INDIAN INSTITUTE OF INFORMATION TECHNOLOGY

Deterministic, Non-Deterministic, Approximation, and Optimization algorithms

Dr. Animesh Chaturvedi
Assistant Professor: IIIT Dharwad
Post Doctorate: King's College London & The Alan Turing Institute
PhD: IIT Indore MTech: IIITDM Jabalpur

Deterministic and Non-deterministic algorithms

Computational complexity theory

- Fields in
  - Theoretical Computer Science
  - Analysis of Algorithms
- An algorithm solves a computation problem by mathematical steps.
- A computational problem (such as an algorithm) is a task solved by a computer.
- Focuses on classifying computational problems according to the resource usage
- Resource usage: amount of resources needed to solve computational problem,
- Resources: such as time and storage.

https://en.wikipedia.org/wiki/Computational_complexity_theory

Single Source Shortest-Paths Implementation

algorithm	restriction	typical case	worst case	extra space
topological sort	no directed cycles	$E + V$	$E + V$	V
Dijkstra (binary heap)	no negative weights	$E \log V$	$E \log V$	V
Bellman-Ford	no negative cycles	$E\ V$	$E\ V$	V
Bellman-Ford (queue-based)		$E + V$	$E\ V$	V

Easy

Medium

Hard

Remark 1. Directed cycles make the problem harder.

Remark 2. Negative weights make the problem harder.

Remark 3. Negative cycles makes the problem intractable.

ROBERT SEDGEWICK | KEVIN WAYNE. *Algorithms* 4th Edition, https://algs4.cs.princeton.edu/44sp/

Single Source Shortest-Paths Implementation

algorithm	restriction	typical case	worst case	extra space
topological sort	no directed cycles	$E + V$	$E + V$	V
Dijkstra (binary heap)	no negative weights	$E \log V$	$E \log V$	V
Bellman-Ford	no negative cycles	$E\ V$	$E\ V$	V
Bellman-Ford (queue-based)		$E + V$	$E\ V$	V

Remark 1. Directed cycles make the problem harder.

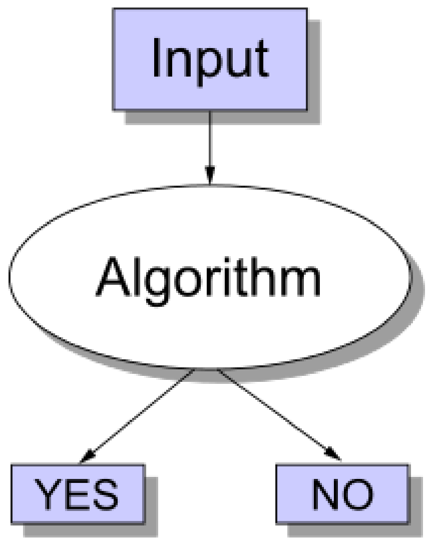
Remark 2. Negative weights make the problem harder.

Remark 3. Negative cycles makes the problem intractable.

ROBERT SEDGEWICK | KEVIN WAYNE. *Algorithms* 4th Edition, https://algs4.cs.princeton.edu/44sp/

Deterministic algorithm

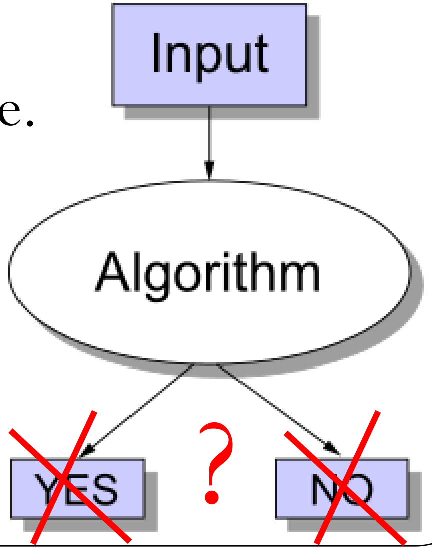
- Given a particular input, will always produce the same output, with the underlying machine always passing through the same sequence of states.
- State machine: a state describes what a machine is doing at a particular instant in time.
- State machines pass in a discrete manner from one state to another.
- Enter the input, initial state or start state.
- Current state determines what will be next state, the set of states is predetermined.



https://en.wikipedia.org/wiki/Deterministic_algorithm

Non-deterministic Algorithms

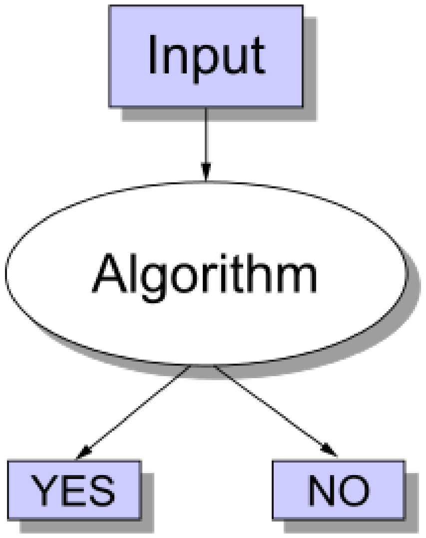
- If it uses external state other than the input, such as
  - user input,
  - a global variable,
  - a hardware timer value,
  - a random value, or
  - stored disk data.
- If it is timing-sensitive,
  - e.g. if it has multiple processors writing to the same data at the same time.
- If a hardware error causes its state to change in an unexpected way.
- The order each processor writes data will affect the result.

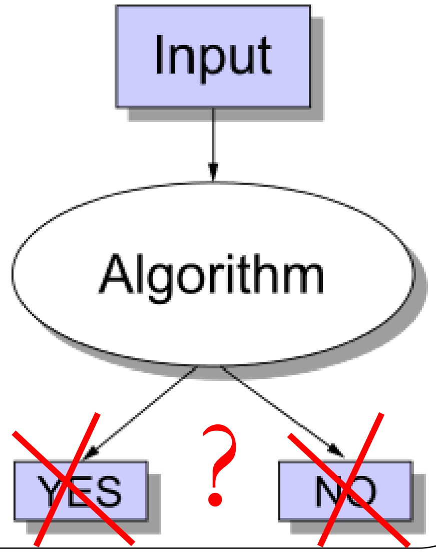


https://en.wikipedia.org/wiki/Deterministic_algorithm

Deterministic and Non-deterministic Algorithms

- Disadvantages of Determinism
  - predictable future by players or predictable security by hacker
  - e.g. predictable card shuffling program or security key
- Pseudorandom number generator is often not sufficient,
  - thus cryptographically secure pseudo-random number generator,
  - hardware random number generator.





