Theoretical Computer Science past, present and future

Amir Daneshgar
Sharif University of Technology

http://www.sharif.ir/~daneshgar

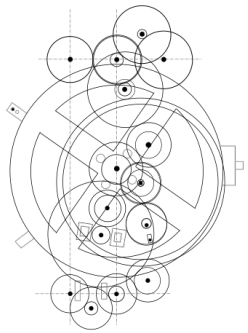
19 February 2018
(30 Bahman 1396)

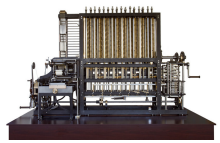
daneshgar@sharif.ir

Outline

1. A concise history of (T)CS
2. ToC: older motivations
3. ToC: current motivations
4. ToC: impacts
   - Randomness
   - Expanders
5. TCS: speculations on the future

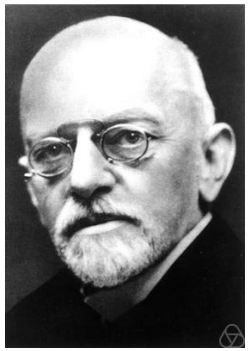
Prehistory < 1900



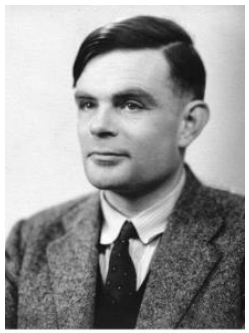


- Abacus (~ 3000 B.C.): Probably existed in Babylonia (present-day Iraq).
- Antikythera mechanism (~ 80 B.C.): Discovered in 1901, within an ancient Greek shipwreck off the island of Antikythera.
- Mechanical adding machine ($\sim 1620 - 1640$): Wilhelm Schickard, Blaise Pascal, Gottfried Wilhelm Leibniz.
- Difference Engine: Charles Babbage ($1791 - 1871$).
- First program: Ada Augusta Byron, Countess of Lovelace $(1815 - 1852)$.

A mathematical boost 1900 – 1940



Kurt Gödel



Alan Turing

- David Hilbert (1900): addressed the International Congress of Mathematicians with three main questions on computability.
- Kurt Gödel (1931): Answered two important questions on consistency and completeness.
- Alan Turing (1936): constructing a formal model of a computer, the *Turing machine* and answered Hilbert's Entscheidungsproblem by introducing the *halting problem*.
- Contributions of Church, Turing, Post, Kleene, ... on the concept of computation and recursion theory motivated by the concept of a proof of a true mathematical statement.

Digital computers $1940 - 1950$

- Mark I electromechanical computer (1944): The calculations required for ballistics during World War II led to this construction by Howard H. Aiken.
- Colossus: Built by British to help Alan Turing breaking the code behind the German machine, the Enigma.
- ENIAC (1946): built at the Moore School at the University of Pennsylvania.
- EDVAC (1944): Mauchly, Eckert, and John von Neumann.
- Z3 (1941): the first operational, general-purpose, program-controlled calculator built by Konrad Zuse.
- Invention of the transistor (1947): By John Bardeen, Walter Brattain, and William Shockley.
- Invention of magnetic core memory (~ 1949): By Jay Forrester.

Compiler design 1950 – 1960

- Invention of the notion of a compiler (1951): By Grace Murray Hopper at Remington Rand.
- First FORTRAN compiler (1957): John Backus and others.
- LISP and ALGOL (1958): John McCarthy and Alan Perlis, John Backus, Peter Naur and others.
- Integrated circuits (1959): Jack Kilby (Texas Instruments) and Robert Noyce (Fairchild Semiconductor).

The CS discipline 1960 – 1970

- Computer science as a discipline (1962): The first computer science department was formed at Purdue University.
- The rise of automata theory and the theory of formal languages $(1960's)$: Noam Chomsky, Michael Rabin and others.

Theory of computation: A culmination of ideas coming from digital design, compiler design, recursion theory and complexity.

The rise of modern TCS 1970 – 1980

- Steve Cook's seminal paper on NP-completeness (1971):
- Design of CRAY-1 (1976): Seymour Cray.
- Theory of computation: Digital circuit design+Compiler design+computability.

Theory of computation: Origins (computability)

Al-Kharazmi
($\sim$ 780-850 AD)

John von Neumann
(1903 - 1957)

The fundamental question in early days

Can we provide a computational solution to any problem?

