Contents

Introduction

Von Neumann said in 1946 "I'm thinking about something more important than the atom bomb: I'm thinking about computers." In the first computers, the total memory was only 5KB, inside 40 cathode-ray tubes. They were built from WWII scrap products. First computer built according to von Neumann architecture was here. Wide-ranging view that they had exploring artificial intelligence first weather prediction done here. First efforts to predict 48 hours with great precision yesterday's weather was predicted.

A history of the theory of computation, Avi Wigderson

1. 1936: Alan Turing, "On computable numbers with an application to the entscheidungsproblem."
    - Mathematical definition of a computer
    - Computer science and technology revolution
    - Natural processes as information processes
    - Not everything is computable. (We knew this from day one, unlike in physics where the uncertainty principle had to be discovered.)
2. 1971: Steve Cook and Leonid Levin hypothesized P≠NP.
    - Mathematical (Clay millennium)
    - CS (limits what efficient algorithms can do)
    - Practical (underlies e-commerce, Internet security)
    - Scientific (limits nature and human processes—we assume that "nature computes.") Scientists should reconsider any model that violates P≠NP, as they do any model violating the 2nd law of thermodynamics.

1 The Computational Universe, Leslie Valiant

We'll talk about connections of the theory of computation to biology and natural sciences.

Some natural phenomenon are naturally computational; computation should be the primary lens to view it: the universe is "computational."

Computation is a way of understanding natural phenomenon just like physics. Computation has its laws. How do they relate to natural phenomenon? The **Church-Turing thesis** says that the mechanically computable functions are exactly those that can be computed by a Turing Machine. This is an assumption about nature. Some functions are not computable.

How do we use it? Can the brain be described as a computation? Can biological reproduction be described as a computation? Von Neumann asked this. The answer is yes because computation is defined to be so broad as to include everything in nature. This hangs on the validity of the Church-Turing Thesis.

Turing published papers Computing Machinery and Intelligence (1950), and The Chemical Basis of Morphogenesis (1952). Von Neumann wrote The Computer and the Brain (1956) and Theorem of Self-Reproducing Automata (1966). There is a possibility of understanding mechanical processes.

How do we use the idea of computation to understand nature better? The impact of the Church-Turing thesis is limited because it is very general. We need more constraining laws: Consider the quantitative resources used by the mechanisms: number of steps and components. This is the study of **computational complexity**.

Problem 1.1: How could Darwinian evolution come this far this fast?

We want to view evolution as a computational process. The time bounds for evolution on earth: Darwin calculated that it needed $> 300 \cdot 10^6$ years, but Kelvin said that $< 24 \cdot 10^6$ years are available. Now physicists say that $< 13800 \cdot 10^6$ years are available. Why doesn't it need many more steps—e.g. exponential in the length of the human genome.

The most naive view is that evolution is a search process. Can we come up with a clearly much more efficient process? *How can circuits evolve in subexponential time?* Darwin wasn't specific on the mechanics of evolution. What can we try? Uniform independent mutations? Genetic algorithms aren't great at this; existing computer simulations are disappointing.

Evidence for Darwin's theory of evolution has been accumulating spectacularly, but our standards of what is an explanation have become more rigorous.

1. We now know what is a "mechanism."

2. We expect the predicted behavior of mechanisms to be confirmed by analysis or computer simulation. "Science is what we can explain to a computer." –Donald Knuth.

This is a harsh criterion for biology.

William Paley was a religious philosopher (and mathematician) who articulated the creationist viewpoint: *"the contrivances of nature surpass the contrivances of art, in the complexity, subtility, and curiosity of the mechanism."*

"I did not at that time trouble myself about Palye's premises, and taking these on trust I was charmed and convinced of the long line of argumentation." –Charles Darwin.

The human genome specifies 20000 proteins, and the conditions under which each is expressed in terms of the concentrations of the others: $h_j(c_1, \ldots, c_{20000})$ for $j \in [20000]$. *The genome acts;* it is a function.

The gene for the jth protein has a description of the expression function h_j and description p_j of te protein. The genome has 20000+ of these, so $4 \cdot 10^9$ bits. Central question: How do the h_i, p_i get updated by Darwinian mechanisms? We know that survival depends on how it acts.

How can a functioning protein network, taking $4 \cdot 10^9$ bits to describe, arise in the course of $4 \cdot 10^9$ years? We need a *polynomial time* algorithm.

Can highly complex sets of such functions evolve into existence without a designer, under a Darwinian process?

This fits very well into an established field of CS, **machine learning**.

1. There is a function $f(x)$ to be computed (ex. translation, distinguishing).
2. The knowledge is not programmed but is learned from experience, e.g., by giving examples.

