

# Research Statement

Paul Valiant

December 7, 2011

Computation has become a fundamental workhorse for progress in the sciences. Across virtually every field of science, computation has opened up new avenues of research. At the immediate level, computers can supervise and direct experiments, collect orders of magnitude more data, and analyze it in ways unthinkable without them. But perhaps much more significantly, what one might call the “algorithmic mindset” is rapidly spreading its reach. Computation is not just about computers, it is a way of thinking about the world. Dijkstra famously said “Computer science is no more about computers than astronomy is about telescopes.” Computational phenomena are all around us, in the sense, for example, that an ant colony embodies a stunningly effective distributed computer. There are many different approaches to studying different aspects of the behavior of an ant colony or other system—ecological, biochemical, genetic, physics-based. However, the spread of computational notions across the scientific community is increasingly encouraging computational views of such phenomena to be recognized, abstracted, and studied as fundamental units.

Computation is taking up the role that mathematics has long enjoyed, that of being *unreasonably effective in the sciences*. In what follows, I will describe several areas of science that I have engaged with a “computational lens”. Further, as the long synergistic history of mathematics and physics attests, the benefits can flow both ways: the history is not just of developments in mathematics enabling fundamental scientific progress, but also of engagement with physics motivating fundamental developments in mathematics. Newton’s mathematical accomplishments, for example, cannot be understood except in light of the developments in mechanics he was pursuing. Thus it may be hoped that when the interconnections between computation and the other sciences have reached a similar level of maturity, corresponding fruits will have been gained on both sides.

I want to emphasize that, while my background in computer science has been from the theory perspective, this vision for computer science is not theoretical in nature. The intersection between computation and the other sciences is not theoretical in nature. To seriously develop the opportunities I describe, one must embrace tools and techniques from all parts of computer science. My philosophy is primarily one of engagement, of willingness to get one’s hands dirty in the service of the right problem. I have worked on a wide variety of problems, and continue to pursue new leads in the search for fundamental problems whose solution might tease apart deep structure in computation, and provide significant benefits for the other sciences.

## 1 Statistics and the quest for data efficiency

Computer hardware and software have advanced to the point where for almost any test of speed or memory in which computers can reasonably compete, they may be set up to outperform people. One area, however, in which our abilities vastly exceed anything obtainable algorithmically is that of *data*

*complexity*: how much data does one need to complete a task? One compelling demonstration of this gulf is provided by the recent “Jeopardy!” appearance of the IBM Watson computing system. On speed and memory, Watson has a huge advantage over humans: it responded to questions with millisecond precision; its memory stored, among other things, the complete contents of a million books and the entirety of Wikipedia. Yet was it able to leverage this vast record of human knowledge to outperform human players? Counting speed, yes; judging on accuracy, no, often drawing guffaws from the studio audience for its missteps, most notably for naming “Toronto” as its “Final Jeopardy” response in the category “US Cities.”

We live in an era of “big data,” where the amount of data that can be brought to bear on questions of biology, climate, economics, and (as Watson demonstrates) trivia, is not only vast, but also expanding rapidly. Scaling up a computational framework to comfortably deal with ever-larger data presents a series of challenges in algorithms, infrastructure, and interfaces. I want to highlight one perhaps often overlooked aspect of this phenomenon, which is paradoxically only becoming more acute as the size of data shoots up: *are we using our data efficiently?* One would naively suspect the opposite, that it is only for tiny data sets that one must be careful to use each datum wisely, and that for huge data sets one can afford to be profligate. However, as a long history of algorithms development demonstrates, the potential losses from choosing an inefficient approach only grow with the problem size. When sorting five numbers, the most natural algorithms are often the best, though no one would use such algorithms to sort a million numbers. Correspondingly, I put forward that the data analysis techniques that spawned from a statistics that matured in an environment of dozens or hundreds of data points might drastically underperform in an era of big data.

For several years, starting with my PhD thesis ([38], to be published in updated form as [40]), I have been investigating the following fundamental instance of this problem: *given independent samples from one or more probability distributions, what is the fewest samples needed to estimate properties of the distribution(s)?* Three widely relevant instances of slightly different forms are: Given a table with  $n$  rows, how many random rows must be examined to estimate the number of distinct rows to within  $\pm \epsilon n$  (with high probability)? Given a distribution on at most  $n$  elements, how many samples are needed to approximate its entropy to within  $\pm \epsilon$  (with high probability)? And, perhaps most basic of all, given samples from two distributions over  $n$  elements, how many samples are needed to estimate the—say,  $L_1$ —distance between the two distributions to within  $\pm \epsilon$  (with high probability)? These questions are fundamental to statistics, and more broadly, the quest for efficiently leveraging big data. The fact that they remained unresolved for so long is perhaps a reflection of the deep need for computational input to the other sciences.

In my PhD thesis, I put these problems under a common framework for the first time, allowing me to develop unified tools for addressing this whole class of problems. In analogy with the *canonical testers* of graph property testing introduced in [3], I described a canonical tester for this class of statistical problems which is guaranteed to perform within a  $n^{o(1)}$  factor of optimal, that is, use a number of samples that is within  $f(n)$  factor of the minimum possible where  $f(n)$  grows more slowly than any fractional power of  $n$ . This  $n^{o(1)}$ -optimal tester has one rather unexpected property: it completely ignores any domain element it hasn’t seen *many* ( $\gg \log n$ ) times among the samples. Such frequently occurring elements are very easy to deal with: the law of large numbers says that the *observed* frequency of such an element will be a good estimate of its true frequency, namely, its frequency under the distribution whose properties we are trying to estimate. Conversely, the “rare” samples from the distribution, those samples which are individually seen only a handful of times, are

traditionally the hardest to analyze. While discarding the delicate information contained in these rare samples clearly has a cost, my thesis demonstrates that this cost is only a  $n^{o(1)}$  factor in the number of samples; this cost is counterbalanced by the significantly simpler picture this perspective allows, yielding for the first time the ability to analyze these problems in general. My thesis yielded lower bounds of  $n^{1-o(1)}$  for all three problems mentioned above: entropy, distinct elements, and  $L_1$  distance, providing the first near-linear lower bounds for two of these problems, and essentially reproducing the lower bounds of [33] for the distinct elements problem while bringing it under this new common framework.

