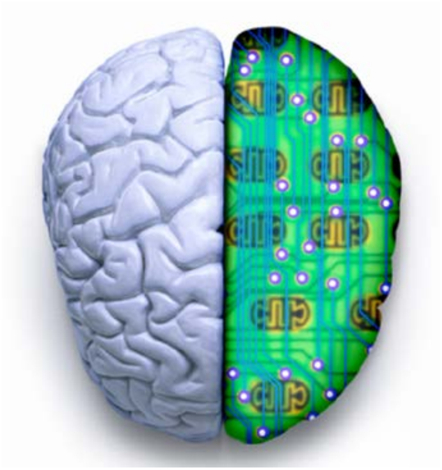
Advanced Artificial Intelligence CM4107 (Week 3)



Searching and Planning, Computational Game Theory

Dr. Yann Savoye
School of Computing Science and Digital Media
Robert Gordon University

Module Information

- **Assessment:**

  - Coursework (2 components)
    - Component 1: literature review
    - Component 2: paper implementation

  - No mid-term or final written exam.

    All deadlines are strong:
    - *It will not be possible to upload material after the deadline.*
    - *No deadline extension will be granted. No excuse.*
    - *Only the content submitted via the Moodle will be mark.*

Coursework

- **Submission of the Coursework Part 1:**

  Deadline - Monday, October 30th, 2017 23:00:

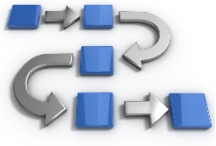
  Activity 1 and Activity 2
    - *2-pages written reports (in PDF format)*
    - *7 slides presentation (in PDF format)*

- You should have selected two papers! And started to read them …

Coursework

- **How to read a paper?** **"Three Passes Technique"**

- ***First Pass*** : *problem, key ideas*



- ***Second Pass*** : *high-level pipeline*

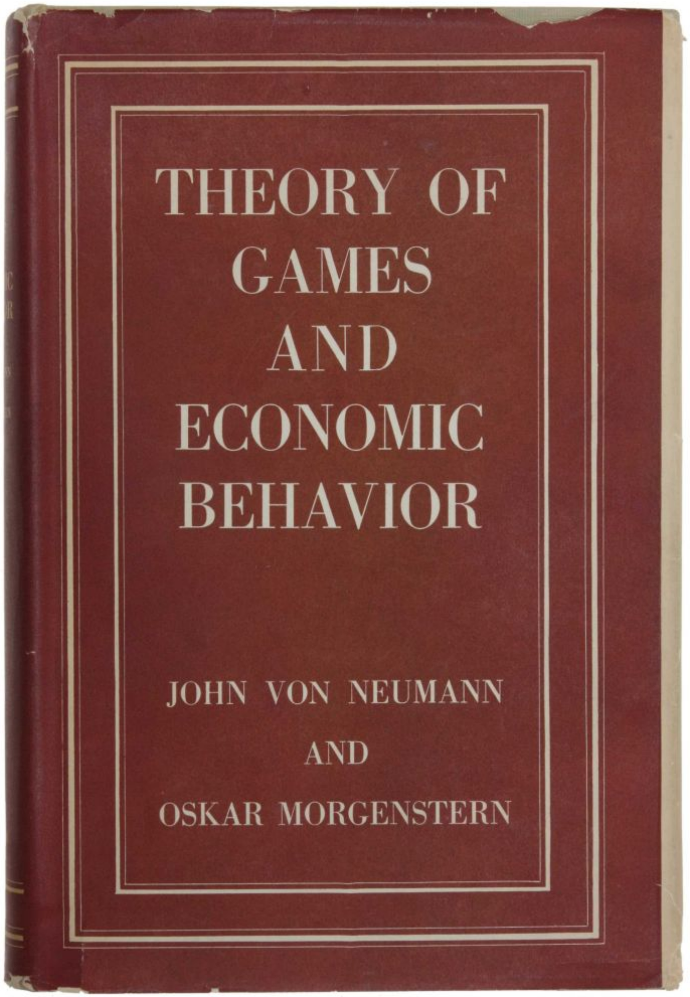
- ***Third Pass*** : *low- level details*

Overview

- Part I – Computational Game Theory
- Part II – Game Formalization and Solving Process
- Part III – Motion Planning and Behaviours
- Part IV – Searching
- Part V – Optimizing
- Part VI – Conclusions

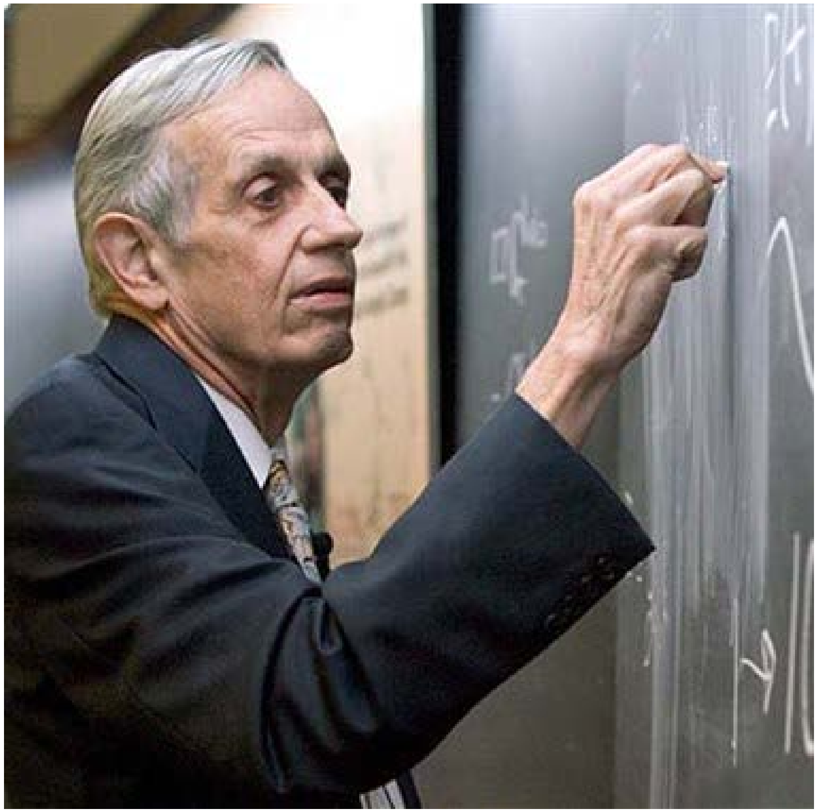
Part I – Computational Game Theory

Theory of Games



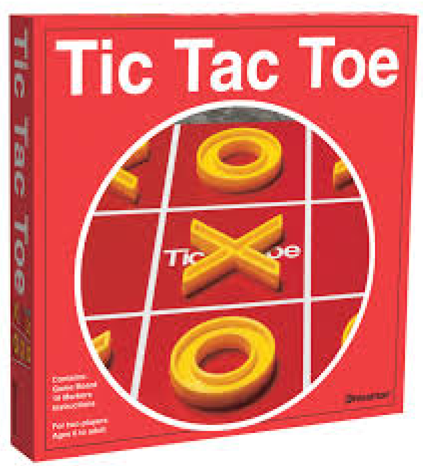
Game Theory *was developed by* ***John Von Neumann*** *in 1944*

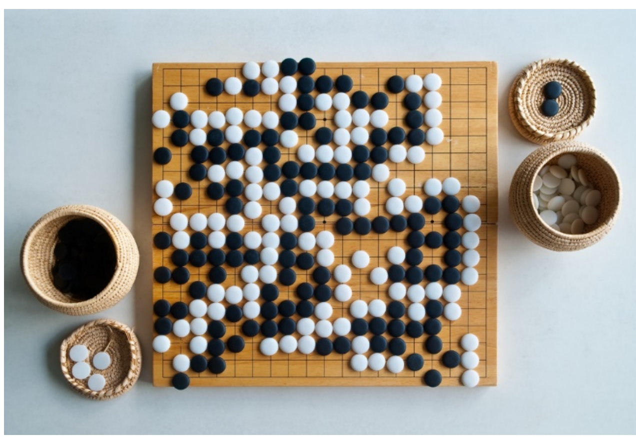
Nash Equilibrium



One of the fundamental principle of ***game theory*** *, the idea of* ***equilibrium*** *was developed by* ***John Nash***