https://en.wikipedia.org/wiki/Deterministic_algorithm

P (Polynomial) Time Problems

- Contains all decision problems that can be solved deterministically using a polynomial time (polynomial amount of computation time).
- A problem is in P-complete, if it is in P
- P is the class of computational problems that are "efficiently solvable" or "tractable".
- Class P, typically take all the "tractable" problems for a sequential algorithm,
- But, in reality, some problems not known to be solved in polynomial P time.

https://en.wikipedia.org/wiki/P_(complexity)
https://en.wikipedia.org/wiki/P-complete

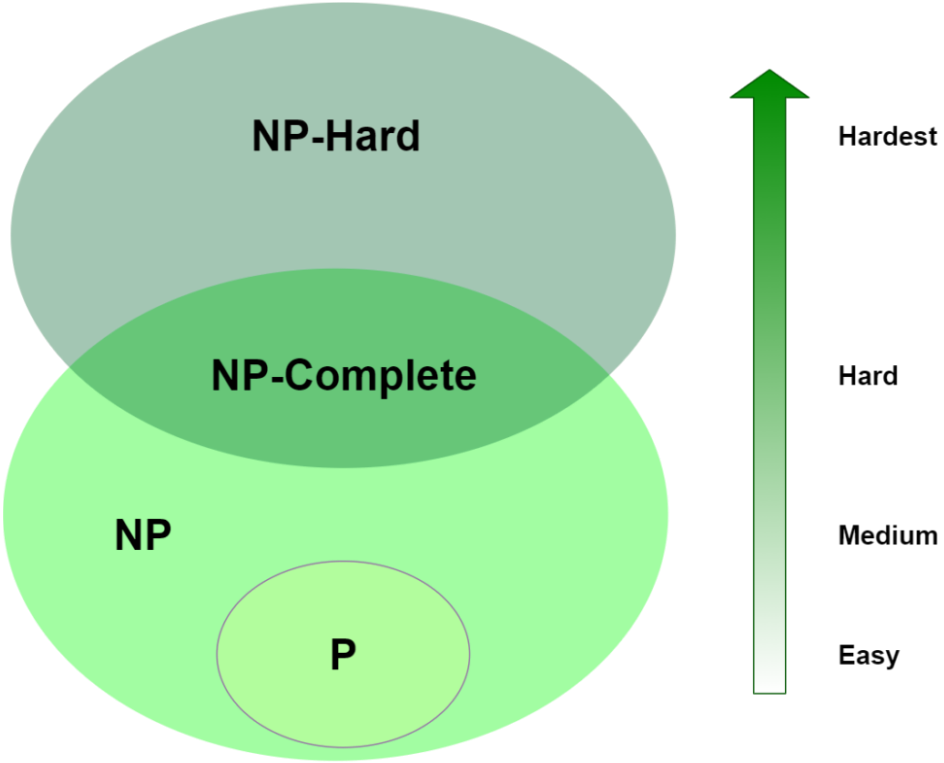
P (Polynomial) Time Problems

- Programmable Function (or method) is polynomial-time
  - if completes in constant-time or polynomial time,
  - then the entire algorithm takes polynomial time.
- Polynomial-time algorithms:
  - Minimum Spanning Tree: Kruskal's **O(*E* lg *V*)** and Prim's **O(*E* + *V* lg *V*)** algorithm
  - Shortest Path Algorithms: Djikstra's **O(E lgV)** and Bellman-Ford's **O(*E V*)** algorithm

https://en.wikipedia.org/wiki/P_(complexity)
https://en.wikipedia.org/wiki/P-complete

NP - Naming convention

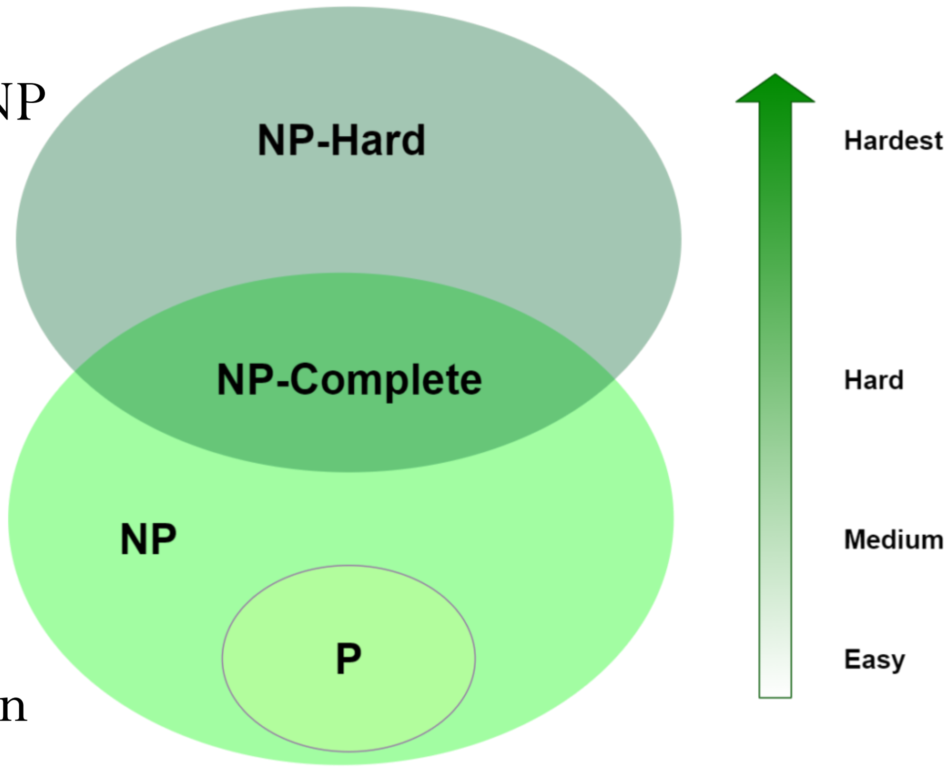
- Classification
  - Hardest → NP-Hard
  - Hard → NP-Complete
  - Medium → NP
  - Easy →P
- Order of N inputs
  - $O(1)$ – constant-time
  - $O(\log_2(n))$ – logarithmic-time
  - $O(n)$ – linear-time
  - $O(n^2)$ – quadratic-time
  - $O(n^k)$ – polynomial-time
  - $O(k^n)$ – exponential-time
  - $O(n!)$ – factorial-time



