Electronic Colloquium on Computational Complexity, Report No. 72 (2018)

On the nature of the Theory of Computation (ToC)*

Avi Wigderson

April 19, 2018

Abstract

[This paper is a (self contained) chapter in a new book on computational complexity theory, called *Mathematics and Computation*, available at https://www.math.ias.edu/avi/book].

I attempt to give here a panoramic view of the Theory of Computation, that demonstrates its place as a revolutionary, disruptive science, and as a central, independent intellectual discipline. I discuss many aspects of the field, mainly academic but also cultural and social. The paper details of the rapid expansion of interactions ToC has with all sciences, mathematics and philosophy. I try to articulate how these connections naturally emanate from the intrinsic investigation of the notion of computation itself, and from the methodology of the field. These interactions follow the other, fundamental role that ToC played, and continues to play in the amazing developments of computer technology. I discuss some of the main intellectual goals and challenges of the field, which, together with the ubiquity of computation across human inquiry makes its study and understanding in the future at least as exciting and important as in the past.

This fantastic growth in the scope of ToC brings significant challenges regarding its internal organization, facilitating future interactions between its diverse parts and maintaining cohesion of the field around its core mission of understanding computation. Deliberate and thoughtful adaptation and growth of the field, and of the infrastructure of its community are surely required and such discussions are indeed underway. To help this discussion I review the history, culture and evolution of the field. I list what I believe are the essential characteristics which allowed ToC to so well embrace and integrate so many external influences, continuously expanding its core mission, and to create one of the greatest success stories in the history of science. I feel that preserving many of these characteristics, primarily its independence, and that educating ToC professionals (as well as other scientists, engineers, and, at appropriate levels, children and the general public too) in these achievements and challenges, are essential for a similarly spectacular future.

*Some prefer the name Theoretical Computer Science (TCS) for this field.

ISSN 1433-8092

Contents

1 Introduction

The great impact of computer technology on our society tends to hide and obscure the intellectual foundations it rests on, especially from laymen but sometimes from experts as well[1]. Even when understood, these theoretical ideas are often viewed only as servants of the technological development. In this survey, I plan to focus and elaborate on the nature of ToC as an independent intellectual discipline, which is summarized below.

The Theory of Computation is as revolutionary, fundamental and beautiful as major theories of mathematics, physics, biology, economics... that are regularly hailed as such. Its impact has been similarly staggering. The mysteries still baffling ToC are as challenging as those left open in other fields. And quite uniquely, the theory of computation is central to most other sciences.

In creating the theoretical foundations of computing systems ToC has already played, and continues to play a major part in one of the greatest scientific and technological revolutions in human history. But the intrinsic study of computation transcends man-made artifacts. ToC has already established itself as an important mathematical discipline, with growing connections to nearly all mathematical areas. And its expanding connections and interactions with all sciences, naturally integrating computational modeling, algorithms and complexity into theories of nature and society, marks the beginning of another scientific revolution!

The sections below discuss at a high level various aspects, mainly intellectual but also social and educational, of the Theory of Computation. This exposition aims at describing the nature and scope of the field, its evolution so far and the important role I expect it to play across the intellectual and societal arenas. I hope that collecting them here will give a useful bird's-eye view of the field to newcomers and to motivated outsiders, but may be useful also to ToC researchers. *This exposition is clearly biased by my personal experience, views and limited knowledge.* Also, it is not uniform across topics.

The first several sections discuss the connections and interactions between ToC with many disciplines. I first discuss, rather briefly, its interaction with its "parent" fields, Computer Science and Engineering (CS&E), and Mathematics. I then move on to describe in more detail interactions with "neighboring" fields like optimization and coding theory. But the bulk of this paper is devoted to the interactions of ToC with many sciences, with philosophy, and technology. Before turning to the connections with more remote sciences, like biology and economics, I make two important detours in stand-alone sections. The first, in Section 3 will discuss a general definition of computation, which will clarify the broad mission of ToC and how it is naturally aligned with the basic missions of all sciences. The second, in Section 4, will give a general description of the central tenets of ToC methodology, which will clarify what new aspects ToC can bring into these joint collaborations with the sciences. After discussing connections and interactions, I will turn to some high level challenges of ToC, and then briefly discuss the role and importance of ToC research and the impact researchers can have in K-12 education. I will conclude with some personal observations on the socio-academic character of the field, which I believe partly explains its remarkable success so far, and should be preserved for future success.

Taking stock of of a scientific field's history, evolution, ideas, contributions and challenges, is very familiar in fields that existed for centuries, and numerous semi-popular books were written on mathematics, physics, biology and others from this viewpoint. ToC, counting from Turing's paper, has existed for 80 years (of which I have personally witnessed the last 40). I believe that there is clearly room for such a book on ToC, which will describe and argue in more detail its achievements, goals and its independent intellectual place in the academic sphere. Moreover, I believe that such education of members of the field, academia and the public will be important for its future development. Here, I have condensed my version of such a treatment of the field into one paper[2]. Besides being short, this survey is only a time capsule. I fully expect that the enormous expansion rate of the role computation plays in every aspect of our lives will likely change and expand the scope, goals and mission of ToC. As computation is ubiquitous, the potential for such expansion

[1] The same is true for past technological revolutions as well.

[2] Twenty years ago, Oded Goldreich and I gave a similar treatment [GW96] of the nature and future of the field, with the exact same conclusions of its success, promise and independence, but one that was much shorter and had far less detail. The two decades that passed, and all the additional research ToC has created since, make it easy for me to add on much more supporting evidence to these claims.

is unlimited.

Throughout this paper, ToC research will refer to the pursuit of understanding computation in all its forms rather than to the disciplinary affiliation of the researchers who perform it; while these researchers are typically affiliated with ToC, notable contributions of this nature were also made by applied computer scientists, mathematicians, and scientists of all disciplines.

Finally, while self-contained, this survey was written as a chapter in the book [Wig17], and so is biased against detail on material present in that book (which I will reference occasionally – being online it is readily accessible).

2 Close collaborations and interactions

It is hard to do justice in a few pages to the breadth and variety of interactions of ToC with its "nearby" disciplines, all much larger than it. I do not attempt to fully exposit this immense body of work. Rather I try to capture the different nature and role these interactions take from each different discipline, and how in turn they shape and reshape ToC and its reach. I start from the closest disciplines, Math and CS&E (Computer Science and Engineering) and then continue into neighboring fields. Both here, and when we turn to interactions with more distant fields, the coverage and level of detail are not uniform.

Meet the parents: CS&E vs. Mathematics As discussed in the introduction, ToC was born out of, and lived within, two extremely different traditions, which shaped its unique character. I consider this combination of genes and environments extremely fortunate! Here are some examples. CS&E is a young and impatient field, while math is old and patient. Research directions in CS&E are mostly driven by practice and utility, while in math they are mostly driven by æsthetics and abstraction. CS&E demands the study of real, functioning computer systems, while math demands any study to be fully rigorous, but allows imagination to run free without regard to reality. Each alone and both together generate an unending host of modeling and algorithmic challenges for ToC. I will be rather brief on both.

2.1 Computer Science and Engineering

The immense technological achievements of the computer industry which surround us blind many, not only laymen and politicians but scientists as well, to the pure intellectual wonders which brought many of them about. ToC, besides creating the many theories underlying the technologies I mention below, has laid and continues to build the conceptual infrastructure for all computer professionals and the problems they address. Most fundamentally, this includes the ability to precisely model diverse computational settings, to formally define resources and analyze their utilization when organizing and processing information by algorithms. ToC provided drastically new and useful ways to account accessibility to knowledge (and thus assess privacy and secrecy), to the power and limits of randomness and interaction, to the organization, utilization and analysis of systems with many components, and to many others computational settings, some covered in detail in this book. These conceptual contributions, some of revolutionary consequences, have become ways of thought so ingrained in basic undergraduate education that they are often taken for granted.

The specific contributions of theory to CS&E activities are very well documented, so I will be telegraphic here, confining myself to name-dropping. ToC has created and is still developing foundational theories which underlie much of the development of computational systems of all kinds. In many cases theory predated practice, and enabled system development[3]. In many cases, theoretical ideas born and developed for a particular technology remained useful after that technology became obsolete[4].

[3]Examples abound, and we give only two primary ones. The early theoretical blueprints of computing machines of Turing and von Neumann which unleashed the digital age is the first. And the theoretical foundations of public-key cryptography without which enabled E-commerce and so enabled the Internet is the second.

[4]Again, examples abound, and we give two representatives of two phenomena. First, theories: the theory of communication complexity, designed primarily to understand area-time trade-offs in VLSI chips, and has gone on to be key in numerous other areas, as we discuss in Chapter **??**. Second, algorithms: the algorithm for pattern matching, designed initially for text processing, was, together with its extensions absolutely essential for the Genome project.

Here is a partial list of research and application areas, each member of which is deep, broad and occupies volumes of *theory* textbooks (which one should study to fully appreciate the underlying theoretical foundations of each). It all starts in 1936 with Turing's definition of computation and algorithms, and proceeds with theories of cellular automata, finite and infinite automata, programming languages, system verification, databases, computational complexity, data structures, combinatorial algorithms, numerical algorithms, cryptography, distributed systems, software engineering, randomness and pseudo-randomness, computational learning, approximation algorithms, parallel computation, networks, quantum computation, online and real-time algorithms, and many more. Needless to say, the rapidly evolving world of computer science and technology continuously supplies many varied situations to model as well as problems to solve, such as the recent sub-linear algorithms for huge datasets, differential privacy for scientific and public databases, delegation in cloud computing, population protocols in sensor networks and of course the struggle for theoretical explanations of machine learning programs like deep networks. Future technologies and applications will undoubtedly lead to new theories, more interaction and better understanding of the nature, applicability, power and limits of computation.

2.2 Mathematics

With math, the nature of interactions developed differently. Again I will be brief, as the book [Wig17] has many examples throughout its chapters, with its *Interlude* chapter specifically devoted to a select few of them. Early on, aside from the early intimate collaborations with logic, ToC was mostly a user of mathematical techniques and results. Dealing mainly with discrete objects, it was natural that most initial interactions were with combinatorics. But as ToC broadened and deepened, its varied theories required tools from diverse mathematical fields, including topology, geometry, algebra, analysis, number theory, algebraic geometry and others. Such tools were often not readily available, or did not even exist, and so had to be developed. This formed many collaborations which led to purely mathematical results in all these areas. Another source of new questions (and the need for further tools) comes from completely new mathematical models for different computational phenomena, as some of the chapters of the book [Wig17] illustrate. These activities were combined with the meta-challenge of making math algorithmic; namely, *efficiently* finding objects whose existence was proved by some indirect or non-explicit arguments. Such questions pervade mathematics, in some cases for centuries. Pursuing them, especially with the newly developed algorithmic and complexity tools, has led to rethinking and expanding many areas, discovering new structures and providing many new questions. These rapidly expanding collaborations with many areas of mathematics has greatly enriched ToC as well, and clarified its own independent role as an important mathematical area. It is heartwarming to see how most young mathematicians are well versed in the basic notions, results, and major open problems of computational complexity, as they would be about any other math area outside their speciality. I have no doubt that the algorithmic and complexity ways of thinking will further penetrate all areas of math and will create many new interactions between other fields.

Meet some neighbors: Optimization, Coding & Information Theory, Statistical Physics I now turn to give some more details and references on the many and growing interactions between ToC and three very large disciplines which are naturally quite close to computer science. These connections are already surprisingly diverse, and in some cases have surprising origins. I find this array of connections stunning in breadth and depth, and note again that the topic list and references we give are very partial.

2.3 Optimization

The connections of the large optimization community[5] with ToC is far from surprising, as efficient algorithms are a core object studied by both (even though initially the types of problems and tools each community focused on were somewhat different). However, what may not have been expected is how major break-

[5]Which includes subareas of Statistics, Operations Research, Control Theory and other fields.

throughs, which mostly arose from *computational complexity* considerations and constructs, would enrich and rejuvenate the study of the power and limits of algorithms. These include:

- The PCP theorem, leading to a theory of hardness of approximation, analogous to (though far more difficult and refined than) the theory of $\mathcal{NP}$-completeness. Some landmarks include [AS98, ALM+98, FGL+96, Kho02, Rag08]. It is interesting to note that the origins and methods leading to the PCP theorem are far from optimization: they include cryptography, interactive proofs, coding theory, program testing and average-case complexity.

- The power, limits and connections of LP and SDP hierarchies, powerful algorithmic paradigms existing in the optimization literature, which were (and are) becoming much clearer with works like [Gri01b, Gri01a, ABL02, LRS14, BS14, CLRS16]. Some of the origins and connections leading to these developments include proof complexity, computational learning theory and random restrictions from circuit complexity.

- Related and more specific to the item above is "extension" of linear programs, namely adding variables to reduce the number of inequalities. The exact power of this technique was fully characterized in [Yan91], and its limits determined first in [FMP+15] (and then further in subsequent work, also mentioned in the previous item). Interestingly and surprisingly, some of the crucial insights and methods came from seemingly unrelated areas including communication complexity and quantum information theory.

- Starting perhaps with Karmarkar's algorithm [Kar84] for linear programming, continuous optimization methods became a mainstay of ToC. But more recently momentum picked up considerably, and the ellipsoid method, interior point method, alternate minimization, multiplicative weights and a variety of first and second order descent methods (as well as their applications), have been greatly extended e.g. in [ST04a, CKM+11, AHK12, Mad13, LS13, LS14, KLOS14, AZO14, GGOW15, AZH16] among many others. In many cases these broke efficiency records held for decades and some resulted in near linear algorithms. Connections and tools range from electrical flows and random walks to permanent approximation, spectral decompositions and functional analysis.

- The Exponential Time Hypothesis (ETH) has been proposed as a natural but much stronger hardness assumption than $\mathcal{P} \neq \mathcal{NP}$ in [IPZ01, IP01]. The ensuing active theory, often called *fine-grained complexity*, (as it introduces also more delicate notions of reductions between optimization problems) enables predicting the precise complexity of problems inside the class $\mathcal{P}$, and in particular ruling out linear time algorithms (see e.g. the survey [Wil15]).

- Smoothed analysis, introduced in [ST04b], provides a radically new way of analyzing heuristics, very different from all previous average-case models, and serves to explain their success on "typical" inputs in a new way.

2.4 Coding and Information Theory

The connections of ToC with the vast field of coding and information theory are extremely broad, ranging from very expected to very unexpected. Most expected connections have to do with the fact that computations (in computers, networks of computers, databases etc.) process information which is subject to noise and error, and so must be designed to be fault-tolerant. This naturally begs the use of error-correcting codes and information theory. The less expected connections include the introduction of *locality* into error-correcting codes (in several ways), the rejuvenation of several old ideas and models (like list-decoding, low-density parity-check codes, and the role of interaction in coding and information theory), with new questions, applications, techniques and results. We list and reference some of these below (the book [Wig17] devotes a special chapter to the interactive aspect of this collaboration).

- The idea of list-decoding, namely that codes can tolerate far more noise if decoding is relaxed to produce a short list of potential messages, as opposed to a unique one, goes back to the very early days of coding theory [Eli57]. Forty years later, starting with the breakthrough work of Sudan [Sud97], a sequence of works including [GS98, PV05, GR08] quickly led to optimal list-decodable codes with efficient encoding and decoding algorithms, and the study of list decoding for many different codes.

- In contrast, the idea of studying *local decoding* arose from complexity theoretic considerations in cryptography, derandomization and PCPs. In local decoding only one (specified) bit of the original message should be recovered from a noisy encoding of it, but only a tiny (random) fraction of it can be inspected, and small error is allowed. Remarkable constructions and applications appear e.g. in [GL89, STV99, KS09, Y+12, Efr12, DSW14, DG16] and in many others. As one example of an application, the related *locally repairable codes* of [GHSY12] and its followers had a tremendous effect on the design of distributed data centers of Internet data, where recovery from fault and consistency across multiple copies became a completely new challenge at these volumes and speeds. Some of the many complexity theoretic applications of local decoding (often combined with list decoding) can be found in the survey [Sud00], and many others, in algebraic complexity, combinatorial geometry and information retrieval appear in some of these references.

- *Local testing* of codes is another new coding problem which was born from computational complexity considerations of program testing, interactive proofs and PCPs was. Here one simply asks if a given word is in a given code or very far from it (in Hamming distance), again by inspecting only a (random) tiny part of it. Once more, remarkable constructions and applications were found, e.g. in [BFL91, BLR93, AS03, RS97, GS00, GS06, IKW12, KMRZS17, DK17].

- In the reverse direction, the idea of *concatenated codes* [For66], developed in coding theory to construct explicit, efficient codes, inspired many of the proof composition constructions of PCPs and related systems. Other notions of robustness as well as actual constructions were borrowed, directly and indirectly, from the recovery from errors in robust codes to recovery from errors in robust proofs.

- "Graph-based" or LDPC (Low-Density Parity-Check) codes were first proposed and studied in the seminal work [Gal62]. However, this direction was left mostly dormant until a surge of interest and results brought it back 30 years later, much due to the rise of *expanders*, graphs with diverse applications in ToC. These initial works include [Spi95, SS96, LMSS01, RU01, Lub02, CRVW02].

- Interactive computation was always part of information theory. Nevertheless, communication/information complexity, and interactive coding theory have emerged and grew to two well-developed theories *within* computational complexity, with major applications within the field. These theories have considerably enriched the interactions with coding and information theory.

2.5 Statistical Physics

It may seem surprising that I include statistical physics a neighboring field to ToC, but many natural connections and common interests between the two were discovered very early on, and have led to significant collaborations (that naturally included also discrete probability and combinatorics). Moreover, many of the problems and methods of statistical physics are naturally algorithmic as we shall presently see. I will briefly survey this common ground, so as to better appreciate the new interactions (making this section somewhat longer).

A central theme of statistical physics is understanding how global properties of a large system arise from local interactions between its parts. A commonly given example is the *global* states of matter: gas, liquid, solid, and the transition between them (i.e. turning ice into water and then into vapor) under temperature change, which affects the movement of molecules and the *local* interactions between neighboring ones. Another common example is magnetization under the influence of an electric field. A system is typically described by a graph or hypergraph whose nodes represent the interacting parts, each of which can be in some

finite number of states, and whose the edges represent the local structure. Prescribed functions determine for each local part its contribution to the total *energy* of the system. For each possible state q of the system – where q is a vector of states of every part – one then adds the local contributions to define the energy of the whole state $E(q)$. Being a *statistical* theory, the state of a system is a random variable q drawn from a probability distribution, which is determined by the total energy, usually proportional to $\exp(-\beta E(q))$. The proportionality constant $Z = \sum_q \exp(-\beta E(q))$ is called the *partition function* of the system; a central object of study in this theory. Here β is a global parameter (like temperature) which controls the strength of local interactions. Determining global properties of the system (mean energy, long-range correlations and others, many encoded in the partition function) is generally reduced to sampling a state according to the above *Gibbs distribution* as it is commonly called. This sampling problem is a natural computational problem! Note however that the number of states is exponential in the size of the system (the number of parts), so it is nontrivial.

This (discrete) formalism captures a large number of models of matter of all types, including spin systems on glassy and granular matter, electromagnetic systems (both classical and quantum), "hard-core" interactions, percolation in porous materials, and others. Many other variants we will not discuss include continuous settings like heat baths, Brownian motion and other diffusive systems, and non-linear interactions, e.g. the Maxwell-Bolzman ideal gas model (which gave birth to statistical physics), billiard models etc.