Games



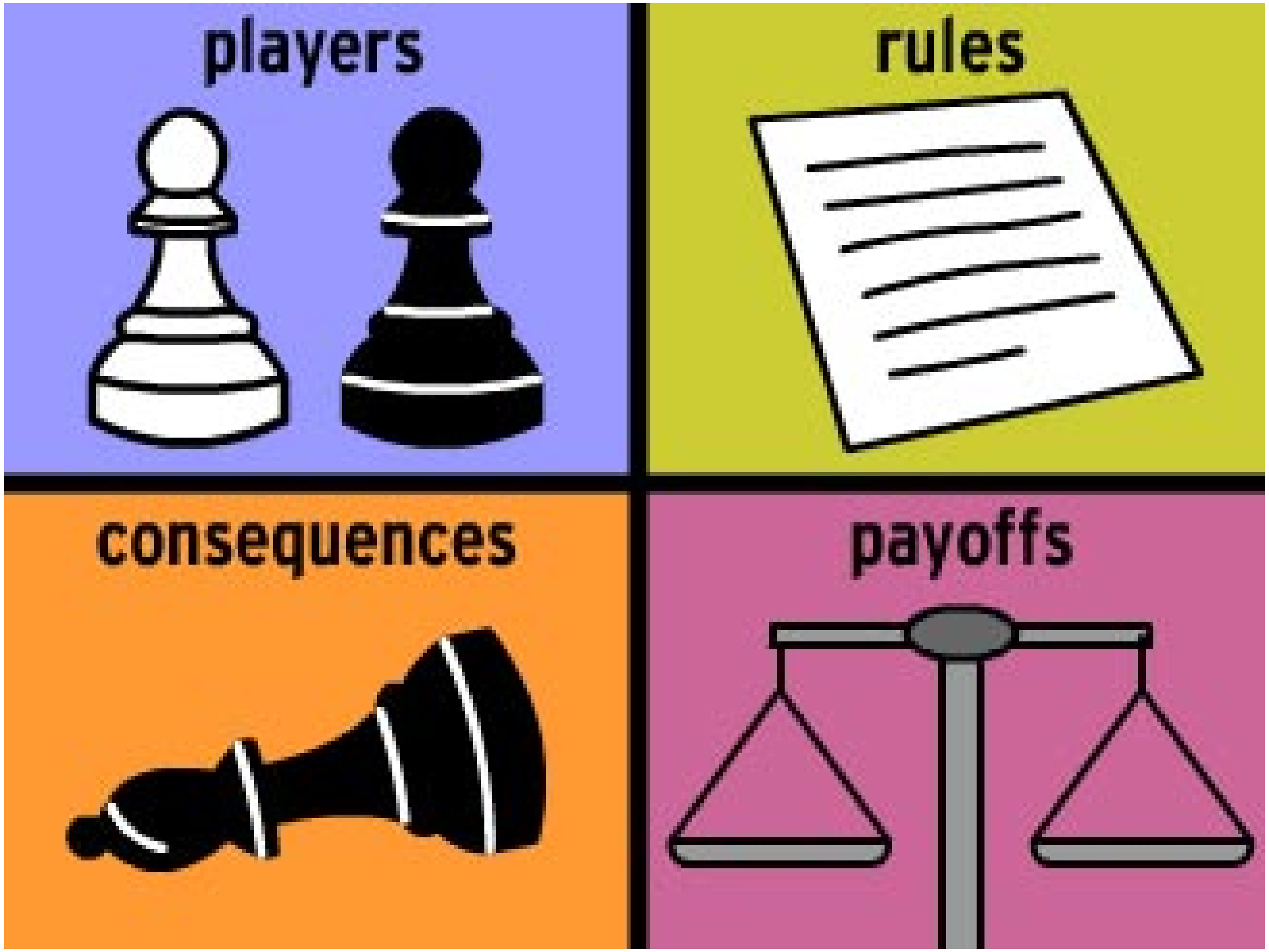


*Beyond what we call '**games**', such as chess, poker, soccer, it includes the **modelling of conflict***

What is a Game?

- A game : an **abstract model** for interacting **self-interested** players.
- **Abstract** = model **relevant to the decisions** that players make
- Focus on **decision-making** influencing the **outcomes**
- **Self-interest**: Players assumed to **act in their own interests**
- **Self-interested agents** : the game models an **optimization process.**
- A **game** consist of a set of **players, strategies, payoff.**
- **Players:** everyone who has an effect on your earnings
- **Strategies**: actions available to each player
- **Payoffs**: **Reflect the interests/**outcome **of the players**
- **Cooperative Game:** players form binding commitments
- **Non-cooperative Game:** player make decision independently.
- **Bayesian** games, **repeated** and **stochastic games**

What is game theory?



Game Theory *provides some interesting lessons in* ***strategic thinking****.*

What is game theory?

- **Game Theory = the theory of rational decision in conflicts.**
- Mathematics of determining **strategies** for optimal play.
- **Strategies for dealing with competitive situations.**
- Anticipation of the adversary’s play and respond accordingly.
- **AI community = dominant formalism for studying strategic.**
- **Game theory in various research discipline.**
- **Game-theoretic mechanisms in Economics.**
- **Computational** Game Theory: address **algorithmic issues.**
- **Optimize an objective function.**
- **Military** battles, **business** interactions, managerial **economics**

Brief introduction to Game theory

- **Game theory is the mathematical modelling of strategy**
- **Decision making**
- Popularized by **movies** such as "**A Beautiful Mind**“.
- **Interaction between agents** to achieve good outcomes
- **Strategic interaction** among rational (and irrational) agents.
- **Social choice theory** (i.e., collective decision making)
- **Representing game and strategy**
- The **optimal strategy** is based on the **minimax concept**
- Each player **maximizes the minimum** values obtainable.
- Designing **mechanisms to maximize** aggregate happiness.

Key Concepts

Cooperative outcome	An equilibrium in a game where the players agree to cooperate
Dominant strategy	A dominant strategy is one where a single strategy is best for a player regardless of what strategy other players in the game decide to use
Nash equilibrium	Any situation where all participants in a game are pursuing their best possible strategy given the strategies of all of the other participants
Tacit collusion	Where firms undertake actions that are likely to minimize a competitive response, e.g. avoiding price-cutting or not attacking each other's market
Whistle blowing	When one or more agents in a collusive agreement report it to the authorities
Zero sum game	An economic transaction in which whatever is gained by one party must be lost by the other.

Applications of Game Theory

- **Economy**
- Politics (vote, coalitions)
- Biology (Darwin's principle, evolutionary)
- **Anthropology**
- **War**
- **Management-labor arbitration**
- **Philosophy (morality and free-will)**
- **National Football league**
- **Trading**

Gamification *is the process of using*
game-based mechanism *to solve problem*

Strategies in Game

- **Kakutani'**s fixed point / **Minimax** / **Arrow** theorem
- **Dominance** solvability
- **Rationalizability**
- **Social choice** theory
- **Evolutionary** stable states and sets
- **Nash equilibrium**

- **Strategies are evaluated along the following dimensions**

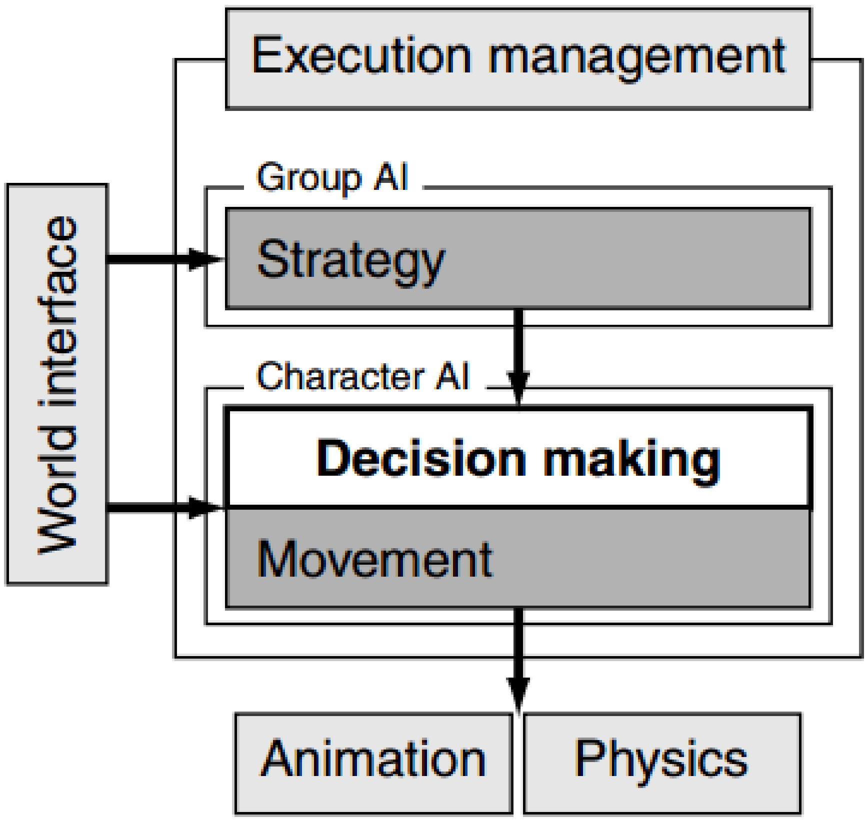
  Completeness*: always find a solution if exists?*