https://www.baeldung.com/cs/p-np-np-complete-np-hard

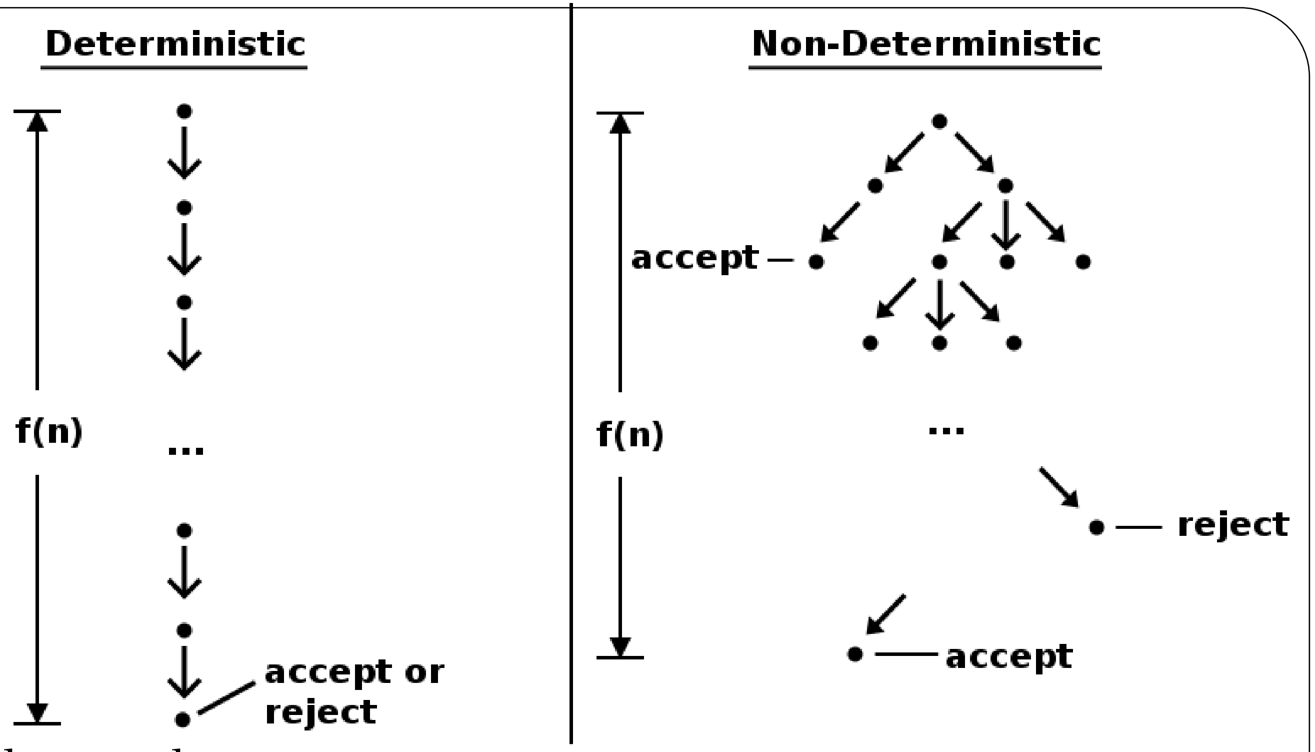
NP - Naming convention

- NP-hard: Class of problems are at least as hard as the hardest problems in NP.
- NP-hard problems do not have to be in NP; means NP hard problem may not even be decidable.
- NP-complete: Class of decision problems which contains the hardest problems in NP. Each NP-complete problem has to be in NP.
- NP: Class of computational decision problems for which any given yes-solution can be verified as a solution in polynomial time
- NP-easy: At most as hard as NP, but not necessarily in NP.



https://en.wikipedia.org/wiki/NP-hardness
https://www.baeldung.com/cs/p-np-np-complete-np-hard

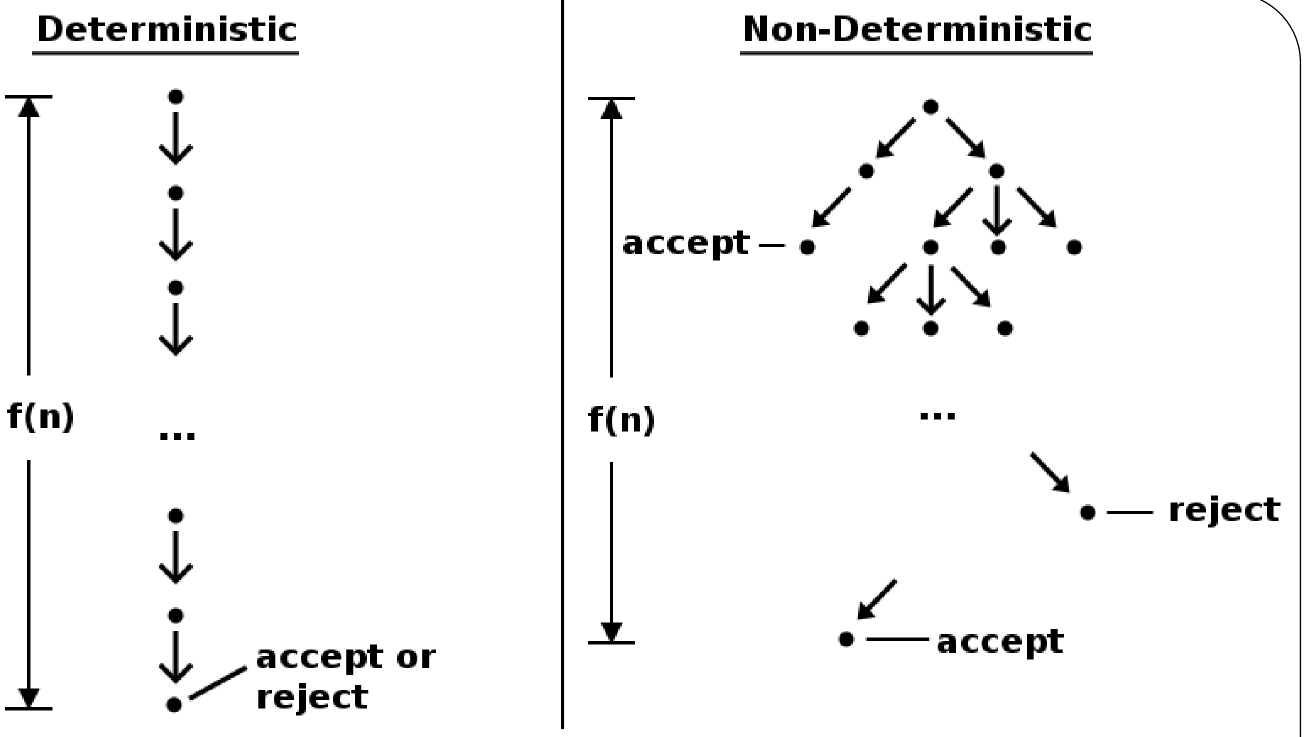
P and NP Problems



- Nondeterministic Polynomial-time
- "Nondeterministic" refers to
  - "luckiest possible guesser"
- "Complete" refers to
  - "in the same complexity class"
- **P** versus **NP** determine
  - whether a problem can be verified in polynomial time
  - whether the problem can also be solved in polynomial time.
- If it turned out that **P** ≠ **NP**, (widely accepted/believed),
- There are problems in **NP** that are harder to compute than to verify:
- NP problems could not be solved in polynomial time, but the answer could be verified in polynomial time.