While these results provide a general framework for deriving  $n^{o(1)}$ -factor tight testing results for statistical properties, which successfully narrowed previous gaps of up to  $\sqrt{n}$ , all the new results of [38] constituted improvements of lower bounds to almost-match known upper bounds, but yielded no new algorithms. For each of the three problems above—entropy, distinct elements, and  $L_1$  distance—the best algorithms then known all require a linear number of samples. In fact, achieving such performance is trivial: take  $\theta(n)$  samples from the distribution(s), and compute the property in question (entropy, distinct elements, or  $L_1$  distance) on the empirical distribution(s) of the samples. This linear barrier thus became a focal point of attention: a hundred years of statistical algorithms had been unable to move it, and the lower bounds all seemed to be rapidly converging towards it. The challenge became even more tantalizing with the nonconstructive demonstration of Paninski that a *sublinear* estimator for entropy exists [29, 30].

In recent joint work with Gregory Valiant, we exhibited the first explicit sublinear estimator for entropy, distinct elements,  $L_1$  distance, and a class of related properties [34, 35]. These estimators all use  $\theta(\frac{n}{\log n})$  samples, a log-factor improvement. In the realm of “big data,” log-factor savings in data size are significant. Are further improvements possible? In this setting, no: we also showed that  $\theta(\frac{n}{\log n})$  is the optimum number of samples up to constant factors. The slackness of  $2^{\theta(\sqrt{\log n})}$  of my thesis is eliminated.

The new perspective which enabled these results was, perhaps surprisingly, that of linear programming. I had been invited to a workshop by Alon Orlitsky outlining an ambitious program of his to characterize (in some sense) the maximum one could learn from samples of a distribution (based on the work [2, 26, 27, 28]). Gregory Valiant and I took up the challenge of reframing certain of these notions in the robustly well-behaved language of linear programming. “Linear programming relaxation” is a famously successful technique to rephrase problems in approximate forms to make them computationally tractable, thus trading accuracy for computational efficiency. By contrast, our concern was *data* efficiency, and, here, the *analysis* had been intractable, as previously no algorithm was known to estimate entropy or distinct elements or  $L_1$  distance in sublinear samples.

The main new tool we used in the analysis is a pair of new multivariate central limit theorems we developed, one proven directly via Stein’s method, the other proven from the first via convolution/deconvolution arguments based on convexity. Central limit theorems are fundamental to statistics—it has been said that the modern history of statistics can be described as a history of central limit theorems—and while central limit theorems are relatively well-understood in the one dimensional setting, adapting one of these to a multivariate setting (or adapting a multivariate central limit theorem from the wrong setting) often incurs an exponential penalty in the dimension. For example, the standard multivariate central limit theorem is that of Götze ([18]) which shows convergence to the normal distribution in the following sense: two distributions are close if for any *convex* set  $C$ , the probability of  $C$  under the two distributions respectively are close; unfortunately, convexity is a much more drastic restriction in many dimensions than it is in a single dimension,

as, in high dimensions an exponential number of convex pieces may be required to approximate otherwise well-behaved sets.

Our first and most general central limit theorem we believe is uniquely natural: a sum of independent (not necessarily identical) random variables can be approximated in the *earthmover distance* sense by a multivariate normal distribution of corresponding mean and covariance. (The earthmover distance between two probability distributions is the minimum over all “schemes of moving the probability mass from one distribution to form the other,” where the cost of moving  $\epsilon$  probability mass a distance  $\delta$  is  $\epsilon\delta$ . We leverage this to prove a much stronger though less general central limit theorem in the discrete setting: sums of arbitrary multinomial distributions can be approximated in the  $L_1$  sense by the multivariate normal distribution of corresponding mean and covariance, rounded to the nearest lattice points. The  $L_1$  sense of convergence is in some sense the strongest possible for algorithmic purposes: if distributions  $X$  and  $Y$  differ by  $\epsilon$  in the  $L_1$  sense, then running an algorithm on a sample from  $X$  versus  $Y$  will produce output that differs only at most an  $\epsilon$  fraction of the time. Since so many processes in computer science can be expressed as sums of samples from multinomial distributions, we hope these results characterizing such distributions in terms of much simpler distributions may find wider use.

This is an ongoing line of work with many further developments. In the 2011 Foundations of Computer Science conference, Gregory Valiant and I showed how to, in a sense, “mechanize” the central limit theorem approach to lower bounds of the above-discussed paper, showing that it was *dual* (in the linear programming sense) to the search for the optimal *linear* estimator—a class of estimators typically considered in statistics that are particularly algorithmically simple, consisting of a single dot product [35]. This duality implies that the best lower bound obtainable by our method essentially matches the performance of best linear estimator, simultaneously implying the optimality of *both* approaches, vindicating both our central limit theorems, and a hundred years of statistical practice searching for linear estimators. Further, this implies that optimal statistical estimates can be obtained at the minimum conceivable computational cost, and our paper further demonstrates how to find such estimators.

There is much further work to be done. In particular, because  $\frac{n}{\log n}$  is optimal in this context, one might hope for a different way to frame these problems that would yield more sharply sublinear performance. Recently submitted work with Costis Daskalakis, Ilias Diakonikolas, and Rocco Servedio considered how introducing the assumption of *monotonicity* (or, more generally, unimodality, or  $k$ -modality) simplifies the problem: we showed that it has an *exponential* effect, reducing sample complexities from sublinear to sub-logarithmic [12]. In general, there are many more aspects of the problem to consider, until the worst-case bound of  $\Theta(\frac{n}{\log})$  becomes just one of a related complex of results that can be leveraged to yield case-by-case optimal performance in practice.