Theory of computation: Origins (computability)

- Hilbert (1900): Can every true statement be proven (in a finite axiomatic system)?
- Gödel's incompleteness theorem (1931): Some true statements are unprovable!
- Can every function be computed?
- Turing's undecidability theorem (1936): Some functions are not computable!

Theory of computation: the coding trick

A problemtype P consists of the following data:

- Constants: .....
- Given input: .....
- Query: .....?

An example

- Constants: 3.
- Given input: the integer n.
- Query: Is n divisible by 3?

Main question: How do you provide the data?

Theory of computation: membership problem

The membership problemtype

- Constants: Σ a finite set of alphabets and a subset $L \subseteq \Sigma^*$.
- Given input: a word $x \in \Sigma^*$.
- Query: Is it true that $x \in L$?

Fact: Any yes-no problem as P can be reduced to a membership problem for some subset (i.e. a language) L_P.

Main question: How do you provide the data?

A computational model (machine)

It is an abstract machine consisting of:

- A hardware (i.e. a control unit).
- A memory.
- A mechanism to read the input.
- A mechanism to write the output.

working as a discrete dynamical system with local property.

Theory of computation: early impacts

There was a controversy regarding the definition of computation during the early days.

- Definition through computational machines.
- Definition through constructional procedures (i.e. grammars).
- Definition through computable functions (i.e. recursion theory).

Church-Turing thesis (1934-1937)

At the level of Turing machines all rational models of computation are equivalent!

Theory of computation: complexity measures

The fundamental questions

- Do natural hard problems exist?
- What are the consequences of answers YES or NO to this question?

Theory of computation: complexity measures

Main Question: How much should we pay for a computation?

Typical cost functions

- Time
- Used memory
- even more complex cost functions!

The variable: is the length of the input!

Main objective: Study the behaviour of the cost function?

Theory of computation: P vs. NP

Acceptable (i.e. effective) cost functions

- Early days (1960's): Polynomially bounded functions.
- Nowadays: low-degree polynomially bounded.
- Trend (big data): $O(n \log n)$ or less!

The complexity class P
(1960's: Cobham, Edmonds and Rabin)

Consists of all decision problems that can be solved by polynomial-time bounded deterministic decider algorithms.

It is good if we can effectively (i.e. fairly easily) solve a problem!

Theory of computation: P vs. NP

Fact

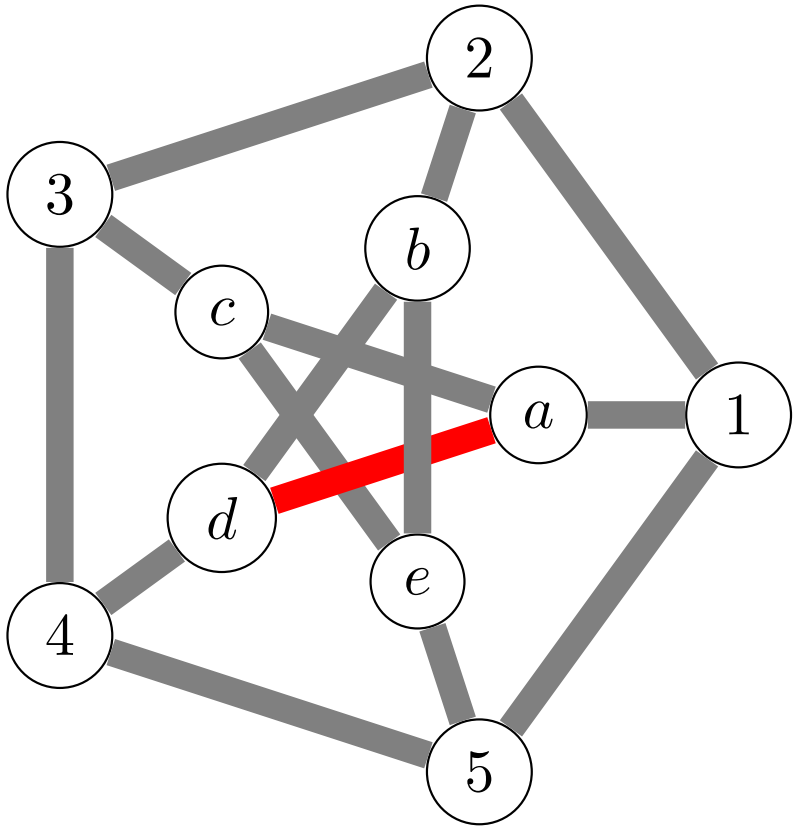
There exists a large number of fundamental decision problems (say more than 2000) for which any claim for a solution can be verified efficiently (i.e. in polynomial time), however, no efficient (i.e. polynomial time) solver is known for any one of these problems!