https://en.wikipedia.org/wiki/P_versus_NP_problem

NP Complete



- Nondeterministic Polynomial-time

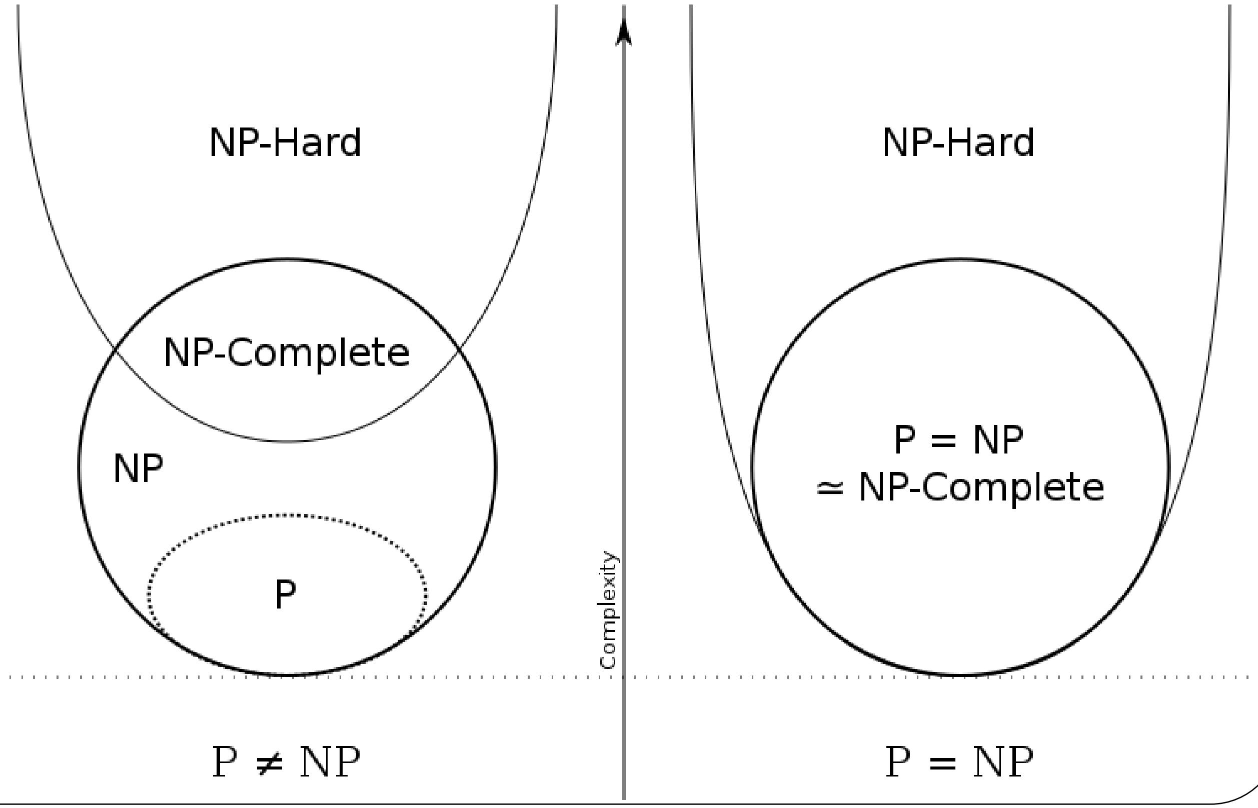
Complete

- A problem is NP-complete when:
  - a brute-force search algorithm can solve it, and the correctness of each solution can be verified quickly, and
  - the problem can be used to simulate any other problem with similar solvability.
- NP-complete problem can be *verified* "quickly",
- There is no known way to *find* a solution quickly.

https://en.wikipedia.org/wiki/NP-completeness

NP - Hard Problems

- Non-Deterministic Polynomial-time hardness
- At least as hard as the hardest problems in NP
- The might be some polynomial-time algorithms for NP-hard problems but might not been discovered yet
- NP-hard but not NP-complete
  - halting problem: "given a program and its input, will it run forever?"
  - traveling salesman problem