## 2 Security and game theory

One of the things computer science is known for in the other sciences is a bent towards considering and controlling for *worst-case* behavior. In some sense, this is only natural: the phrase “one in a million” that is typically used as an exaggeration of inconceivable rareness, for us is hardly a term of scarcity at all—in light of gigahertz computation rates, “one in a million” happens thousands of times a second. While various parts of computer science have absorbed the lessons of worst-case thinking and are now looking for subtler, more “realistic” ways of capturing behavior, there are

many other sciences which the worst-case mindset has barely touched, and which might benefit substantially from the influence of computer science towards this kind of robustness.

Economics, in light of recent history, seems particularly in need of more robust approaches. Here I will discuss the sub-field of game theory, which aims to describe how people behave in “strategic” situations, namely, situations where they stand to gain or lose, in a manner depending on the actions of themselves and others. The very related field of mechanism design looks at the flip side of this coin: how can one design a game to elicit certain behavior from the players? Underlying these fields is the assumption that players are *rational*, which often takes the form that players will play strategies in a *Nash equilibrium*.

A computer science perspective reveals at least two grounds for caution here: first, does this notion of rationality make too-stringent demands on its players—can finding a Nash equilibrium be an intractable problem? Such a result would throw the more general assumption of “rationality” into doubt, or at least require us to “move the goalposts.” Second, and more directly, any worst-case analysis of several players in a competitive setting has to allow for the possibility that the players do not behave as expected, are “irrational,” whether for reasons of computational complexity or otherwise. These two challenges, of computational hardness, and “irrationality”, have come to become focal points in the two very new fields of *algorithmic* game theory, and *cryptographic* game theory respectively.

## 2.1 The Computational Hardness of Games

What is the computational hardness of being “rational” in the game-theoretic sense? A long line of work on the computational complexity of *Nash equilibria*—the solution concept upon which much of game theory rests—has pushed this question to prominence.

It has been known since the 1989 paper of Itzhak Gilboa and Eitan Zemel that computational hardness lurks all around the Nash problem: uniqueness of equilibria is NP-complete; existence of equilibria that must include or avoid certain strategies is NP-complete; existence of equilibria with bounds on the expected payoffs to players is NP-complete [17]. But these results only make more pressing the question: *can a single equilibrium be computed?*

In joint work with Tim Abbott and Daniel Kane, I investigated whether a simpler case of this general problem could be analyzed to yield insights into the full problem [1]. We considered the special case of two-player games where the outcome for each player is restricted to be binary—a win or a loss. We asked how this simplification affected the nature of these games’ Nash equilibria, and their computational tractability. The surprising answer is that restricting the outcomes to win/lose has *no effect* on the difficulty of the Nash problem for two player games.

Over the two years that followed, further pieces of the puzzle were assembled, revealing a vast gamut of problems that all have identical complexity: Costis Daskalakis, Paul Goldberg, and Christos Papadimitriou showed that restricting general  $n$ -player games to 4-player games retained all the complexity of finding a Nash equilibrium, and was in fact as hard as any problem in the complexity class PPAD, which is conjectured to contain problems of inherent exponential complexity [13]; Xi Chen and Xiaotie Deng, and the original authors independently, extended this result to 3-player games soon after [6, 14]; and finally Chen and Deng completed the chain to show that the reduction in fact applies all the way down to 2-player games [7]. Xi Chen, Xiaotie Deng, and Shang-Hua Teng further showed that even allowing approximate equilibria does not simplify the problem [8].

I joined with Xi Chen and Shang-Hua Teng to combine the techniques in this line of work with

those of my original paper to conclude that even the simplest combination of these restrictions, namely finding *approximate* equilibria in 2-player *win/lose* games is exactly as hard as the fully general problem [9].

There is much more work still to be done, first because this collapse of the complexity of all these versions of the Nash question to the complexity class PPAD still fails to provide strong evidence either way: is this class hard or not? Second, proving this class to be computationally intractable would only make more pressing the fundamental economic question: how do strategic players interact in a computationally intractable landscape?

## 2.2 The Cryptographic Perspective on the hardness of PPAD

The class PPAD, which unexpectedly characterizes the hardness of games, is defined by the following problem: given an exponentially large directed graph that consists of paths and cycles (in-degree and out-degree is at most 1 everywhere), defined in terms of code that for every vertex returns the in-edge and out-edge, and given a *source* (that is, a node with in-degree 0 and out-degree 1) find any other source or sink (see [31]).

One way to intuitively appreciate the structure of this class is via a “hardness proof” based on a primitive we do not know how to construct, and which is in fact known to be impossible to construct in general: code obfuscation [4]. Obfuscation is the notion—obvious to anyone who has looked at someone else’s code—that it is possible to write code in a way that is hard to understand. Supposing we had a general technique for obfuscating code, then, given a scheme (such as that of [23]) for constructing pseudorandom permutations  $\pi$  on  $n$ -bit strings, a hard instance of PPAD consists of the graph defined on ordered pairs of  $n$  bit strings, such that directed edges exist only between  $(i, \pi(i))$  and  $(i + 1, \pi(i + 1))$  for integers  $i < 2^n$ . The unique source of the graph is the pair  $(0, \pi(0))$  which is given as part of the problem definition, along with an obfuscated version of the code which implements the previous sentence. The objective of this instance of the PPAD class is to find the unique sink,  $(2^n - 1, \pi(2^n - 1))$ , which is trivial if, given the code, one can figure out how to run  $\pi$ ; but given obfuscated code, one has little choice but to query for the next edge  $O(2^n)$  times.