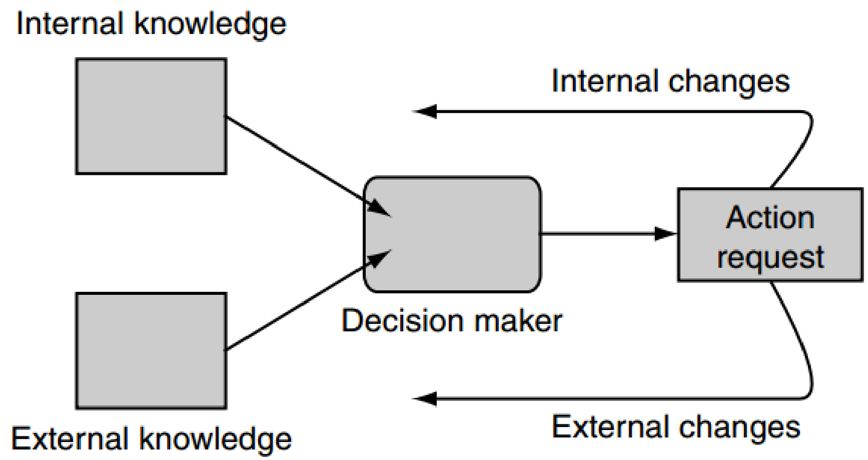
  Time complexity*: time a solution*

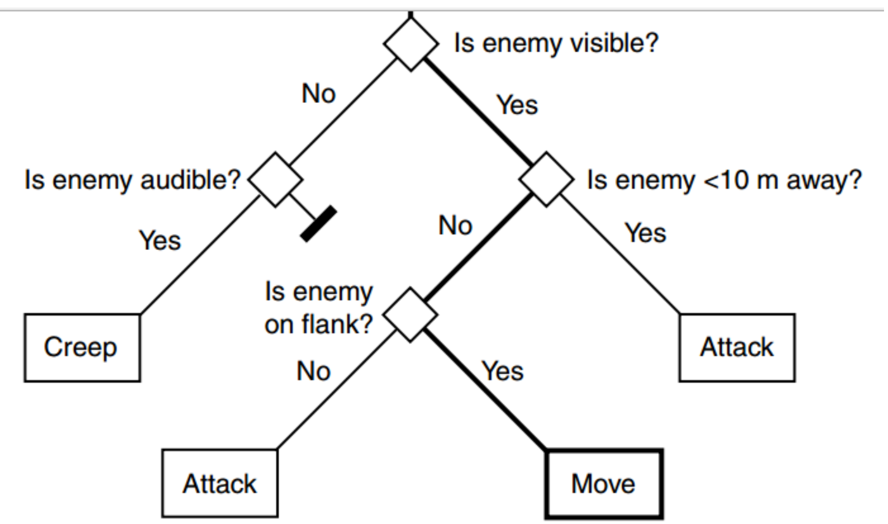
  Space Complexity*: enough memory ?*

  Optimality *: is the solution improvable ?*

Decision Making







Importance of Game Theory in AI

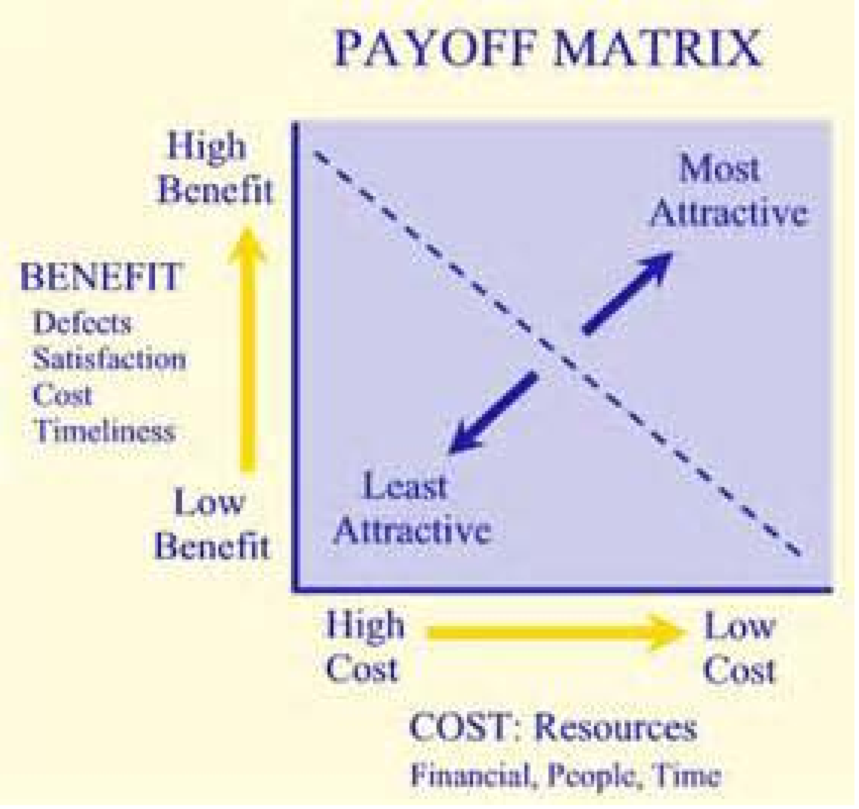
- **Game theory** and AI is a way to think and **model systems.**
- **Games** = the most **areas of progress** in the **AI space**
- Lot of **games mastered by AI** systems using break through algos.
- The **success of AI** is really tied to the progress on **game theory**.
- **Games** are a **materialization of game theory**.
- **Collaborating** or **competing** to accomplish a task can be gamified
- Several **aspects of game theory** that could help to **understand AI**
- Helps agents **select strategies**
- Guarantees about **artificially designed mechanisms**
- Automated analysis of **strategic models**

Dominance and Payoff Matrix

- **Dominance** = when one strategy is better than another strategy

- A strategy is dominant if a player earn a **larger payoff** than any other.

- **A payoff matrix** describes the **outcome** for each player and for each **set of strategic choices.**



The 3-doors Monty Hall Problem

You're given the choice of three doors: *Behind one door is a* ***car****; behind the others,* ***goats****.*

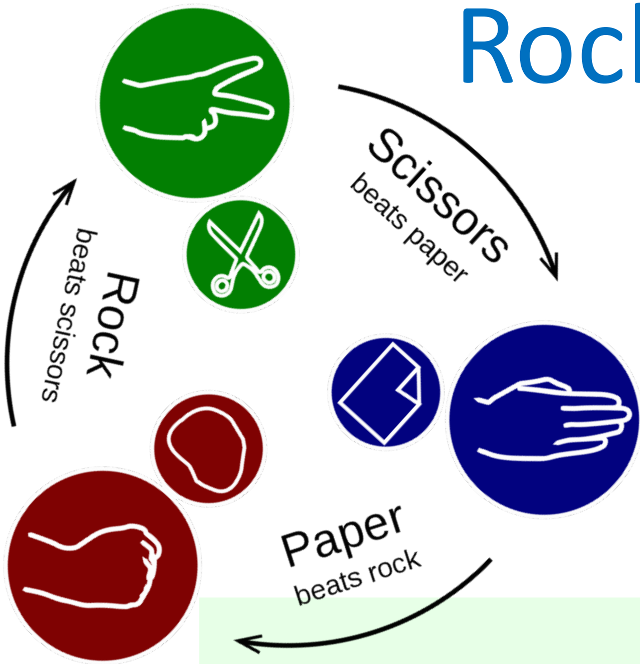
Is it to your advantage to switch your choice*?*

	DOOR 1	DOOR 2	DOOR 3	RESULT
GAME 1	Auto	Goat	Goat	Switch → **LOSE**
GAME 2	Goat	Auto	Goat	Switch → **WIN**
GAME 3	Goat	Goat	Auto	Switch → **WIN**
GAME 4	Auto	Goat	Goat	Stay → **WIN**
GAME 5	Goat	Auto	Goat	Stay → **LOSE**
GAME 6	Goat	Goat	Auto	Stay → **LOSE**

the dominance implies that a strategy maximizing the probability of winning the car.

always-switching strategies

Rock Paper Scissors



Gains of Player 1		Player 2 chooses...		
				