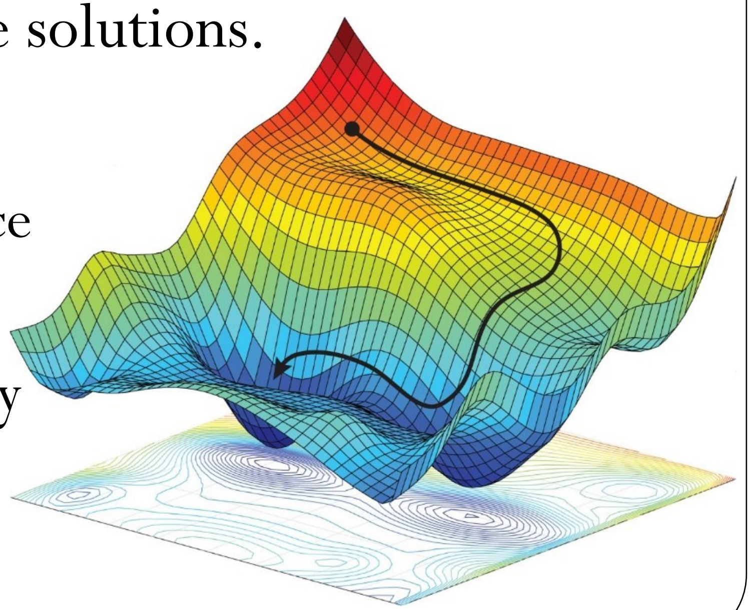


https://en.wikipedia.org/wiki/NP-hardness

Optimization, Approximation, and Probabilistic

Optimization

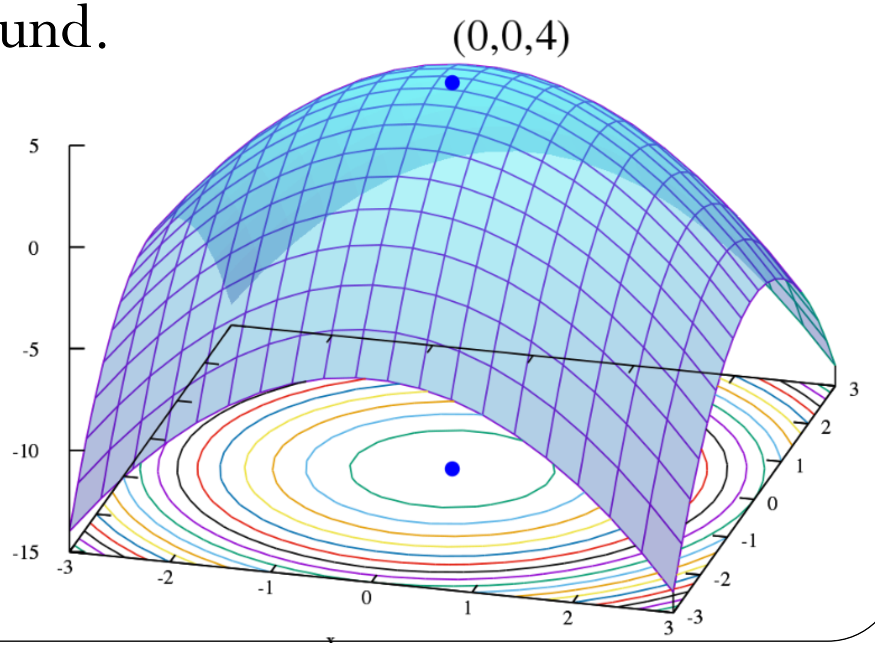
- An optimization problem can be represented in the following way:
  - Given: a function $f : A \longrightarrow \mathbb{R}$ from some set A to the real numbers
  - Goal: an element $x_0 \in A$ such that $f(x_0) \leq f(x)$ for all $x \in A$ ("minimization") or such that $f(x_0) \geq f(x)$ for all $x \in A$ ("maximization").
- The domain A of f is called the search space or the choice set
- The elements of A are called candidate solutions or feasible solutions.
- A is some subset of the
  - Euclidean space $\mathbb{R}^n$, Geometry represent N-dimensional space
  - Specified by a set of constraints,
  - Equalities or inequalities that the members of A have to satisfy



https://en.wikipedia.org/wiki/Mathematical_optimization

Optimization

- Problem of finding the best solution from all feasible solutions.
- **Discrete optimization:** A problem with discrete variables in which an object must be found from a countable set like integer, permutation or graph
  - **Combinatorial optimization**
- **Continuous optimization:** A problem with continuous variables in which an optimal value from a continuous function must be found.
  - **Constrained problems**
  - **Multimodal problems**



https://en.wikipedia.org/wiki/Optimization_problem

Combinatorial optimization

- finds an optimal object from a finite set of objects, where the set of feasible solutions is discrete or can be reduced to a discrete set.
  - Exhaustive search uses algorithms that quickly rule out large parts of the search space,
  - Or use **Approximation or Probabilistic algorithms**.
- A combinatorial optimization problem A is a quadruple (I, f, m, g), where
  - I is a set of instances; given an instance $x \in I$, $f(x)$ is the set of feasible solutions;
  - Given an instance x and a feasible solution y of x, $m(x, y)$ denotes the measure of y, which is usually a positive real.
  - g is the goal function, and is either min or max.
  - Goal is to find for some instance x an optimal solution, that is, a feasible solution of y

$$m(x,y) = g\{m(x,y') \mid y' \in f(x)\}.$$

https://en.wikipedia.org/wiki/Combinatorial_optimization

Constrained optimization

- is a process of optimizing an objective function with respect to some variables in the presence of constraints on those variables.
  - primarily equality constraints, inequality constraints, and integer constraints
  - set of candidate solutions that satisfy all constraints is called the feasible set
- A general constrained minimization problem may be written as follows:

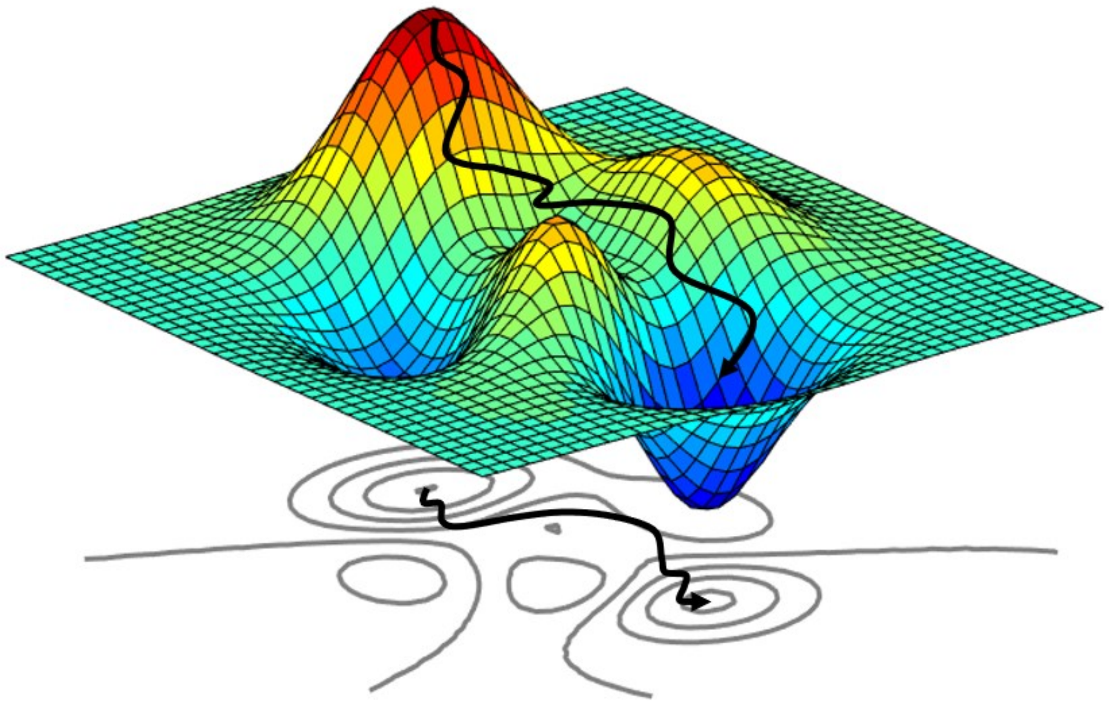
$$\begin{array}{rlll} \min & f(\mathbf{x}) & & \\ \text{subject to} & g_i(\mathbf{x}) = c_i & \text{for } i = 1, \ldots, n & \text{Equality constraints} \\ & h_j(\mathbf{x}) \geqq d_j & \text{for } j = 1, \ldots, m & \text{Inequality constraints} \end{array}$$

where $g_i(x)$ and $h_j(x)$ are constraints that are required to be satisfied, and $f(x)$ is the objective function that needs to be optimized subject to the constraints