While appealing to the intuition of obfuscation is tenuous, the recent success of *homomorphic encryption* ([16]) may provide some justification by analogy: until recently, homomorphic encryption’s best justification was perhaps also the fact that it is trivially achievable via an appeal to obfuscation. More generally, I think there is great potential for cryptographic primitives that are more “computational” in nature. Homomorphic encryption is an incredibly powerful primitive, offering the ability to securely run arbitrary code on other people’s computers; obfuscation, if something like it exists, would be an even more powerful way of encrypting computation. There is one more primitive along these lines I want to mention, which, unlike obfuscation, has plausible candidates for its construction, and which, in strengthened form, would imply the hardness of PPAD.

*Computationally sound proofs* (CS proofs) are a notion introduced by Silvio Micali that in some sense combine the features of probabilistically checkable proofs with those of interactive proofs [24]. CS proofs provide a way to condense ordinary proofs into proofs of only *logarithmic* size, that, up to security assumptions, remain impossible to fake. In a 2008 paper I showed how to use a variant of these to construct what I called *incrementally verifiable computation* [39]. The notion is as follows: given a computation that takes super-polynomial time but only polynomial memory, adapt it into a new computation that frequently (i.e., every polynomial number of steps) outputs a proof of the

correctness of its current memory state. The analogy with PPAD is as follows: a long computation is encoded as a path embedded in a PPAD instance, each node representing one of the sequence of memory states produced when the computation is run, along with the incrementally-generated proof of the correctness of these states. (Care must be taken to ensure that this path is traversable in both a forward and backward direction; see [5].) Locally, a node is interpreted as being part of the path if the node’s label, when considered as an ordered pair, consists of a memory state, and a valid proof that the memory state is indeed a legitimate snapshot of the computation being simulated. Thus to find a source or sink in the graph beyond the given one, one must either compute the final state of the arbitrary PSPACE task being simulated, or break the proof system, both of which are by assumption very hard. Thus, this version of incrementally verifiable computation, and by extension, a version of CS proofs, would imply the hardness of PPAD and thus the hardness of Nash equilibria.

This is a speculative direction of inquiry. But, bolstered by Craig Gentry’s recent success constructing homomorphic encryption, I think it is worth paying further attention to the many related notions of “encrypting computation.” This breakthrough may suggest some new ones.

### 2.3 Cryptographic Game Theory

It seems that cryptography and economics have much to gain from each other. Cryptography defines all participants as either rule-abiding or capable of arbitrary antagonistic action; economics defines all participants as merely selfish: clearly reality reflects some aspects of both models. Put another way, cryptography takes a sometimes unrealistically broad view of people’s possible actions, while economics takes a sometimes unrealistically narrow view.

Together with Silvio Micali, I have undertaken to investigate these issues in the arena of *auction design*, motivated by the rather startling fact that the established auction mechanism for *multiple* goods (Vickrey-Clarke-Groves) can, under certain circumstances, be hijacked by two players who collectively win *all* the items for sale, at a price of *zero*, despite the presence of arbitrarily many other players bidding arbitrarily high. The strategy behind this extraordinarily fortuitous outcome for these two players—cooperating, in secret—happens to lie outside the economics definition of “rational behavior”. Nevertheless, it is perhaps fears of this peculiar kind of “irrationality” that lead to this auction almost never being run in practice.

We took as our goal to investigate the effects of “cooperating in secret” on auctions. While the introduction of such collusion to a classical auction can completely destroy it, we showed that—borrowing a notion from computer science—it is possible to “benchmark” auctions, specifically, guarantee performance relative to a benchmark. Crucially, we obtained a *monotone* benchmark, namely, one which can only increase when new players join the auction. Thus the introduction of a group of bidders cooperating in secret, far from destroying the auction, in fact can only *increase* our revenue guarantee. This is only the first in an ongoing series of works on this topic. I hope this line of work will encourage a broader collaboration between economics and computer science, producing more robust models of economics, and more economically-aware approaches to cryptography and security.

### 3 Fluid dynamics—the Navier-Stokes problem

An ancient and fundamental problem in mathematics, which I am convinced will move forward only with the help of computational tools, asks: are the differential equations governing liquids and gases well-behaved? The most basic and still unresolved form of this problem was written down by Euler in 1757; the Euler equations describe a frictionless and incompressible fluid. The more commonly discussed Navier-Stokes equations consist of the Euler equations plus a term for friction. To model and simulate physical fluids, terms taking account of compressibility, temperature, and other effects, are often added. However, these equations, in all their forms, are famously badly behaved. Even the most sophisticated automobile and aerospace designers rely on wind tunnels to test their products—streamers glued to the surface of the product to indicate the direction of motion—because they cannot rely on computers to accurately implement a mathematical understanding of the underlying equations.

The catch-all term for the “badness” that arises in the equations of fluid dynamics is *turbulence*. Turbulence describes the breakdown of order in a fluid, how energy moves from large-scale structures to ever finer structures. Almost every natural system of fluids exhibits turbulence. Further, the naive computational approach of simulating each of the interacting structures in a fluid rapidly breaks down as the size of the typical structures shoots towards zero. What is needed is an understanding that this rapid race-to-the-bottom is well-behaved, a way to hide away the unmanageable complexity of turbulence behind a veil of abstraction. This is the essential approach to a computational understanding of virtually any physical phenomenon, the ability to make the claim “while I am unwilling to simulate the phenomenon beyond this certain level of detail, I know that anything happening below this level of detail is well-behaved in the following ways, and thus can have only minimal influence on the bulk properties of interest.” Unfortunately, this approach to fluid dynamics is stymied at the outset by our continued inability to resolve, for even the simplest Euler equations, the fundamental problem: given smooth and bounded initial conditions, can the fluid “blow up,” that is, might some ever-smaller pocket of fluid manage to accelerate itself in a runaway process that leads towards infinite velocity in finite time? (See [10] for a review.) I want to emphasize that the concern is not explicitly with “infinite velocity”, but rather, that it is hard to imagine any sense of saying “we understand small-scale fluid dynamics and it is well-behaved” while remaining unable to confirm, deny, or analyze the “blow-up problem.”