It is not hard to see that the discrete formalism above naturally captures and indeed generalizes *constraint satisfaction problems* (CSPs) like e.g. $3-SAT$, which I discussed in several sections of the book [Wig17]. Description of such systems (the interaction graph and the local energy functions) comprise the input data to many optimization problems of algorithmic interest, except that typically one asks not to sample a random state, but rather to find one that (exactly or approximately) optimizes the energy. Similarly, in combinatorics, the same data is considered as input to many enumeration problems, i.e. counting the number of states with optimal (or specific) energy. It turns out that all these computational problems, *sample, search, count*, are related, which naturally fosters interaction that we will discuss.

A sampling method originating with von Neumann and Ulam in the 1950s, developed partly for the Manhattan project, is the so-called *Monte Carlo* algorithm, which was subsequently further developed. In the framework above, one sets up an (exponentially large, implicitly described) *Markov Chain* on the states of a system, whose stationary distribution is the Gibbs distribution (this is called the *MCMC method*, for Markov Chain Monte Carlo). Then, starting from an arbitrary state, one proceeds with the natural random walk on the chain *in the hope* of converging quickly to a typical state as required. Natural algorithms of this type include the Metropolis algorithm and Glauber dynamics. Numerous simulations of such chains were and are carried out to determine properties of numerous models. However, few tools existed, for precious few systems, that could rigorously quantify the convergence rate of these algorithms, and so heuristic arguments (and resource limitations) were used to cut off the simulation time. Needless to say, if the random walk did not converge, the information deduced from about the global state of the system could be completely wrong. And of course, MCMC may not be the only way to sample! Interaction with ToC started when the field began investigating *probabilistic algorithms* in the 1970s, of which MCMC is a perfect example. This has led to significant progress on the sets of problems described above regarding local interacting systems, which we briefly summarize.

- Valiant's complexity theory of counting problems [Val79a, Val79b], introduces the complexity class $\#\mathcal{P}$, and establishes that Permanent of Boolean matrices is complete for this class. This allows us to demonstrate hardness of almost all natural enumeration problems above, and with them, the hardness of computing the probabilities in the associated Gibbs sampling problems (indeed, dichotomies distinguishing hard and easy counting were established - see the comprehensive book [CC17]).

- To sweeten the pill, Jerrum, Valiant and Vazirani [JVV86] prove that *sampling* is equivalent, for most problems of interest, to *approximate counting*. This provided a key connection that makes sampling algorithms applicable for solving enumeration problems.

- A series of papers by Jerrum and Sinclair [JS89, SJ89] provided general methods of *canonical paths* and *conductance* to bound the convergence of general Markov chains, which they used to rigorously

prove polynomial convergence for problems in statistical physics like the Ising and monomer-dimer models, and enumeration like counting matchings in graphs. These papers had a flood of follow-ups, developing these and other methods, including rigorous ways of using various forms of coupling. Many early examples are summarized in [JS96].

- The important work of Jerrum, Sinclair and Vigoda [JSV04] gave a polynomial time probabilistic algorithm to approximate the permanent of non-negative matrices, which by completeness (above) captures many other enumerations problems. Efficient *deterministic* algorithms for the permanent, albeit with only an exponential-factor precision [LSW00, GS14], exposed connections of these problems to matrix scaling and hyperbolic polynomials. These aspects and many more are exposed in Barvinok's book "Combinatorics and complexity of partition functions" [Bar16].

- The connection between *spatial*, structural properties of local systems (e.g. long-range correlations in the Gibbs distribution, phase transition) and *temporal*, complexity theoretic properties (e.g. convergence time of natural Markov chains like Glauber dynamics) has been studied by physicists in spin systems since the 1970s. This connection was expanded by the work Weitz' [Wei06] to the *hard-core* model; he developed a *deterministic* algorithm (very different from the Markov chain approach) which is efficient up to the phase transition. This was complemented with a a hardness result of Sly [Sly10] just above the phase transition. This triggered further collaboration and better understanding of this deep relationship between spatial and temporal mixing (see [DSVW04] for a survey).

- The *Lovasz Local Lemma* (LLL) enables us to establish the *existence* of rare "global" events. Efficient algorithmic versions of the LLL were initiated by Beck [Bec91], and starting with the work of Moser [Mos09] (and then [MT10]), have led to approximate counting and uniform sampling versions for rare events (see e.g. [GJL16]). These new techniques for analyzing *directed, non-reversible* Markov chains are a new powerful tool for many more applications. A completely different deterministic algorithm of Moitra [Moi16] in the LLL regime promises many more applications; it works even when the solution space (and hence the natural Markov chain) is not connected!

- Finally, Markov chains like the Metropolis algorithm, guided by an optimization objective like energy, have been used for optimization in heuristics like *simulated annealing*, to generate Gibbs distributions that favor states of high objective value. Some of the techniques above allow analysis of convergence time for classical specific optimization problems (see e.g. [JS93] for upper bounds and [Jer92] for lower bounds).

3 What is computation?

As we now move away from interactions of ToC with neighboring fields to (seemingly) more remote ones, we take a higher view. It is fitting at this point that we should explain first, as we discuss the theory of computation, what we mean by the term *computation*. One of the broadest ways to informally "define" computation, indeed the view which underlies the celebrated *Church-Turing Thesis* (which is discussed more later) is as follows. *Computation is the evolution process of some environment via a sequence of "simple, local" steps*. If this definition seems to you as capturing practically any natural process you know of, that's precisely how I want it to sound!

Of course, the definition above calls for a specification of the evolving environment, and a notion of granularity that will distinguish local and global, simple and complex. The most basic setting (from which the word "computation" originally arises), in which bits evolve in a Turing machine or Boolean circuit, offers one example of the granularity notion, and the rules/steps of evolution. Another, still with actual computation, but completely different granularity and basic steps, arises as we can consider the evolution of the states of processors in a network under, say, pairwise communication. And there are many other choices which capture other (existing or imagined) computing systems.

The main point I wish to make is that this viewpoint of processes as computation extends to numerous other settings, vastly removed from computers. In each setting, many different choices of granularity, and

simple and local rules (which may be given by nature, or made up by us), will lead to different evolution processes. All of these are worth studying (even when physical) as *information processes* from a computational perspective, using the methodology of ToC that we elaborate on in the next section.

Here is a partial list of environments with such interacting parts, which in all cases can shed their physical characteristics and be viewed as transformations of pure information. All of these are playgrounds where theorists of computation[6] have a role to play! The computational modeling and quantitative study of the resources expanded by *all* such processes, what they "compute" and other properties of this evolution, is the bread and butter of ToC.

- Bits in a computer.
- Computers in a network.
- Atoms in matter.
- Neurons in the brain.
- Proteins in a cell.
- Cells in a tissue.
- Bacteria in a Petri dish.
- Deductions in proof systems.
- Prices in a market.
- Individuals in a population.
- Stars in galaxies.
- Friends on Facebook.
- Qubits in entangled states.

These and many other similar examples clarify that the notion of computation far transcends its (natural and essential) relevance to computer technology, and demonstrate the need for a theory of computing even if computers did not exist at all!

4 ToC methodology

As we have seen above, many natural and man-made phenomena present us with processes we would like to understand, and the many problems of practical and intellectual importance present us with the need to develop efficient ways to solve them. The theory of computation has created a powerful methodology and language with which to investigate questions of this type. Here are some of its (interrelated) important principles, which are demonstrated repeatedly book [Wig17], and should be considered in the general context of computation we discussed above. Let me stress that most of these principles are not new; they have been in use in many studies across science and mathematics for ages. I feel however that they are handled in a more systematic way in the context of the theory of computation, and in a sense, some of these principles themselves become objects of study of the field. I expect that their more systematic use in math and other sciences would be rewarding, as existing interactions (some of which are discussed in the following sections) already reveal.

[6] And these can come from every discipline.

1. **Computational modeling**: *Uncover and formally articulate the underlying basic operations, information flow and resources of given processes*. The book [Wig17] has many examples of computational processes with a large variety of basic operations, like Boolean or arithmetic gates, which can be deterministic, randomized or quantum. We have seen e.g. geometric, algebraic and logical deductions in proofs. In all of them we have studied time, space, communication, randomness and other resources. Beyond CS and math lie a vast number of natural processes expanding different resources, many of which can be viewed as information processes. Modeling their basic steps and resources *as* computation, and applying the computational language, methodology and results may be extremely beneficial in the sciences. Moreover, the abstract, computational understanding of natural processes may feed back into computer technology by integrating algorithms and hardware used by nature, as initial attempts in nano computing, quantum computing, DNA computing, swarm computing and others promise.

2. **Algorithmic Efficiency**: *Attempt to minimize relevant resources used by computational processes and study their trade-offs.* It should be stressed at the outset this principle applies equally to algorithms designed by a human to solve a problem, or by deep networks trained on massive data to self-improve, or by algorithms which evolved in nature over eons to guide the behavior of living organisms. In all, economizing resources is primary (even inanimate physical objects seem to prefer low energy states). Experience shows that studying the limits and trade-offs of efficiency is a great classification guide for problems and processes. Moreover, developing general analytic tools to find the limits and trade-offs of efficiency in one computational setting can be extremely powerful in others.

3. **Asymptotic thinking**: *Study problems on larger and larger objects, as structure often reveals itself in the limit*. There is a natural, practical tendency to focus on understanding (or developing algorithms for) concrete, specific objects which we really care about, which have specific sizes (e.g. the human genome, the Facebook graph, the roadmap of the US, Rubik's cube, the proof of Fermat's last theorem, etc...). However, viewing such concrete objects as parts of infinite families, to which the same process or algorithm applies, may lead to more efficient and more general methods of treating even these particular individual ones. Some of the major successes of the field stem from applying the asymptotic viewpoint, and numerous algorithms, reductions and complexity classes demonstrate this clearly. This approach has parallels in other fields. In coding theory, Shannon's asymptotic approach revolutionized digital communication and storage (despite the fact that all technological applications have very specific finite parameters). In physics this approach is often called the *thermodynamic limit*. In mathematics the asymptotic view is natural in discrete settings like set systems and graphs (although some parameters of continuous structures are viewed asymptotically, like dimension, genus, and continuity itself). Making discrete structures continuous and studying various limits uncovers hidden structure as well; perhaps more of that can be applied to the study of computation. The recent development of a limit theories of combinatorial objects (see e.g. the comprehensive book [Lov12]) is a promising example.

4. **Adversarial thinking**: *Prepare for the worst, replacing specific and structural restrictions and constraints by general, adversarial ones—more stringent demands often make things simpler to understand!* A recurring theme across the theory of computation is that surprising algorithms and protocols are discovered when the models at hand allow stronger (rather than weaker) adversaries[7]. While seemingly counterintuitive, such a handicap (or worst-case view) often shines a light on an idea that may be obscured by specific details. And clearly, positive results, namely upper bounds and algorithms, if they exist in such generality, are much preferable. Of course, equally often lower bounds are found in general adversarial settings; such negative results call for formulating more specific assumptions, under which the problem we care about does have a solution. But even then, guidance in modeling on how to restrict adversaries and avoid hardness can often be gleaned from understanding how and why algorithms fail in the presence of more general adversaries. The theory of cryptography in particular, where adversaries are typically restricted solely in their computational power, has been extremely successful due to this very choice, which perfectly fits computational complexity theory.

[7] These can be inputs, distributions on inputs, schedulers, noise, eavesdroppers, etc., depending on the context.

5. **Classification**: *Organize computational tasks into (complexity) classes according to the amounts of various resources they require in various models*. Classification is natural across the sciences, but it is most often based on structural properties of objects. What is seemingly surprising in computational complexity is how a huge number and variety of problems snugly fits into relatively few complexity classes characterized by resource requirements, as well as the strong connections between classes defined by different resources. Of course, central open questions of the field are mostly about proving that certain pairs of classes are actually distinct. One could imagine the potential power of this type of computational complexity based classification in other sciences, e.g. of synchronization or coordination processes in natural distributed systems, or e.g. of word problems and isomorphism problems on groups and algebras and manifolds in mathematics.

6. **Reductions**: *Ignore your ignorance, and even if you can't efficiently solve a problem, assume that you can, and explore which other problems it would help solve efficiently*. Despite furious arguments between philosophers on the power of "reductionism" to understand the world, understanding complex phenomena by breaking it into simpler parts is often responsible for great scientific discoveries. In computer science, a standard programming practice calls for using subroutines for a certain solved problem A as part of an algorithm for another more complex problem B. Algorithmic techniques make important use of such reductions. The main twist in the theory of computation is the use of reductions for proving *hardness* as well as easiness: in the above example, not only does the easiness of A imply the easiness of B, conversely, the hardness of B implies the hardness of A. These relations produce partial orders of computational tasks under various notions of difficulty, which often also relate different models to each other. The sophistication and intricacy of reductions (which are themselves algorithms) has been raised to an art form in areas like cryptography and pseudo-randomness, where many other properties besides efficiency restrict the processes A and B. Different, mathematically rich sophistication arises in hardness of approximation reductions and PCP constructions. I have no doubt that far more structure and connections can be uncovered in fields like mathematics and biology when the language of reductions is systematically used to relate disparate problems and models.

7. **Completeness**: *Identify the most difficult problems in a complexity class*[8]. Combining the two items above, the following has been amazing initially, and became a natural expectation by now. Whole complexity classes of problems that can be solved in certain limited resources and environments, can be "captured" by a single "complete" problem in the class. Completeness promises (via reductions) that better algorithms for that single problem will immediately entail the same improvements for all others in the class. And conversely, to separate the class from another, it suffices to prove hardness for that single problem. For both directions, one is free to focus on any complete problem in studying the whole class. Some notions of completeness, especially $\mathcal{NP}$-Completeness, turn out to be very widespread phenomena across many disciplines. Finding more examples of such phenomena in mathematics and science for other notions would be extremely interesting and useful.

8. **Hardness**: *Prove intractability results — these are useful!* The potential utility of tractability results, namely efficient algorithms, is obvious. However, knowing that a task is difficult (has no efficient algorithms, for some model and resource) can be as useful! It can suggest changing the model and the definition of the task in numerous practical and scientific applications. And as we have seen, hard problems can be directly used to yield positive applications, as in cryptography and pseudo-randomness. Finally, failed attempts at proving hardness have suggested surprising algorithms that may not have been discovered otherwise (a famous example is Barrington's algorithm [Bar86]). Hardness of course is typically hard to prove, and conditional hardness is the next best thing; in this case one naturally strives to minimize and simplify the assumptions needed.

9. **Barriers**: *When stuck for a long time on a major question, abstract all known techniques used for it so far, and try to formally argue that they will not suffice for its resolution*. Introspection has

[8] Namely those which all other problems in the class reduce to in the sense above.

been a strong suit in computational complexity, and the formal study of barriers to progress on major questions of the field became part of the field itself. It is interesting that these investigations have often led to *computational* characterizations of proof techniques (despite the fact that these are what we researchers develop, not the computational models we study). This results in some unexpected connections between proofs and computations, and surprising unconditional results, e.g. that no *natural*[9] proof exists of the hardness of factoring integers. Barriers are good not only as explanations (or excuses) for failure in resolving major open problems — they will hopefully direct us to develop new, different techniques which bypass them. This has happened in the past, both in barriers to lower bounds (such as diagonalization), and barriers to upper bounds (like integrality gaps for linear or semi-definite programming). It would be interesting to see mathematical analogs of barriers for proving e.g. the Riemann Hypothesis using current techniques, which are similar in spirit to the barriers we have in computational complexity to proving $\mathcal{P} \neq \mathcal{NP}$ using current techniques.

10. **Play** *Forget reality, ask for the impossible*. Despite being grounded and strongly connected to the practice (and demands) of computer technology, some of the greatest advances and innovations of the theory of computation came from completely ignoring realistic constraints and intuitive biases about what is possible, and making up toy models and problems to play with and explore. It paid handsomely, both intellectually and practically, to seriously and thoroughly explore ideas and notions that seemed unreasonable or even outrageous when introduced. Examples include non-deterministic machines, proofs which are not always sound, placing inputs on players' foreheads, playing Poker over the telephone, playing Chess and Go on an $n \times n$ board, pricing anarchy, counting without counters, conviction without knowledge, quantum post-selection, perfect randomness, anonymous ownership, putting Sudoku and Theorem Proving on equal footing, and many many others that need more than a few words to describe. These lead to continuously making up new games and rules and tasks for Alice and Bob, Arthur and Merlin, Byzantine generals and dining philosophers, multi-arm bandits and numerous players with less catchy names occupied with, and then playing and analyzing these games. Of course, many of these were inspired by external, realistic considerations, and strong intuition of their inventors, but their abstract and free exploration were often essential for progress and understanding of the original applications and often completely unexpected ones. The fearlessness of making assumptions and exploring their consequences, or asking the impossible and investigating the minimal assumptions that will make it come true, has been a constant driving force of the field, and will undoubtedly continue.

5 The computational complexity lens on the sciences

I now enter a topic that was hardly discussed in the book [Wig17] at all, with two important exceptions: $\mathcal{NP}$-completeness and Quantum Computing, each occupying a chapter there. So I will very briefly summarize both before elaborating on many others. First, it is remarkable how and why the notion of $\mathcal{NP}$-completeness has invaded all sciences and been essential and ubiquitous in all. "Ubiquity" may be a huge understatement in describing this rare scientific phenomenon. Scientists of all fields find the need to understand and use a very *technical* notion of Computer Science, and publish numerous[10] articles which associate NP-completeness with whatever they are studying. Now on to Quantum Computing, by now an exciting, well developed area which sparked remarkable scientific interactions as well as billions of dollars in the development of technologies for building a quantum computer. This field serves a perfect demonstration to what happens when the ToC methodology of the previous section is applied full force to the initial suggestions of the physicists [Ben80, Fey82, Deu85] about building quantum mechanical computers.

We now proceed far beyond these important examples.

[9] In the formal sense of Razborov and Rudich [RR97].

[10] You can amuse yourselves with Google Scholar, to see that "numerous" may be an understatement as well. Searching when is the phrase "$\mathcal{NP}$-completeness" occurs *concurrently* with *each* of "physics", "biology", "chemistry" in scientific publications gets about a million hits. To make these numbers even more meaningful, these pairs *each* occur within a factor of 5 from the respective occurrence of such natural pairs as "energy, physics", "atom, chemistry", "life, biology" and "function, mathematics".

A famous article of Eugene Wigner [Wig60], whose title captures its essence, is *The unreasonable effectiveness of mathematics in the natural sciences*[11]. Well, the effectiveness of ToC in the sciences *is* extremely reasonable, expected and natural, and we will give plenty of demonstrations below. This view is completely consistent with the mechanistic view of the world, underlying most scientific theories. It became known as the *computational lens on the sciences*, or sometimes, when quantitative evaluation of efficient resource use is primary, the *computational complexity lens on the sciences*.

We will discuss numerous connections and interactions of ToC with diverse scientific disciplines, which focus on the integration of algorithmic and computational complexity considerations as an essential part of *modeling* and understanding nature. (This focus goes far beyond the vast use of computation as a *tool* in science, which as mentioned is of major importance in itself.) I see these activities as promising radical changes in many existing theories, and major progress in better understanding many natural phenomena.

