constraint programming," in *15th European Conference on Artificial Intelligence (ECAI)*, 2002, pp. 111–115.

[10] T. Teige and M. Fränzle, "Resolution for stochastic boolean satisfiability," in *Logic for Programming, Artificial Intelligence, and Reasoning*, 2010, pp. 625–639.

[11] ——, “Generalized Craig interpolation for stochastic boolean satisfiability problems,” in *Tools and Algorithms for the Construction and Analysis of Systems - 17th International Conference, TACAS*, 2011, pp. 158–172.

[12] N. Lee, Y. Wang, and J. R. Jiang, “Solving exist-random quantified stochastic boolean satisfiability via clause selection,” in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2018, pp. 1339–1345.

[13] S. Albers, “Online algorithms: a survey,” *Mathematical Programming*, vol. 97, no. 1, pp. 3–26, Jul 2003.

[14] L. Babai, “Trading group theory for randomness,” in *17th ACM STOC*, R. Sedgewick, Ed. ACM, 1985, pp. 421–429.

[15] S. Goldwasser, S. Micali, and C. Rackoff, “The knowledge complexity of interactive proof systems,” *SIAM J. Comput.*, vol. 18, no. 1, pp. 186–208, 1989, earlier version in STOC ‘85. [Online]. Available: https://doi.org/10.1137/0218012

[16] S. Goldwasser and M. Sipser, “Private coins versus public coins in interactive proof systems,” in *18th ACM STOC*, 1986, pp. 59–68.

[17] C. Lund, L. Fortnow, H. J. Karloff, and N. Nisan, “Algebraic methods for interactive proof systems,” *J. ACM*, vol. 39, no. 4, pp. 859–868, 1992.

[18] A. Shamir, “IP = PSPACE,” *J. ACM*, vol. 39, no. 4, pp. 869–877, 1992.

[19] M. Snir, “Lower bounds on probabilistic linear decision trees,” *Theoretical Computer Science*, vol. 38, pp. 69 – 82, 1985.

[20] M. E. Saks and A. Wigderson, “Probabilistic boolean decision trees and the complexity of evaluating game trees,” in *27th IEEE FOCS*, 1986, pp. 29–38.

[21] R. Santhanam and R. Williams, “Beating exhaustive search for quantified boolean formulas and connections to circuit complexity,” in *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, 2015, pp. 231–241.

[22] M. L. Littman, “Initial experiments in stochastic satisfiability,” in *AAAI/IAAI*, 1999, pp. 667–672.

[23] S. M. Majercik and M. L. Littman, “Contingent planning under uncertainty via stochastic satisfiability,” in *AAAI/IAAI*, 1999, pp. 549–556.

[24] R. Williams, “Strong ETH breaks with Merlin and Arthur: Short non-interactive proofs of batch evaluation,” in *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, 2016, pp. 2:1–2:17.

[25] R. Impagliazzo and R. Paturi, “On the complexity of k-sat,” *J. Comput. Syst. Sci.*, vol. 62, no. 2, pp. 367–375, 2001.

[26] L. Chen, S. Goldwasser, K. Lyu, G. N. Rothblum, and A. Rubinstein, “Fine-grained complexity meets IP = PSPACE,” in *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, T. M. Chan, Ed. SIAM, 2019, pp. 1–20.

[27] A. Abboud and A. Rubinstein, “Fast and deterministic constant factor approximation algorithms for LCS imply new circuit lower bounds,” in *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, 2018, pp. 35:1–35:14.

[28] S. Aaronson and A. Wigderson, “Algebrization: A new barrier in complexity theory,” *ACM Trans. Comput. Theory*, vol. 1, no. 1, pp. 2:1–2:54, 2009.

[29] A. Rubinstein, S. Seddighin, Z. Song, and X. Sun, “Approximation algorithms for LCS and LIS with truly improved running times,” in *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, 2019, pp. 1121–1145.

[30] M. Hajiaghayi, M. Seddighin, S. Seddighin, and X. Sun, “Approximating LCS in linear time: Beating the $\sqrt{n}$ barrier,” in *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, 2019, pp. 1181–1200.

[31] M. E. Saks and C. Seshadhri, “Estimating the longest increasing sequence in polylogarithmic time,” *SIAM J. Comput.*, vol. 46, no. 2, pp. 774–823, 2017.

[32] M. Leshkowitz, “Round complexity versus randomness complexity in interactive proofs,” in *RANDOM 2018*, 2018, pp. 49:1–49:16.

[33] M. Santha, “On the monte carlo boolean decision tree complexity of read-once formulae,” *Random Struct. Algorithms*, vol. 6, no. 1, pp. 75–88, 1995.

[34] B. Monien and E. Speckenmeyer, “Solving satisfiability in less than 2^n steps,” *Discrete Applied Mathematics*, vol. 10, no. 3, pp. 287–295, 1985.

[35] R. Paturi, P. Pudlák, and F. Zane, “Satisfiability coding lemma,” *Chicago J. Theor. Comput. Sci.*, 1999, earlier version in FOCS ’97.

[36] U. Schöning, “A probabilistic algorithm for k-sat and constraint satisfaction problems,” in *40th Annual Symposium on Foundations of Computer Science, FOCS ’99, 17-18 October, 1999, New York, NY, USA*, 1999, pp. 410–414.

[37] D. Dubhashi and A. Panconesi, *Concentration of Measure for the Analysis of Randomized Algorithms*, 1st ed. New York, NY, USA: Cambridge University Press, 2009.

[38] R. Santhanam, personal communication.

[39] F. Magniez, A. Nayak, M. Santha, J. Sherman, G. Tardos, and D. Xiao, “Improved bounds for the randomized decision tree complexity of recursive majority,” *Random Struct. Algorithms*, vol. 48, no. 3, pp. 612–638, 2016.