My interest in the blow-up problem stemmed from the impression that, from a distance, there seem to be two rather separate communities working on this problem, from two rather different directions. There is a pure math community, with roots going back hundreds of years, trying to prove that the differential equations are well-behaved, and blow-up cannot happen. Meanwhile, the community responsible for simulating fluids on supercomputers sees their computations apparently coming up to the brink of blow-up whenever they run a simulation for long enough that runaway small structure develops. These two approaches offer rather complementary benefits. Simulations of delicate and unintuitive phenomena can yield a rich array of data, shedding light on unexpected angles; simulations can provide quick and dirty refutation or support for very complicated conjectures. Mathematics, on the other hand, relentlessly simplifies and abstracts. While the simulation approach pushes for evidence of blow-up, and (typically) the mathematical approach pushes for evidence of regularity, one can get the feeling that these two approaches are in some sense “pushing past each other.” I believe that a much tighter integration between investigations of these two sides of the question has huge potential.

I have been looking into these issues for three years now, starting with a paper of Pelz which

generated wide interest by proposing a highly symmetric framework under which blow-up was conjectured to be possible [32]. Evidence for this was provided by the results of various simulations, starting with initial conditions chosen by hand from among the space of suitably symmetric possibilities within the proposed framework. As Pelz’s framework seems in some sense the best proposed mechanism for blow-up, I have investigated it from at least three different perspectives beyond the “pick initial conditions and simulate” presented in the original papers. While Pelz’s proposed mechanism for blow-up is uniquely simple, all attempts to simulate candidate initial conditions rapidly devolve into much more complicated behavior. My first approach, then, aimed at the *inverse problem*: searching for initial conditions that would provably maintain their simplicity throughout the trajectory of Pelz-style blowup. The computational demands of this type of investigation are almost the inverse of the classical approach: very little memory is needed because simplicity is preserved throughout; but computational demands are much higher, as, instead of simulating from a chosen initial condition, the entire space of possibilities must be searched. This setup is ideal for “graphics card supercomputing,” leveraging the unusual combination in consumer graphics cards of a mere gigabyte of RAM, paired with hundreds of processing cores and hundreds of gigabytes per second of internal bandwidth. The search space was parameterized by roughly 250,000 variables, representing fluid flow centered around an intended site of blow-up via the rational Chebyshev basis radially, and the spherical harmonic basis on each spherical shell; the search problem was set up as a minimization problem in a delicate way so that not only could I compute “distance from Pelz-style blowup” explicitly and efficiently, but further, I could compute each of the 250,000 components of the *gradient* of this quantity. Minimization was conducted using a low-memory version of the conjugate gradient method [25]. All the computationally-intensive portions of this were adapted to run via Nvidia’s CUDA system on a graphics card, including Fourier transforms [42], spherical harmonic manipulation, and conjugate gradient search.

This approach, and two recent more mathematical approaches I have taken to understand potential Pelz-style blowup have yielded evidence of increasing weight that such blow-up is not possible. As I am still trying to make sense of the intuitions gained here, nothing has appeared in print yet. However, the hope is that these novel reasons why Pelz’s proposal seems to fail might apply much more generally than just to ruling out this particular form of blow-up, that the same intuitions that rule out Pelz-style blowup might rule out blow-up in general.

## 4 Protein folding and evolution

The fantastic processes seen to occur in everyday biology have served to inspire vast amounts of science and engineering, and computer science is no exception. Alan Turing, when he was formulating the Turing machine, famously used his impressions of the human thought process as a basis for what would come to define the modern computer. It is thus no surprise that biology continues to be a font of inspiration about the amazing things that are possible, as well as a source of very hard problems whose solutions may have profound consequences. I will briefly discuss one example of each kind.

Protein folding is a famously hard problem, whose solution would potentially revolutionize medicine by enabling drastic improvements in our understanding of the biological processes that underpin the human body. Proteins are the workhorses of biology, a toolbox of 22 constituent parts (amino acids) strung together in arbitrary sequences described in DNA, which are then assembled into biochemical devices of almost endless functional variety. The sequence of amino acids that

comprises any given protein can be essentially read directly off the DNA; the challenge, however, is to understand the shape of the final 3-dimensional form that this sequence is assembled into—a machine is more than just a sequence of parts, it is an understanding of the parts plus how they fit and function together in their final assembled form.

The natural way to attempt to learn the structure of a protein is to observe it directly. Of such measurement techniques, the most successful is x-ray crystallography, whereby, if it is possible to crystallize a protein so that it occupies regular sites of a lattice, oriented identically, then the exact structure of the protein can be recovered by essentially doing a Fourier transform of the x-ray diffraction pattern. However, it is a great challenge to embed new proteins in a regular lattice, as proteins may change their shape in response to their environment, and a solid crystal lattice is far from the typical environment of most proteins.

It has long been hoped that computational tools will enable a unified way of predicting structure of a much wider class of proteins. Individual atoms at body temperature are well beyond the reach of direct observational techniques, in part because such atoms vibrate with a frequency on the order of  $10^{14}$  Hertz, however, such a frequency is no barrier to simulation by computers. Much work since the 1960s has worked to abstract out the behavior of atoms in proteins at the  $10^{-15}$  second time scale into a system of what is essentially “balls and springs” governed by Newtonian mechanics, with a few kilobytes of parameters describing, for example, the spring constants between all pairs of atoms that could appear bonded together in a protein.