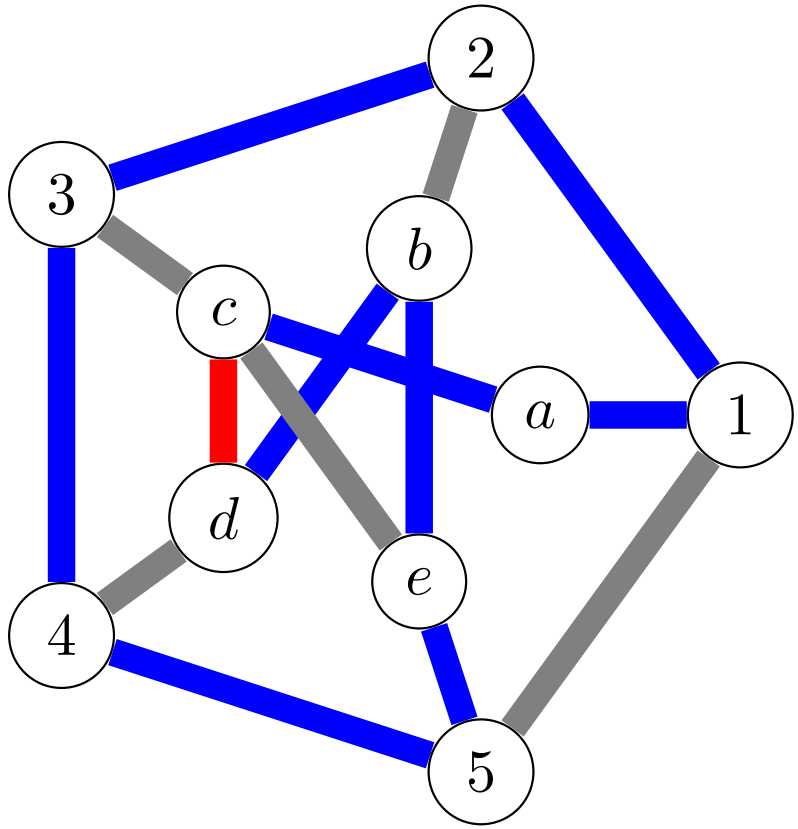
The complexity class NP
(1970's: Cook, Levin, Karp)

Consists of all decision problems that can be solved by polynomial-time bounded nondeterministic acceptor algorithms.

NP-Complete: The class of hardest problems in NP.

Example for an NP-complete problem: Hamiltonian Cycle





An important question!

The fundamental questions

Do natural hard problems exist?

- One has to formulate easy and hard!
- We are not Zeus: hence it is generally believed that the answer is YES!
- It is astonishing that existence of hard problems is quite important in modern technological applications!

A fundamental problem: Do hard problems exist?

i.e., $P \stackrel{?}{=} NP$.

Theory of computation: 1 M$ Millennium Problems (Clay institute 2000)

Seven problems each gives you $1 million at least!

P vs. NP

Determine the answer to the question $P \stackrel{?}{=} NP$.

Riemann Hypothesis

The prime number theorem determines the average distribution of the primes. The Riemann hypothesis tells us about the deviation from the average. Formulated in Riemann's 1859 paper, it asserts that all the 'non-obvious' zeros of the zeta function are complex numbers with real part 1/2.

There are 5 more problems and 4 more unsolved ones!

Randomness and usefulness of hard problems

A couple of fundamental problems:

- Can one produce (i.e. simulate) almost ideal random bits?
- Is randomness useful in computation? Can one use randomness to get easier solutions?
- Can one reproduce a large number of random bits using a small number of ideal ones?
- What does Riemann Hypothesis say about the set of natural numbers?

On the concept of a "Proof"

How to make sure that a claim is true?