Before starting, let me point out that the computational element in scientific theories and models was always present (though often only implicitly). This followed both philosophical and practical considerations that long predate computational theory. First, while philosophers and scientists debated for centuries the precise properties of a scientific theory, all agree it should be *predictive*, namely be able to supply (or guess well) the result of a new experiment before it is carried out. Without such predictive power a theory is typically useless. Moreover, this predictive ability is necessary for the requirement of *falsifiability*: the existence of potential experiments that prove the theory wrong. But prediction is patently a computational task! Initial data is fed to the model as input, and the expected result must be supplied as output (e.g. "at what time will the sun or the moon rise tomorrow over New York?"). If this computational task is impossible or even merely intractable, then again the theory is useless. The need to solve computational problems in order to make scientific predictions was an essential component of the efforts of great scientists (as one famous example, Newton's *Principia* contains ingenious efficient algorithms for such prediction tasks). What has changed with the arrival of ToC and computational complexity was the ability to mathematically formalize and assess the tractability of these computational tasks.

Again, the pioneers of the field, Turing and von Neumann, already had a clear vision of natural processes as computational ones, and that this view is essential to understanding nature. And as was typical of these giants, they did not deal with trifles. In one of the most cited papers in biology, "A chemical basis for morphogenesis", Turing [Tur52] set out to give a "model of an embryo", that will explain the emergence of *structured* asymmetry and inhomogeneity (dominant in every living organism, with catchy examples like the color patterns on zebras, or left- and right-handedness) in a process which starts from symmetric, homogeneous initial conditions and follows symmetric evolution rules. von Neumann was even bolder. In his book "Computation and the brain" [VN58] he built the first significant bridge between computer science and neuroscience. In this remarkably prescient document von Neumann compares and contrasts computers and brains, and uses Turing's universality, the digital nature of neurons' outputs and most up-to-date knowledge in both (young) fields to show how much they can gain from interaction. In his book "Theory of self-reproducing automata" [VNB+66] he takes on the ultimate challenge, comparing machines and living creatures, and modeling life, evolution and reproduction. His *universal constructor*, a 29-state (universal) cellular automaton capable of self-reproduction is essentially using Turing's duality of program and data for replication, predating the soon-to-be-discovered[12] double-strand structure of DNA in which the same duality is used in natural reproduction. It seems almost sinful to give one-sentence summaries of these detailed and ambitious seminal works, and I hope the reader will find out more, and especially appreciate how natural computational modeling is for complex biological systems.

It took the theory of computation several decades before plunging back in earnest to continue in these giants' footsteps and integrate the computational approach into a much wider variety of scientific disciplines. These decades were essential for the internal development of ToC: many new computational models and types were studied, and the focus of computational complexity created a far better understanding of efficient utilization of various resources by various algorithms, and of arguing limitations and trade-offs on that efficiency. This fit perfectly with the modeling of nature, in which efficient utilization of resources

[11] This article was debated and interpreted by many after being published.

[12] This book was published long after his death.

is the law of the land. The past three decades have seen a boom in the invasion of algorithmic thinking and computational modeling into many fields in the natural and social sciences. In many cases, visionary leaders of these fields have recognized the importance of the computational lens and suggested integrating its study; famous examples include Herb Simon's *Bounded Rationality* (see [Sim57]) in Economics and Richard Feynman's *Quantum Computing* (see [Fey82, Fey86]) in Physics. In many other cases, either following such cues or independently of them, visionary leaders of algorithms and computational complexity, armed with the powerful methodology and knowledge of this field and with a passion for understanding another, have turned to propose the integration of the computational view to classical scientific questions.

What is exciting is that some of these have turned into real interactions, with growing collaborations, joint works, conferences and experiments. Of course, the difference in cultures and language, knowledge barriers, and simple caution and conservatism make some of these interactions move slower than others. Also, note that each of these fields is typically vast compared to ToC; this often limits interaction to specific subfields and problem areas. Still, there are already plenty of stories to tell.

There is no doubt in my mind that the algorithmic lens on sciences is the global scientific paradigm shift of the 21st century. We are only witnessing its beginning. The computational models proposed will evolve and new ones will be born, with more interaction, with novel experiments they will suggest, and with the absorption of computational methodology. These will naturally allow experimental sciences to become more theoretical, create further connections with mathematics, and interact even better with the complementary revolution of automated scientific discovery. This development will necessitate the education of every future scientist in the theory of computation.

The following stories give some illustration of computational and algorithmic modeling in a variety of different areas and problems across a remarkably diverse set of scientific disciplines. This selection is based on my biases and limited knowledge, and focuses mostly on outreach of ToC researchers to these fields. It should suffice to give the reader a sense of the scope of this interaction and its potential. There is no reason to think that any of the models or theories proposed is "correct" or final. Rather, the point is that computational complexity is an essential ingredient in them, and that this is typically a novelty which adds a new perspective to important problems in each field. As these collaborations expand, I view the stories here as a few drops in a huge bucket to fill. And yet, the combined impact of very few decades of such interactions is, in my view, amazing!

Note that I will *not* discuss at all the many important ways, in which the direct use of algorithms, computer systems and technology interact and impact the sciences, including analyzing and interpreting vast amounts of scientific, social and medical data, simulating scientific models and processes, etc. This interaction is yet another revolution, which is beyond the scope of this paper.

5.1 Molecular Biology

Perhaps the fastest, largest and most important growth of interaction between biologists and computer scientists has started with the Human Genome Project at the turn of the millennium. The combination of new sequencing technologies with new efficient algorithms to analyze the outputs these produced gave birth to *Computational Biology* (or *Bioinformatics*) (see e.g. [Pev00, JP04] for earlier works on this collaboration). These fields grew to encompass all aspects of molecular biology and genetics, disease discovery and drug design, with similar collaborations combining modeling and algorithms, both man made and of the machine learning variety. These collaborations, in academia and industry, probably dwarf in size everything else we mention combined!

In a reverse twist, computer scientists [Adl94, Lip95] initiated and studied *DNA computing*, the potential use of large molecules like DNA to faster solve difficult computational problems. Despite the slower speed of molecular processes in comparison with electronic ones, it seemed to offer immense (albeit, constant) parallelism at very low cost. This was indeed demonstrated in the lab by Len Adleman in his pioneering paper [Adl94]. Note that universality of this computational model was not obvious, but was demonstrated in vivo by [BGBD+04], with potential medical applications to cellular-level diagnosis and treatment!

Besides computing functions in the usual sense, Ned Seeman [See04] envisioned the design and physical construction of nano-materials from initial DNA and other molecules. The pictures demonstrate our ability

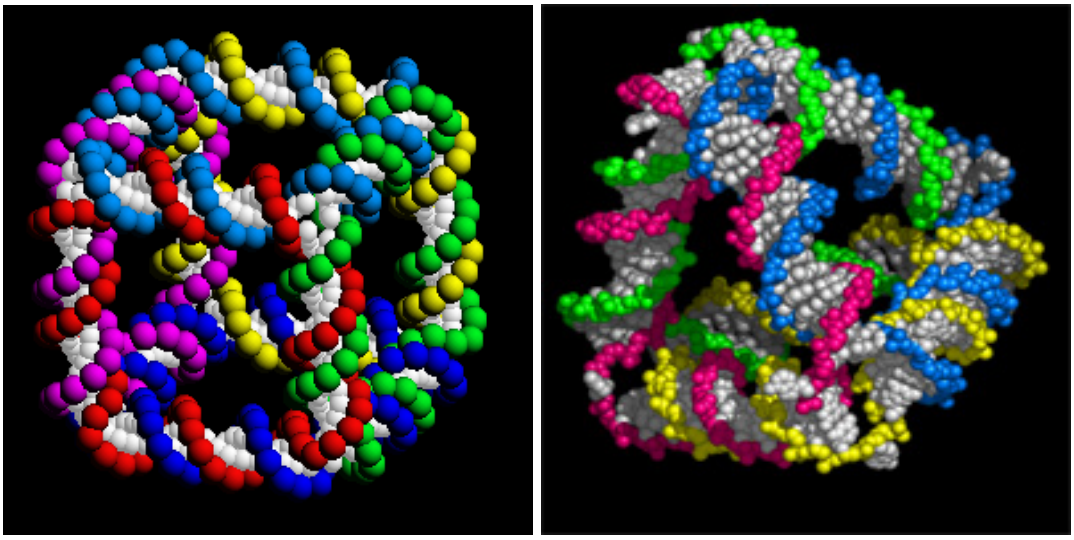

Figure 1: Cube and tetrahedron from DNA strands, from Ned Seeman's lab

to program DNA-like molecules to have certain atoms in certain positions which will interlock Lego-style with other ones to yield many different structures which are completely unlike anything nature does on its own (see figures of a cube and tetrahedron made by the carefully designed DNA strands in Figure [**?**]). See also e.g. [RPW04, BRW05] for more algorithmic techniques and resulting structures. The ability to program such nano-level organic designs for a variety of functional and structural medical purposes is extremely promising (see e.g. [KSZ+10]), especially as one can build not only rigid structures but also moving machines (see e.g. [GV08]). The same may be said of nano self-assembly for other materials (see carbon-based, e.g. Chapter 11 of [TPC15]). And indeed, there has been a growing flurry of joint activity of biologists and computer scientists on models and algorithms for "programmable matter" (see e.g. this workshop report [DK16] and this thesis [Der17] for recent examples). In all technologies the tolerance to errors is a key issue, and fault-tolerant self-assembly is a major thrust in this effort - see the survey [Win06].

It is clear that these exciting abilities beg a general theory of the underlying "programming languages" whose basic building blocks are nano-materials, and whose basic operations use chemical or physical bonds. Studying the expressive power (namely, which "architectures" are possible and at what cost) of such languages (and ultimately, designs) is certainly on its way as exemplified above, in parallel with the technological developments. These create new challenges for ToC researchers, and I fully expect much more collaboration in this area.

5.2 Ecology and Evolution

The fact that computational complexity is essential for models of nature was already clear to Charles Darwin in his *Origins*, and this was at the heart of one of the most interesting controversies in the history of science! Darwin took pains to approximate the age of the Earth, in order to check if his theory of evolution is consistent with the time it took life on Earth to reach such a level of diversity. In his day the (religious) dogma was still that the Earth is 6000 years old, and it was clear to Darwin this was far from sufficient for mutations, reproduction and natural selection to reach that diversity from one common origin. Luckily, the science of geology was being developed just then, and Darwin used it to estimate Earth's age at a few hundred million years, which seemed satisfactory to him[13]. But he became greatly upset (as he confessed in a letter to Wallace) when learning that the great physicist William Thomson (later Lord Kelvin) had

[13]Needless to say, this time estimate and its sufficiency for Darwin's theory relies on many assumptions, which he gives in detail.

estimated the age of the sun (based on its energy, according to the best physical theories of the period) only at a few tens of million years (roughly a factor of 10 *lower* than Darwin). This time estimate seemed to Darwin as having the potential to falsify his theory. Of course, as we know, Thomson was very wrong (as nuclear forces were discovered much later), and the age of the Earth is roughy 10 times *higher* than Darwin's estimate. But the message of this story is completely clear. Any natural process expends resources (like *time* in the story above), and so "computational complexity" must be an integral part of the relevant theory's consistency (and falsifiability). Indeed, better computational modeling of evolution processes and their complexity may lead to a revision and better understanding of this celebrated theory.

Building a quantitative, computational theory that will explain the complex mechanisms which evolved (e.g. in the cell) from simple ones through variation and selection has been the quest of Les Valiant in the past decade, starting with his paper "Evolvability" [Val09]. He views evolution as a restricted form of his PAC *learning* methodology. Rather than going into any details, let me suggest Valiant's exciting and extremely readable book *Probably, Approximately Correct* [Val13] in which he expounds his theory, and in particular the notion of *Ecorithms*: algorithms which interact with their environment.

From this very general question and model for evolution we turn to discuss two specific conundrums, whose resolution I believe will involve computational complexity. Some initial suggestions in this direction have been made.

The celebrated *"problem of sex"* (called the queen of problems in evolutionary biology by Graham Bell), asks to explain the prevalence of sexual reproduction in nature, in the face of its cost in energy and the loss of genetic material (which may have been improving by selection) in the process. After all, asexual reproduction seems far cheaper energetically, and can continuously improve fitness. Of course, these simplistic considerations were only the beginning of a lengthy historical debate and many theories that are far from resolved. Here I would only like to point out, again without detail, recent contributions towards a quantitative, computational, and very different proposal by a combined team of evolutionary biologists and complexity theorists [LPDF08], which was further explored and expanded in these (and other) papers [LPPF10, CLPV14, MP15, LP16].

Yet another mystery of evolutionary biology is the *"problem of conflict"*. Roughly it asks how is it that creatures, optimized by evolution over millions of years experience, can "freeze" by an inability to resolve a conflict between very different alternatives. This phenomenon has been observed in many species and situations (humans are extremely familiar with it), and as the one above, received plenty of debate and differing theories. Another collaborative work [LP06] suggests an intricate computational model in which evolution of a system towards optimal behavior may organically create within itself two subsystems which can be at odds with one another! Computational limitations play a crucial role in this model.

Concluding this section, let me discuss the very general problem of understanding some very different algorithms of nature from a very different angle.

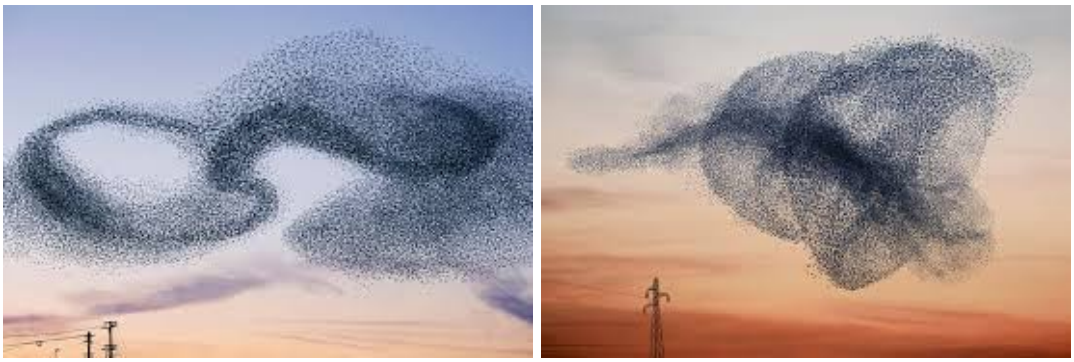

Figure 2: Search "starling murmurings" for amazing videos!

It is nearly impossible to watch the videos of many thousands of birds (the still pictures in Figure /ref-Starling do not begin to make this impact) without, following disbelief and awe, seeking to understand

the mechanism that allows this distributed system to produce such spectacular complex yet coordinated behavior. And indeed, for decades these (and numerous other forms of remarkable coordination and action by many other species) have been studied by scientists and mathematicians. I would like to point out one very general discrete model of such a phenomena, called *influence systems*, that was proposed by Bernard Chazelle (see his surveys [Cha12, Cha15]). It attempts to capture the dynamics of a distributed network in which the topology is changing by that dynamic itself. This model is relevant not only to the natural systems and algorithms but also to many dynamic social situations. Chazelle's work develops techniques to study such non-linear models, and gives the first quantitative upper and lower bounds on convergence times in bird flocking and similar systems.

5.3 Neuroscience

Unlike many other scientific problems discussed in this section, the quest to understand the brain (especially the human brain!) always had a significant computational component, as the brain is generally regarded as an (albeit terribly complex) computational device. Very early works in computational modeling of the brain include McCulloch and Pitts' *nerve nets* [MP43] (a collaboration of a neuroscientist and a logician) and the aforementioned book of von Neumann on the subject [VN58]. Great advances in biological techniques and in computational power have led to better and better understanding of neurons, the connections between neurons, and the activities of neurons during a variety of behaviors. These in turn have led to many (neural) network models of the brain, and of parts of the brain, attempting to explain various cognitive and other functions at different levels of specificity. It is interesting that the nature of many such models and the challenges in their analysis drew the interest and collaboration of physicists with biologists, which has been going on for several decades. All this represents vast and still rapidly growing work. Among numerous texts on the subject, an excellent comprehensive book surveying descriptive, mechanistic and interpretive computational models of the brain, at all levels is Dayan and Abbott's *Theoretical Neuroscience* [DA01].

In what follows I would like to describe some recent interactions, taht stress the integration of computational complexity into models of the brain. I expect these interactions to grow, and lead to more collaboration and understanding[14].

In terms of a general computational model of the brain, Les Valiant [Val00] pioneered a comprehensive design from a computational complexity perspective. It postulates precise mathematical assumptions about the interconnection patterns between neurons and the computational capabilities of individual neurons, which are consistent with known physiological results. It then proceeds to describe detailed algorithms which could be implemented in this model, and carry out some basic cognitive functions of memorization and learning. These details include how and what information is stored in (collections) of neurons, and careful accounting of the physical and computational resources expanded by these algorithms. To me, more than an actual model of the brain (which will probably take decades to figure out), Valiant's is a model for modeling the brain. In particular, it suggests further experiments, and is eminently falsifiable. Curiously, it even predicted the need of certain physiological phenomena (e.g. strong synapses, neurogenesis) that some of the algorithms required, and these predictions were indeed verified to some extent after the first publication of the book. Valiant further articulates principles for general modeling of the brain in [Val06].

More works attempt to computationally address more specific areas of the brain, and focus on the implementation of more specific cognitive functions. The hippocampus and memorization are further studied e.g. in [Val12, MWW16], the cortex and learning in [Val14, PV15], and the compression of information in neural tissue in [AZGMS14]. While these papers are mostly by computer scientists, they appear in journals read by (and presumably reviewed by) neuroscientists, and promise future collaboration.

One of the greatest computational and algorithmic challenges in neurobiology arises from *connectomics*, the field which (in some analogy with the Human Genome Project, only orders of magnitude harder) attempts to map out the entire human brain, complete with all synaptic connections. Even partial success will hopefully give the ability to simulate its activities on a computer, and to better understand its processes and diseases. The very cross-disciplinary *Blue Brain Project* [Mar06] represents perhaps the most massive such

[14] See e.g. https://simons.berkeley.edu/programs/brain2018

collaborative attempt today. The "big data" challenges arising even in the decisions of which data to collect, which to keep, how to organize it, and how to go about extracting the required structure out of it are great and far from agreed on (of course, these challenges are present and increasingly so in many other areas in which unprecedented amounts of data can be easily collected). Joint work of neuroscientists and computer scientists [LPS14] attempts to articulate principles guiding this task, and possibly build a computational theory of connectomics.

5.4 Quantum Physics

In contrast to the section above, where I focused on contributions of complexity theorists to neuroscience, here I focus on contributions of physicists, who use complexity theoretic results and methodology to suggest resolutions of conundrums of physics! *As we shall see, the physicists' quest for understanding the fundamental structure of physical space and time may need the fundamental understanding of the computational resources of space and time!*

We have devoted a whole chapter of the book [Wig17] to Quantum Computation, which was started by physicists, picked up by computer scientists, and has led to a remarkably intense and productive collaboration between the two groups, and produced new results and fundamental questions in quantum mechanics and quantum information theory. A byproduct of this interaction is the integration of complexity theory concepts and results by physicists which allow for the proposals I will now sketch. I will focus solely on Quantum Gravity, or the long-sought theory which will enable marriage of quantum mechanics and general relativity, towards Einstein's dream of a *unified theory*. And I will only discuss a few (related) examples. I fully expect far more integration of computational considerations to theories in all areas of Physics, and far more interaction between the fields.