The state of the art is thus: there are proteins with a few thousand atoms, pairs of which interact via simple Newtonian forces at a time scale of  $10^{-15}$  seconds; meanwhile we know that, left alone, such a protein will “fold up” into its final structure in a second (plus or minus several orders of magnitude depending on the protein); the challenge is to predict the final structure, *without* evaluating each pairwise interaction of  $10^4$  atoms at each of  $10^{15}$  time slices. In short, there is an obvious algorithm that takes time  $10^4 \cdot 10^4 \cdot 10^{15} = 10^{23}$  computational steps. As  $10^{23}$  computational steps is just beyond the edge of what is currently feasible, the challenge is to find an even slightly more efficient algorithm than the standard “compute everything at every time slice.”

There are several grounds for optimism. The most fundamental is the widely-held belief that in typical proteins, the final configuration is essentially the *minimum energy* configuration. Thus protein folding is an optimization problem which nature solves by a  $10^{15}$ -step random walk. But our algorithmic toolbox is large, and there are many optimization algorithms that have the potential to dramatically outperform blindly random walks. In particular, while the random walk approach needs to examine  $10^{15}$  states, any evidence of structure in the search space—of abstractions that characterize the shape of the search space—might be reasonably expected to enable a much more efficient traversal of the search space.

Dramatic evidence for this comes from a remarkable initiative to “crowdsource” protein folding. The *FoldIt* project has created a videogame-like interface by which non-experts from across the internet can attempt to rearrange components of a 3D protein model using their own intuitions, which are scored in real time by displaying to the user the energy of the current configuration, namely, the quantity to be optimized. The FoldIt project has made significant progress on several biologically relevant proteins [11, 20]. While the FoldIt platform is clearly very far from the level of success one would ultimately hope for protein structure prediction, it is remarkable, and enlightening, that they have been able to make any progress at all: one might imagine a user considering a new configuration every second, for several minutes or hours, thus using only a few hundred or thousand configurations to (in some cases) solve a problem that the standard approach needs

$10^{15}$  configurations to solve. This is very strong evidence of exploitable structure in the “protein landscape”.

What kinds of abstractions might enable such a substantial reduction in the search complexity? As it turns out, biologists already use an enticingly plausible abstraction to summarize the observed development of simulated protein folding (no protein has been simulated for  $10^{15}$  time steps, but simulations of  $10^9$  steps are becoming common, and biologically relevant). The amino acids that comprise proteins hold a dozen or two atoms, but biologists typically depict them via a single line segment, the “backbone.” Further, backbones are often found to form “sheets” or “helices,” and thus a protein with several thousand atoms may be abstracted down to, say, five helices and 2 sheets, in a biologically rigorous fashion. All this would be in vain, however, were it not for one additional crucial fact: when protein folding is simulated (or in the rare instances when snapshots of real protein folding can be observed in the lab), the structure of helices and sheets changes at a much slower rate than the configurations of individual atoms. These two facts in conjunction: that the “helices+sheets” abstraction essentially describes the state of a protein, and that the helix and sheet configurations evolve perhaps a million times slower than individual atoms, suggests that this could be a basis for a significant speedup of the protein folding enterprise.

The FoldIt project thus provides evidence that protein folding can be done in far fewer than  $10^{15}$  conceptual steps, and the biological abstractions of helices and sheets give a strong indication of the “concepts” that should be at play here. This seems to me a highly significant opportunity for algorithmic thinking to contribute to science.

## 4.1 Evolution

I want to end with a description of the most ambitious problem on my mind, one which at first glance is not the subject of computer science, but which is perhaps just starting to be recognized as being a fundamentally computational problem, and one which is fundamental to the development of computer science: evolution. I speak of biological evolution, that so-obviously-implausible claim that life, with all its ingenuity, and perfectly adapted solutions to complex problems, arose via purely mechanistic means.

Of course biologists have studied every aspect of evolution accessible to them, and each mystery they have unraveled has been found to have a perfectly ordinary solution—this protein mutated in its 17th amino acid, which made it interact differently with, etc. etc., which led to a slight fitness advantage because the creature is now less susceptible to this chemical which is newly present in its environment. But in some sense this is no different from being confronted with a machine-code implementation of a fantastic algorithm, and finding that sufficiently detailed investigation of any particular aspect of it yields nothing more complicated than “and then it adds the contents of registers 3 and 5, storing them in register 2, etc. etc.” Fundamentally, biologists are succeeding admirably at the problem of breaking down complicated biological phenomena into their constituent mechanistic pieces; the mystery, however, is an algorithmic one: how do mechanistic pieces assemble into a whole that serves to provide innovative solutions to complex problems as diverse as metabolism, vision, or the many problems whose solutions characterize “intelligence”?

In fact, it was the artificial intelligence question which first brought my attention to evolution. Artificial intelligence is such a challenging problem that one must hope to benefit as much as possible from the example of natural (biological) intelligence. The obvious approach of studying the intelligence of adult humans immediately reveals that it is a very complicated phenomenon and that, perhaps in order to understand the “finished product,” one must analyze the path by

which it got there. Machine learning theory thus seeks to understand, both with algorithms and lower bounds, the ways in which one can acquire knowledge. From the vantage point of this now relatively mature field, one can perhaps state with confidence that the incredible effectiveness with which humans (and many animals) learn from and perform in the world (as compared with, say, current computers) *cannot possibly* result from superior learning algorithms, but *must* result from a previous advantage: a huge platform of in-built knowledge. It was perhaps due to realizations along these lines that there has been a surge of effort in the last few decades to understand the mental capabilities of human infants. While the infant brain was once thought of as a “blank slate” capable of no meaningful interaction, an increasingly sensitive toolkit for investigating infant cognition is revealing to us a continually more impressive gamut of abilities, which appear so early in life that they must be, essentially, granted us by our DNA. And now one must ask how our DNA came to be, came to describe our (infant) brains, which take in experience from the world to yield intelligent adults. This ladder of explanations has thus led us back where we started: evolution is responsible for intelligence, must be investigated for the sake of understanding intelligence, and may in fact be the only *simple* component in the entire intelligence phenomenon.