https://en.wikipedia.org/wiki/Constrained_optimization

Multimodal optimization

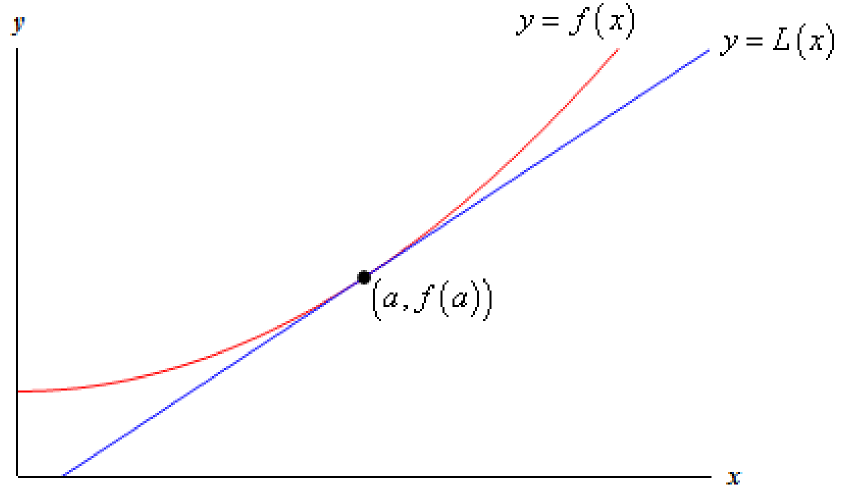
- finds all or most of the multiple (at least locally optimal) solutions of a problem, as opposed to a single best solution.
  - Evolutionary Multimodal Optimization is a branch of **Evolutionary Computation**
- **Evolutionary algorithms:**
  - Genetic Algorithms (GAs),
  - Evolution Strategy (ES),
  - Differential Evolution (DE),
  - Particle Swarm Optimization (PSO) etc.



https://en.wikipedia.org/wiki/Evolutionary_multimodal_optimization

Approximation

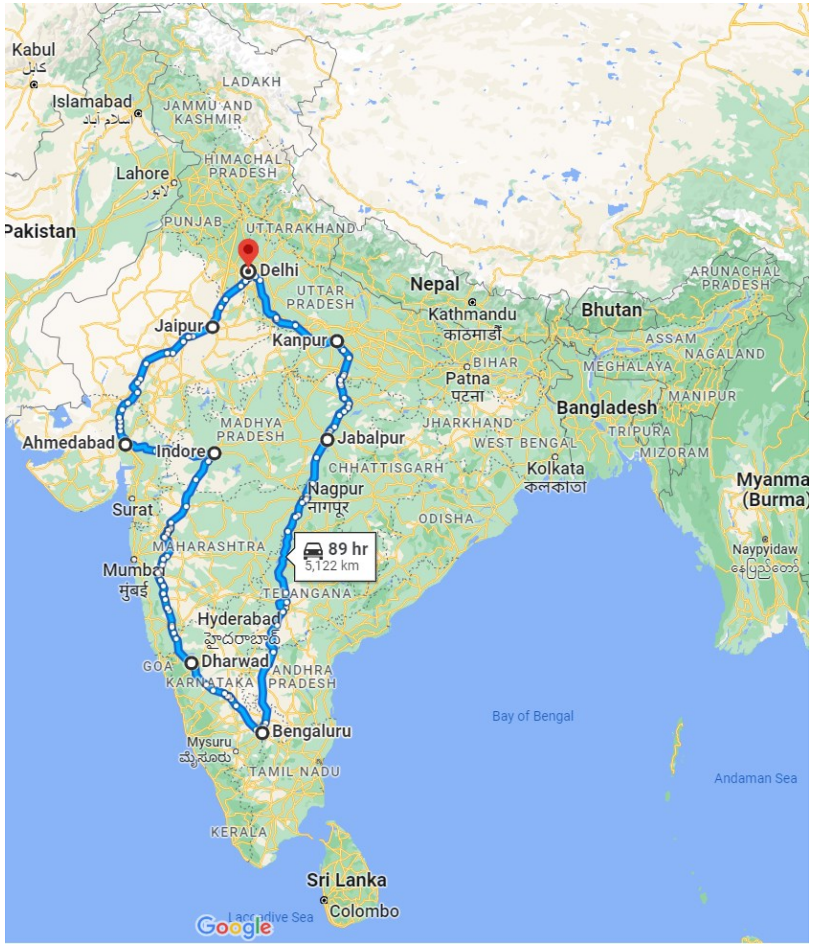
- Efficient algorithms that find **approximate solutions** to NP optimization problems.
- Solution with provable guarantees on the distance of the returned solution to the optimal one.
- Example:
  - Vertex Cover Problem,
  - Traveling Salesman Problem,
  - Set-covering problem (resource-selection problems)
  - Subset-sum problem



https://en.wikipedia.org/wiki/Approximation_algorithm

Approximation

- Karp (1972) proved the TSP to be NP-hard, but effective heuristic approximation methods were developed (Lin and Kernighan, 1973).
- The traveling-salesperson problem (TSP) is a standard combinatorial problem in theoretical computer science (Lawler et al., 1992).
- Arora (1998) devised a fully polynomial approximation scheme for Euclidean TSPs.