It is all about black holes. If the following sounds too mysterious and far-fetched to you, know that I feel the same, and so do some physicists. We discuss physical theories that are far from being testable in the foreseeable future, and arguments regarding them rest heavily on thought experiments. Moreover, many of these ideas are recent, and are extensively debated. I will be lax and informal. For the brave who wish for detail, a technical but very readable survey, with plenty of intuition, detailing most aspects of the discussion below was written by Harlow [Har16]. It also contains all relevant references. A more informal survey that addresses these issues (with more computational complexity background) are these lecture notes of Aaronson [Aar16]. Now brace yourselves for two stories full of buzzwords, which you should accept and read on as you do with fantasies, to the (computational) punchline at the end.

A fundamental connection of gravity and quantum mechanics is the *Hawking radiation* (while Hawking was the first to fully develop it mathematically, he used earlier ideas of Zeldovich and of Bekenstein). They discovered that, while black holes are so heavy as to swallow everything, including light around them (which is the reason we don't see them directly), and seem doomed to grow forever, quantum effects nevertheless cause them to leak particles, and despite the extremely slow pace of this effect, black holes will eventually evaporate altogether!

Hawking radiation, which is well accepted, makes it extremely difficult to stitch the seemingly conflicting constraints imposed on it by general relativity (e.g. the singularity at the center of the black hole, the smooth structure of space-time at the event horizon[15] of the black hole) on the one hand, and those imposed by quantum mechanics (e.g. unitary evolution of the radiation, complementarity, no-cloning of quantum states and monogamy of entanglement). These potential conflicts between constraints (not all of which share the same acceptance status) are explored through ingenious thought experiments. The central one at hand, called *the Firewall Paradox*, was suggested by Almheiri et al. in [AMPS13]. *Oversimplifying*, the paradox can be roughly summarized by the ability of an observer sitting outside the event horizon to extract a qubit of information from the leaking Hawking radiation, then jump into the black hole and extract that *same* qubit after crossing the horizon[16], which is a contradiction (it would violate a basic theorem of quantum information theory called *monogamy of entanglement*)!

[15] The "boundary" of the black hole; anything inside it is trapped and falls into the black hole.
[16] I did not make this up!

Many proposals to resolve the paradox were suggested, but the one of Harlow and Hayden [HH13] (see elaboration and simplifications in [Aar16], Lecture 6) is unique in introducing a computational viewpoint which is relevant for many other issues regarding such processes. It essentially views both the black hole producing the radiation from the information that fell into it, as well as the observer who extracts the paradoxical bit from that radiation as efficient (polynomial time) quantum algorithms. This view leads to a nontrivial computational analysis whose upshot is that if *Quantum Statistical Zero Knowledge is hard for quantum algorithms*[17], then *the black hole will evaporate long before the observer can compute any paradoxical bit*! In short, computational complexity offers one resolution of this paradox: time is too short for the experiment to take place, simply since the observer's task is computationally hard[18]. This explanation (that in itself is conditional) is still debated with the others, but its computational spirit entered the discussion, and with it the understanding that viewing nature and observers as algorithms is illuminating!

Understanding the processes in the interior of a black hole, which are subject to general relativity, is of course problematic almost by definition—there is no way to peek inside. Would it not be wonderful to deduce it from what we observe on its boundary? A remarkable proposal of Maldacena [Mal99] suggests something of that kind: a general method for accessing information about the behavior of particles governed by a quantum gravitational theory (like string theory) inside the "bulk" (or interior) of space-time region, from studying the dynamics of a quantum mechanical system on its boundary. This theory has become known as the *AdS/CFT correspondence* where AdS stands for "Anti de-Sitter" space[19], and CFT for Conformal Field Theory[20]. Conformal field theories are completely accepted and computations of their evolving parameters are by now standard in some cases. Using these to infer various parameters inside the AdS universe (or indeed go the other way, use general relativity in AdS to bypass complex CFT computations[21]), one needs a "dictionary" that relates AdS to CFT quantities. Indeed, Maldacena provides an extensive such dictionary. While there is no proof of this duality, it has become an extremely useful tool in many areas of theoretical physics, and allows making predictions about the bulk behavior which may perhaps be tested.

Now, we will focus on *the Susskind puzzle*, which asks for the analog of an AdS quantity called *wormhole length*. This idea of "wormholes" goes back to Einstein and Rosen, and according to another exciting new theory of Maldacena and Susskind they are supposed to explain the nature of entanglement. Anyway, all we need to know is that in appropriate AdS universes the "length" of wormholes grows linearly with time. Susskind's puzzle is to fill the dictionary with the analog quantity of the corresponding CFT. It is hard for any physicist to imagine *any* quantity in quantum evolution which scales linearly with time! But Susskind finds an answer by considering computational complexity. Discretize time and space in the given CFT, and you get a quantum circuit on some n qubits, which the evolution iterates at every step (say, with some fixed initial state ψ_0, e.g. all zeroes). This iteration produces a sequence of quantum states, ψ_t after t steps. Susskind proposes that the quantum circuit complexity of ψ_t in the CFT[22] *is* the dictionary analog of the wormhole length after t steps in the AdS evolution! In other words, a fundamental, *physical* component of a theory of quantum gravity corresponds under this proposal to a *computational* parameter in CFT. Again, while still fresh and very debated, the view of processes in these theories as quantum circuits, and studying their computational complexity, is entering the discourse of theoretical physics.

[17] No need to understand what that means, other that it is a computational assumption about the intractability of a natural problem, like $\mathcal{P} \neq \mathcal{NP}$. But I stress that this strange computational assumption, merging cryptography and quantum computing has made it to the consciousness of physicists.

[18] In the same way as in the section about Lord Kelvin's calculations of the age of the universe could have killed Darwin's evolution theory, if they were correct.

[19] To stress that the theory of gravity we employ happens on a negatively curved space, namely has a negative cosmological constant. As an aside, notice that this certainly is *not* the case in our universe, whose expansion is accelerating, and so has a positive cosmological constant; this does not stop physicists pursuing first the full understanding of this "toy model".

[20] Namely a quantum field theory equipped with a strong symmetry called conformal invariance. Importantly, there is no gravity in CFT!

[21] In general these require quantum computation

[22] Namely the size of the smallest quantum circuit generating that state from the zero state.

5.5 Economics

One of the great successes of ToC interactions with a "remote" discipline has been with economics, particularly with game theory. This field was born in the late 1940s with the celebrated magnum opus *Theory of games and economic behavior* of John von Neumann and Oscar Morgenstern, and with John Nash's foundational papers on equilibria. During the following decades game theory has developed sophisticated and elaborate theories of markets and strategic behavior of rational agents in them. These theories manifestly aim to be predictive, and inform decision making of strategic entities (from von Neumann's original motivation to play well the (frightening) cold war game with possible strategies including nuclear attacks, through the (potentially vicious) competition of different firms over market share, where strategies may be simply the prices of products, all the way to the personal and social interactions in numerous situations). In many theories and papers of the field, it was intuitively clear that real people or organizations, with computational limitations, must actually implement the suggested strategies. Indeed, some algorithms were proposed, but relatively little was done to formally investigate their performance, and more generally to integrate computational complexity into these models, partly since the relevant computational theory was missing.

This void started filling in the 1980s, when a theory of algorithms and complexity was quite developed, and interest in ToC to apply it to economics grew. This work accelerated since the 1990s with the advent of the Internet, creating new markets in which processing time of participants became a primary concern. The interaction with ToC and the integration of the computational and algorithmic elements into economic theories and models encompasses many aspects of the field. It is lucky for me that an excellent collection of surveys appears in the book *Algorithmic game theory* [NRTV07], and so I will briefly recount below some of the major themes (by now, full fledged research areas) covered there; far more detail and missing references can be found in this book's chapters. Plenty more research and collaboration has happened in the decade since this book's publication, and I will conclude with one (very different) example of interaction between the fields.

- *Complexity of equilibria.* Equilibria in a variety of strategic situations (games, markets) embody rational solution concepts: they guarantee each individual satisfaction with their payoff given the behavior of other parties, and so are sought as a notion of stability. Central theorems of game theory prove that equilibria exist in very general situations. But how to achieve them was rarely discussed. The pioneering work of Papadimitriou [Pap94] observes that the arguments used in such proofs are purely existential (e.g. topological fixed-point theorems), and that natural algorithms to compute equilibria require exponential time. He develops a complexity theory for economics and mathematical problems of this nature, namely where solutions are guaranteed by various existential principles. He raises the possibility that computing Nash equilibria is a complete problem for the class PPAD, namely as hard as finding a Brouwer fixed point. This was settled in the affirmative 15 years later [DGP09, CDT09], even for 2-player games, giving the first indication of hardness of Nash equilibria, possibly limiting its value as an economic solution concept[23]. Proving this result required the introduction of the family of *graphical games* into game theory in [KLS01]. The "need" for efficient equilibrium concepts which may salvage this hardness has led to approximate versions of Nash equilibria which do have near-polynomial time algorithms [LMM03]. Moreover, hardness results and algorithms were then extended to a variety of other equilibria, most notably the Arrow-Debreu market equilibria. Approximation algorithms for such problems connected this field with optimization, leading to proofs that in many "real-life" situations (restricting utilities, preferences, etc.), equilibria *can* be computed or approximated efficiently.

- *Price of anarchy.* This is a completely different take on equilibria, one that could have been part of economic theory but somehow was not. Proposed in the seminal paper by Koutsoupias and Papadimitriou [KP99], the *Price of anarchy* is the ratio between the cost to society of a "distributed" solution, given by the worst equilibrium that individual (selfish and non-cooperating) agent strategies achieve,

[23] Further hardness results based on various (so far, nonstandard) *cryptographic* assumptions have recently been pursued (see e.g. [BPR15]), that heavily use the reductions and completeness foundations of [Pap94] and their subsequent development.

and the optimal "centralized" cost, attained by strategies assigned to them by a (benevolent and fully informative) entity. Numerous situations in which agents need to make decisions in utilizing shared resources (e.g. cars on roads, packets on the Internet), affect performance (and thus individual and societal cost), and raise the question how costly is this freedom of choice ("rational anarchy"). This paper has led to very surprising results showing that in many general situations the price of anarchy is far less than what might be expected. One major general result of this type is [RT02], establishing that this ratio is at most 4/3 for the traffic flow on *any* network of any size, in which delay on a link is a linear function of its congestion. This bound remarkably matches the tight example known for decades in economics as Braess' paradox (showing that adding a link to a network can actually *increase* congestion), that is achieved on a 4-node network. Another example of a similarly surprising result, in its generality and in the bound on the price of anarchy it supplies, was discovered for the completely different setting of Walrasian equilibrium [BLNPL14]. We stress that while the questions of distributed versus centralized solution costs are non-algorithmic in nature, they, as well as their solutions, arise naturally from ToC methodology (here mainly the use of reductions and of approximation guarantees, neither common in classical game theory). These important works have naturally led also to many algorithmic questions, again connecting with optimization, on-line algorithms and distributed computation, connected with research questions in game theory and generated new collaborations between the fields.

- *Mechanism design* (sometimes called "inverse game theory"). A major goal of game theory is understanding the behavior of rational, strategic agents participating in games which are already *prespecified*. Mechanism design is the field of game theory which asks the meta-question: how to specify (or engineer) these games, incentives and rules of behavior so as to force similarly rational, strategic agents to achieve certain global goals. Examples abound: governments setting up laws and distributing resources for various societal benefits, auctions aimed at selling items and maximizing profit, voting systems for best representations of public opinion, healthcare and other social systems balancing services and costs, etc. Many basic principles and theorems were discovered in this study, explaining e.g. how to elicit agents' actions to have various properties as truthfulness, maximizing profit or some global social outcome. Again, the advent of the Internet has added numerous new applications for the design of mechanisms, which are also far more complex than many previously considered due to the potential huge number of participants, speed of action and the distributed nature of interaction (the market of on-line advertisements is a perfect example). Now algorithms are the agents, and their advantages and limitations must be taken into account. This has greatly enriched the theory of mechanism design in manifold ways. For example, the use of communication complexity and other tools allowed for hardness results limiting the power of mechanisms. For another, the view of (distributed) algorithms and protocols as executed by rational agents has added a completely new ingredient into algorithmic theory; its design should ensure behavior which makes it correct, efficient, fair, etc. when implemented by selfish rational players. The collaboration between ToC and game theory in this field is perhaps the most extensive, and is manifest in industry as well, as one can observe even restricting attention to the theory and practice of auctions for advertisements motivated by a multi-billion dollar search-engine advertising industry.

- *Cryptography and game theory*. We mentioned above the negative news to economics following the computational complexity limitations on the discovery and implementation of desired strategies (e.g. in equilibria). On the other hand, plenty of positive news follows from cryptography on the implementation of strategies which seem impossible or hard to achieve in the (generally assumed) information theoretic environment of game theory. As one example, take the notion of *correlated equilibrium* of Aumann, which in contrast to Nash equilibrium is easy to compute, *if* all information about all players' utilities is known to some central trusted player. This "unrealistic" assumption renders this notion almost useless in situations where no such trusted authority exists or players wish to protect their privacy, until you realize that the *secure multi-party computation* solves just that! Indeed, the papers [Yao86, GMW87, BOGW88] and their followers do so (in different settings), eliminating the need

of a trusted party in any such situation, and making Auman's solution concept (and many others) eminently realistic and useful! The many differences of axioms of both fields (in crypto, privacy is a goal, while in game theory it is often the means; in crypto, players may be honest or malicious while in game theory they are rational, etc. etc.) guarantee great synergy and collaboration as we integrate the point of view of one field into the other.

Let me conclude with an example of interaction between the fields that tackles a completely different, fundamental issue, that probes the assumption of transparency and efficiency of markets. Mountains of articles were written about the financial crisis of 2008, proposing numerous reasons and models for that colossal collapse. There is a general agreement that mis-pricing of financial derivatives (such as CDOs) played a central role, and there are debates about how to fight it. One paper which proposes a computational complexity perspective on this situation is the collaborative [ABBG10], *Computational complexity and information asymmetry in financial products*. In it, the authors propose a simple model which demonstrates how information asymmetry between buyer and seller, and hence the cost they assign to the same derivative, can be extremely high, even in a fully transparent market, and even if the buyer (or any inspector of the derivative contract) is extremely powerful computationally, e.g. a large bank, regulator or the government. Creating such *artificial* examples is easy for anyone versed in cryptography, and indeed, this very asymmetry of information underlies all Internet security and e-commerce systems! The paper above constructs simple, *natural* CDOs, made up from mixing "normal" and "junk" assets, and shows that distinguishing those in which the mixture is random from those in which the mixture biases for "junk", is a computationally difficult task under well-studied complexity assumptions. This paper has gotten some traction and reference in the economics literature (see e.g. the very different [CKL13, BCHP17][24]). I feel that the many financial models (e.g. like those of markets and risk in the papers above), invite possible revisions and extensions in light of computational complexity theory. Unlike game theory, the finance side of economics has had precious few collaborations with ToC so far, but I expect these to grow.

5.6 Social Science

The advent of the Internet and the many types of social activities and networks it supports has opened up both a host of new problems for social scientists, as well as, for the first time, the ability to run social experiments on vast numbers of participants. This has created many new collaborations between computer and social scientists, which is enriching many aspects of social science and making it much more quantitative. Aside for the one (early) famous example below, which illustrates the nature of ToC influence, I will simply refer the reader to the excellent and extensive book *Networks, crowds, and markets: Reasoning about a highly connected world* [EK10]. This book itself is such a collaboration, and I note again that plenty more has happened in the seven years since its publication. The variety of topics of interaction between ToC and Social Science described there is staggering (especially given the short amount of time since interaction started), and we give just a taste. These topics highlight a major evolution within ToC of the study of distributed networks. Most of this large field focused initially on man-made design of networks and algorithms for them. The interaction with social science has expanded this field greatly! Among the many new aspects the book describes are the growth and change of different (physical, information, social) networks, the organic appearance and evolution of power structures within them (e.g. hubs, authorities, institutions), the nature of dynamic processes (gossip, influence, incentives, connectedness and others) on networks. Many of the network models include game-theoretic and economic aspects, when viewed as markets whose participants are strategic agents placed at the nodes. This connects the research here with that of the previous section. In all, the ease of getting real-world data on many of these aids the testing of new models, theories and algorithms. Of course, many other aspects and models of social interactions are not covered by the book [EK10], e.g. the work of Chazelle on *influence systems* [Cha12, Cha15] mentioned above.

Our example is Jon Kleinberg's paper [Kle00], titled *The small world phenomenon: an algorithmic perspective*. We only sketch it here; the reader is encouraged to read its lucid, detailed exposition and

[24] Be cautious that the word "complexity" can mean many different things in economics.

references. This paper is a take on Stanley Milgram's famous experiment on *The small world problem*, first described in [Mil67]. Following previous work pursuing the popular cliche *Six degrees of separation*, Milgram designed an experiment to test the connectivity and distance between pairs of people in the US by asking people to forward a letter from one to the other using only intermediaries who are on "first name basis" with each other. The required chains were typically indeed very short, and a sequence of analytical network models were suggested and analyzed which might at the same time explain social relations in society and support such short distance. But it took over 30 years until Kleinberg pointed out that there is another fundamental problem to be explained in Milgram's experiment: *even if short paths exist, why were such paths found by a purely local algorithm*. Kleinberg went on to give a thorough answer. First, he proved that many networks models (including most suggested in past work) have an abundance of short paths, which *no* local algorithm will find! Next he proposed extended natural models, and identified the unique one in this class for which a local algorithm will succeed. This single work was key to the collaborations mentioned above, and to numerous follow-up works.

6 Conceptual contributions; or, algorithms and philosophy

Let me move up another level of abstraction in the nature of ToC impact. We describe some of the following important contributions in the book [Wig17] but there are plenty more we didn't!

Alan Turing not only laid out the groundwork for the mathematical foundations of computer science and for the computer revolution which followed, but has also demonstrated the powerful light that computation can shine on fundamental concepts. His article, *Computing machinery and intelligence* [Tur50] takes on one of the most difficult and controversial notions, *intelligence*. With remarkable clarity and brevity, rare in arguments on such philosophically, socially and scientifically charged concepts[25], he proposes a completely fresh and original approach to defining and understanding it.

Over the years, the theory of computation was presented with the need to understand, and thus also firstly to define, concepts and notions which have occupied intellectuals from many disciplines over centuries. This need arose sometimes from scientific or technological origins, and sometimes indeed from the philosophical inclinations of thinkers in the field. With the computational focus and language at heart, they gave birth to novel definitions of such concepts, breathing fresh ideas into arguments over their meaning and utility. This is far from saying that the computational lens is better than other views—it sometimes complements others. But rather, it points to the intellectual depth *inherent* in the field, and to the relevance of computation, and even more, computational complexity, at the philosophical level. I feel proud to belong to a field which has seriously taken on defining (sometimes re-defining, sometimes in several ways) and understanding such fundamental notions that include:

collusion, coordination, conflict, entropy, equilibrium, evolution, fairness, game, induction, intelligence, interaction, knowledge, language, learning, ontology, prediction, privacy, process, proof, secret, simultaneity, strategy, synchrony, randomness, verification.