Player 1 chooses...		0, 0	1, -1	1, -1
		-1, 1	0, 0	-1, 1
		-1, 1	1, -1	0, 0

strict dominance

weak dominance

Prisoners' Dilemma

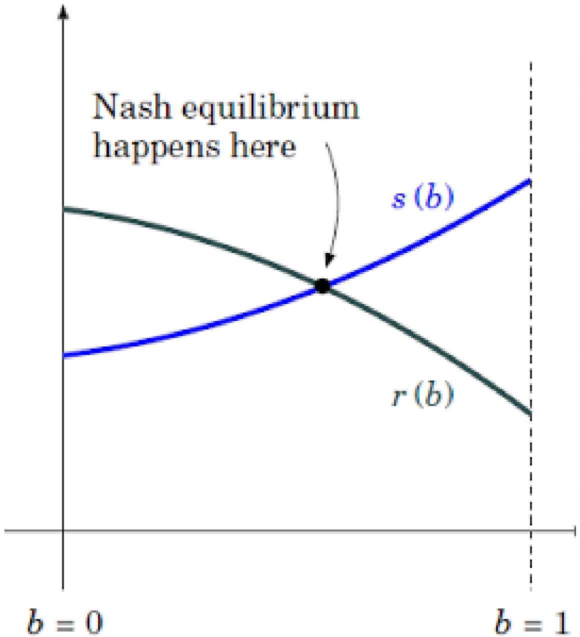
Betray

Cooperate

		Prisoner B	
		Confess	Keep quiet
Prisoner A	Confess	Both go to jail for ten years	Prisoner B gets life imprisonment, A goes free
	Keep quiet	Prisoner A gets life imprisonment, B goes free	Both go to jail for one year

*The **prisoner's dilemma** is a paradox in which two individuals acting in their own **self-interest pursue** a course of action that does not result in the **ideal outcome***

Nash Equilibrium



EQUILIBRIUM POINTS IN N-PERSON GAMES

BY JOHN F. NASH, JR.*

PRINCETON UNIVERSITY

Communicated by S. Lefschetz, November 16, 1949

One may define a concept of an n-person game in which each player has a finite set of pure strategies and in which a definite set of payments to the n players corresponds to each n-tuple of pure strategies, one strategy being taken for each player. For mixed strategies, which are probability

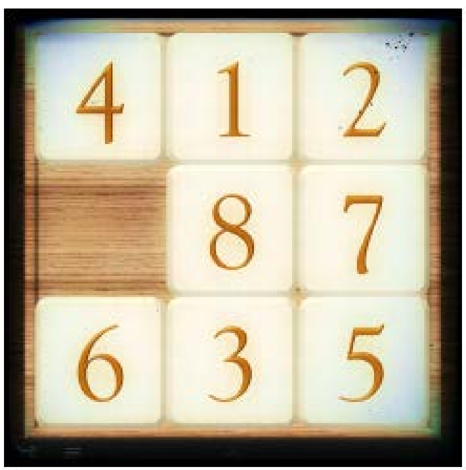
Nash Theorem : every game with a finite number of players and strategies . has at least one equilibrium (in pure or mixed strategy).

- **John Nash** = proved the existence of an equilibrium point in non-cooperative games
- **The defense adjust its pressure until the efficiencies of the offensive are equal**

Part II – Game Formalization and Solving Process

Formulating the 8-Puzzle Problem

- **8-puzzle** is a **sliding puzzle** that consists of a frame of **numbered square tiles in random order** with one tile missing.
- **States:** each represented by a 3x3 array of numbers In [0…8], where value 0 is for the empty cell.

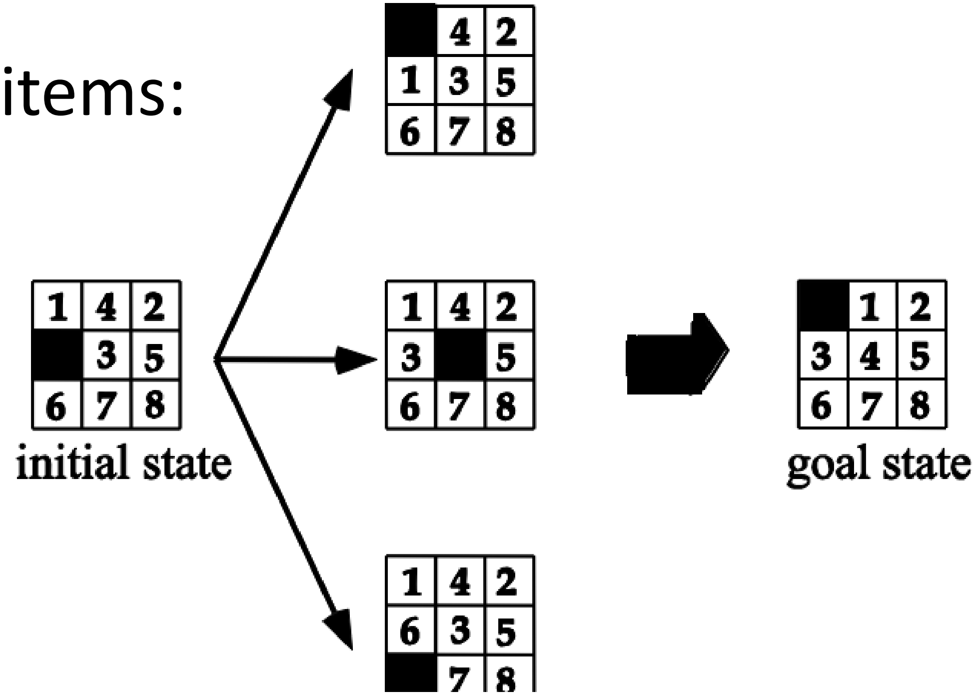


$$A = \begin{matrix} \mathbf{4} & \mathbf{1} & \mathbf{2} \\ \mathbf{0} & \mathbf{8} & \mathbf{7} \\ \mathbf{6} & \mathbf{3} & \mathbf{5} \end{matrix}$$

Problem Space Model

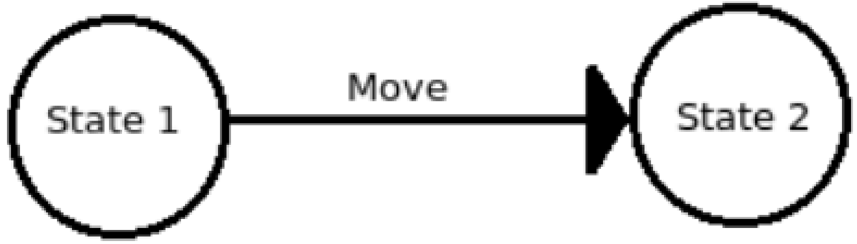
A problem is defined by 5 items:

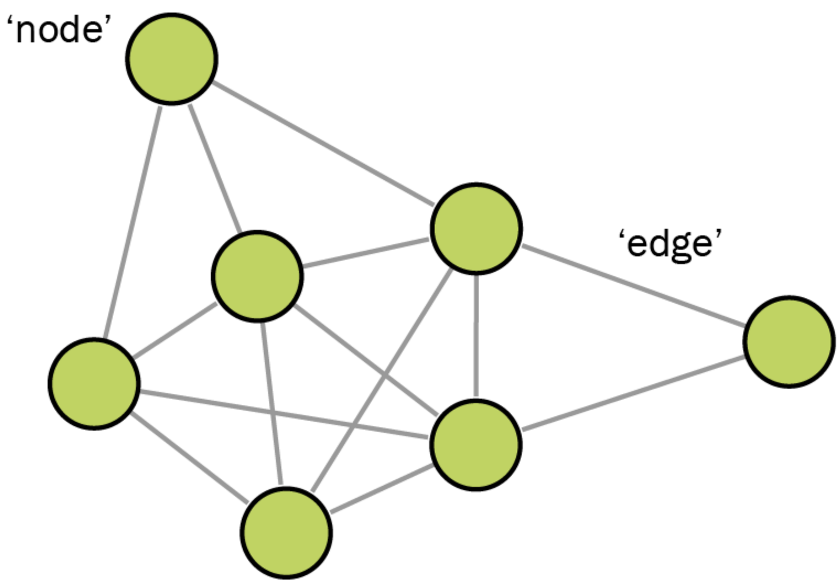
- Space of states
- **Initial state** (given)
- **Goal State** (given)
- Transition model
- Path