“Simple” needs to be emphasized, for evolution must be simple. Not quite as simple as “survival of the fittest,” but simple nonetheless, for nothing fundamentally complex could have arisen by chance, expressed itself through the entire gamut of organisms across the four billion year history of life, and been responsible for solving all the enormously diverse challenges of life.

One clearly daunting barrier to a computational explanation of evolution is the vast computational resources which would be necessary to simulate evolution as it occurred on earth. But from another perspective, the individual numbers are high but not unapproachable: DNA length in the billions, life around for billions of generations, species populations in the billions, etc. One way of caricaturing what I have described is: nature blindly chose and ran an unknown algorithm for  $10^{50}$  computational steps that yielded amazing results beyond the understanding of current science, among which is all of biochemistry including the mechanisms for every disease along with every immune system developed for fighting them; the regenerative capabilities of each species capable of it; and the cognitive capabilities of every creature, including humans. The question is, can we, with the perhaps  $10^{25}$  computational resources readily available, abstract away the superfluous elements of nature’s approach to evolution, attain a “quadratic speedup” over nature’s  $10^{50}$  approach, yielding a computationally tractable analog of biological evolution that is capable of results of similar scope and power?

Motivated by the promise of a computational understanding of evolution, but at the concrete end of the spectrum, several recent projects have aimed at understanding fundamental aspects of biological evolution that have been phenomenologically evident from the beginning, but which have lacked any rigorous (computational) justification. I will briefly discuss two such efforts, one of which I have been involved with.

Christos Papdimitriou and Adi Livnat have taken aim at the phenomenon of sexual reproduction, and, more generally, the complicated and costly practice of nearly every living organism to combine its DNA with that of other organisms to produce offspring, instead of producing offspring directly [21, 22]. This practice runs counter to the notion of “survival of the fittest” since many of the ingredients for success embodied in the DNA of the fittest individual of the previous generation will be immediately wiped out when half of its DNA is replaced with that of another member of the population to produce the next generation. The surprising finding of this line of research is that genetic recombination is *not* a good solution for long-term or short-term fitness, but rather,

excels at a certain kind of “medium-term” fitness. They found that genetic recombination encourages “mixability,” that is, genes that do well in many different combinations. This seems like an important insight: that evolution uses genetic recombination as a means to insure against future changes in the environment—paying a constant small cost, to save from future calamity.

In another direction, Leslie Valiant created the model of *evolvability* [37] in an effort to rigorously investigate an aspect of evolution so basic its epithet is often taken to describe the entire process: “evolutionary hill climbing.” Namely: evolution is often thought of as an optimization algorithm, whereby the results of many small mutations are evaluated and combined to yield ever-fitter creatures, as measured against the potentially arbitrary “fitness landscape” that nature imposes. To what degree can the distributed “survival of the fittest” process perform as an effective optimization algorithm, and how much is it handicapped by the many apparent limitations of evolution as compared with standard optimization algorithms: no central control; no feedback on performance beyond survival-versus-death; no ability to pass (genetic) information save through the DNA of surviving creatures, while only “nature” controls which creatures survive, etc? Initial investigations were framed by the *probably approximately correct* (PAC) model [36], thus, “creatures” are presented instances of a binary classification problem drawn from some distribution, and are expected to evolve so that a high percentage of the creatures will with high probability correctly classify further instances drawn from the distribution. In the initial paper on evolvability, Leslie Valiant showed that evolvability is strictly weaker than the general PAC model, and, in fact, allows successful evolution only of those problems solvable via *statistical queries* (SQ), as defined by Michael Kearns [19]. In particular, learning arbitrary *parity* functions is beyond the reach of evolvability, as, intuitively, determining which subset of bits is being XOR’ed involves solving a matrix equation consisting of the examples seen, whereas “survival of the fittest” has no mechanism for collating survival data across a species and solving to yield DNA for the next generation. Subsequently, Vitaly Feldman showed that the class SQ is not just an upper bound on evolvability, but in fact characterizes a strengthened variant of the Boolean evolvability model [15].

In a forthcoming paper, I generalized the evolvability model from the Boolean domain to the real domain: functions take real inputs and yield real outputs [41]. This has clear advantages to modeling real biological phenomena such as the expression levels in protein circuits. Adapting the techniques from Vitaly Feldman’s paper ([15]) yields results of a rather different, but surprisingly natural form: if the expected fitness of creatures, as a function of its DNA, is a function that is optimizable given *approximate* access to the function, then it is also optimizable in the evolvability framework. Thus “evolutionary hill climbing” is in fact a general phenomenon, and any optimization algorithm that needs only approximate function evaluation can be recast as an evolution algorithm. This yields an evolution algorithm for linear and fixed-degree polynomial functions of great generality. It works for all distributions, and all convex loss functions.

Evolution would be impossible if results like these did not hold. And thus, in a small sense, these results achieve my general goal of advancing both computation and the other sciences by examining their fundamental problems through a computational lens.