It is worthwhile reading this list again, slowly. I find it quite remarkable to contrast the long history, volumes of text written and intellectual breadth the concepts in this list represent, with small and young such a field that has added so much to their understanding. Moreover, for many of these there was little or no expectation that the science of computing will need to deal with them, and that if it does, will find meaningful and illuminating ways to do so. It is only with hindsight that it all looks so natural now. The book [Wig17] discusses some of these notions, and when it does, I have tried to articulate the way computational complexity illuminates them, often borrowing directly from the very articulate people who originated these definitions. Scott Aaronson [Aar13] notes that while some of us convey extremely well the philosophical implications of our work to our own community, we perhaps should broadcast more of it to the outside world, in particular to philosophers. His paper tries to do just that—explain the potentially important role that computational complexity can play in understanding some philosophical and scientific conundrums.

[25] But typical to all of Turing's arguments on many other issues, including his definition of a computer!

This whole direction, abstracting the role of computation and complexity in the intellectual discourse probably deserves another book, far less technical and more accessible than this one. I will content myself here by concluding with two very high level philosophical issues, possibly meta-philosophical principles. The first is *subjectivity* and the second is *interaction*. Again, neither is new, but armed with the computational complexity lens they inform and revise many of the notions above.

Subjectivity The first is the role of computational complexity in establishing *subjectivity* in the view of reality. Properties (including many of the notions listed above), were for centuries defined and treated as intrinsic, *objective* properties of objects, agents or interactions. Now they are treated by computational complexity theory as *subjective*, highly dependent on the observer! This should be reminiscent of Einstein's special relativity, and its radical (at the time) predictions that different observers may see the same object have different lengths, or that may disagree on whether two events were simultaneous or not. Einstein derived such predictions by adding a single axiom: that *the speed of light is constant*. The axiom taken by computational complexity is that *certain problems are intractable*. This has no less radical consequences. A random event can be completely unpredictable by one observer, and predictable to another. A message can be clear to one receiver, and completely incomprehensible to all others. These consequences of the simple computational axiom underlie Internet security and E-commerce, as well as many other applications. What is more, differences in the computational powers of different observers (who may actually be participants in a certain activity) allow us to *quantify* "how much" of a given property is present from their point of view. For example we can quantify precisely, given the computational complexity of an observer, how predictable some event is, or how much knowledge it can glean about a secret message, or how accurately it can evaluate a complex economic commodity. In short, having different computational powers imply having different inference, analysis and reasoning powers, and sophisticated use of these limitations, together with appropriate versions of our computational axiom make this subjective view of reality the source of some of the most exciting and important applications of computing systems.

Interaction Interaction enhances many aspects in life. The importance of interaction for *understanding* easily follows by comparing a class in which students are allowed to ask questions, to a class in which they are not. Interaction of this nature underlies Turing's "Imitation Game" which underlies his approach to *intelligence*. Simple examples also show that the *cost* (e.g. the amount of communication) of natural computational tasks between two parties can be greatly in bi-directional interaction versus uni-directional ones. The study of interaction in computational complexity has uncovered extremely surprising powers which were discussed in some chapters of this book [Wig17]. Here are some examples. One major impact was on the central notion of *proof*. Classically, proofs are uni-directional communication from a prover to a verifier. Allowing bi-directional *interactive proofs*[26] has led to a complete revision of what is the capability of proofs. First, more can be proved interactively: the "$\mathcal{IP} = \mathcal{PSPACE}$" theorem [Sha92] basically shows how a prover can convince a verifier of having a winning strategy in e.g. Chess, something unimaginable in classical proofs. Second, proofs can have paradoxical properties: the "$\mathcal{NP}$ in Zero-Knowledge" theorem [GMW91] basically shows that every statement which has a proof, has an interactive proof which is equally convincing, but reveals no new information to the verifier! Other fundamental discoveries extend results from the uni-directional to the interactive setting. Shannon's famous result [Sha48] proves that uni-directional communication can be protected from constant noise-rate with only constant redundancy (via his error-correcting codes). The same result holds (via Schulman's new ingenious codes [Sch92, Sch93]) for bi-directional communication, despite the adaptivity of such conversations.

Let me conclude with a a higher level contribution, where the study of interaction (in the surprising context of information privacy), actually informs the *scientific method* itself. A key axiom of this method demands that a scientist decides the questions to ask about the data *before* she collects it. This forces a uni-directional communication with nature: first data is received or collected, and only then processed. Of course, findings lead to new questions and require new experiments and data acquisition. Scientists may be (indeed, some are) tempted, for efficiency reasons, to use existing data to answer the new questions,

[26] and incorporating randomness, error and complexity.

rather than collecting more data. Are the results of such adaptive use of data valid? As it happens, an interactive definition of *differential privacy*, and efficient algorithms for maintaining it, surprisingly inform the fragile validity of adaptive data analysis [DFH+15], showing that in certain cases the above temptation may be justified, and how! Such deep connections, which go far beyond the intended utility of the models and algorithms developed, further shows the depth of the very concept of computation.

7 Algorithms and Technology

We now switch gears to discuss several aspects of the intense and continuous interaction between the theory of algorithm design and the industry of computing systems.

Given its focus on computational complexity, the book [Wig17] has paid little attention to *the* major effort of the Theory of Computation, namely algorithm design. A major product of the theory of computation is the design and analysis of efficient algorithms for specific problems, and more fundamentally, the design of algorithmic techniques and paradigms for broad classes of computational tasks in a variety of models. Here I address a few issues related to algorithms, their design and impact, and interaction with technology. One word of caution: in much of the ensuing discussion the boundary between algorithms and technology is not always completely clear (as is the case between theory and practice, science and engineering, pure and applied math).

7.1 Algorithmic heroes

The combination of algorithms and technology has provided computational power and its myriad applications to society in a way that has totally transformed it in just a few decades. No doubt, this trend will continue and accelerate. While both ingenious algorithms and remarkable technological advances played central roles in this revolution, it seems to me that the general public is not quite aware of the role algorithms play in enabling the applications they experience, and attribute it almost solely to technological advance. Of course, in many cases algorithmic and technological advance are intertwined, but this ignorance exists even when *a single algorithm has enabled a whole industry*, and moreover the essence of that algorithm can be explained to any undergraduate in a few pages or a few hours.

In popular talks I give about algorithms and complexity I show two list of names, and ask the audience which of them is familiar. The first list has names like Johannes Gutenberg, Joseph Jacquard, Thomas Edison and James Watt. These were presented as cultural heroes when I was growing up for inventions that dramatically advanced society at the time: respectively the printing press, the weaving loom, the light bulb and the steam engine. Most general audiences recognize them and their inventions (although it is less so among young audiences). The second list I show has names like Edsger Dijkstra, Donald Knuth, John Tukey and Elwyn Berlekamp. Here I mostly draw blanks. I then show, just for impression, the few lines of pseudo-code for each of the respective algorithms: Shortest Paths, String Matching, Fast Fourier Transform and Reed-Solomon decoding that these algorithmic heroes and their collaborators developed, and others extended. I note that like the older inventions of physical devices mentioned, these tiny intellectual creations are extremely brilliant and efficient, but that their impact may be far larger. Demonstrating this I finally discuss (some of) the industries which have of variants these algorithms enabling them, respectively: automatic navigation, computational biology, medical imaging and all storage and communication devices.

It seems to me that such stories are exciting and accessible school material even in middle school, and the theory community should have every incentive to bring it there. Of course, these are just examples and there are many others; prime examples of such algorithms include RSA, underlying most of E-commerce, and PageRank, underlying Internet search, and Back-Propagation, without which the "deep-networks" revolution in machine learning (that we will soon discuss) would be impossible. In these examples, some people may recognize the names of the algorithms (or the buzz surrounding them), but not necessarily their inventors. Finally, there are even more basic algorithmic principles and structures, for efficient data organization and access, including hashing, caching, sketching, heaps, trees, bags-of-words, among many others[27], all gems

[27] It is convenient for me to know that the reader can find details on Wikipedia and other Internet sources.

underlying many algorithms and applications.

7.2 Algorithms and Moore's Law

Another point to discuss is the relative contribution of technology[28] and algorithms to applications, especially in the future. Clearly, a computational application becomes available only if it is efficient enough to be useful to its users and profitable to its developers and manufacturers. How much more efficient can we make hardware and software? Focusing on the most basic resources whose efficiency is paramount, speed and size, clarifies how incredible the science and technology of hardware design has been. In what became known as *Moore's Law*, Gordon Moore predicted in the 1960s the doubling of the number of components on integrated circuits every 18 months. Similar predictions were made regarding the increase in computational speed. Miraculously (at least to me), this exponential growth persisted at the predicted rate for half a century! But not anymore...

Whether the recent slowing of Moore's law is a sign of its imminent death, or it will stay with us for a while longer, technology has *absolute* physical limits. Components will never be smaller than atoms, and communication between them will never exceed the speed of light. At that point, and likely much earlier, the *only* source of efficiency will be the invention of better algorithms; which computational applications and products we will have available, and which we will not, will depend on algorithmic advances far more than on technological ones. Here, for numerous problems, the best time and space bounds we know seem far off from optimal, and there seems plenty of room for improvements!

7.3 Algorithmic gems vs. Deep Nets

So, it is pretty certain that algorithms will rule the Earth, but which algorithms? The remarkable computing power which technology has already provided us (together with very important algorithmic advances!) has enabled a new breed of algorithms, which arise in machine learning. In short, algorithms as technological products. Tech and medical companies, as well as governments invest enormous funds and effort in these new algorithms, which I turn to describe (and are discussed more generally in Section 8.2).

In stark contrast to the elegant, concise algorithmic gems mentioned above, which were man-made, many new algorithms simply "create themselves", with relatively little intervention from humans, mainly through interaction with massive data. This contrast with "classic" algorithm design is heightened as these new algorithms are typically (and perhaps necessarily) gigantic, and very poorly understood by humans. I am referring of course to the "deep networks" (see [BGC15] for an extensive text on their uses in learning and optimization), a common name to heuristics modeled after networks of neurons[29]. I will conclude this section with a few remarks on these algorithms, and the challenges and questions they raise.

These self-taught algorithms attempt to solve numerous problems which are often hard to formally define, such as finding "significant signals" in huge data sets that are often extremely noisy—be they financial, astrophysical, biological or the Internet. The structure one may want to extract/uncover may be clusters, correlations, geometric or numerical patterns etc. or completely unexpected ones. These may represent e.g. familiar faces or objects in pictures, groups of friends and interests in social media, market performance of companies in the buying and selling of stocks, effects of treatments or genetic defects in biological data and illuminating interactions of matter and energy in physical observations.

This is no place to give a proper description of the deep nets and their training process. It suffices to say that today their size can get to millions of gates and wires between gates. Each connection has a strength parameter that the training process attempts to optimize. In short, training is a huge optimization problem with an ill-specified goal and numerous degrees of freedom in the heuristics driving the attempted optimization. This "black magic" works extremely well only in relatively few cases so far. But in a growing number of cases it greatly outperforms any human designed algorithm. It is extremely interesting when such a self-taught program can label by name the essential content of arbitrary pictures taken by humans, nearly

[28] Here, in the sense of physical infrastructure of computing systems.

[29] There are many other types of heuristics, past or future, to which this discussion is relevant as well. However, deep nets seem to have become dominant in the past decade, so I'll focus on them.

as well as humans would. It is very impressive that another such program, after playing against itself a billion games of Go, can now beat the world's best human players[30]. Great progress by such programs is made on human language understanding and translation, and perhaps their fastest growth is felt in "Data Science", where these programs play ever more important roles in actual scientific discovery. Indeed, one wonders at what point they will be able to better articulate the new scientific laws uncovered and pursue their consequences like medical drugs and treatments, food and energy supply, etc. also without human help[31].

The main point I find fascinating and challenging on this front, and for which the theory of computation can and should contribute, is a theoretical understanding of such algorithms. This understanding should include ways to make their training more principled and efficient, and developing models and tools to assess their performance and output quality. A major challenge is to understand why and for what tasks are deep networks successful, and what are their limitations. Further, it is of crucial importance given their growing prevalence in systems humans crucially depend on, to explore how to make them *fair* (and what this means), how susceptible they are to adversarial data, and how to protect against it. Of course, the size and complexity of deep nets may put limits to how well they can be theoretically understood (much like massive creations of nature as the brain and other biological systems). Indeed, it is not unlikely that the theoretical study of deep nets (on which, unlike animals, experimentation is free from the Declaration of Helsinki guidelines) will help in better understanding biological systems and vice versa.

Given our current ignorance (namely, gaps between upper and lower bounds) regarding many well posed important problems, I have no doubt that there is plenty more room for the discovery of algorithmic gems which we can easily understand, appreciate and use. To drive this point home, let me note in conclusions that no deep-nets would exist without a few great algorithmic gems embedded in practically all of them, including the extremely efficient *back propagation* and *gradient descent*

8 Some important challenges of ToC

Even at a high level, there are just too many important challenges of ToC to list here, especially in the wide contexts of its connections to the practical world of computing, and the broad connections to the sciences. Also, quite a number of important conjectures appear in the different chapters of the book [Wig17]. What I would like to single out here is four meta-challenges in the complexity theoretic core of ToC, which have many incarnations and formulations. These (naturally interrelated) challenges are certifying intractability, understanding heuristics, strengthening the foundations of cryptography and exploring the Church-Turing thesis.

8.1 Certifying intractability

By far, the greatest challenge of computational theory (and a major challenge of mathematics), is to establish that *some*, indeed any, natural computational task is *nontrivially* difficult for a general computational model. This question has many incarnations, but the sad state of affairs is that even though almost every natural task we really care to perform seems exponentially difficult, we cannot establish even super-linear lower bounds (namely, that a given task is harder than just reading the input)! The $\mathcal{P} \neq \mathcal{NP}$ conjecture is of course the most popular manifestation of this challenge (e.g. proving a lower bound for some $\mathcal{NP}$-complete problem like satisfiability of Boolean formulas or 3-coloring of maps). Similarly, establishing a non-trivial lower bound (if true) for even more basic (and practically burning) questions like integer multiplication or the discrete Fourier transform elude us as well. That for 80 years we had a formal mathematical model in which to prove such lower bounds, and certainly great motivation for doing, demonstrates the difficulty of this major problem!

[30] This happened with Chess too, but there theory and human involvement were an essential part of the program design, whereas for Go such involvement seems to have been minimal.

[31] Feel free to extrapolate (as some have already done in intellectual discussions and in popular culture) on the many natural societal questions which arise given such abilities of machines.

Essentially the same sad state of affairs exists in questions of *proof complexity*: we know no nontrivial lower bounds on the length of e.g. Frege proofs of *any* propositional tautology; this quest is manifested by the $\mathcal{NP} \neq \text{co}\mathcal{NP}$ conjecture. Likewise, in *arithmetic complexity*, we know no nontrivial lower bounds on the number of arithmetic operations (sums and products) needed to compute natural polynomials like the Permanent; this quest is manifested by the $\mathcal{VP} \neq \mathcal{VNP}$ conjecture. And the same holds for many other models and modes of computation; we can't prove hardness!

We do have "excuses", in the form of *barrier results*, which give partial explanations of why current techniques fail to deliver nontrivial Boolean lower bounds. But, we do not really understand why we have not yet found techniques to bypass these barriers. Furthermore, in proof complexity and arithmetic complexity we don't even have any excuses in the form of barrier results. In all models, the lower bounds can be expressed quite neatly in various forms as combinatorial and algebraic questions which do not betray any clear distinction from many other such questions that were resolved in these or other fields of mathematics (and in particular, the same lower bound questions for restricted models). So, understanding *why* proving nontrivial lower bounds is so difficult is one of the greatest challenges of the field, and probing this mystery abstractly may lead to progress on proving them. Of course, there are two explanations which must be mentioned, even though few (if any) believe them. First, that these conjectures are false, no lower bound exists, and everything is extremely easy to compute and prove. Second, that these conjectures are actually independent of mathematics, say of ZFC, and so can be true or false "as we please" from a provability standpoint, even though they have a truth value in the "real world"[32]. Ruling out the second option, without resolving these conjectures, is an extremely interesting direction as well.

The simplest explanation (which is probably the correct one), is that proving lower bounds for general models is truly one of the deepest and most difficult mathematical problems we know. Thus, probably far more groundwork, on a variety of limited computational models, is needed to gain a true understanding of general computational models and their limitations. Of all challenges of the field, this is the one I care about most, and hope that it will continue to attract excellent researchers to develop the necessary structural theory and techniques towards such lower bounds. I am certain that such understanding will not only expose the limitations of computation, but also its full power.

8.2 Understanding heuristics

We often get much more than we (theoretically) deserve. Many algorithms are known to behave extremely badly in *some* instances, but have excellent performance on "typical" ones. Let's discuss some families of examples.

- Some algorithms for diverse problems *seem to* perform far better (in time spent or output quality) than their theoretical analysis (if any) guarantees. Prototypical examples commonly given of such *heuristics* include SAT and TSP solvers on huge instances, the Simplex algorithm for linear programming. Of course, these heuristics are very clever, carefully optimized and fine-tuned.

- As discussed in more detail in Section 7.3 above, an explosion of heuristics arises from the new ability to train *deep networks* to solve learning, optimization and game playing problems. Also here there is a process of fine tuning, but it is largely automated in that mysterious, self-teaching "training" process. In many (but certainly not all) problems motivated by a variety of applications, these heuristics outperform carefully designed algorithms, or human experts, on many natural data sets.

- Another family of examples, of a very different nature, is the following. Probabilistic algorithms are designed to work correctly (with high probability) assuming that the random coin tosses they use are perfect (namely, independent and unbiased). But they often *seem to* perform equally well when fed a sequence of bits generated in some arbitrary deterministic fashion, or taken from physical sources like Internet traffic or user keystrokes. This (possibly surprising) phenomena happens, for example, in numerous Monte-Carlo simulations used by physicists.

[32] A way this can happen is e.g. the existence of a polynomial-time algorithm for SAT, whose correctness is independent of ZFC.

- Finally, nature itself provides us with many processes that *seem to* quickly solve "hard problems", like protein folding, formation of foams, market equilibria and coordinated action of swarms. Here of course we typically don't really know nature's algorithms, but may have models for them, which in many cases fail to predict or guarantee the observed (high quality) performance.

The general but vague explanation to such phenomena is that the *inputs* to all these algorithms and processes come from certain sets or distributions (which nature or humanity generate), for which the given heuristic is far more efficient than for arbitrary inputs. The suitability of a heuristic to the "real-world" inputs it processes can have many reasons. In some cases algorithm designers managed to put their fingers on the actual (or related) input distribution and design the algorithm to be super efficient on such inputs (even though in general it may be slow), as happens for some TSP and SAT solvers. In some cases algorithms were designed to be natural and elegant, like the Simplex algorithm, where its superb performance on real-life linear programs was surprising and indeed lucky; our understanding of it is lacking despite various later attempts to explain it. In yet other cases algorithms *evolved* to suit their input distribution, as happens for many neural-nets and other (supervised and unsupervised) learning frameworks. Moreover in nature, where such learning and evolution abound, it is quite possible that in many settings both the algorithm and the inputs *co-evolved* to yield superior performance; e.g. perhaps proteins which survived are in particular those which fold easily by the algorithm in our cells.