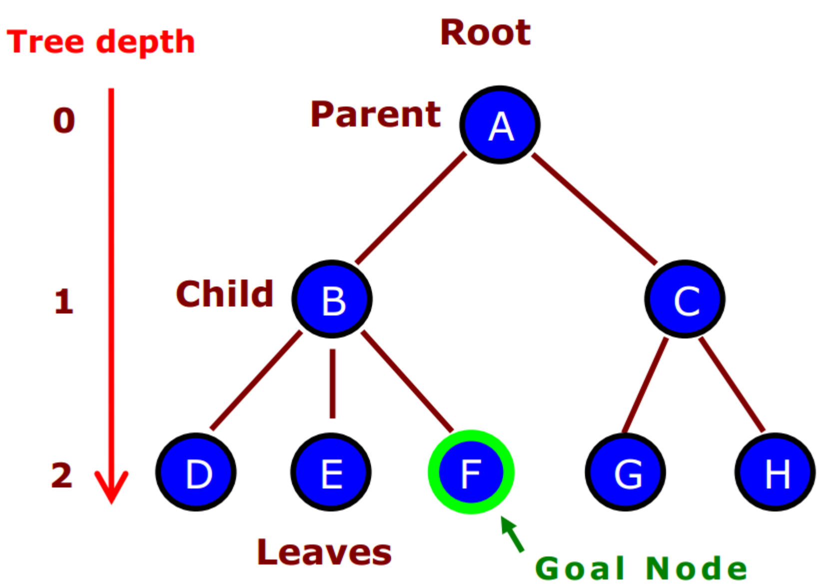
*A **solution** is a **sequence of actions** leading from the initial state to a goal state.*

Representation of States



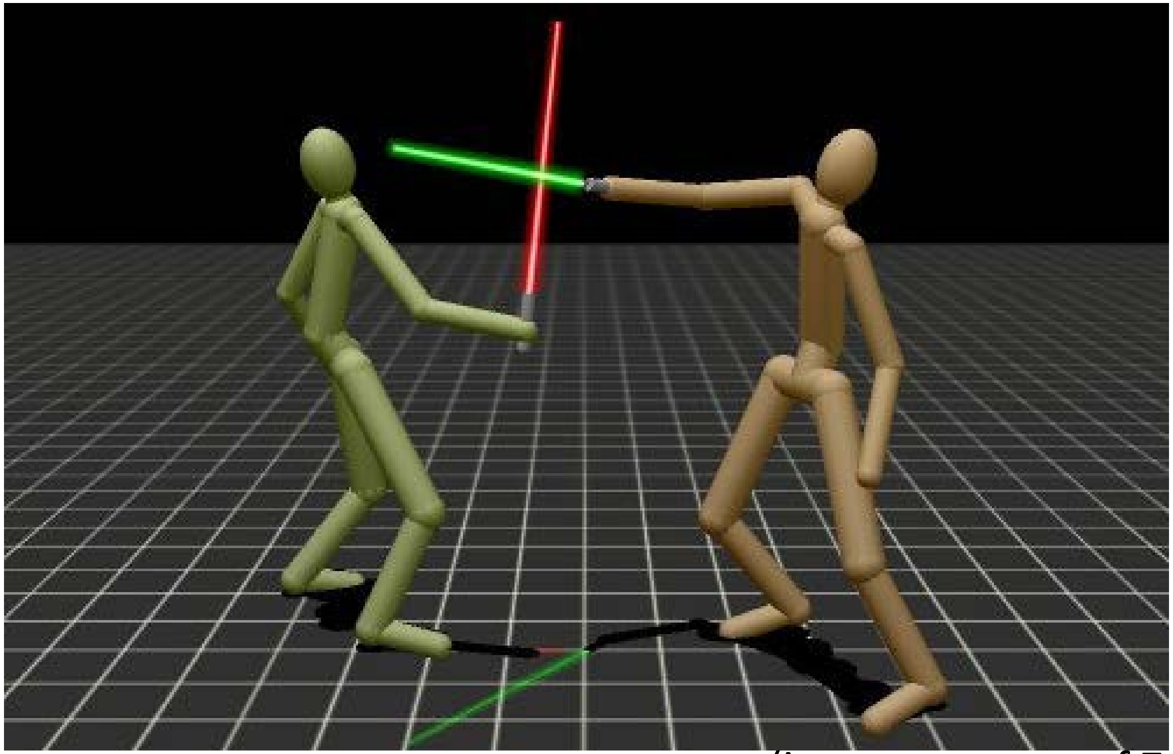


Graph



Tree

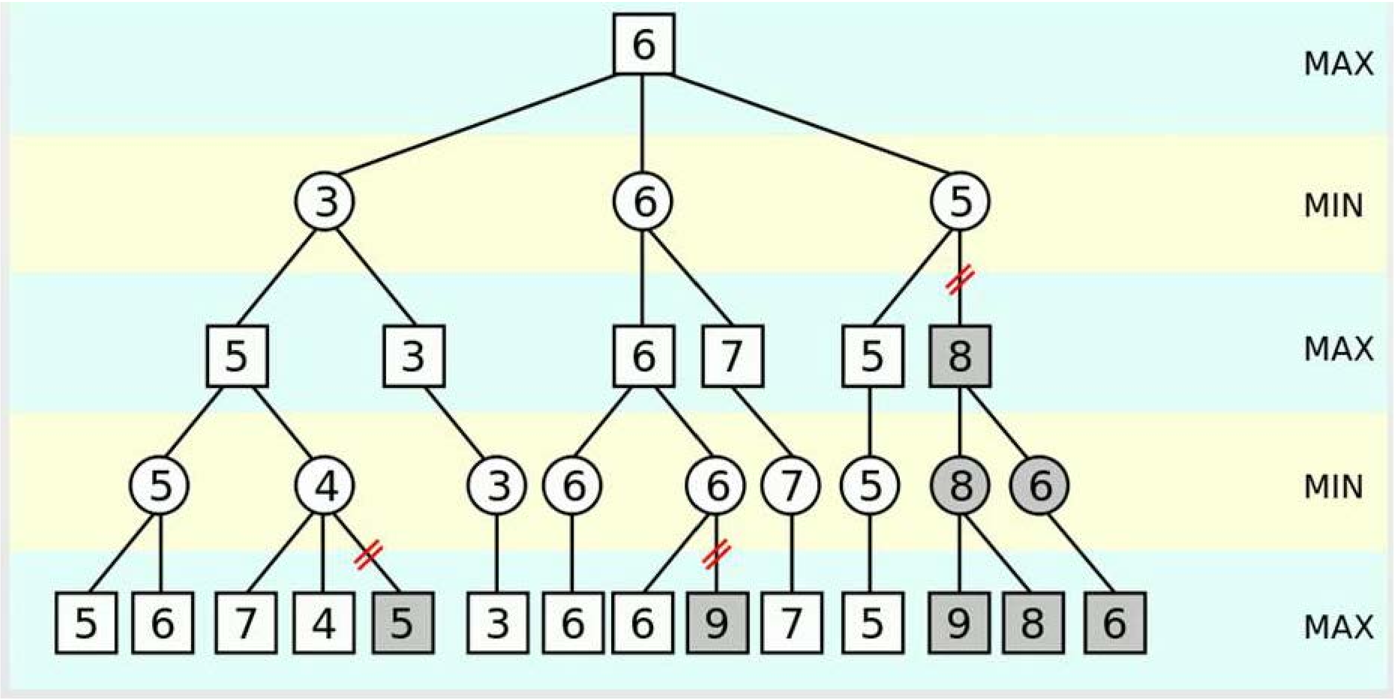
Adversarial Games



(image courtesy of Zoran Popovic)