The **perceptron algorithm** (Rosenblatt, 1958) assumes some linear inequality distinguishes between 2 groups. It updates after each example. (Ex. add the coefficients (x, y, z) when group A, and subtract the equation when group B.)

In learning, there is a possible space; some points are true positives and negatives, and a hypothesized divider. In PAC learning,

1. With high probability error is small
2. Examples are from arbitrary distribution, but testing is from the same distribution.
3. Computation, inverse errors all bounded polynomially.

A correspondence between machine learning and evolution. hypothesis.

hypothesis	genome
examples	experiences
learning algorithm	mechanism of variation
correctness	survival (feedbck from environment)

This doesn't look like evolution because it says on a single outcome you can change the genome. What really happens is Darwinian: offsprint have some variation and are selected. Machine learning is Lamarkian!

We put a Darwinian restriction on machine learning.

1. Darwinian feedback: makes variants of current genome *independent of experiences*, and lets one with good performance on real world experiences win.
2. The goal is the ideal function that specifies the best things to do in every situation. The performance of a genome measures how often it does the best thing.
3. The variants are generated by polynomial time randomized Turing machine (any feasible computation). (Even given this, evolution is not trivial.)

4. Polynomial size populations, number of generations, number of experiences. (Honesty.)

Suppose we have a sequence of 0's and 1's representing whether a set of proteins are expressed or not, and a function of this string says whether it's best to do X or Y.

Monotone disjunctions are evolvable for uniform distributions ("at least one of x_1, x_3, x_7 is true"). Proof: Explicit algorithm for variation.

Parity functions are not evolvable for uniform distribution. ("An odd number of x_1, x_3, x_7 are true.") Proof: Kearn's result on statistical queries.

Let

- $f(\mathbf{x})$ be idea behavior.
- $\mathbf{x}$ from arbitrary distribution D.
- Current genome $H(\mathbf{x})$.
- $\text{Loss}(f(\mathbf{x}), h(\mathbf{x})) = |f(\mathbf{x}) - h(\mathbf{x})|$.

The statistical view is that each gene has a fitness as do combinations of them; analyze how such genes mix and compete.

The computational view (the genome acts) is that the gene specifies a function h; the performance has a best behavior.

What has been found so far?

1. V. Feldman: Distribution-free evolution of boolean conjunctions is not possible with yes/no feedback.
2. Distibution-free evolution is possible for sets of fixed degree polynomials for all convex loss functions.
3. Chemical kinetics says that h_i are of the form polynomial/polynomial.
4. Angelina/Kanade: Attribute-efficient evolution of linear functions is possible for smooth distributions and squared loss.

Some remarks.

1. This talk was not about: evolution as history of life on earth, or widely debated themes (game theory, sex, peacocks). It is about how evolution can achieve novel functionality as fast as it can.
2. Baldwin effect: Do adaptive phenomena duing life speed up Darwinian evolution? Is there a combined learning model? (Does adaptation during life *combined with* evolution speed up the process?)
3. Epigenetics: More information is handed down to the next generation than the DNA sequence; some of it is influenced by experience. Does this have a decisive influence on the speed of evolution? Is there a model for it?
4. Facilitated Variation (Marc Kirschner)

5. Is our enterprise premature? We care about genotype, phenotype, and the environment.

   No: machine learning can produce efficient mechanisms without understanding or *knowing about* what it is doing.

Questions:

1. We are able to manipulate the genome. Could this lead to be faster evolution? (Learning when you do have an expert.)

   Yes.

2. Right now we're just looking for *any* explanation, and haven't been able to account a lot of subtleties.

3. "This talk is not about." Von Neumann: by axiomatizing the automata we've thrown half the problem out the window; it may be the more important half—how it is actually realized in biology.

4. In machine learning you can define a loss function and ideal function. There is an ambiguity on what is the best response.

   We want the same algorithm to work for a wide range of loss functions and distributions.

5. In circuit complexity, computers have found more complex solutions that humans have trouble understanding. How can we develop resources which can explain this to humans?

6. It's easier to evolve a real-valued functions. A population of individuals getting boolean-valued feedback similar to getting real-valued feedback? Probably yes.

2 Equilibria, Computation, and Compromises: Some points of contact between theoretical computer science and economics, Tim Roughgarden

The origins of game theory:

- "Zur Theorie der Gesellschaftsspiele"
- Theory of games and economic behavior (1944, Oskar Morgenstern)

He supervised ENIAC (UPenn, 1945) and the IAS machine (1945-1951). The economic and CS community didn't interact that much though.