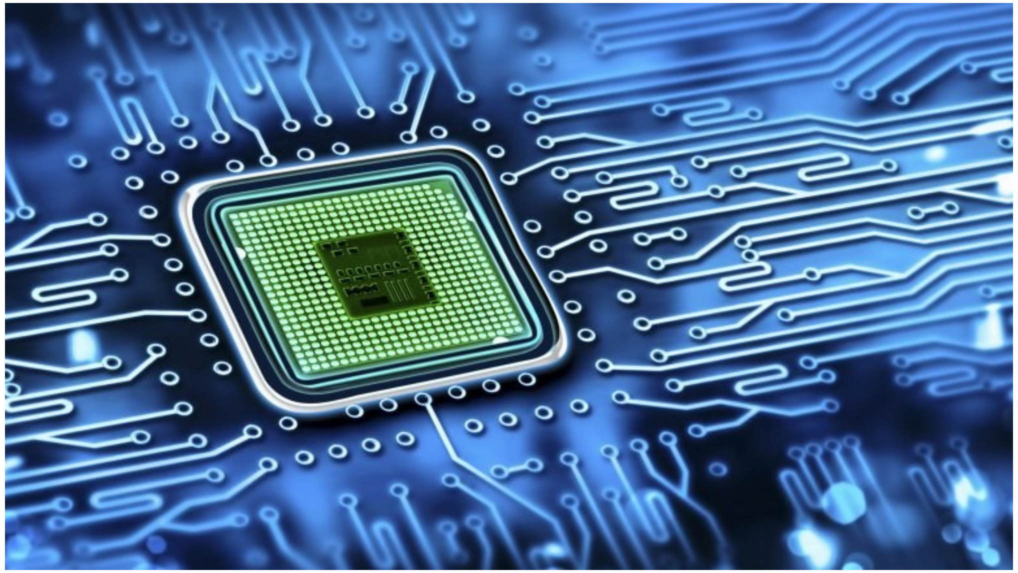


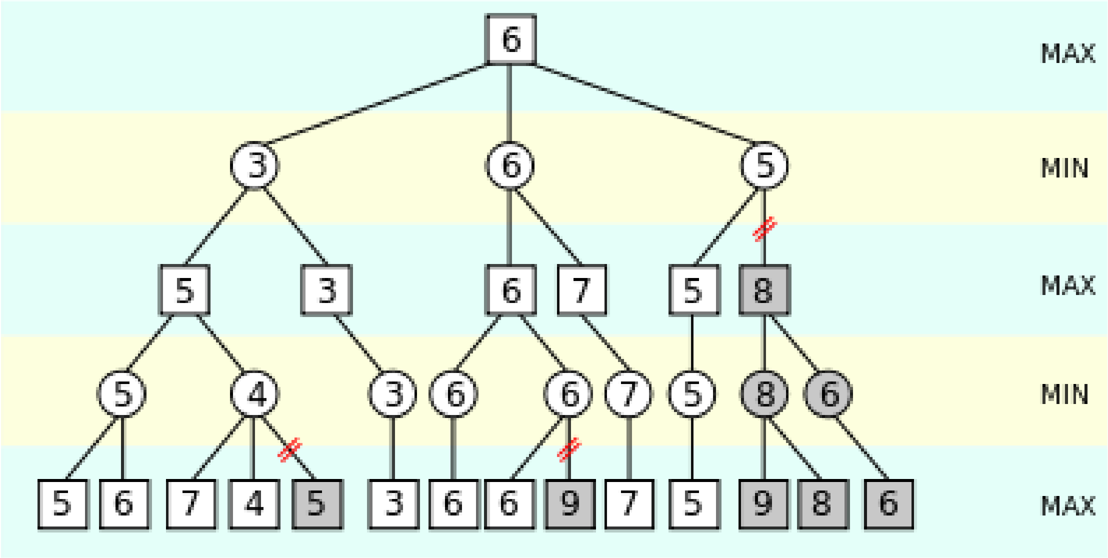
Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms* (Vol. 3, pp. 624-642). Cambridge: MIT press.

Stuart Russel, and Peter Norvig. "Artificial intelligence: A modern approach. Third edit." Upper Saddle River, New Jersey 7458 (2015). http://aima.cs.berkeley.edu/

Approximation

- VLSI layout methods are surveyed by Shahookar and Mazumder (1991), and many layout optimization papers appear in VLSI journals.
- Approximation in Mini-Max is to cut the search off at some point





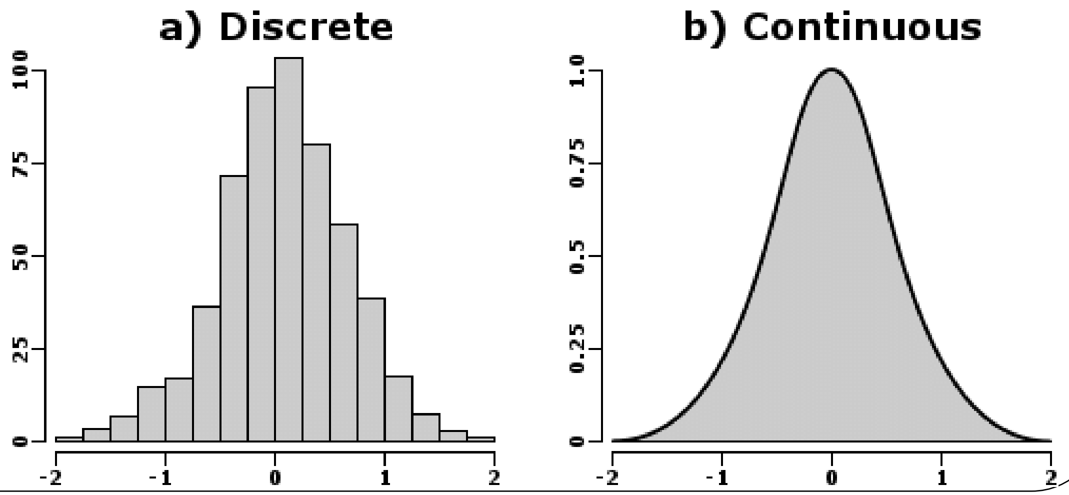
Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms* (Vol. 3, pp. 624-642). Cambridge: MIT press.

Stuart Russel, and Peter Norvig. "Artificial intelligence: A modern approach. Third edit." Upper Saddle River, New Jersey 7458 (2015).

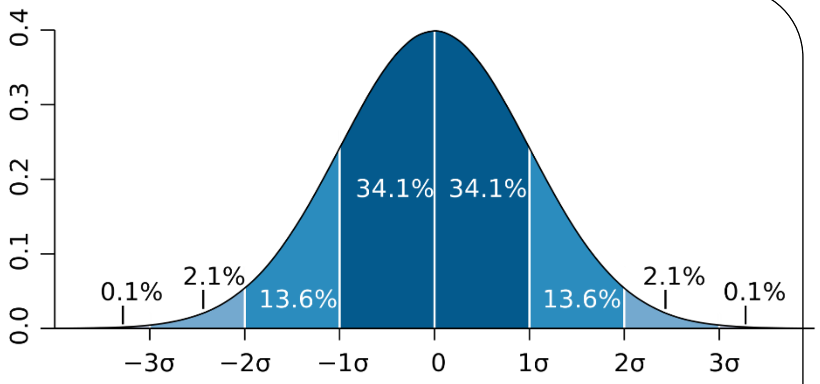
http://aima.cs.berkeley.edu/

Probabilistic Vs Stochastic

- Deterministic system are predictable.
  - The state of the system can be forecasted for given input, constraints, and mathematical model.
  - States of a deterministic system are pre-determined.
- Non-Deterministic system are unpredictable
  - Stochastic and Probabilistic are usually used interchangeably.
  - Both represent the randomness present in the system.
  - Probabilistic is superset concept of stochastic.