## References

- [1] T. Abbott, D. Kane, and P. Valiant. On the complexity of two-player win-lose games. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 113–112, 2005.
- [2] J. Acharya, A. Orlitsky, and S. Pan. The maximum likelihood probability of unique-singleton, ternary, and length-7 patterns. In *IEEE Symp. on Information Theory*, 2009.
- [3] N. Alon, E. Fischer, I. Newman, and A. Shapira. A combinatorial characterization of the testable graph properties: It’s all about regularity. *SIAM Journal on Computing*, 39(1):143–167, 2009.
- [4] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. *Advances in Cryptology*, 2001.
- [5] C. H. Bennett. Time/space trade-offs for reversible computation. *SIAM Journal on Computing*, 18(4):766–776, 1989.
- [6] X. Chen and X. Deng. 3-Nash is PPAD-complete. *Electronic Colloquium on Computational Complexity*, TR05-134, 2005.
- [7] X. Chen and X. Deng. Settling the Complexity of 2-Player Nash-Equilibrium. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 2006.
- [8] X. Chen, X. Deng, and S.-H. Teng. Computing Nash equilibria: approximation and smoothed complexity. In *IEEE Symposium on Foundations of Computer Science*, 2006.
- [9] X. Chen, S.-H. Teng, and P. Valiant. The approximation complexity of win-lose games. *Symposium on Discrete Algorithms (SODA)*, 2007.
- [10] P. Constantin. On the Euler equations of incompressible fluids. *Bulletin of the American Mathematical Society*, 44(4):603–621, 2007.
- [11] S. Cooper, F. Khatib, A. Treuille, J. Barbero, J. Lee, M. Beenen, A. Leaver-Fay, D. Baker, Z. Popović, and Foldit players. Predicting protein structures with a multiplayer online game. *Nature*, 466:756–760, 2010.
- [12] C. Daskalakis, I. Diakonikolas, R. Servedio, G. Valiant, and P. Valiant. Testing  $k$ -modal distributions: optimal algorithms via reductions. Manuscript, 2011.
- [13] C. Daskalakis, P.W. Goldberg, and C.H. Papadimitriou. The Complexity of Computing a Nash Equilibrium. In *ACM Symposium on Theory of Computing (STOC)*, pages 71–78, 2006.
- [14] C. Daskalakis and C.H. Papadimitriou. Three-player games are hard. *Electronic Colloquium on Computational Complexity*, TR05-139.
- [15] V. Feldman. Robustness of Evolvability. *Conference on Learning Theory (COLT)*, 2009.
- [16] C. Gentry. Fully homomorphic encryption using ideal lattices. *ACM Symposium on Theory of Computing (STOC)*, 2009.
- [17] I. Gilboa and E. Zemel. Nash and correlated equilibria: Some complexity considerations. *Games and Economic Behavior*, 1989.
- [18] F. Götze. On the rate of convergence in the multivariate CLT. *Annals of Probability*, 19(2):724–739, 1991.
- [19] M. Kearns. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM*, 25(6):983–1006, 1998.
- [20] F. Khatib, F. DiMaio, Foldit Contenders Group, Foldit Void Crushers Group, S. Cooper, M. Kazmierczyk, M. Gilski, S. Krzywda, H. Zábranská, I. Pichová, J. Thompson, Z. Popović, M. Jaskolski and

- D. Baker. Crystal structure of a monomeric retroviral protease solved by protein folding game players. *Nature Structural and Molecular Biology*, 18:1175–1177, 2011.
- [21] A. Livnat and C. Papadimitriou. A mixability theory for the role of sex in evolution. *Proceedings of the National Academy of Sciences*, 105:19803–19808, 2008.
- [22] A. Livnat and C. Papadimitriou. Sex, mixability, and modularity. *Proceedings of the National Academy of Sciences*, 107:1452–1457, 2010.
- [23] M. Luby and C. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, 17(2):373–386, 1988.
- [24] S. Micali. Computationally Sound Proofs. *SIAM Journal on Computing* 30(4), pp. 1253–1298, 2000.
- [25] J. Nocedal. Updating Quasi-Newton Matrices with Limited Storage. *Mathematics of Computation*, 35:773–782, 1980.
- [26] A. Orlitsky, N. Santhanam, K. Viswanathan, and J. Zhang. On modeling profiles instead of values. *Uncertainty in Artificial Intelligence*, 2004.
- [27] A. Orlitsky, N.P. Santhanam, and J. Zhang. Always Good Turing: Asymptotically optimal probability estimation. *IEEE Symposium on Foundations of Computer Science (FOCS)*, 2003.
- [28] A. Orlitsky, N.P. Santhanam, and J. Zhang. Always Good Turing: Asymptotically optimal probability estimation. *Science*, 302(5644):427–431, October 2003.
- [29] L. Paninski. Estimation of entropy and mutual information. *Neural Comput.*, 15(6):1191–1253, 2003.
- [30] L. Paninski. Estimating entropy on  $m$  bins given fewer than  $m$  samples. *IEEE Trans. on Information Theory*, 50(9):2200–2203, 2004.
- [31] C.H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, pages 498–532, 1994.
- [32] R. B. Pelz. Locally self-similar, finite-time collapse in a high-symmetry vortex filament model. *Physical Review E*, 55(2):1617–1626, 1997.
- [33] S. Raskhodnikova, D. Ron, A. Shpilka, and A. Smith. Strong lower bounds for approximating distribution support size and the distinct elements problem. *SIAM Journal on Computing* 39(3): 813–842 (2009).
- [34] G. Valiant and P. Valiant, “Estimating the unseen: An  $n/\log(n)$ -sample estimator for entropy and support size, shown optimal via new CLTs,” *ACM Symposium on Theory of Computing (STOC)*, 2011.
- [35] G. Valiant and P. Valiant. The power of linear estimators. *IEEE Symposium on Foundations of Computer Science (FOCS)*, 2011.
- [36] L. Valiant. A theory of the learnable. *Communications of the ACM*, 27, 1984.
- [37] L. Valiant. Evolvability. *Journal of the ACM*, 56(1), 2009.
- [38] P. Valiant. “Testing symmetric properties of distributions.” MIT PhD thesis, June 2008.
- [39] P. Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. *Theory of Cryptography Conference*, 2008.
- [40] P. Valiant. Testing symmetric properties of distributions. *SIAM Journal on Computing*, special issue for *STOC 2008*, to appear.
- [41] P. Valiant. Distribution free evolvability of polynomial functions over all convex loss functions. *3rd Conference on Innovations in Theoretical Computer Science*, 2012.
- [42] V. Volkov and B. Kazian. “Fitting FFT onto the G80 architecture.” Available at [http://www.cs.berkeley.edu/~kubitron/courses/cs258-S08/projects/reports/project6\\_report.pdf](http://www.cs.berkeley.edu/~kubitron/courses/cs258-S08/projects/reports/project6_report.pdf).