Theorem 2.1 (Nash's Theorem (1950))**:** Every finite noncooperative game has at least one (Nash) equilibrium.

This gave economists an universal existence result, "everything they wanted." Think of equilibrium as "no one wants to move." The one catch is that we allow probabilistic strategies.

On the CS side, the Cook-Karp-Levin Theorem showed that many fundamental problems are NP-complete, "very difficult to solve computationally." When a problem is found to be NP-complete, we must compromise! Use heuristics, solve approximately, narrow your domain (special instances...). CS theory has evolved in the long shadow of this theory.

Outline:

1. Introduction, von Neumann, Nash, Cook-Karp-Levin
2. Approximation

Why computer scientists need to talk to

2.1 The price of anarchy

Communication networks. Engineers had been thinking about routing data. Now we have multiple self-interested users. Think of vehicular traffic: the fundamental principles are the same.

Pigou's example: One unit of traffic wants to go from s to t. 101 is more prone to congestion. Each edge is labeled with a cost function, as a fraction of the congestion. For example, two highways go from s to t, with costs $c(x) = x$ and $c(x) = 1$. "It's a no brainer": everyone has the dominant strategy to take 101. We expect 101 to be fully congested; that is the only equilibrium.

But you can do better! Drivers don't take in account the extra amount of congestion they cause. An altruistic dictator would split the traffic $\frac{1}{2}, \frac{1}{2}$: the average commute drops to 45 minutes.

Kousoupias and Papadimitriou '99 call this the "price of anarchy," $\frac{4}{3}$ in this example.

Example 2.2 (Braess's Paradox, 1968)**:** 2 routes, $s \xrightarrow{x} a \xrightarrow{1} t$, $s \xrightarrow{1} b \xrightarrow{x} t$.

Suppose there is a teleporter from a to b with infinite capacity. Everyone would make use of the technology and go $s \xrightarrow{x} a \to b \xrightarrow{x} t$, and the average has gone up from 90 minutes to 2 hours! When you add resources, the outcome is worse for everyone. The price of anarchy is $\frac{4}{3}$ (there is no way to use the device to improve).

The same equations govern physical equilibrium. Take a bunch of strings and springs; tie them together. Tie to the bottom of the table and tie a heavy of weight at the bottom. Snip a string, and the weight goes *up!* Inelastic strings correspond to 0, 1, and springs correspond to x. Travel time is distance; flow is force. Cutting the string corresponds to deleting the 0 edge. It frees up 2 strings to carry a string in parallel.

By changing the example to $s \xrightarrow{x^d} t$ and $s \xrightarrow{1} t$ and taking $d \to \infty$, the price of anarchy is unbounded. With dictatorial control, route $1 - \varepsilon$ through the top and ε through the bottom.

When is the price of anarchy bounded? We hope that imposing additional structure on the cost function helps. We worry that bad things happen in larger networks.

Is no higher-order function enough? Yes!

Theorem 2.3 (Roughgarden, Tardos, 2000)**:** For every network with affine cost functions,

$$(\text{cost of eq flow}) \leq \frac{4}{3}(\text{cost of opt flow}).$$

We'll give a "moral" proofs. Think about electrical currents. It's an equilibrium, equalizing the voltage drop. Thompson's principle says it minimizes the dissipated energy (optimal). This is a special case of traffic equilibrium, where all cost functions are linear, ax. The price of anarchy is 1. With added constants it still minimize a energy function, just not the one we care about. They don't differ very much, so they almost optimize the function we care about.

We want to go beyond affine cost functions. The structure of the worst case example is still true. No matter the cost function, the worst network is going to be 2-node networks. You can compute the price of anarchy for any family of cost functions. For the M/M/1 cost function $c(x) = \frac{1}{u_e - x}$, the price of anarchy is at most $\frac{1}{2}(1 + \frac{1}{\sqrt{\beta}})$. (Let β be the fraction that is unused (overprovisioning).) Moral: even modest (10%) over-provisioning is sufficient for near-optimal routing.

The classical model and results: the traffic model and its equilibria, and the inefficiency of equilibria. Contributions from the theory CS perspective:

1. approximation as a constructive compromise: lesson learned from coping with NP-completeness
2. "pricec of anarchy" small in many cases
3. identification of worst-case networks.

2.2 Auction design: the rubber meets the road

9/28/2012: The FCC backs a proposal to realign airwaves. The government is selling the spectrum, but the spectrum that is most valuable is already accounted for. The people who are owned. A lot of it is owned by TV broadcasters. The government proposed repurposing it for more valuable use: buying it back and selling it to wireless telecoms for broadband.