- A sound proof is a valid argument for the correctness of a claim.
- To make sure that a claim is true it is quite sufficient to have/see/verify a proof of it!
- However, to make sure that a claim is true it is also sufficient to make sure that there exists a proof of it!!!!

Example: the blind and the twins

	Dishonest	Honest
Yellow Blue		
Blue Yellow		
Blue Yellow		
Yellow Blue ✗ ✗		

Example: BPP and secure communication

BPP is the randomized counterpart of P.

The class BPP

A language L is in BPP if there exists a randomized algorithm A such that

- $x \in L$ implies that $Pr(A(x) = accept) > 3/4$.
- $x \notin L$ implies that $Pr(A(x) = reject) > 3/4$.

A fundamental problem: Do hard problems exist?

i.e., $NP - BPP \stackrel{?}{=} \emptyset$.

Example: BPP and secure communication

In secure communication:

- It is assumed that everyone knows about the details of algorithms ENC and DEC except the security parameter κ (i.e. the key).
- The adversary problem: $\{p \mid \exists\ \kappa \ \ ENC(\kappa, p) = c\} \in NP$. (oversimplified!)

$NP - BPP = \emptyset \ \Rightarrow$ there is no secure communication system!

Study of a possible converse is the subject of modern provable cryptography!

Probabilistic algorithms

Question 1:

Does randomness fundamentally help in computation? i.e. are there problems with probabilistic polynomial-time algorithmic solutions but no deterministic one?

Question 2:

Does NP require strictly more than polynomial time?
i.e. there exist natural hard problems!

At least one of the answers is NO!!

Probabilistic algorithms

More on the story of random bits in computability!

On the existence of hard problems

Assuming factorization of integers has no efficient algorithm implies $P \neq NP$.

[Blum, Micali, Yao, Nisan, Impagliazzo, Wigderson]

Existence of hard problems (say $P \neq NP$ or something similar!) implies the existence of Pseudo-random generators.

Probabilistic proof system

A proof is an argument for a claim.

Main question: Is it valid?

$\exists$ a probabilistic verifier $V(claim, arg)$ for the claim, such that

- If the claim is true then $V(claim, arg^*) = true$ for some argument arg^*.
- If the claim is false then $V(claim, arg) = false$ for every argument arg with probability more than 0.99.

Probabilistic checkable proofs (PCP's)

$\exists$ a probabilistic verifier $V(claim, arg)$ for the claim, such that

the verifier only reads at most 10 bits of the argument at random.

[Arora-Safra, Arora-Lund-Motwani-Sudan-Szegedy-Hastad]

Every proof can be efficiently transformed into a PCP!

Applications

Assuming the existence of natural hard problems:

Non-approximability

Some NP-complete problems (e.g. MaxClique, MaxSat, ...) are non-approximable!

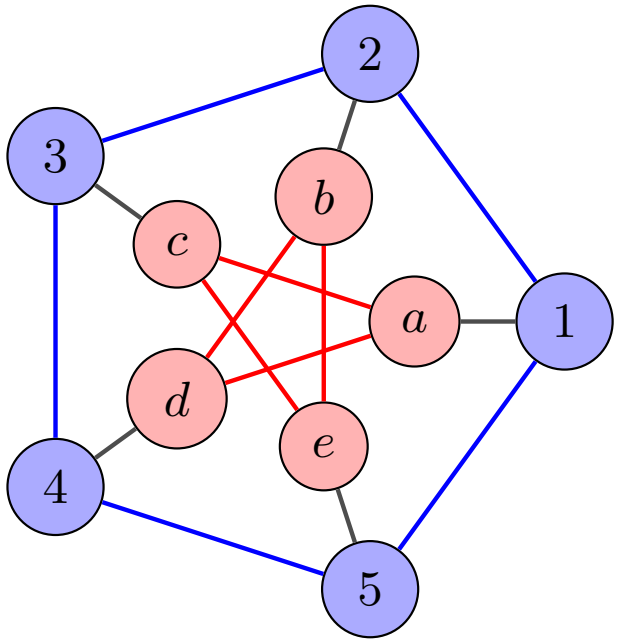
Grading answer sheets

There exists a randomize procedure using which you can grade the answer sheets of your exam in which you only read at most 10 random characters from each sheet and the maximum probability of giving a wrong grade is less than 0.0001!