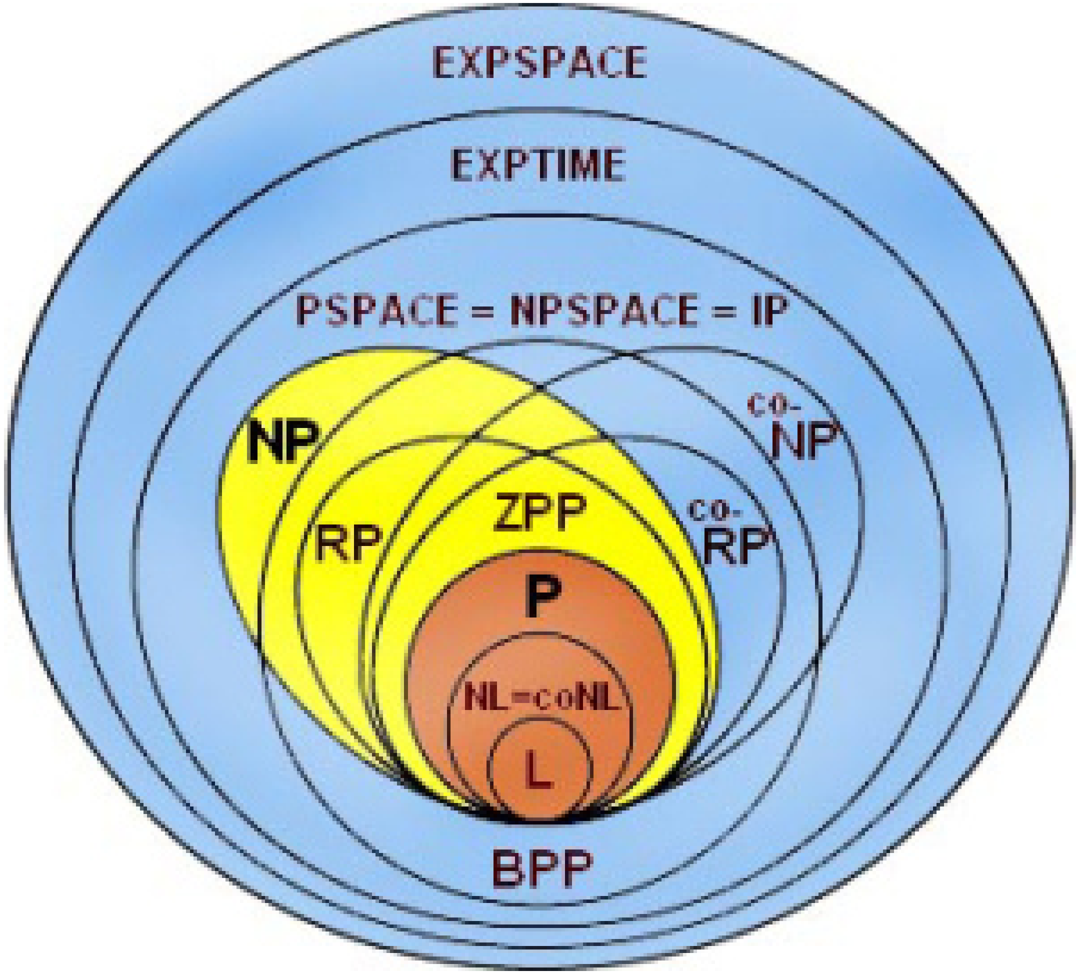
- **Exploit** the possible **strategies of an adversary**
- Controllers for characters in **competitive games.**
- Results **goals are in conflict**, giving rise to **adversarial search problem**
- Finds the move **that minimizes the maximum** utility of an **opponent**.

Alpha-Beta Pruning



- **Minimax:** to **minimize** the maximum **loss** (defensive)
- **Maximin**: to **maximize** the minimum **gain** (offensive)
- Minimax = Maximin
- Minimax search = **exponential in depth** of the tree.
- **Pruning**: **Eliminate some part** of the tree.

Complexity of Solving Games

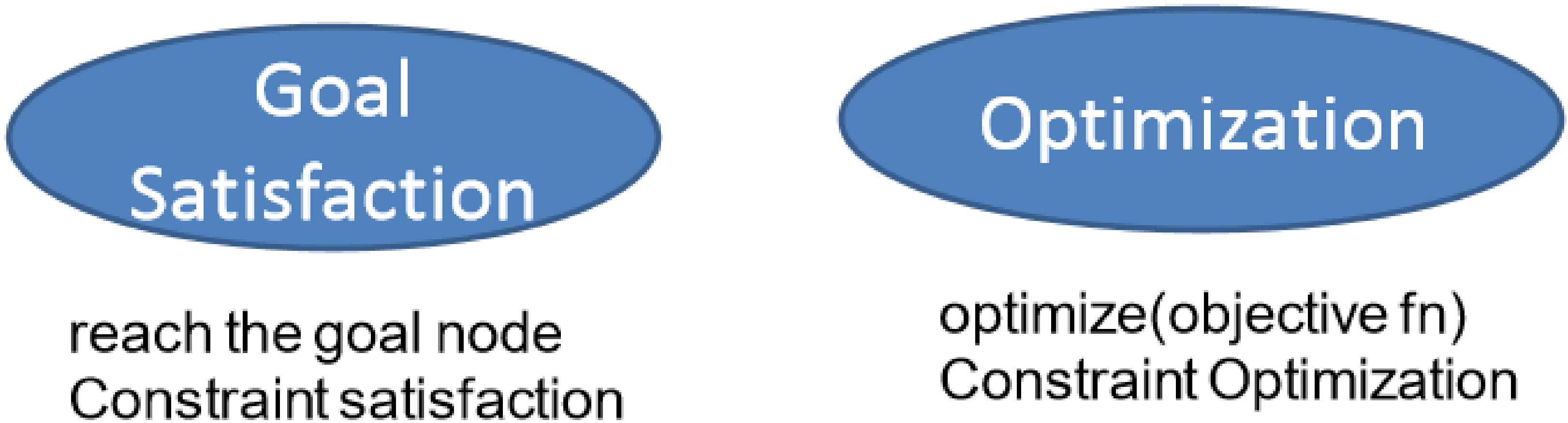


Optimal solution *of n-Puzzle family is* ***NP-hard.***

Complexity of Solving Games

- **Computational complexity of solving games**
- Determine if **game theory** can be used to **model real-world** settings
- **Solving** requires the use of **computational power**
- **Solution concept is realistic**
- **Complexity of solving** gives a **lower bound on complexity**

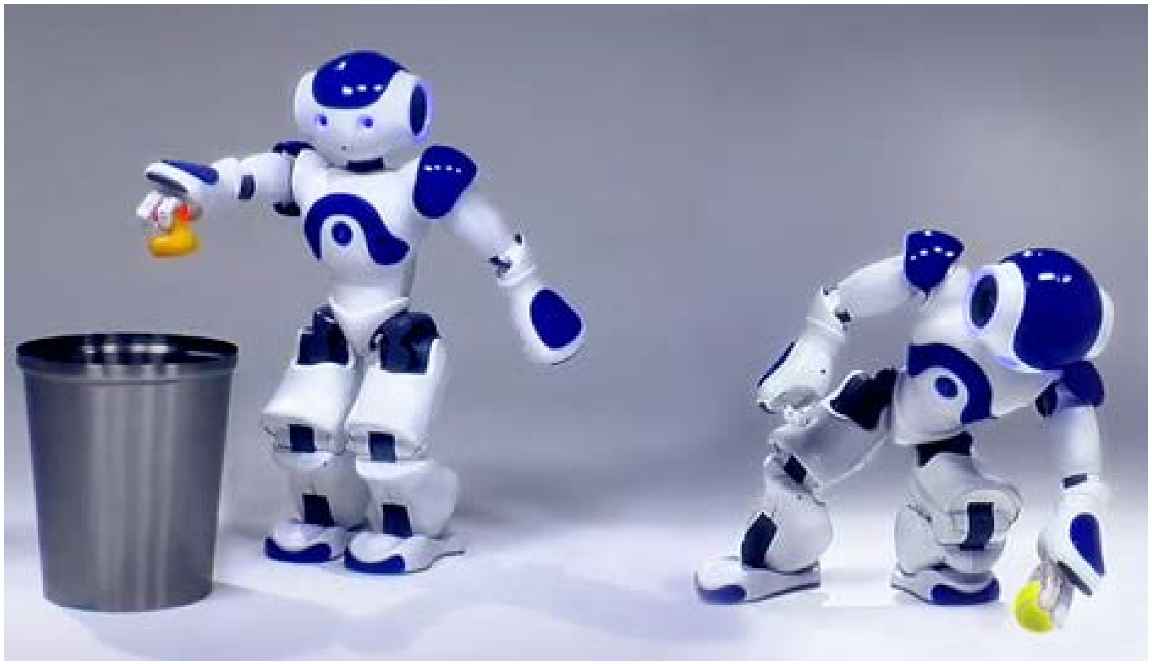
Solution Concepts

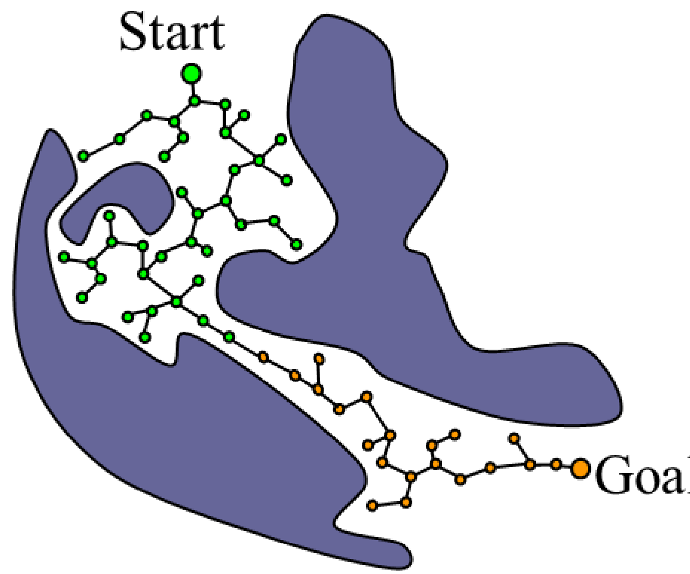


- **Existence:** Does a solution guarantee that the game is rational?
- **Uniqueness:** Does the solution is unique?
- **Tractability:** Is it computationally easy to verify?
- **Comprehensibility:** Is it easy to understand?
- **Invariance:** Do a small change in the setup lead to different solution?
- **Search** and **Optimization**