The government accepted bids, and find the highest bidder. This is the first time they do a reverse auction: buy TV broadcast licenses back. They estimated buying for \$15 billion and selling for \$40 billion revenues. They want to use the profit to reduce the defecit: "Middle class tax relief and job creation act."

If the auction is designed badly, disaster happens.

Example 2.4: New Zealand, 1990: 10 interchangeable TV broadcasting licenses. Simultaneous sealed-bid second-price auction.

The projected revenue is $\$250,000,000$. Buyers bid high on one and super-low on a bunch. If you have an auction where it's not clear how to play, expect bad results. They made $\$36,000,000$; there was a huge difference between the top 2 bids. In one of the auctions, the second highest bidder was \$6.

Reverse auction format. The **descending clock auction**. Start high and go low.

- Each round, each broadcaster offered a buyout prime which decreases over time.

  Decline: exits and retains license.

  Accepted: moves to next round.

  In each round, they lower the buyout price. Eventually, some broadcasters refuse.

- Intuition: stop auction when prices are as low as possible, subject to clearing enough spectrum.

  Buy a target amount as cheaply as possible. Example goal: from channels 38–51 (14), clear 10 of them nationwide.

  After you buy out, repack remaining TV stations into a smaller subset of channels (e.g. 38–41).

- Repacking problem: color regions so that no 2 adjacent ones are a different color.

What is the influence of theory CS?

1. High-level auction format an extension of Mehta/Roughgarden/Sundararajan '09. We want computational efficiency, using approximation algorithms.
2. Proposed methods for discriminatory pricing inspired by Lehmann/O'Callaghan/Shoham '02. (Treat different people differently. Some regions interfere very much, others not much.) Use approximation.
3. State-of-the-art algorithms for solving NP-complete problems necessary and sufficient to solve repacking problem quickly [Leyton-Brown, 13, 14]: encode as SAT and use presolvers, solver configurations tuned to interference constraints. (10,000 variables, 100,000 constraints.) Add domain knowledge.

1. reverse auction format builds on previous designs.
2. greedy heuristics for decreasing buyout prices (approximations).
3. State-of-the-art SAT solvers needed for fast solutions of repacking problems.
4. Formalize rules of thumb for forward auction design using approximation (positive results) and communication complexity (negative results).

   CS gives vocabulary to turn rules of thumbs into theories.

   When lots of complementaries—things not valuable individually but in concent. Hard to bid because you need to simultaneously win all of them. You need complicated options. This is communication complexity!

Nash's Theorem "spoils" you in game theory. But in practical applications, it's not enough to know it exists. Someone is responsible for calculating the equilibria! We want a constructive Nash's Theorem. Nash's proof was based on fixed-point theorems that don't give construction.

There was evidence against it:

1. Michael Rabin (1957): "It is quite obvious that not all games considered in the theory can actually be played by human beings." Intractibility of game theory equilibria. He exhibited a game with countable decision space with no decidable procedure to give equilibrium. There are big gaps between existence theorems and constructive existence theorems.

2. (Once the language of NP is in play) Gilboa/Zeme (1989) show that many problems about Nash equilibria are NP-complete, followed by similar results.

3. Is computing a Nash equilibrium NP-complete? The problem with this question: There is a "type-checking problem": there is guaranteed existence. We need to refine NP to obtain the right complexity class. Papadimitriou '94 introduced PPAD. (Polynomial periodic, directed version.)

4. (After 10 years) Computing a Nash equilibrium of a bimatrix game is PPAD-complete. It is also intractable in other senses.

   Interpretation: there is no general constructive version of Nash's theorem. Compromises are required: tractable special cases, approximation, etc.

What do we get from the computational lens?

1. Nash's theorem gives universal existence.

2. Complexity theory shows impossibility of universal computational tractability.

3. Contributes to skepticism over universal predictive power of the Nash equilibrium.

2.3 Conclusions

- Many points of contact between theory CS and economics/game theory over the past 15 years.
- Many 21st-century CS applications require economic reasoning.
- The theory CS toolbox articulates computational barriers and offers constructive compromises.

Many big companies have chief economists.

CS is uniquely situated to talk about computational intractibility, and whether equilibria are going to be realized.

Questions.

1. Ongoing debate about neutrality. Is there an application from CS to resolving public policy debate?

   For network routing there is a clean parameter that translated to policy advice. There are models where the status quo can be significantly/not significantly improved, and it's hard to tell.

2. Google did an public IPO with an auction; results were mixed.

   Why did not one else do it since Google?