Authentication

There exists a randomize procedure using which you can prove to your bank on the Internet that you are YOURSELF without revealing your electronic signature at all!

Graphs and Matrices



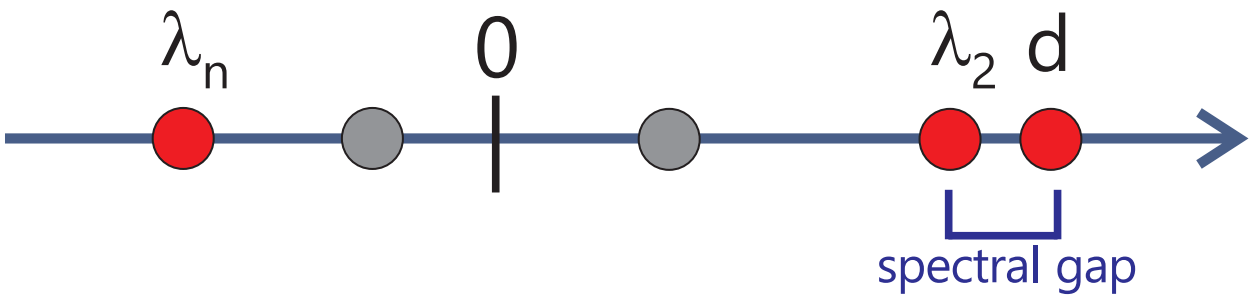
$$
\begin{array}{c}
 \\
1 \\
2 \\
3 \\
4 \\
5 \\
a \\
b \\
c \\
d \\
e
\end{array}
\begin{array}{c}
\begin{array}{cccccccccc} 1 & 2 & 3 & 4 & 5 & a & b & c & d & e \end{array} \\
\left[\begin{array}{cccccccccc}
0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0
\end{array}\right]
\end{array}
$$

Random bits

A fundamental problem

Design of pseudo-random generators, and extractors are among the most fundamental problems in ToC, Engineering and Science.

Random regular graphs: main questions



- What can be said about the spectral gap?
- What can be said about other connectivity related parameters as chromatic number, expansion, Hamiltonicity .....
- Analysis of the extremal cases are usually quite challenging problems.

Primes and Zeta Functions: definitions

- Consider a geometric space made of primes and their amalgams.
- e.g. $\mathbb{Q}$, number fields, function fields, Riemannian manifolds, graphs, ...
- The connectivity of the space is naturally related to number of primes and how they are mixed together.
- Connectivity is a fundamental concept that can be studied and measured in many different ways.
- A zeta (in general L) function is a mathematical concept that is supposed to present and reflect all these aspects in a reasonable way!

Primes and Zeta Functions

Rational numbers

The Riemann zeta function $\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s} = \prod_{p\ prime} (1 - p^{-s})^{-1}$ is related to Hecke operators but possibility for relation to a natural diffusion is not fully understood yet.

Graphs

The Ihara zeta function $\zeta(u) = \prod_{[P]\ prime} (1 - u^{\ell([P])})^{-1}$ is related to the adjacency operator and this relation is fully understood.

Apply $u := q^{-s}$ to compare!

Ramanujan Graphs

Riemann Hypothesis (RH) for $\mathbb{Q}$

If $\zeta(s) = 0$ and $0 < Re(s) < 1$ then $Re(s) = 1/2$.

Riemann Hypothesis (RH) for graphs (Ihara zeta func.)

If $\zeta(q^{-s})^{-1} = 0$ and $0 < Re(s) < 1$ then $Re(s) = 1/2$.

This is equivalent to the following:

Ramanujan graphs

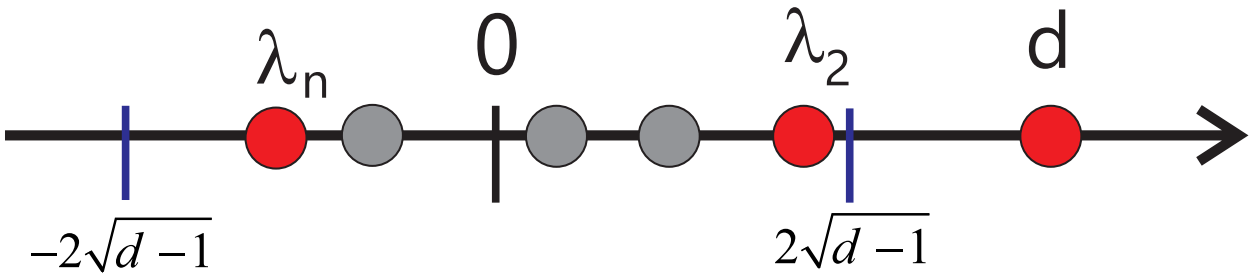
A $(q+1)$-regular graph with adjacency matrix A satisfies RH iff it is Ramanujan, i.e. if