Understanding actual input distributions arising "in practice" for various computational tasks, and designing algorithms which handle them well (or explaining why certain heuristics do) is a very old and fundamental conundrum of the field. The many decades of algorithms research has provided theoretical models and frameworks which address this important issue from various viewpoints. General approaches include probabilistic analysis of algorithms [Kar76], semi-random models [FK01], smoothed analysis [ST04b] stable instances [BL12,BBG13], and input distributions with certain moment bounds [HS17,KS17] and many others[33]. Some of these approaches were developed, extended and applied significantly. Beyond such general approaches, which apply to wide families of algorithmic tasks, there are numerous papers which address specific tasks, targeting and taking advantage of their "natural" input distributions. This interaction between modeling inputs and algorithmic design is extremely important, and will hopefully lead to a better understanding of performance (and other behavioral properties) of such algorithms on "relevant" inputs.

With the advent of machine learning algorithms that interact and improve with the data, there are more and more heuristics whose performance (which in many applications is extremely impressive) is hard to explain. These, and other existing heuristic mysteries present a great challenge of developing evaluation criteria for the actual performance and behavior of such algorithms, that we discussed already in Section 7.3 above. Possibly missing is a theory of benchmarking, which will allow comparing heuristics and suggest rational guidelines for choosing the many parameters of learning systems before setting them loose on the data. Certainly missing is a theory explaining the kinds of problems and kinds of data that e.g. deep-nets are good at solving, those which are not, and why. The need for such theory is not only theoretical! We will soon have to *trust* such algorithms (when they'll replace humans e.g. in driving cars and in providing medical diagnoses and treatment), and *trusting* algorithms we *do not* understand is a very different matter than trusting algorithms we *do* understand. Most computer systems in existence are of the later type, but the balance is shifting. I believe that (beyond performance) theories modeling "social aspects" of program output should and will be developed, and some guarantees will have to be supplied by heuristics employed in the public sphere. These issues are already entering public policy and legal discourse (for one fascinating example, on the idea of delivering better justice by replacing judges by algorithms).

It is clearly hard to expect a fully predictive theory of algorithmic design which would yield the ideal, *instance optimal* performance. On the other hand, the growing ease by which one can completely neglect any theoretical analysis and resign oneself to the output given by general heuristics such as deep networks can be dangerous. My personal feeling is that there is plenty more theoretical understanding to be gained on modeling classes of real-world data, designing general algorithmic techniques to efficiently and accurately solve them and evaluating the performance of heuristics. This challenge is central to the field, and to society.

[33] Including average-case analysis [Lev86] which focuses on hardness rather than easiness.

I expect that the growing interactions with the natural sciences will add both new questions and new ideas to this study.

8.3 Resting cryptography on stronger foundations

Crypto is by far the most extensive and repeatedly surprising study of remarkable consequences of intractability assumptions. Furthermore, these consequences (and thus, the assumptions needed for them) form the current foundations of computer privacy and security and electronic commerce, which are used by individuals, companies, armies and governments. One would imagine that society would test these foundations and prepare for potential fragility at least as well as it does for, say, nuclear reactors, earthquakes and other potential sources of colossal disasters.

While most of cryptography requires the existence of *trap-door* functions, we have precious few candidates for these, of which integer factoring is still the a central one (other important candidates are based on elliptic curves, lattices and noisy linear equations). Moreover, short of the fact that these problems were not found to be easily solvable yet, we have no shred of evidence that any of them is really difficult. For example, an efficient algorithm for any of these few problems will not imply efficient algorithms for a large collection of many other problems which are believed hard, and will not imply the collapse of any complexity classes.

I find this state of affairs quite worrisome, and feel that far more effort should be devoted to the challenge of finding more secure foundations for cryptography. The world has too much to lose if these foundations collapse. And it should be stressed that cryptographic research is thriving, mainly expanding the frontier of applications and consequences of even stronger (so, less believable) intractability assumptions than trap-door functions[34]. Naturally, applications are often first discovered under strong or specific assumptions, which are later weakened. Still, caution is necessary, certainly when resting the security of actual systems on shaky assumptions.

Of course, part of this state of affairs, namely of having few reliable, useful assumptions, is not for lack of trying, and this challenge is indeed formidable. Given the first challenge of proving super-linear lower bounds for, well, anything, we can't expect yet super-polynomial lower bounds for e.g. factoring. One natural direction is simply developing more candidates for trap-door functions, in case others break. Here the difficulty seems to be the (often algebraic) structure these problems seem to "demand". Another is developing reductions between trap-door functions, to show that efficient algorithms for any of them have non-trivial consequences. Here again the rigid structure seems to resist reductions. Finally, resting cryptography on more believable intractability, preferably $\mathcal{P} \neq \mathcal{NP}$ (or at least its average-case version) runs into impossibility results against black-box use of such assumptions. Barak [Bar01] was the first to go beyond such black-box use; much more has followed. But it seems that further development of non-black-box crypto techniques (in which there definitely is exciting activity) is essential to this challenge.

8.4 Exploring physical reality vs. computational complexity

The celebrated Church-Turing thesis always struck me as an amazingly bold challenge. After all, it basically asserts that

everything computable is computable by a Turing machine.

where "everything" is any well defined function with discrete input and output domains. This thesis challenges anyone, scientists most prominently, to come up with such a well defined computational task, which can be performed somehow in the real world (including the ability to prepare the input and read-off the output), with any physical means whatsoever, but which cannot be computed by the Turing machine (one of the simplest physical devices imaginable). Still, in the 80 years that have passed, no one has come forth with a serious response to this challenge.

[34] The choice of proper cryptographic assumptions is itself a field of study in cryptography, discussed e.g in [GK16] and its references.

Soon after complexity theory was developed in the 1970s, people quickly extended this thesis to the far bolder thesis that it holds even if we restrict ourselves to *efficient* computation. This is sometimes called the "strong" or "feasible" Church-Turing thesis, and essentially states

everything efficiently computable is efficiently computable by a Turing machine.

Here "efficient" is usually equated with "polynomial" (although one can adopt more stringent efficiency measures). This means that physical resources expended in the computational process, like energy and mass, as well as time and space, which are used in the computational process are allowed to grow only polynomially with the input size.

This thesis, together with the $\mathcal{P} \neq \mathcal{NP}$ conjecture, presents a seemingly much more tangible challenge: simply find a natural process that can solve some $\mathcal{NP}$-complete problems with polynomial resources. And indeed, here came numerous suggestions, many (but certainly not all) explained and discussed in Aaronson's beautiful survey [Aar05]. Many such proposals strengthen the case for adding $\mathcal{P} \neq \mathcal{NP}$ (and more generally computational complexity considerations) as a guiding principle for modeling natural phenomena[35].

However, here I want to turn around the challenge provided by the strong Church-Turing thesis, and consider it not only as a challenge to scientists to either refute it or use computational complexity in modeling natural processes, but rather as a challenge to complexity theorists, to computationally abstract and model those (potential) gifts of nature which may eventually be used to enhance our laptops. Rising to this challenge (or rather challenges, as they arise from all walks of nature) has already been extremely important to foundational questions about efficient computation and algorithms, as well as to practice. Two of the main examples of exploring such gifts occupied several chapters in the book [Wig17]: the gift of randomness and the gift of "quantumness". I'll start with these, and then mention a few others that are less developed.

First, nature seems to provide us with plenty of unpredictable phenomena, leading to the extremely important (theoretically and practically) theories of probabilistic algorithms, probabilistic proofs, pseudo-randomness and randomness extraction. The first two theories postulate access to *perfect randomness*, namely a sequence of unbiased, independent coin tosses, for the benefit of enhancing computational power. The last two theories explore relaxing (in different ways) this strong postulate (of having access to perfect randomness), which may be violated in the real world. One explores having no access to randomness *at all* (and replaces it with a universally believed hardness assumption for "generating unpredictability"). The other assumes we can only access very weak random sources, e.g. sequences of highly biased and correlated bits (and explains how to enhance their quality).

Now, practical use of "random" sequences in computer programs, e.g. in Monte Carlo simulations for weather prediction and nuclear interactions predates all these theories, and moreover it seemed that in many cases it did not matter how arbitrary (including deterministic!) was the "random" source that was used to yield "useful" results. But of course, there were no guarantees in these situations, nor understanding when such sources "worked" and when they did not. The two theories above provide a formal basis to the study of randomness requirements of probabilistic algorithms, showing them provably applicable in settings way beyond the original requirement of perfect randomness. These theories justified and guided the choices of "random" sources to be used, identified sources which should not be used, and found efficient means to enhance weak ones. The de-randomization results (especially $\mathcal{BPP} = \mathcal{P}$) which follow from these theories impact the Church-Turing thesis directly: they justify replacing deterministic Turing machines with probabilistic ones (that err with small probability), as a legitimate definition of what efficient computation *is*[36]. Beyond all of that, as we discuss in the book [Wig17], the abstract study of randomness has been incredibly useful to the theory and practice of computation (and mathematics) in many unexpected ways, way beyond the original intent.

Next, nature actually seems to provide us with quantum mechanical phenomena, of which spitting out perfectly random coin tosses is among the simplest powers. Indeed, this power seems to promise a possibility of efficiently manipulating (some) exponentially long pieces of information! This promise led to the theoretical

[35] Which in many of these proposals also bring out the need to understand the actual inputs fed to these "heuristics of nature" in the sense discussed in the "heuristics challenge" above.

[36] This does not belittle the importance of using randomness nonetheless, when it gives polynomial speed-ups.

study of using such quantum phenomena in computation, formulation of the quantum mechanical Turing machine and the study of its power and limits. Here theory is far from fully developed. While we have formal universal models and equivalences between them, important understanding of certain quantum (decoherence) error correction, quantum cryptographic protocols and numerous interactions with physicists, we have only precious few problems such a quantum computer can solve efficiently that a classical (probabilistic) one cannot[37]. Fortunately or not, these few problems include integer factoring and discrete logarithms, so this model undermines the current foundations of most cryptographic security systems. Of course here (unlike with probabilistic computing) theory predated practice, and the algorithmic potential has led to the many, extremely challenging (and expensive) practical projects of building a quantum computer; literally attempting to add this quantum enhancement to our laptop and revising yet again our understanding of what an efficient Turing machine is.

The jury is out on which of the win-win outcomes will materialize: either we will have such quantum laptops (significantly enhancing our computational abilities), or we will discover a fundamental gap in our models of nature which prevent their existence (and with it, hopefully revise them and formulate better ones). But beyond motivating and assisting the practical project, the developed theories of quantum computing, proofs, advice, learning, games and more have been extremely illuminating to the *classical* theory of computation in many unexpected ways (leading to lower bounds, quantum reductions between classical problems, no-signaling PCPs for delegation of computation, certification of randomness and many others). Here, unlike with probabilistic computation, the belief is that quantum computation *does* add power to Turing machines (namely that $\mathcal{BQP} \neq \mathcal{BPP}$). I find the discovery of more problems in the gap, as well as reasons why despite it $\mathcal{NP} \not\subseteq \mathcal{BQP}$ as well, to be an extremely challenging direction for computational complexity theory.

What else can we add to our laptops, or networks, or other algorithms, which will enhance the notion of what can be efficiently computed? There are many other gifts of nature, possibly more specific than the two above, which were already turned into computational models and paradigms whose algorithmic power and limits are explored, and many others being pursued. As discussed, I find that besides the natural scientific interest in such (and many other) models for the benefit of understanding nature, their purely theoretical pursuit as computational models will be beneficial for broader understanding of computation itself, and perhaps to the discovery of new algorithmic techniques, possibly leading to the creation of new computational devices and systems. Many examples of natural phenomena with such potential are mentioned in this paper on the connections of ToC to the sciences above. I thus look forward to the discovery of the power, limits and relationships emanating from the study of such computational settings, formalized perhaps e.g. as DNA machines, nano machines, selfish networks, natural algorithms (among many others), enriching both the practice of computing as well as its theory. Some of these may lead to classes like nano-$\mathcal{P}$, selfish-$\mathcal{P}$, etc., and possible probabilistic, quantum, non-deterministic, space-bounded, non-uniform, ... variants, and new connections to previously studied models that will deepen our understanding of computation.

9 K-12 Education

This is a huge subject, so the discussion here is quite superficial and scattered, and this is aggravated by the lack of proper references (which may well exist). It can be regarded perhaps as a short wish-list, and a call-to-arms for the participation of ToC professionals in this effort.

The theory of computation has so much to offer in making the education curriculum (at all levels) of every person so much better, richer, more valuable and fun that it would take a separate book to discuss this. Of course, I am not original here, and there are plenty of thoughts and projects underway (with articles describing them) which suggest such efforts of integrating "computational thinking" into classrooms at various ages in various ways. As usual, even very good ideas have to eventually be implemented. As with other subjects, besides deciding what and how to teach kids, the most nontrivial task is figuring out how to train teachers to successfully deliver this material, with enthusiasm, to students. To make better decisions

[37] Beyond the basic problem of simulating quantum systems.

on both principles and implementation, I strongly hope that members of our community, who are inclined to do so, will be involved in these developments—this challenge is at least as important as exploring the scientific challenges of our field.

Below I list a small sample of truths I wish all kids would learn (at the appropriate age) in a way that will excite them and make them (via appropriate examples and demonstrations) remember them forever, no matter what life interests they end up pursuing. I believe these capture examples of the kind of minimal knowledge and understanding about computation that all educated people should possess, in the same sense that we expect them to possess elementary knowledge and understanding about other topics such as the basic sciences, history, social systems, etc.

- Turing, like Einstein, was one of the giants of 20th century science. His influence on technology and society was far greater.
- Computers cannot do everything. Indeed, there are basic, desirable computational tasks they can never accomplish. This is an absolute truth (a mathematical theorem), just like Pythagoras' theorem.
- What computers can do relies on the discovery of clever, efficient algorithms. One such algorithm can create a new industry, or predict and prevent an epidemic. The great discoverers of algorithms sit in the same hall of fame with great inventors, like Gutenberg, Edison, Pasteur,...
- Theories of nature should be predictive. Prediction is an algorithm! Predicting the weather, or natural disasters *before* they happen requires these algorithms to be very efficient, *faster* than nature!
- Most math problems don't have only one right answer (with all others "wrong"). As in life, many mathematical tasks have solutions of different quality. Finding any solution is a good start, and trying to improve it is even better (this is discussed more below).
- The place value (e.g. decimal or binary) system that we all use to represent numbers was invented, and survived for millennia, only since it afforded extremely efficient storage and algorithms for performing arithmetic. Compare it to the unary system, or Roman numerals, ...
- The method we learn to multiply numbers in grade school is firstly, an *algorithm*, and moreover, it is *not* the fastest algorithm known. There is a much faster one, which is still slower than the algorithm we have for adding numbers. It is a great intellectual mystery, with great practical implications, whether multiplication is inherently harder than addition.
- The world economy assumes computational hardness (at this point in time, this is the assumption that "inverting multiplication"– finding the prime factors of a large number – is an impossibly hard problem). Moreover, world economy may collapse if this (or related) computational assumptions are false.

A bigger issue, in which I believe that ToC education has much to offer is math education in elementary and high school. A lot has been written on the subject, and I will be brief on this point. For a more elaborate text, see this article by Fellows [Fel91], who has also invested much effort in developing classroom material for elementary schools and moreover has experimented with delivering it.

A recognized major malady of math education is the focus on arithmetic and algebra, with repetition ad nauseum of numerous, equivalent, boring numerical exercises to which the answer can be either right or wrong. In contrast, algorithmic problems on discrete objects offer, from kindergarten age on, an enormous variety of problems on e.g. games, puzzles, mazes, which offer solutions of varied qualities and properties. For example, in optimization problems, every solution may be "correct", but their cost differ. Or, many different algorithms, strategies and methods may solve a problem, but the amount of time or other resources they expand differ. In such problems, solutions of students invite the challenge *can you do better?* that is so much more constructive than *wrong!* Such problems also far better relate to the basic human experience all students feel resolving life conflicts, from scheduling their activities to maximizing something (interest, pleasure, their

parents' satisfaction, ...), spending their allowance or playing better chess or soccer. Such problems also better relate to the view expanded on above of natural *processes* (themselves algorithms typically solving optimization problems), which should be highlighted in science courses, making a connection to math rather than through equations alone. Problems in which solutions of different qualities exist, and the quest to find better and better ones, allow students deeper understanding and, I believe, much more fun. Moreover, such problems expose math much better as a *modeling* science, which allows precise formulation of such life and science situations, as well as aspects like chance, uncertainty, partial information and adversity. I feel that incorporating these ideas into math education, first for teachers and then for students is very important (and I am not underestimating the difficulty of this task). This material should be part of every young person's knowledge in this information age, but at the same time teaching it can also help transform the sad, negative attitude of many generations towards the very subject of mathematics. Repeating myself, these ideas are not original and indeed are attempted in many places - I am simply calling here for more participation of ToC professionals, to whom this way of thinking is so natural, in developing the materials and implementation details for this important endeavor.

10 The ToC community

I believe that the remarkable success of the field, partly summarized above, was not purely due to the intellectual content of ToC and the raw talents of its researchers, but also the way the ToC community organized and built itself. This section is devoted to this process and character as I have experienced it, through very active participation in all relevant activities, in all stages of my career. I allow myself generalities when the exceptions I know of are rare, fully realizing that my view is biased and that I tend to idealize.

Since the 1960s much activity of ToC has been centered at and propelled by two yearly conferences, FOCS (Foundations Of Computer Science) and STOC (Symposium on the Theory Of Computing)[38]. Taking place in the Fall and Spring respectively, these forums became the meeting place (and the beating heart) of the whole community, a good fraction of which came to hear accounts of the most exciting developments in the field in the last 6 months. These presentations are determined by a program committee, which has no time to referee submissions, but rather to evaluate their novelty and impact on progress in the field. These have two effects. One is in the committee meetings themselves: 20 or so experts, of different research areas, opinions (and different ages!) meet periodically to discuss the submissions and make tough selection decisions, help clarify contributions and shape directions. Second is the periodical meetings of a significant chunk of the ToC community, including many graduate students, at these conferences. The constant intellectual excitement about the challenges supplied in abundance by the field, and undoubtedly the personalities of its elders, have created a culture whose characteristics I'd like to list (and idealize). They undoubtedly play an important role in the success of the field so far, and it will be great to preserve and strengthen many of these as the growth and diversification of the field continue (which may be nontrivial). Needless to say, many of them are part of other academic disciplines, but I find their combination in ToC quite unique.

- *Level of discourse*. The need to convince a program committee, many members of which are not experts on your submission, has brought submissions to include long introductions with high level motivation for the research described. Further, it has impacted the inclusion of intuitive explanations of technical developments. Restricted to being short, submissions therefore focus on ideas and concepts, which eased their propagation to non-experts. The exact same effect has amplified in the lectures. As many members of the audience are non-experts, motivation and intuition occupy significant parts of the short lectures. Thus papers and lectures help highlight ideas and concepts, which can easily be understood by many. Moreover, this level of discussion allowed the use and generalization of ideas and concepts developed in one context, possibly for a particular problem or application in completely different ones. This common high-level language, and the frequent exchange of the most recent important findings,

[38]These conferences generally take place in North America, and I have attended most of these since 1980. Other conferences which encompass most areas in ToC, with some biases or different geographic locations were later added with a similar broad appeal, including ICALP, MFCS, SODA, ITCS, and what I say below is relevant to them as well.

has undoubtedly been responsible for the fast progress of the field. I believe that this too has an impact on the way ToC faculty teach their courses and write their books and surveys, putting stress on highlighting motivation, ideas, concepts and intuition before and beyond technical definitions and proofs.