   For something like wireless spectrum, there is a need to centralize the information.

   eBay has shifted from auctions to posted prices. At the dawn there was novelty to auctions (fun). Now for commodity of goods there's not much point to running an auction. To be foolproof, give a price. (Sometimes auctions give ways of gaming the system.)

3. "A lot of TCS is economics done in a different department." Which parts of CS are just economics and which part aren't? "Optimization is equivalent to economics."

   A lot of these applications are: formally proving what we can't do and using that as a guide to approach the problem. The traveling salesman problem is fundamentally different from shortest-path. There wasn't the vocab to say "we have to make compromises" formally. The two themes are (a) when are we using approximation because of computational intractibility, (b) when are we using approximation for some other reason?

4. Consider the packing problem. Are these results robust to small imperfections in measurements?

   Graph coloring is in general not robust. In this case there is not much uncertainty; it's built in. There are detailed physical models of interference. There is leeway built in: FCC promises at most 2% of customers will be affected.

   Robustness is in general important. People measure imperfection in different ways. The CS tradition is to do "worst case analysis," and economicsts do average-case or Bayesian analysis. I'm trying to find a sweet spot between them.

5. Would crowdsourcing help?

   It doesn't hurt. A different way to express the difference between NP and P problems is: if I double the computing power, how much bigger instances can I solve? For hard problems, it grows slowly.

   For people making a living off solving NP problems, throwing massive hardware at a problem is a key tactic.

6. Are there approximation possibilities in PPAD? This is a major open question. You can't get the absolute best, can you get close?

7. Is anybody working on designing games that don't represent the real game but the properties and complexity of the game and then crowdsourcing to get economists looking at *how people behave?*

   Michael Kearns in Penn studied how well humans do solving NP-problems. Intractibility results from NP-completeness are very robust. You would not expect a switch from intractible to tractible from crowdsourcing.

When you see a gap between theory and practice, usually the reason is it's a structured special case. As scientists, it's our duty to articulate the structure and give provable bounds.

8. For the auction: the calculations are done by the buyer. There is no burden on participants. There is target time for asking questions to participants: 1 hour between questions. One has to solve the problem in an hour. You have to solve many instance of the repacking problem before you get the next set of prices to give.

3 Computational Phenomena in Social Interaction, Jon Kleinberg

The Turing machine is patterned on someone computing by hand. We've thought about people matching wits with computer in chess. In the 1990's things changed again: the field is undergoing radical transformation: there is an outpouring of information. "What is your working metaphor for the web?" Back then, people would say a vast library of human knowledge. 10 years later, 2004-2006, we have speciation events. We move from the metaphor of the library to the crowd, people interacting. Assemble events from millions of individual perspectives. Something new is emerging, not under our control.

"The emergency of 'cyberspace' and the World Wide Web is like the discovery of a new continent." –Jim Gray, 1998, Turing Award address.

Even mudane things in our lives is something that can be looked at computationally, because it's online. These aren't like information systems where you write the code and check; they follow rules of their own, and can react in unexpected ways. For example, there are unexpected consequences of creating incentive systems.

- The online world is a phenomenon to be studies: new computational perspectives on social-science questions.
- Think of them as organic systems. But there is a big dose of design in the systems.

The terrain of the online world is geographic, social, and graph-theoretic. Computations on graphs form a central part of our understanding.

For example, people have studies political blogs form 2 dense clusters (Adamic-Glance, 2005), anti-war chain letters (Liben Nowell-Kleinberg 2008)—branching tree as people recruit others. All these systems are shot through with graphs. It becomes a conceptual mapping exercise: what does the social network look like?

There is the "shock of the familiar," seeing them cast in new ways, looked at through algorithms. It's like some alien intelligence able to detect subtle patterns but having no common-sense knowledge. It's like what aliens would learn looking from the outside in.

Is there life on earth? "A search for life on Earth from the Galileo spacecraft," by Carl Sagan. Exobiology programs, send a spaceship to find alien life; it passed over the Earth. His team did a sanity check to make sure the spacecraft could detect life on earth.

The emissions are more digital: what can we see from raw data?

One experiment Kleinberg did was look at photos from photosharing sites (flickr), putting dots at the location where photos appear. It's recognizable (map of Europe). This is a helpful picture to think about when we enter places where data is more abstract.

There are dark places—can't live there, or aren't on flickr. There is 1-D fuzz (rivers).

What does the social network of the world look like? We need building blocks. How are social networks organized?