$$\mu \stackrel{\text{def}}{=} \max\{|\lambda| \mid \lambda \in Spec(A) \ \& \ |\lambda| \neq q+1\}$$

then $\mu \leq 2\sqrt{q}$.

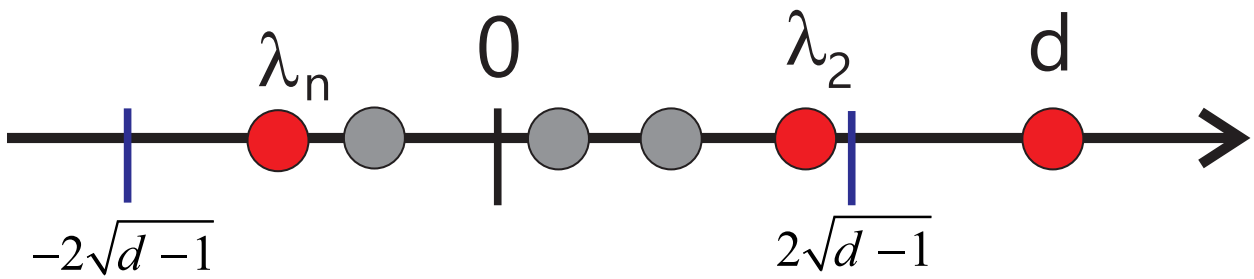
Spectrum of a Ramanujan Graph

Note: Regularity is $d = q + 1$.



- Nontrivial eigenvalues are small
- The graph is sparse but highly connected
- It is a good sparse approximation of a complete graph
- Alon-Boppana 1986: We can not beat the bound $2\sqrt{d-1}$ asymptotically
- The bound $2\sqrt{d-1}$ is the spectral radius of the infinite d-regular tree (i.e. the universal cover!)

Expanders in TCS



Expanders are sledgehammers of TCS! They are used in:

- Derandomization
- Complexity theory
- Error correcting codes
- Compressed sensing
- Communication networks
- Approximate counting
- Measure theory
- Number theory
- ...

Ramanujan graphs of arbitrary degree

A. W. Marcus, D. A. Spielman, N. Srivastava, 2013+
Published in *Annals of Mathematics* (2015)

There exist (arbitrarily large enough) bipartite regular Ramanujan graphs of arbitrary degree.

The proof is based on the fundamental technique of interlacing families of polynomials which is also used by the same authors to prove Kadison-Singer Problem.

A Royal Road to Mathematics

Propaganda!

Euclid of Alexandria (about 300 BC):

There is no Royal Road to geometry.

Graph theory zoo

Graph theory is a Royal Road to the heart of modern mathematics that is free to be used by any curious scholar!

The road passes through Computer Science land of Oz!

Some hot topics

- Big data and fast ($O(n \log n)$) algorithms.
- Machine learning and foundations of AI.
- Provable cryptography.
- Quantum computation.
- Theoretical biology.
- Network theory and modeling.
- Image processing and computer graphics.
- Compressed sensing
- Foundations of mathematics and programming.

A quotation!

In the prelude of “Récoltes et Semailles", Alexandre Grothendieck makes the following points on the search for relevant geometric models for physics and on Riemann’s lecture on the foundations of geometry.
It must be already fifteen or twenty years ago that, leafing through the modest volume constituting the complete works of Riemann, I was struck by a remark of his “in passing".

... it could well be that the ultimate structure of space is discrete, while the continuous representations that we make of it constitute perhaps a simplification (perhaps excessive, in the long run ...) of a more complex reality; That for the human mind, “the continuous" was easier to grasp than the “discontinuous", and that it serves us, therefore, as an “approximation" to apprehend the discontinuous.

Thank you!

Comments are Welcome

daneshgar@sharif.ir