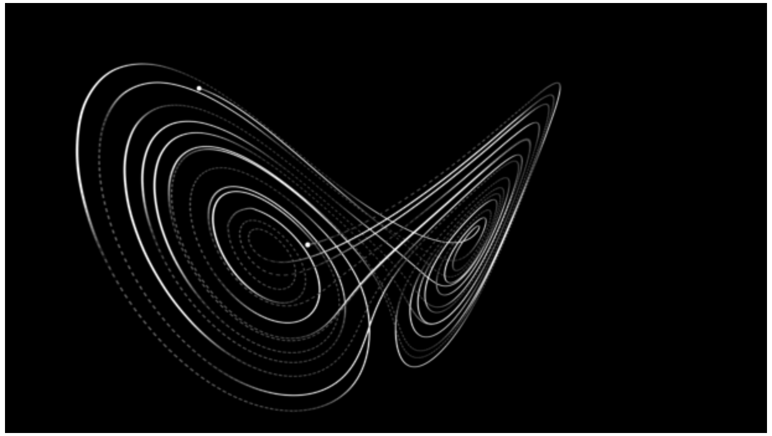
Probabilistic Vs Stochastic



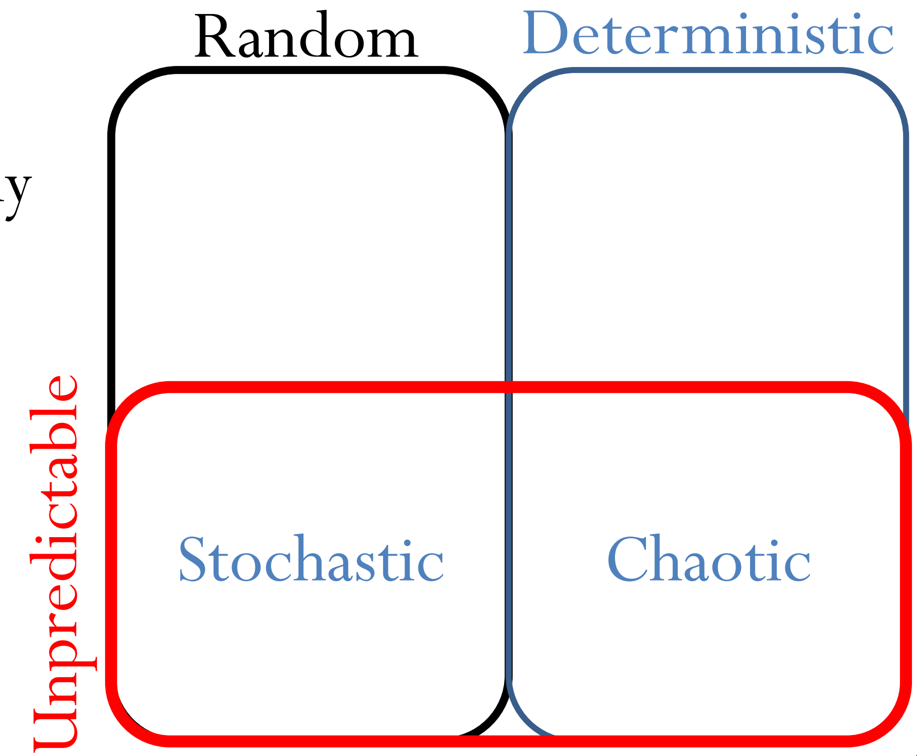
- Probabilistic models are independent of time,
  - which describes system with numerical chances or likelihood of an event to occur.
  - E.g., Lottery numbers are independent of each other.
  - Each instance is determined by the same probability distributions, but with no memory of older instance.
- Stochastic models are time dependent systems,
  - whose changes are described by its past and probabilities for successive changes.
  - E.g., Price of a stock is its old price and an uncertain change.
  - The uncertainty are small, which is semi-predictable.
  - If the stock was closed at 100,
    - then its opening value is unpredictable around 90 or 110.

https://qr.ae/pvU18E
https://qr.ae/pvU11n

Stochastic and Chaotic



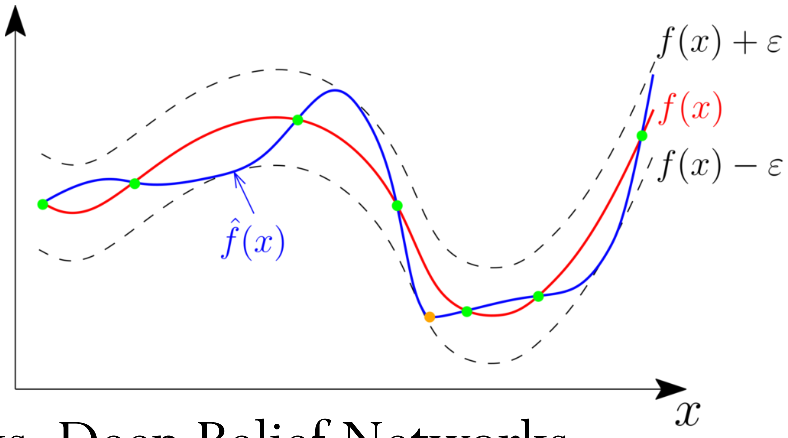
- A **chaotic** system is deterministic in theory.
  - It responds drastically to infinitesimal changes in initial and boundary conditions, making it in practice unpredictable and unstable.
- Deterministic chaos, or simply Chaos.
- **Chaos:** “When the present determines the future, but the approximate present does not approximately determine the future.”
- A **stochastic** system is a random phenomenon.
- Stochastic and Chaotic two terms interchangeably.
- It is hard to distinguish between chaotic and stochastic systems.

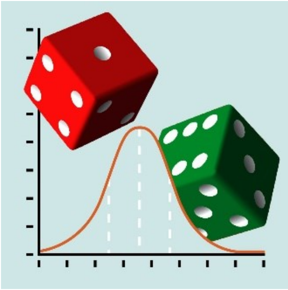


https://en.wikipedia.org/wiki/Chaos_theory
https://qr.ae/pvU11A

Optimization, Approximation, and Probabilistic

- Deterministic nature of a system does not make it predictable.
- Exhaustive search is intractable for NP-hard problems
- Used Optimization, Approximations, and Probabilistic algorithms in
  - Travelling Salesman Problem (TSP), Minimum Spanning Tree (MST), and Knapsack
  - Minimax, Alpha–Beta pruning,
  - Monte Carlo, Markov chain Monte Carlo,
  - Constraint Satisfaction Problems,
  - Likelihood weighting, Maximum-Likelihood,
  - Stochastic Differential Equations (SDEs),
  - Belief states, Belief propagation, Bayesian Networks, Deep Belief Networks,
  - Various Machine Learning, Deep Learning, Reinforcement Learning algorithms,
  - and several other problems





https://mattiheino.com/2016/12/27/deterministic-not-predictable/

References

- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms* (Vol. 3, pp. 624-642). Cambridge: MIT press.
- Stuart Russel, and Peter Norvig. "Artificial intelligence: A modern approach. Third edit." Upper Saddle River, New Jersey 7458 (2015). http://aima.cs.berkeley.edu/
- Wikipedia pages https://www.wikipedia.org/
- Images are from several sources e.g. movies, TV serials, internet, miscellaneous links, slides, blogs, etc.
- Quora
  - https://qr.ae/pvU18E
  - https://qr.ae/pvU11n
  - https://en.wikipedia.org/wiki/Chaos_theory
  - https://qr.ae/pvU11A

ขอบคุณ

Thai

Grazie

Italian

תודה רבה

Hebrew

Gracias

Spanish

Спасибо

Russian

English

Thank You

Obrigado

Portuguese

شكرا

Arabic

Merci

French

https://sites.google.com/site/animeshchaturvedi07

多謝

Traditional Chinese

Danke

German

धन्यवाद

Hindi

நன்றி

Tamil

Tamil

多谢

Simplified Chinese

ありがとうございました

Japanese

감사합니다

Korean