1. Homophily: tendency to associate with people like yourself.

2. Polarization, conflict, and structural balance. (Heider 1946, Cartwrigt-Harary 1956, Davis 1963)

3. Global organization: triadic closure, strong vs. weak ties. If A knows B and C, there is a good chance B, C will meet each other. There are dense clusters with other links between them that are weaker. Graphs are inhomogeneous—edges have weights. The places where traingles fail to form are those weak ties. (Rapoport 1953, Granovette 1973)

Stanley Milgram's "6 degrees of separation" was one of the first experiments to look at the social network as a whole. ($680 to test.) Fridus Karinthi (?) wrote a short story in the 20's where characters wondered how many links it would take to link up with a celebrity.

Milgram's small world experiment: choose a target in Boston, starters in Nebraska. A letter begins at each starter, must be passed between personal acquaintances until target is reached. Mail it to someone you're on a first-name basis with. About $\frac{1}{3}$ made it. It took six steps on average, "six degrees of separation."

He didn't have big data, but he got this striking finding. This got an enormous amount of attention by Strogatz and John Watts. They produced models for small-world networks.

There is a search problem: we're operating under severe constraints—people can tunnel their way through it. We learn that the knowledge is findable; people can go through it without a map!

Imagine a grid graph with random long-range links.

1. Routing in a social network: when is local information sufficient? (Kleinberg 2000)

2. Network models based on Watts-Strogatz...

3. Add edges to lattice where u links to v with probability $d(u,v)^{-\alpha}$. The optimal exponent for search is $\alpha = 2$ (the dimension of the grid). You know your own such edges, but not other people's.

We end up with a prediction: $\alpha = 2$, which is surprisingly specific. Is this right? How could we ever test this? We need to convince people to go online and voluntarily say where their friends live...

We need more general notions of distance. The population density is non-uniform. (For the 2-D case, rank $\sim d^2$.)

- Each node ranks all other nodes by closeness. Node v connects to w with probability $\frac{1}{\text{rank}(v,w)}$.

- Now compare to data from online social networks. LiveJournal (East/West coast) got $\frac{1}{\text{rank}(u,v)^{1.05}}$. Facebook got $\frac{1}{\text{rank}(u,v)^{0.95}}$.

This is optimal for search. But we're not trying to optimize search; this is just a byproduct. What drives the network toward this exponent?

A deeper look into network structure: network structure is a crucial ingredient in new approaches to studying collective human behavior. Let's look locally: think of Facebook not as a billion-node network, but instead as a collection of a billlion (relatively dense) small networks (a billion perspectives from single nodes).

1. Can we link local behavior to global structure?
2. What does the local structure tell us about individuals?
3. To what extent is the collective outcome predictable?

The local neighborhoods (your friends, which pairs of your friends are friends) are similar. There are dense networks–friends from high school, from college, coworkers from family. You're the only one they have in common, so you're at the intersection.

What do we expect a graph to look like? Think of the Erdős-Rényi model $G_{n,p}$. The deficiencies are

- Doesn't produce nodes with a enormous numbers of neighbors (more a problem with web graphs than social networks).
- Real social networks are rich in triangles: triadic closure.

We can draw on extremal graph theory: create a thin set of coordinates for the graphs. Choose a small value of k (3, 4). For each k-node graph H, let $f_G(H)$ be the fraction of k-nodes sets inducing H.

- Triad consensus
- Network motifs
- Frequent subgraph mining
- Subgraph density and homomorphism
- Characterizing neighborhoods

Ex. look at triad frequencies, and plot in 3-D. For every neighborhood on facebook, put it on the graph. We get an interesting shape. There are places (a river) where it's dense and where it's blank. Are they blank because can't live there or don't live there? Some are blank because it's mathematically impossible (ex. frequence of 1-edge triad is $\leq \frac{3}{4}$). There aren't big complete bipartite graphs, which makes sense.

What are feasible regions? That's a hard question in extremal graph theory.

What is the river that runs through the 1-dimensional fuzz? $G_{n,p}$ is the "river" that runs through the points—it's a cubic curve. The deviations are based on triadic closure.

Applications.

1. Finding significant people: Given a person's network neighborhood, can we identify their most significant social ties?

   Facebook friends are generating too much content—there is 1000-2000 pieces of data per day. Facebook needs to prioritize content to show you.

   Triangles play a central role. The embeddedness of edge e is the number of triangles containing e, equivalently, the number of mutual friends shared by e's endpoints. There are people who you want to know what they ate for breakfast, vs. just know big events like getting engaged.

   Hypothesis: If an edge is highly embedded, it is likely to be a stronger tie. Tie strength and embeddedness are wrapped up with each other.

   What goes wrong: In practice, embeddness finds many nodes from the largest clusters. It's almost a giant clique. For any random coworker, there will be like 30 friends in common, even though the people don't know each other that well.