- *Openness and evolution*. The horn of plenty which computation is, the diversity of its sources and manifestations, together with the varied interests of its researchers, has generated completely novel models, questions and research directions. These new interests competed for attention of the community with making progress on existing and more established directions and open problems. It fell on the same bodies, program committees of these conferences, to become the arbiters of the balance of new and old, and which parts of each, to select for acceptance and presentations. These frequent discussions and debates, among researchers of different ages and tastes, made more valuable by the common language above, not only inform the actual participants about the broad activities in the fields and their colleagues' opinions on them, but has created a forum which had repeatedly to make tough choices on which of these trends will be visible and in which proportions. Looking back at the historical evolution of trends and directions one can observe a great agility to pursue and investigate, as well as great openness to embrace and advertise, completely new ideas, models, connections and problems (emanating from technological, scientific or purely abstract considerations). This has created a wonderful dynamic, giving birth to many new research directions that were incubated at these forums. Many of these (e.g. most chapter titles in the book [Wig17] are perfect examples) grew sufficiently large to establish their own specialized conferences and communities, often operating in the same style. Of course, some other research directions have shrunk, died out or merged with others.

- *A common core*. Despite this dynamic of topics entering and leaving this spotlight, central themes, mainly in the core algorithmic and complexity theoretic areas of the field, maintained a long term presence even if they grew as above. Essential to the discovery of many fundamental principles (of broad applicability across ToC) came from studying purely mathematical, *seemingly* "un-motivated" models and problems, which *seemed* "impractical" or "un-natural" generalizations of existing ones, or just came out of left field. In short, following internal logic and aesthetics of the field, common in mathematics but much less so in application-driven areas. The openness of the community to such ideas paid back big time, and kept re-establishing the value of the core. Another key factor that made possible, as well as strengthened, the core was maintaining the level and style of discourse described above, despite the tremendous expansion in diversity of areas covered. Naturally, this diversity necessitated many changes in the structure of the conferences and the sizes of program committees, but still allowed the exchange of ideas, concepts and methods across and between these very differently motivated and structured research directions. Moreover, the rate, fluidity and quality of this exchange has clarified to what extent ToC is a cohesive field, and how important it is to have a core and an infrastructure to utilize it. In many cases this flow of ideas, methods and results was extremely natural, whereas in many others it required remarkable ingenuity, and of course knowledge of what was going on in other areas. In a lecture on the value of such knowledge I dubbed this phenomenon *depth through breadth*, the value and strength of which I find unique to our field. Some of the commonality of these many threads is captured in the methodology section above.

- *Community and attraction of talent*. Challenging and important long term goals, a rich tapestry of concepts, ideas and tools with exciting progress in connecting and expanding them, the building of theories to gradually access such understanding, and many other intellectual facets of a scientific field are necessary prerequisites to continuously attract young talent in the field , who will further this work. And indeed, the theory of computation has all that and more, as this paper and indeed the whole book [Wig17] exposit. But I would argue that the social and educational structure of the field makes it even more attractive for young talent to enter, stay, grow and hopefully also discover, educate and lead. I have already explained how the level of discourse makes it easy and inviting to newcomers, especially those exposed to it in undergrad and grad schools, but also for researchers of different fields who are interested. Another important aspect is the "democracy" and social welcoming of the field,

which generally considers raw ideas and not the seniority of those generating them. One of my formative experiences, as a first year graduate student, was being introduced in the FOCS conference of 1980 to Richard Karp, *the* most eminent leader of the field. It was "Dick, this is Avi, Avi, this is Dick", which followed by five or ten minutes in which Karp, with genuine interest, explored what areas of theory I like and might be pursuing. I quickly learned that no introductions were even needed, and that leaders of the field were equally accessible when you come to them with a problem or idea. I believe that this is the experience felt to this day by newcomers to the field, and that it has created a community of much collegiality, collaboration and friendship. This atmosphere persists despite the natural and much welcome competitiveness and drive of ToC researchers.

Critique

After these many positive aspects, I think it makes sense to mention a couple of issues where I feel that our community has behaved sub-optimally, in the hope that these will improve over time. They are all related, and probably stem from a single root cause, namely an internal focus on the intellectual problems of the field, neglecting to some extent a necessary investment in the external spheres in which we live and function. By far the most significant neglect is in communicating with the outside world about ToC achievements, both their intellectual contents and their practical impact. Of course, some ToC members invested in popular lectures, surveys and books for the general CS community, the general scientific community and even the public over the years. But I fear that what was achieved by them is far smaller than could have been achieved with much more (well justified) effort. There are many ways to demonstrate the general low level of understanding, acknowledgement and appreciation of the amazing body of work created by ToC, outside the field. This state of affairs could easily improve, as many of our discoveries has such appealing conceptual contents, and so are easy to relate to almost any audience. In short, I feel that the field should make it a priority, and hope that much more will be done on this important front.

One way which makes it easier to communicate a topic, especially for the general public, but also for people outside one's domain, is having an evocative vocabulary to describe central notions and results. By far the field that excels over all others in this domain is physics. Of course, it starts with a huge natural advantage over most other fields, as their objects of study have aroused our curiosity and fascination from a very early age, like the stars, rainbows, ocean waves, flying birds and so many more. But even for objects of study to which we have no direct access they pick intriguing and captivating names like "big bang", "black holes", "dark matter", "super novae", etc., and call some elementary particles "strange" and "charm". In contrast, not only does ToC start with no one having any computational or algorithmic experience in a formative age, we call our main concepts by boring, inexplicable acronyms like P, NP, BPP, SC, NC... (and these are just a few of the most famous animals in the "complexity zoo"[39]). There are numerous other choices made which I believe could be much better, and perhaps the best example is "complexity" itself (even in the phrase "computational complexity"), which is far too general and is constantly confused with other uses and meanings of this word. Much better phrases for what we are actually studying would have been "computational efficiency" and "computational intractability". Of course, some choices are excellent in being both meaningful and intriguing, like the concepts "zero knowledge proofs", "Monte Carlo algorithms", "randomness extractors", "price of anarchy" and the problems "dining philosophers", "Byzantine generals", "perfect matching" and "maximum clique". I would argue that bad naming choices hinder our internal education of undergraduate and graduate ToC students, not only our ability to communicate our work outside the field! While some of the bad choices are too engrained and thus possibly too late to change, I feel that being conscious of this issue may give birth to better names of new concepts, and inspire inventive names for old concepts which the community can adopt in popular lectures (where using professional jargon is not necessary).

[39] A website of most complexity classes maintained by Scott Aaronson.

11 Conclusions

Scientific disciplines are born (and transformed) in different ways; volumes have been written on the subject. In the case of ToC it is remarkably easy to determine the precise moment of birth. It happened in a singular event, following an age-old process. Computation, in its manifold forms, has been a subject of inquiry since the dawn of man, simply since the efficient utilization of resources is a second (or first) nature to people (and, to all life forms, and indeed to inanimate matter subject to the laws of physics). But this inquiry was missing a formalism that would make it precise. This formalism was supplied in 1936 by Turing's seminal paper, which gave us the *Turing machine* model. It struck gold, finally enabling the creation of a theory! Computation revealed itself as a bottomless goldmine, and what was already uncovered by ToC research stands as one of the greatest achievements in scientific history.

In the sections above I discussed many facets of the Theory of Computation, especially its past and potential future contributions to human inquiry and to society. Its pure academic, mathematical pursuit probes some of the deepest intellectual questions asked by man. And as a bonus, it is of fundamental importance to most other sciences, and to technology and human life. In short, this paper present the clear case that ToC is an independent, important academic discipline. I feel that this view should be the basis of any discussion (internal or external) regarding the field and its future.

This position of ToC in the intellectual arena comes with great responsibility. To continue and develop both internally and in the expanding and dynamic collaborations with other disciplines, I believe that the ToC community needs to grow, and such growth presents new challenges. Some of these will be adequately met by the spirit and social culture of this young field, which I hope will be preserved. Others have to do with the changing nature of the field and its increased diversity, and I feel require significant effort concentrated on education and on cohesion. Let me elaborate a bit.

ToC is in the beginning phase of what I expect will be a long-lasting endeavor. Like other centuries-old disciplines such as mathematics, physics, biology and philosophy, ToC may change form and focus many times in the future. I find that the current period is definitely one of such change, possibly a phase transition, due to the explosion of connections of the field to other scientific disciplines, and the ever growing demands for theory from the even larger explosion of CS&E, generating new technologies and applications of computing systems at an unbelievable pace. These present enormous challenges, as both the nature and size of ToC are changing. I find a couple of (related) predictions inevitable, and each calls for action on the part of the ToC community. First, as the need for computational modeling and algorithm design throughout the sciences and industry expands, many more professionals will have to be trained in its principles and methodology. These will become prerequisite in most undergraduate education, and I strongly feel that ToC must play an active role in the design of these courses and curricula. Second, computational theory research and researchers will become integral parts of disciplines outside CS, and at the same time ToC research itself (which started as a pure math pursuit of computer system motivated problems) will need to adapt, and adopt experimental aspects of scientific and social theories studying the real world.

This great expansion and diversity, not only of research topics but also of methods and culture, is certainly a blessing, and indeed part of the calling of ToC. But it also means that maintaining a viable, central core of the field will become much harder. Hard as it is, however, I strongly believe that preserving such a core, which will allow the flow and exchange of fundamental ideas, techniques and insights about computation relevant across disciplines, will be at least as important as it was in the past. Conversely, I fear that fragmentation of the field can lose this important advantage! The challenge of preserving the cohesion of ToC is certainly nontrivial, and will require thoughtful reorganization and adaptation. Besides creating physical and on-line forums to maintain this cohesion and exchange of ideas, I find that it can greatly benefit from administrative changes in undergraduate education. Specifically, the creation of majors, certificates and other undergraduate programs dedicated to ToC is natural. There is a wealth of material regarding foundations and connections of the field, and it is ripe for creating a highly rich and challenging curricula. *The value, to both academia and industry, of a student graduating with this knowledge and understanding of theory, is extremely high.* Looking further, I feel that given the growing intellectual centrality of ToC to so many academic departments, (and even though CS will probably remain the strongest connection), it may become natural and beneficial, for universities and the field itself, to create *separate* ToC departments; this

will reflect and reinforce both the field's independence and its centrality to others.

Building and maintaining the necessary structures, like those suggested above, to preserve cohesion of ToC, can only exist if the community is confident in its independence, value and mission. Namely, it can happen only if ToC members fully appreciate the meaning, importance, scope and impact of past achievements, and the even greater potential of future ones. These conceptual messages should be an essential part of the education we provide and propagate to our students (and each other), well beyond the technical know-how. Such a foundation will hopefully facilitate further constructive discussion on the organization of the field, which I find so important for its future.

Acknowledgements

I am grateful to the following people who read earlier versions of this survey and gave me extremely useful comments: Scott Aaronson, Oded Goldreich, Pravesh Kothari, Toni Pitassi, Tim Roughgarden, Alistair Sinclair, Arpita Tripathy, Edna Wigderson, Yuval Wigderson and Rich Zemel. I am especially indebted to Oded Goldreich for his careful and constructive scrutiny of every word in this manuscript, and for our long partnership in debates, discussions and promotion of ToC. Finally, I am grateful for so many members of our community for discussions and lectures which helped shape and clarify my views about our field.

References

[Aar05] Scott Aaronson. Guest column: NP-complete problems and physical reality. *ACM Sigact News*, 36(1):30–52, 2005. 32

[Aar13] Scott Aaronson. Why philosophers should care about computational complexity. *Computability: Turing, Gödel, Church, and Beyond*, pages 261–328, 2013. 24

[Aar16] Scott Aaronson. The complexity of quantum states and transformations: From quantum money to black holes. *arXiv preprint arXiv:1607.05256*, 2016. 19, 20

[ABBG10] Sanjeev Arora, Boaz Barak, Markus Brunnermeier, and Rong Ge. Computational complexity and information asymmetry in financial products. In *ICS*, pages 49–65, 2010. 23

[ABL02] Sanjeev Arora, Béla Bollobás, and László Lovász. Proving integrality gaps without knowing the linear program. In *Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on*, pages 313–322. IEEE, 2002. 6

[Adl94] Leonard M Adleman. Molecular computation of solutions to combinatorial problems. *Nature*, 369:40, 1994. 15

[AHK12] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012. 6

[ALM+98] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998. 6

[AMPS13] Ahmed Almheiri, Donald Marolf, Joseph Polchinski, and James Sully. Black holes: complementarity or firewalls? *Journal of High Energy Physics*, 2013(2):62, 2013. 19

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM (JACM)*, 45(1):70–122, 1998. 6

[AS03] Sanjeev Arora and Madhu Sudan. Improved low-degree testing and its applications. *Combinatorica*, 23(3):365–426, 2003. 7

[AZGMS14] Zeyuan Allen-Zhu, Rati Gelashvili, Silvio Micali, and Nir Shavit. Sparse sign-consistent johnson–lindenstrauss matrices: Compression with neuroscience-based constraints. *Proceedings of the National Academy of Sciences*, 111(47):16872–16876, 2014. 18

[AZH16] Zeyuan Allen-Zhu and Elad Hazan. Variance reduction for faster non-convex optimization. In *International Conference on Machine Learning*, pages 699–707, 2016. 6

[AZO14] Zeyuan Allen-Zhu and Lorenzo Orecchia. Linear coupling: An ultimate unification of gradient and mirror descent. *arXiv preprint arXiv:1407.1537*, 2014. 6

[Bar86] David A. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 1–5. ACM, 1986. 12

[Bar01] Boaz Barak. How to go beyond the black-box simulation barrier. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 106–115. IEEE, 2001. 31

[Bar16] Alexander Barvinok. *Combinatorics and Complexity of Partition Functions*. Springer, 2016. 9

[BBG13] Maria-Florina Balcan, Avrim Blum, and Anupam Gupta. Clustering under approximation stability. *Journal of the ACM (JACM)*, 60(2):8, 2013. 30

[BCHP17] Sylvain Benoit, Jean-Edouard Colliard, Christophe Hurlin, and Christophe Pérignon. Where the risks lie: A survey on systemic risk. *Review of Finance*, 21(1):109–152, 2017. 23

[Bec91] József Beck. An algorithmic approach to the lovász local lemma. i. *Random Structures & Algorithms*, 2(4):343–365, 1991. 9

[Ben80] Paul Benioff. The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines. *Journal of Statistical Physics*, 22(5):563–591, 1980. 13

[BFL91] László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational complexity*, 1(1):3–40, 1991. 7

[BGBD+04] Yaakov Benenson, Binyamin Gil, Uri Ben-Dor, Rivka Adar, and Ehud Shapiro. An autonomous molecular computer for logical control of gene expression. *Nature*, 429(6990):423–429, 2004. 15

[BGC15] Yoshua Bengio, Ian J Goodfellow, and Aaron Courville. Deep learning. *An MIT Press book in preparation. Draft chapters available at* http://www.iro.umontreal.ca/~bengioy/dlbook, 2015. 27

[BL12] Yonatan Bilu and Nathan Linial. Are stable instances easy? *Combinatorics, Probability and Computing*, 21(05):643–660, 2012. 30

[BLNPL14] Moshe Babaioff, Brendan Lucier, Noam Nisan, and Renato Paes Leme. On the efficiency of the walrasian mechanism. In *Proceedings of the fifteenth ACM conference on Economics and computation*, pages 783–800. ACM, 2014. 22

[BLR93] M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences*, 47(3):549–595, 1993. 7

[BOGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 1–10. ACM, 1988. 22

[BPR15] Nir Bitansky, Omer Paneth, and Alon Rosen. On the cryptographic hardness of finding a nash equilibrium. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 1480–1498. IEEE, 2015. 21

[BRW05] Robert D Barish, Paul WK Rothemund, and Erik Winfree. Two computational primitives for algorithmic self-assembly: Copying and counting. *Nano letters*, 5(12):2586–2592, 2005. 16

[BS14] B. Barak and D. Steurer. Sum-of-squares proofs and the quest toward optimal algorithms. In *Proc of ICM*, 2014. 6

[CC17] Jin-Yi Cai and Xi Chen. *Complexity Dichotomies for Counting Problems: Volume 1, Boolean Domain*. Cambridge University Press, 2017. 8

[CDT09] Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Settling the complexity of computing two-player Nash equilibria. *Journal of the ACM (JACM)*, 56(3):14, 2009. 21

[Cha12] Bernard Chazelle. Natural algorithms and influence systems. *Communications of the ACM*, 55(12):101–110, 2012. 18, 23

[Cha15] Bernard Chazelle. An algorithmic approach to collective behavior. *Journal of Statistical Physics*, 158(3):514–548, 2015. 18, 23

[CKL13] Bruce Ian Carlin, Shimon Kogan, and Richard Lowery. Trading complex assets. *The Journal of finance*, 68(5):1937–1960, 2013. 23

[CKM+11] Paul Christiano, Jonathan A Kelner, Aleksander Madry, Daniel A Spielman, and Shang-Hua Teng. Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 273–282. ACM, 2011. 6

[CLPV14] Erick Chastain, Adi Livnat, Christos Papadimitriou, and Umesh Vazirani. Algorithms, games, and evolution. *Proceedings of the National Academy of Sciences*, 111(29):10620–10623, 2014. 17

[CLRS16] Siu On Chan, James R Lee, Prasad Raghavendra, and David Steurer. Approximate constraint satisfaction requires large LP relaxations. *Journal of the ACM (JACM)*, 63(4):34, 2016. 6

[CRVW02] Michael Capalbo, Omer Reingold, Salil Vadhan, and Avi Wigderson. Randomness conductors and constant-degree lossless expanders. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 659–668. ACM, 2002. 7

[DA01] Peter Dayan and Laurence F Abbott. *Theoretical neuroscience*, volume 806. Cambridge, MA: MIT Press, 2001. 18

[Der17] Zahra Derakhshandeh. *Algorithmic Foundations of Self-Organizing Programmable Matter*. PhD thesis, Arizona State University, 2017. 16

[Deu85] David Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 400, pages 97–117. The Royal Society, 1985. 13

[DFH+15] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Leon Roth. Preserving statistical validity in adaptive data analysis. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, pages 117–126. ACM, 2015. 26

[DG16] Zeev Dvir and Sivakanth Gopi. 2-server PIR with subpolynomial communication. *Journal of the ACM (JACM)*, 63(4):39, 2016. 7

[DGP09] Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The complexity of computing a Nash equilibrium. *SIAM Journal on Computing*, 39(1):195–259, 2009. 21

[DK16] Vincent Danos and Heinz Koeppl. Self-assembly and self-organization in computer science and biology (dagstuhl seminar 15402). In *Dagstuhl Reports*, volume 5. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016. 16

[DK17] Irit Dinur and Tali Kaufman. High dimensional expanders imply agreement expanders. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 24, page 89, 2017. 7