Part III – Motion Planning and Behaviors

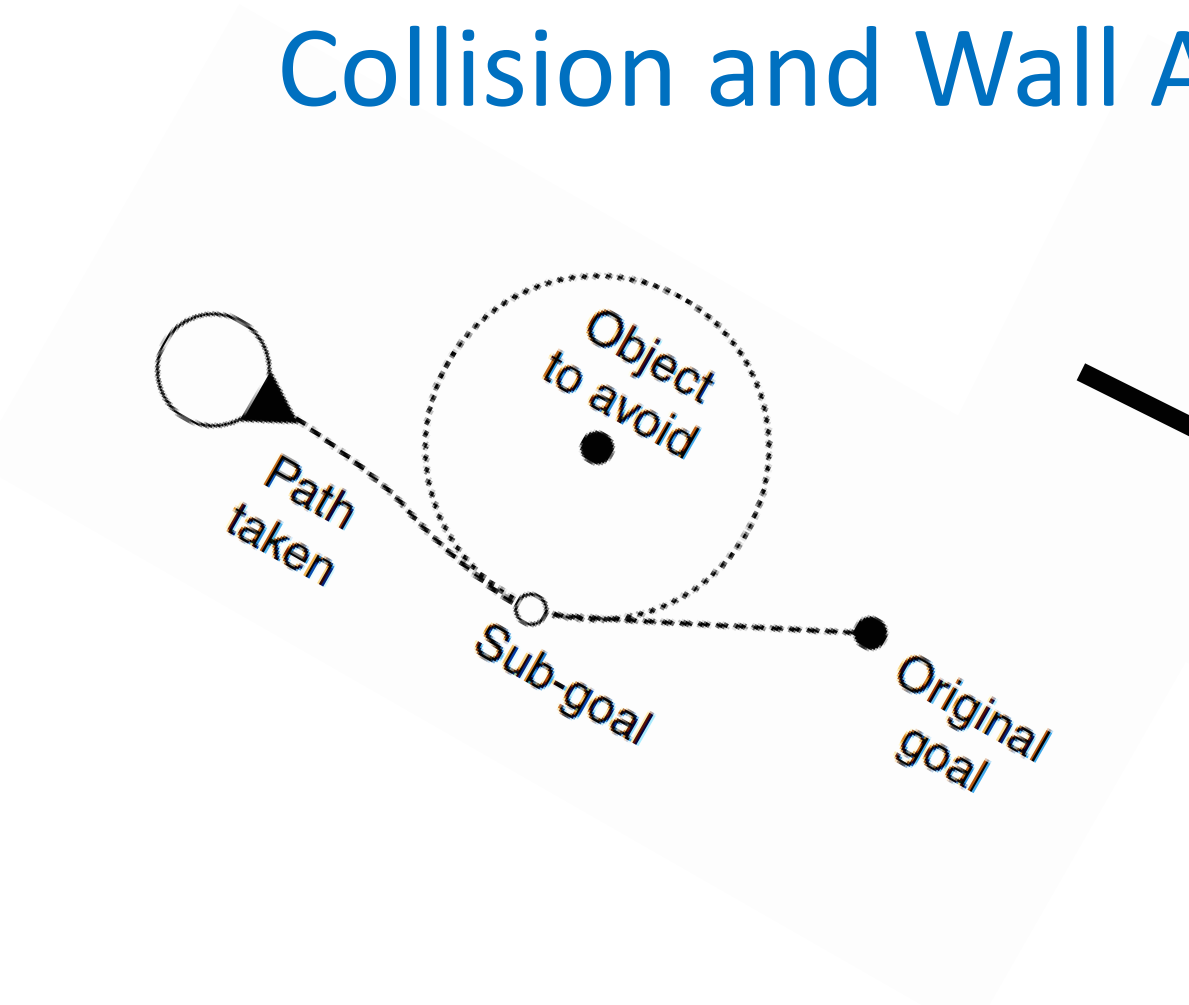
Motion Planning

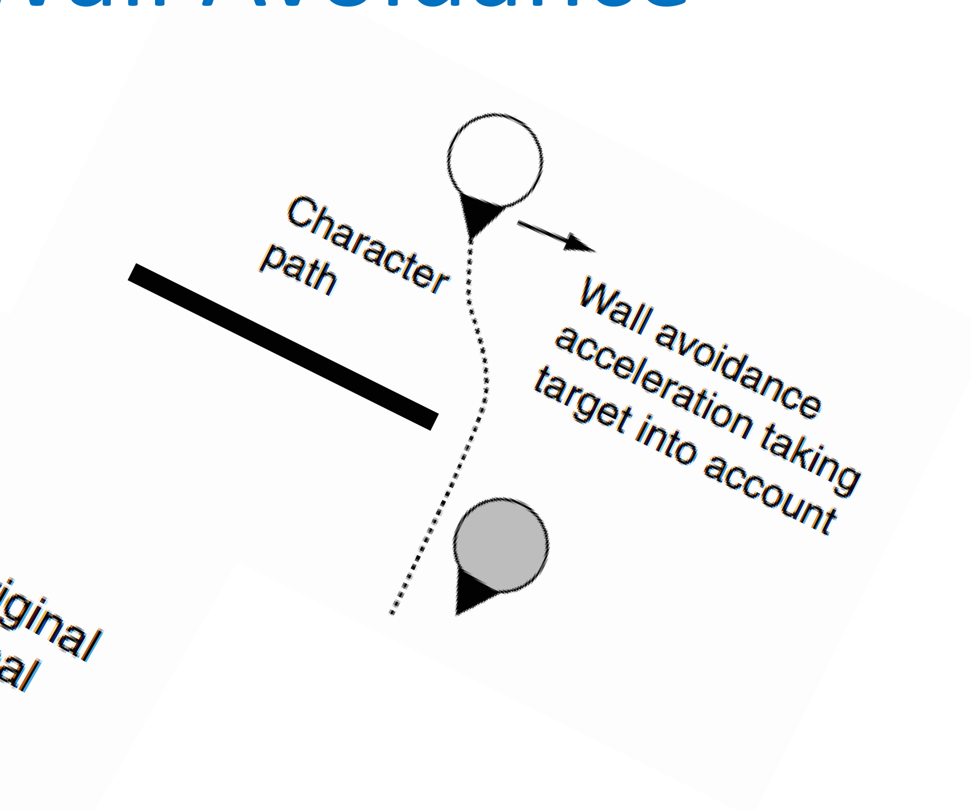




- **Motion planning** = breaking down a desired movement task
- Satisfy **movement constraints** and **optimize** the movement.
- **Motion planning** = algorithm about the movement.
- Development of **motion planning algorithms** for **physical robots**