   (Picture: spring layout. Each is spring with fixed tension.)

   What is this model missing? Often this is a large collection of coworkers or college alumi friends.

   When you encounter a problem that you can't solve with current techniques, make a challenge problem that encapsulates the difficulty. Given a facebook user in a relationship, find the partner just from network structure.

2. Alternatives to embeddedness: Instead of just counting mutual friends, look at their structure. v and w as a bridge between different crowds. "The two of you are the only reason they know each other."

   Go back to measure of separation. A bridge is decreasing distances. We measure by **dispersion**,

   $$\sum_{s,t \in C_{v,w}} d_{G-(v,w)}(s,t) :$$

   the sum of distances between pairs in C_{vw} (common neighbors) after deleting v and w. (Need to normalize, etc.)

   For evaluation, use 1.3 million facebook users who declare a relationship partner in their profile, have between 50 and 2000 friends, and are at least 20 years old. (Failure modes: teenage girls married to their best friends.) For each user v rank all friends w by competing metrics.

   - embeddedness of vw edge
   - dispersion of vw edge
   - number of photos in which v, w are both tagged
   - number of times v viewed w's profile in last 90 days

   For what fraction of all users v is the top-ranked w the relationship partner.

Embeddness works 0.247 of the time, dispesion works .506 of the time, photo is .415, and profile view is .301. Having the right algorithm is important! 2nd question: "how to use this features."

Easier to find for male. We can combine all of these via machine learning, giving .716 for married, .682. Approximately $34 - 38\%$ of incorrect guesses are family members. 12 months cutoff.

When wrong, there is a 50% higher chance the couple will break up in the next 2 months!

A general structure for network neighborhoods: Lots of deep theorems in grpah theory have the form: every grpah is a a constant big pieces of small complexity and a large number of small pieces of high complexity. Ex. Szemerédi regularity lemma. Is there a meta-principle for real graphs? There's a constant number of homogeneous clusters plus a constant number of nodes that defy classifications.

Ex. Siblings and partners are different: one you grow up with, the other you "time travel" with introducing to other firneds.

Further directions. Other information latent in network neighborhoods?

"MySpace is doubley awkward because it makes public what should be private. It doesn't just create social networks, it anatomizes tem. It spreads them out like a digestive tract on the autopsy table. You can see what's connected to what, who's connected to whom." – Toronto Globe and Mail, 6/2006. Social networks are being recorded at high resolution. What is the right framework? What are the dangers? There is opportunity for fundamental computational and mathematical models.

4 Computational phenomenon in physics

Scott got interested in physics as a kid because it was like computer games but with "much more realistic special effects."

Von Neumann was one of the systemizers of quantum mechanics. Why did he not invent quantum computing? He had a lot on his plate.

There's a lot of tech that would be cool but we don't see, like warp drives, and perpetuum machine, and the Übercomputer which tells you the answer to any math problem. The impossibility of the first two machines reflects fundamental primciples: special relativity and the second law. What about the third case? What are the ultimate physical limits on what can be feasibly computed and do those limits have any implications for physics?

The first pass we can make is the extended Church-Turing thessi: any physically-realistic computing device can be simulated by a determinisitic or probabilistic Turing machine with at most polynomila overhead in time and memory. This bridges CS and physics. It's ultimately a question about nature. A universe where the halting problem can be solved for free is not this universe. What would a challenge to ECT look like? Let's look at classical challenges.

1. Dip two glass plates with pegs between them into soapy water. Let the soap bubbles form a minimum Steiner tree connecting the pegs, thereby solving a NP-hard problem

instantaneously. (Find a proof of the Riemann hypothesis in first order logic with at most 10 million symbols—this can be translated to pegs.)

"I bet computer scientists never tried it."

Scott tried it. 3-4 pegs: the soap relax. With more pegs, the bubbles get stuck in a suboptimal configuration. Sometimes it contains a cycle. "I didn't try every brand of soap." Nature is not solving NP-hard problems magically.

Proteins are not solving NP-hard problems in finding optimal folding. What's the catch? There is extremely strong selection pressure to evolve so they fold easily. (A protein solving Riemann hypothesis wouldn't evolve.) There are proteins that fold into suboptimal configurations (prions, mad cow disease). Ex. rocks rolling down a mountain can get stuck in a crack.

2. Relativity computer: Board a spaceship traveling $(1-\varepsilon)c$. Billions of years later you arrive back to read the question to your hard problem. Why hasn't this been done? If you're worried about your friend being dead, bring them on the spaceship!