[DSVW04] Martin Dyer, Alistair Sinclair, Eric Vigoda, and Dror Weitz. Mixing in time and space for lattice spin systems: A combinatorial view. *Random Structures & Algorithms*, 24(4):461–479, 2004. 9

[DSW14] Zeev Dvir, Shubhangi Saraf, and Avi Wigderson. Improved rank bounds for design matrices and a new proof of Kelly's theorem. In *Forum of Mathematics, Sigma*, volume 2. Cambridge University Press, 2014. 7

[Efr12] Klim Efremenko. 3-query locally decodable codes of subexponential length. *SIAM Journal on Computing*, 41(6):1694–1703, 2012. 7

[EK10] David Easley and Jon Kleinberg. *Networks, crowds, and markets: Reasoning about a highly connected world*. Cambridge University Press, 2010. 23

[Eli57] Peter Elias. List decoding for noisy channels. 1957. 7

[Fel91] Michael R Fellows. *Computer science and mathematics in the elementary schools*. University of Victoria, Department of Computer Science, 1991. https://larc.unt.edu/ian/research/cseducation/fellows1991.pdf. 34

[Fey82] R. P. Feynman. Simulating physics with computers. *International journal of theoretical physics*, 21(6):467–488, 1982. 13, 15

[Fey86] Richard P Feynman. Quantum mechanical computers. *Foundations of physics*, 16(6):507–531, 1986. 15

[FGL+96] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Interactive proofs and the hardness of approximating cliques. *J. ACM*, pages 268–292, 1996. 6

[FK01] Uriel Feige and Joe Kilian. Heuristics for semirandom graph problems. *Journal of Computer and System Sciences*, 63(4):639–671, 2001. 30

[FMP+15] Samuel Fiorini, Serge Massar, Sebastian Pokutta, Hans Raj Tiwary, and Ronald De Wolf. Exponential lower bounds for polytopes in combinatorial optimization. *Journal of the ACM (JACM)*, 62(2):17, 2015. 6

[For66] G. D. Forney. *Concatenated codes*, volume 11. Citeseer, 1966. 7

[Gal62] Robert Gallager. Low-density parity-check codes. *IRE Transactions on information theory*, 8(1):21–28, 1962. 7

[GGOW15] Ankit Garg, Leonid Gurvits, Rafael Oliveira, and Avi Wigderson. A deterministic polynomial time algorithm for non-commutative rational identity testing. *arXiv preprint arXiv:1511.03730*, 2015. 6

[GHSY12] Parikshit Gopalan, Cheng Huang, Huseyin Simitci, and Sergey Yekhanin. On the locality of codeword symbols. *IEEE Transactions on Information Theory*, 58(11):6925–6934, 2012. 7

[GJL16] Heng Guo, Mark Jerrum, and Jingcheng Liu. Uniform sampling through the Lovász local lemma. *arXiv preprint arXiv:1611.01647*, 2016. 9

[GK16] Shafi Goldwasser and Yael Tauman Kalai. Cryptographic assumptions: A position paper. In *Theory of Cryptography Conference*, pages 505–522. Springer, 2016. 31

[GL89] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 25–32. ACM, 1989. 7

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. *Proceedings of the 19th annual ACM Symposium on Theory of Computing, ACM Press, New York*, pages 218–229, 1987. 22

[GMW91] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity, or all languages in NP have zero-knowledge proof systems. *J. ACM*, 38(1):691–729, 1991. 25

[GR08] V. Guruswami and A. Rudra. Explicit codes achieving list decoding capacity: Error-correction with optimal redundancy. *Information Theory, IEEE Transactions on*, 54(1):135–150, 2008. 7

[Gri01a] D. Grigoriev. Complexity of Positivstellensatz proofs for the knapsack. *Computational Complexity*, 10(2):139–154, 2001. 6

[Gri01b] D. Grigoriev. Linear lower bound on degrees of Positivstellensatz calculus proofs for the parity. *Theoretical Computer Science*, 259(1):613–622, 2001. 6

[GS98] Venkatesan Guruswami and Madhu Sudan. Improved decoding of Reed-Solomon and algebraic-geometric codes. In *Foundations of Computer Science, 1998. Proceedings. 39th Annual Symposium on*, pages 28–37. IEEE, 1998. 7

[GS00] Oded Goldreich and Shmuel Safra. A combinatorial consistency lemma with application to proving the PCP theorem. *SIAM Journal on Computing*, 29(4):1132–1154, 2000. 7

[GS06] Oded Goldreich and Madhu Sudan. Locally testable codes and PCPs of almost-linear length. *Journal of the ACM (JACM)*, 53(4):558–655, 2006. 7

[GS14] Leonid Gurvits and Alex Samorodnitsky. Bounds on the permanent and some applications. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 90–99. IEEE, 2014. 9

[GV08] Anita Goel and Viola Vogel. Harnessing biological motors to engineer systems for nanoscale transport and assembly. *Nature nanotechnology*, 3(8):465–475, 2008. 16

[GW96] Oded Goldreich and Avi Wigderson. Theory of computation: A scientific perspective. 1996. http://www.wisdom.weizmann.ac.il/~oded/toc-sp2.html. 3

[Har16] Daniel Harlow. Jerusalem lectures on black holes and quantum information. *Reviews of Modern Physics*, 88(1):015002, 2016. 19

[HH13] Daniel Harlow and Patrick Hayden. Quantum computation vs. firewalls. *Journal of High Energy Physics*, 6:085, 2013. 20

[HS17] Samuel B Hopkins and David Steurer. Bayesian estimation from few samples: community detection and related problems. *arXiv preprint arXiv:1710.00264*, 2017. 30

[IKW12] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. New direct-product testers and 2-query pcps. *SIAM Journal on Computing*, 41(6):1722–1768, 2012. 7

[IP01] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62:367–375, 2001. 6

[IPZ01] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001. 6

[Jer92] Mark Jerrum. Large cliques elude the metropolis process. *Random Structures & Algorithms*, 3(4):347–359, 1992. 9

[JP04] Neil C Jones and Pavel Pevzner. *An introduction to bioinformatics algorithms*. MIT press, 2004. 15

[JS89] Mark Jerrum and Alistair Sinclair. Approximating the permanent. *SIAM journal on computing*, 18(6):1149–1178, 1989. 8

[JS93] Mark Jerrum and Gregory B Sorkin. Simulated annealing for graph bisection. In *Foundations of Computer Science, 1993. Proceedings., 34th Annual Symposium on*, pages 94–103. IEEE, 1993. 9

[JS96] M. Jerrum and A. Sinclair. The Markov chain Monte Carlo method: an approach to approximate counting and integration. *Approximation algorithms for NP-hard problems*, pages 482–520, 1996. 9

[JSV04] M. Jerrum, A. Sinclair, and E. Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *Journal of the ACM (JACM)*, 51(4):671–697, 2004. 9

[JVV86] M. R. Jerrum, Leslie G. Valiant, and V. V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43:169–188, 1986. 8

[Kar76] Richard M Karp. The probabilistic analysis of some combinatorial search algorithms. *Algorithms and complexity: New directions and recent results*, 1:19, 1976. 30

[Kar84] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–394, 1984. 6

[Kho02] S. Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 767–775. ACM, 2002. 6

[Kle00] Jon Kleinberg. The small-world phenomenon: An algorithmic perspective. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 163–170. ACM, 2000. 23

[KLOS14] Jonathan A Kelner, Yin Tat Lee, Lorenzo Orecchia, and Aaron Sidford. An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 217–226. SIAM, 2014. 6

[KLS01] Michael Kearns, Michael L Littman, and Satinder Singh. Graphical models for game theory. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 253–260. Morgan Kaufmann Publishers Inc., 2001. 21

[KMRZS17] Swastik Kopparty, Or Meir, Noga Ron-Zewi, and Shubhangi Saraf. High-rate locally correctable and locally testable codes with sub-polynomial query complexity. *Journal of the ACM (JACM)*, 64(2):11, 2017. 7

[KP99] Elias Koutsoupias and Christos Papadimitriou. Worst-case equilibria. In *Stacs*, volume 99, pages 404–413. Springer, 1999. 21

[KS09] Neeraj Kayal and Shubhangi Saraf. Blackbox polynomial identity testing for depth 3 circuits. In *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*, pages 198–207. IEEE, 2009. 7

[KS17] Pravesh K Kothari and David Steurer. Outlier-robust moment-estimation via sum-of-squares. *arXiv preprint arXiv:1711.11581*, 2017. 30

[KSZ+10] Seung Hyeon Ko, Min Su, Chuan Zhang, Alexander E Ribbe, Wen Jiang, and Chengde Mao. Synergistic self-assembly of RNA and DNA molecules. *Nature chemistry*, 2(12):1050–1055, 2010. 16

[Lev86] L. A. Levin. Average case complete problems. *SIAM J. Comput.*, 15(1):285–286, 1986. 30

[Lip95] Richard J Lipton. DNA solution of hard computational problems. *Science*, 268(5210):542, 1995. 15

[LMM03] Richard J Lipton, Evangelos Markakis, and Aranyak Mehta. Playing large games using simple strategies. In *Proceedings of the 4th ACM conference on Electronic commerce*, pages 36–41. ACM, 2003. 21

[LMSS01] Michael G Luby, Michael Mitzenmacher, Mohammad Amin Shokrollahi, and Daniel A Spielman. Improved low-density parity-check codes using irregular graphs. *IEEE Transactions on information Theory*, 47(2):585–598, 2001. 7

[Lov12] László Lovász. *Large networks and graph limits*, volume 60. American Mathematical Soc., 2012. 11

[LP06] Adi Livnat and Nicholas Pippenger. An optimal brain can be composed of conflicting agents. *Proceedings of the National Academy of Sciences of the United States of America*, 103(9):3198–3202, 2006. 17

[LP16] Adi Livnat and Christos Papadimitriou. Sex as an algorithm: the theory of evolution under the lens of computation. *Communications of the ACM*, 59(11):84–93, 2016. 17

[LPDF08] Adi Livnat, Christos Papadimitriou, Jonathan Dushoff, and Marcus W Feldman. A mixability theory for the role of sex in evolution. *Proceedings of the National Academy of Sciences*, 105(50):19803–19808, 2008. 17

[LPPF10] Adi Livnat, Christos Papadimitriou, Nicholas Pippenger, and Marcus W Feldman. Sex, mixability, and modularity. *Proceedings of the National Academy of Sciences*, 107(4):1452–1457, 2010. 17

[LPS14] Jeff W Lichtman, Hanspeter Pfister, and Nir Shavit. The big data challenges of connectomics. *Nature neuroscience*, 17(11):1448–1454, 2014. 19

[LRS14] James R. Lee, Prasad Raghavendra, and David Steurer. Lower bounds on the size of semidefinite programming relaxations. *arXiv preprint arXiv:1411.6317*, 2014. 6

[LS13] Yin Tat Lee and Aaron Sidford. Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 147–156. IEEE, 2013. 6

[LS14] Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in o(vrank) iterations and faster algorithms for maximum flow. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 424–433. IEEE, 2014. 6

[LSW00] Nathan Linial, Alex Samorodnitsky, and Avi Wigderson. A deterministic strongly polynomial algorithm for matrix scaling and approximate permanents. *Combinatorica*, 20(4):545–568, 2000. 9

[Lub02] Michael Luby. LT codes. In *Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on*, pages 271–280. IEEE, 2002. 7

[Mad13] Aleksander Madry. Navigating central path with electrical flows: From flows to matchings, and back. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 253–262. IEEE, 2013. 6

[Mal99] Juan Maldacena. The large n limit of superconformal field theories and supergravity. In *AIP Conference Proceedings CONF-981170*, volume 484, pages 51–63. AIP, 1999. 20

[Mar06] Henry Markram. The blue brain project. *Nature reviews. Neuroscience*, 7(2):153, 2006. 18

[Mil67] Stanley Milgram. The small world problem. *Phychology Today*, 1:61–67, 1967. 24

[Moi16] Ankur Moitra. Approximate counting, the Lovász local lemma and inference in graphical models. *arXiv preprint arXiv:1610.04317*, 2016. 9

[Mos09] Robin A Moser. A constructive proof of the Lovász local lemma. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 343–350. ACM, 2009. 9

[MP43] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943. 18

[MP15] Reshef Meir and David Parkes. On sex, evolution, and the multiplicative weights update algorithm. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 929–937. International Foundation for Autonomous Agents and Multiagent Systems, 2015. 17

[MT10] Robin A Moser and Gábor Tardos. A constructive proof of the general Lovász local lemma. *Journal of the ACM (JACM)*, 57(2):11, 2010. 9

[MWW16] Wenlong Mou, Zhi Wang, and Liwei Wang. Stable memory allocation in the hippocampus: Fundamental limits and neural realization. *arXiv preprint arXiv:1612.04659*, 2016. 18

[NRTV07] Noam Nisan, Tim Roughgden, Eva Tardos, and Vijay V. Vazirani. *Algorithmic game theory*, volume 1. Cambridge University Press Cambridge, 2007. 21

[Pap94] C. H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *J. Comput. System Sci.*, 48(3):498–532, 1994. 21

[Pev00] Pavel Pevzner. *Computational molecular biology: an algorithmic approach*. MIT Press, 2000. 15

[PV05] F. Parvaresh and A. Vardy. Correcting errors beyond the Guruswami-Sudan radius in polynomial time. In *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on*, pages 285–294. IEEE, 2005. 7

[PV15] Christos H Papadimitriou and Santosh S Vempala. Cortical learning via prediction. In *Conference on Learning Theory*, pages 1402–1422, 2015. 18

[Rag08] P. Raghavendra. Optimal algorithms and inapproximability results for every CSP? In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 245–254. ACM, 2008. 6

[RPW04] Paul WK Rothemund, Nick Papadakis, and Erik Winfree. Algorithmic self-assembly of dna sierpinski triangles. *PLoS biology*, 2(12):e424, 2004. 16

[RR97] A. A. Razborov and S. Rudich. Natural proofs. *J. Comput. System Sci.*, 55(1):24–35, 1997. 13

[RS97] Ran Raz and Shmuel Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 475–484. ACM, 1997. 7

[RT02] Tim Roughgarden and Éva Tardos. How bad is selfish routing? *Journal of the ACM (JACM)*, 49(2):236–259, 2002. 22

[RU01] Thomas J Richardson and Rüdiger L Urbanke. The capacity of low-density parity-check codes under message-passing decoding. *IEEE Transactions on information theory*, 47(2):599–618, 2001. 7

[Sch92] Leonard J. Schulman. Communication on noisy channels: A coding theorem for computation. In *Foundations of Computer Science, 1992. Proceedings., 33rd Annual Symposium on*, pages 724–733. IEEE, 1992. 25

[Sch93] Leonard J. Schulman. Deterministic coding for interactive communication. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 747–756. ACM, 1993. 25

[See04] Nadrian C Seeman. Nanotechnology and the double helix. *Scientific American*, 290(6):64–75, 2004. 15

[Sha48] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948. 25

[Sha92] Adi Shamir. IP = PSPACE. *J. ACM*, 39:869–877, 1992. 25

[Sim57] Herbert A Simon. Models of man; social and rational. 1957. 15

[SJ89] Alistair Sinclair and Mark Jerrum. Approximate counting, uniform generation and rapidly mixing markov chains. *Information and Computation*, 82(1):93–133, 1989. 8

[Sly10] Allan Sly. Computational transition at the uniqueness threshold. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 287–296. IEEE, 2010. 9

[Spi95] Daniel A. Spielman. Linear-time encodable and decodable error-correcting codes. In *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, pages 388–397. ACM, 1995. 7

[SS96] Michael Sipser and Daniel A Spielman. Expander codes. *IEEE Transactions on Information Theory*, 42(6):1710–1722, 1996. 7

[ST04a] Daniel A Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 81–90. ACM, 2004. 6

[ST04b] Daniel A Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM (JACM)*, 51(3):385–463, 2004. 6, 30

[STV99] Madhu Sudan, Luca Trevisan, and Salil Vadhan. Pseudorandom generators without the xor lemma. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 537–546. ACM, 1999. 7

[Sud97] Madhu Sudan. Decoding of reed-solomon codes beyond the error-correction bound. *Journal of complexity*, 13(1):180–193, 1997. 7

[Sud00] Madhu Sudan. List decoding: Algorithms and applications. *Theoretical Computer Science: Exploring New Frontiers of Theoretical Informatics*, pages 25–41, 2000. 7

[TPC15] Ashutosh Tiwari, Hirak K Patra, and Jeong-Woo Choi. *Advanced theranostic materials*. John Wiley & Sons, 2015. 16

[Tur50] Alan M Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950. 24

[Tur52] Alan M Turing. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 237(641):37–72, 1952. 14

[Val79a] Leslie G. Valiant. The complexity of computing the permanent. *Theoretical computer science*, 8(2):189–201, 1979. 8

[Val79b] Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979. 8

[Val00] Leslie G Valiant. *Circuits of the Mind*. Oxford University Press on Demand, 2000. 18

[Val06] Leslie G Valiant. A quantitative theory of neural computation. *Biological cybernetics*, 95(3):205–211, 2006. 18

[Val09] Leslie G Valiant. Evolvability. *Journal of the ACM (JACM)*, 56(1):3, 2009. 17

[Val12] Leslie G Valiant. The hippocampus as a stable memory allocator for cortex. *Neural computation*, 24(11):2873–2899, 2012. 18

[Val13] Leslie Valiant. *Probably Approximately Correct: Nature's Algorithms for Learning and Prospering in a Complex World*. Basic Books, 2013. 17

[Val14] Leslie G Valiant. What must a global theory of cortex explain? *Current opinion in neurobiology*, 25:15–19, 2014. 18

[VN58] John Von Neumann. *The computer and the brain*. Yale University Press, 1958. Reprinted in 2012 with a forward by Ray Kurzweil. 14, 18

[VNB+66] John Von Neumann, Arthur W Burks, et al. Theory of self-reproducing automata. *IEEE Transactions on Neural Networks*, 5(1):3–14, 1966. 14

[Wei06] Dror Weitz. Counting independent sets up to the tree threshold. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 140–149. ACM, 2006. 9

[Wig60] Eugene P Wigner. The unreasonable effectiveness of mathematics in the natural sciences. *Communications on pure and applied mathematics*, 13(1):1–14, 1960. Richard courant lecture in mathematical sciences delivered at New York University, May 11, 1959. 14

[Wig17] A. Wigderson. *Mathematics and Computation*. Princeton University Press, 2017. To appear. Draft available here: https://www.math.ias.edu/avi/book. 4, 5, 6, 8, 10, 11, 13, 19, 24, 25, 26, 28, 32, 36

[Wil15] Virginia Vassilevska Williams. Hardness of easy problems: Basing hardness on popular conjectures such as the strong exponential time hypothesis. In *Proc. International Symposium on Parameterized and Exact Computation*, pages 16–28, 2015. 6

[Win06] Erik Winfree. Self-healing tile sets. *Nanotechnology: science and computation*, pages 55–78, 2006. 16

[Y+12] Sergey Yekhanin et al. Locally decodable codes. *Foundations and Trends® in Theoretical Computer Science*, 6(3):139–255, 2012. 7

[Yan91] Mihalis Yannakakis. Expressing combinatorial optimization problems by Linear Programs. *Journal of Computer and System Sciences*, 43(3):441–466, 1991. 6

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *Foundations of Computer Science, 1986., 27th Annual Symposium on*, pages 162–167. IEEE, 1986. 22

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il