   It looks like you get unbounded computational speedups. Other things. We've ignored energy: how much energy does it take to accelerate to near-light speed distance? The difference has to be exponentially small. It takes an exponential amount of energy. Fueling your spaceship takes exponential time. We've traded one exponential blowup in resources for another.

3. Zeno's computer: Build a computer that does step n in $\frac{1}{2^n}$ time. People do try it. They try to overclock their processors. The danger is that it will melt. This is why your computer have a fan and supercomputers are cooled using liquid nitrogen. There's a tradeoff. The faster you can run your microchip the more it has to be cooled, the closer to absolute 0, the more energy it takes. What is the fundamental limit? Physicists have a good idea of the fundamental limit: 10^{43} computations per second: you need to concentrate so much energy that your computer would exceed the Schwarzchild limit and collapse to a black hole—nature's way of telling you not to do something.

4. Time travel computer. Tell Shakespeare the plays. (Not Grandfather's paradox.) The only paradox is computational complexity: Hamlet appears without anyone taking the trouble of coming up with it. Cf. Interstellar. We could exploit closed timelike curves. Feed in 1 guess solution. A computer checks the solution: if it is send it back in time, else increment by 1 and send back in time. To prevent a grandfather parameter, nature has to find a consistent solution. The only consistent solutions are the one where a solution to NP-problem appears in the CTC.

   Aaronson and Watrous show that PSPACE-complete problems can be solved in a CTC with a polytime computer, and nothing more.

What about quantum mechanics? It's "like probability but with minus signs." (And complex numbers, and the 2-norm.) It's simple once you take the physics out of it. Mathematicians could have come up with this by pure thought (in an alternate universe).

Probability theory concerns linear transformations that conserve 1-norm of probability vectors, stochastic matrices. Quantum mechanics concern linear transformations that conserve 2-norm of amplitude vectors, unitary matrices.

Quantum computing: assign an amplitude to every possible configuration:

$$|\psi\rangle = \sum_{x\in\{0,1\}^n} \alpha_x |x\rangle.$$

This presents an obvious practical problem when using conventional computers to simulate quantum mechanics. (Bra-ket notation is the single biggest hurdle for CS-ists to overcome in learning physics.)

Nature, off to the side somewhere is writing down 2^{1000} complex numbers on scratch paper to keep track of 1000 particles! This is more numbers than particles. This is a staggering claim! It took a while for the full enormity of this to sink in. People thought of it just as a practical problem in simulation. People invented lots of clever density theory, some like the density functional theory which won a Nobel prize.

Feynman asked in 1981: is there any way to avoid this exponentiality? If not, why not turn things around and bulid computers that themselves exploit superposition? Feynman only proposed one application: they would be good for simulating quantum mechanics. If we get quantum computers that would still be the most important application. Design better photovoltaics, super high temperature conductivity, design biomolecules, understand high-energy physics...

Can they solve classical problems fasters? Shor in 1994 showed a computer can factor integers in polynomial time: find the periods of exponentially long sequences in polynomial time. This would break public-key cryptography. This made people interested.

But can QC's actually be built? Quantum computers have factored $21 = 3 \cdot 7$ with high probability after a billion dollars (Martin-López 2012). Why is scaling up so hard? Because of decoherence. Unwanted interaction between QC and external environment, "prematurely measuring" the quantum state, forcing us back into classical mechanics before we're ready.

The idea of QC is to choreograph a series of constructive/destructive interferencce, so that for wrong answers positive/negative amplitude cancel out, and for right answers they constructively interfere. If a computer interacts too much before the end, it would no longer be in a superposition. It would be a classic probabilistic mixture: it's just one thing or the other thing.

Popular books talk about Schrödinger's cat. The interesting part is that you can do a different experiment where the branch where the cat is alive and dead interact. If someone is looking at the quantum computer, the cat is merely alive or dead, i.e., you have a classical computer with access to RNG.

Popular literature doesn't distinguish this. "Try each solution in a different parallel universe." But then when you measure you just get a random answer. Interference depends on QC being extremely well-isolated.

A few skeptics in CS and physics argue that building a QC is fundamentally impossible. In the 90's this seemed true. Then people discovered quantum fault tolerance: to do QC, you don't have to keep it perfectly isolated; it's enough to keep it well-enough isolated. If it has a $\frac{1}{1000}$ chance of collapsing, you can correct that: spread across qubits using a clever quantum error-correcting codes.

You just need to get decoherence down to a certain point. We're already several orders of magnitude closer.

It's like building a nuclear weapon: half the critical mass doesn't give half an explosion.

If physicists understand QM, then at that point you can just scale up. Some skeptics still say there's some highly correlated decoherence. QM breaks down to prevent QC from working...