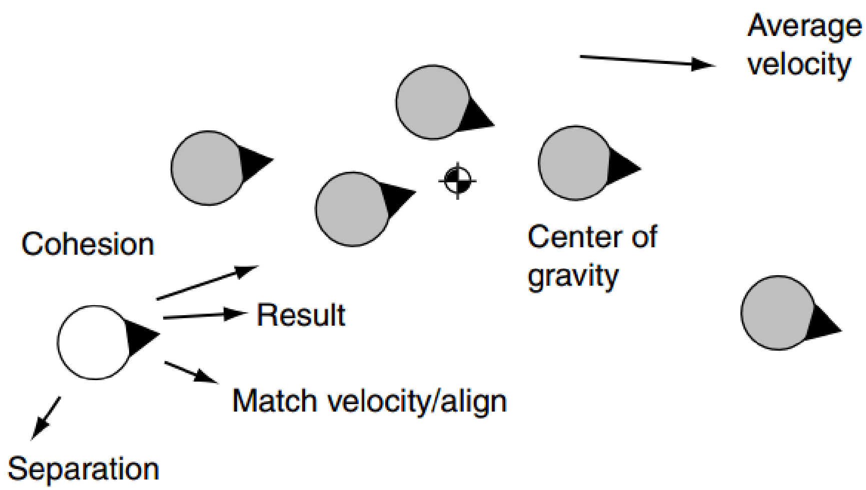
Collision and Wall Avoidance

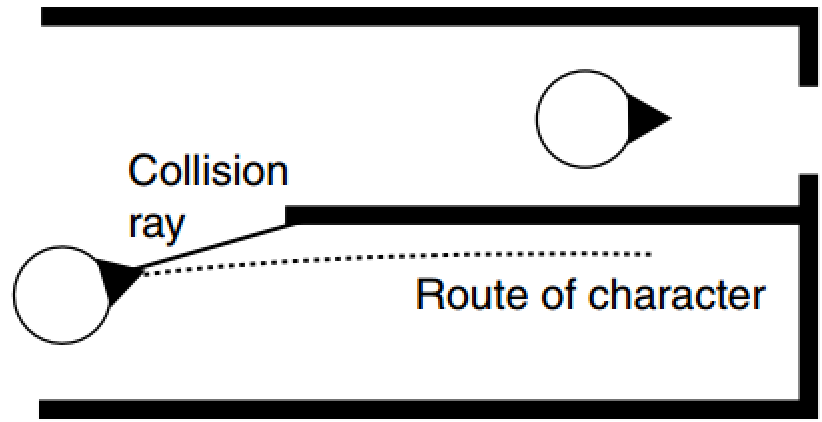




In robotics, **obstacle avoidance** is the task of satisfying some control objective subject to non-intersection or non-collision position constraints.

Flocking/ Steering Behaviors





- **Drives the flock** in the direction of a target/goal.
- Rules of Flocking Behaviors**: Alignment, Cohesion, and Separation**
- **Steering behaviours** control **goal-directed motions.**

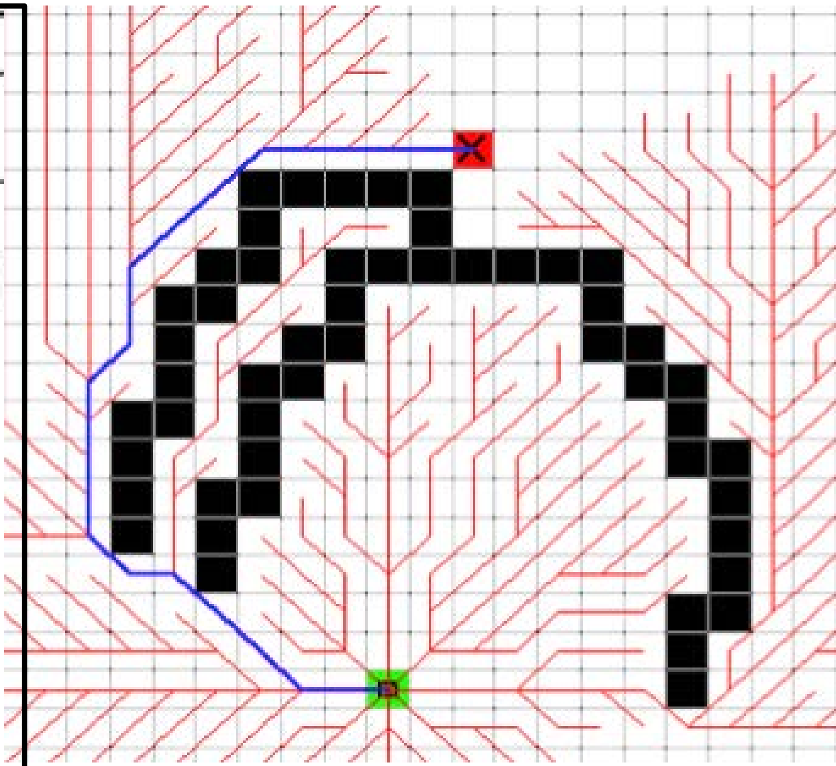
A* Path Finding Algorithm

Algorithm 24 A^* Algorithm

Input: A graph

Output: A path between start and goal nodes

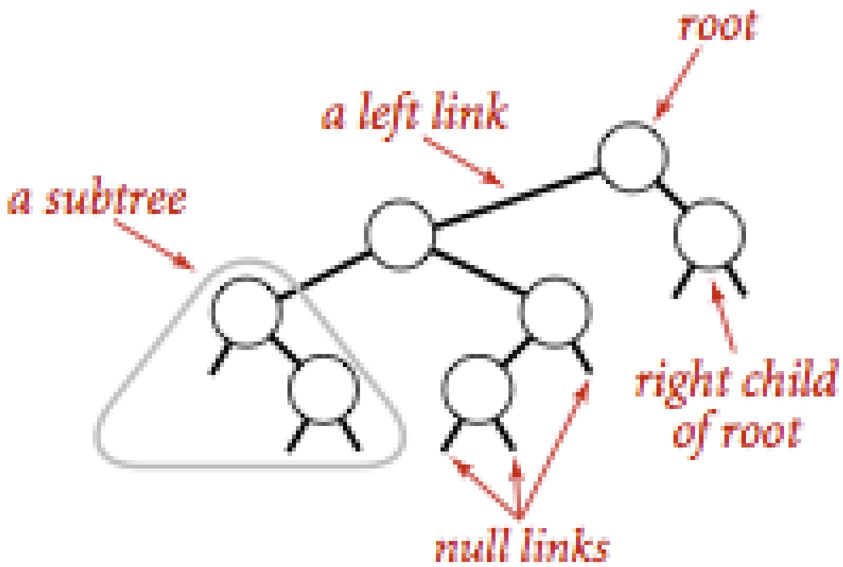
repeat
    Pick n_{best} from O such that $f(n_{best}) \leq f(n), \forall n \in O$.
    Remove n_{best} from O and add to C.
    If $n_{best} = q_{goal}$, EXIT.
    Expand n_{best}: for all $x \in \mathrm{Star}(n_{best})$ that are not in C.
    if $x \notin O$ **then**
        add x to O.
    else if $g(n_{best}) + c(n_{best}, x) < g(x)$ **then**
        update x's backpointer to point to n_{best}
    end if
until O is empty



- Finding **The Shortest Path**
- **Problem of finding a path** between two vertices (or nodes) in a graph
- **Cost function:** the sum of edges' weights is minimized.

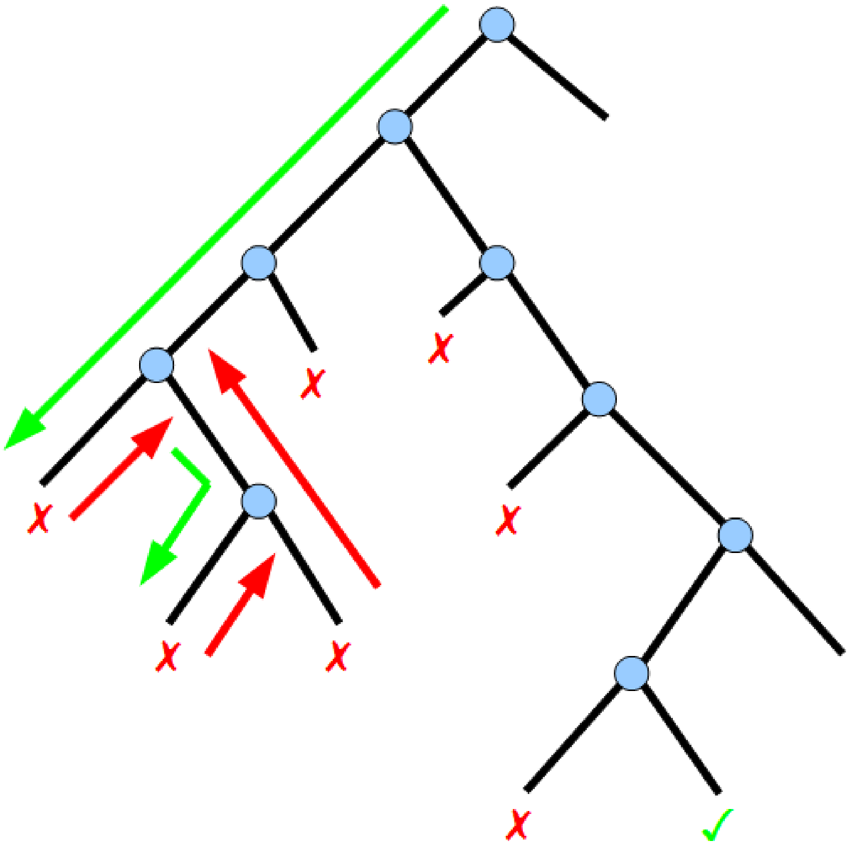
Part IV – Searching

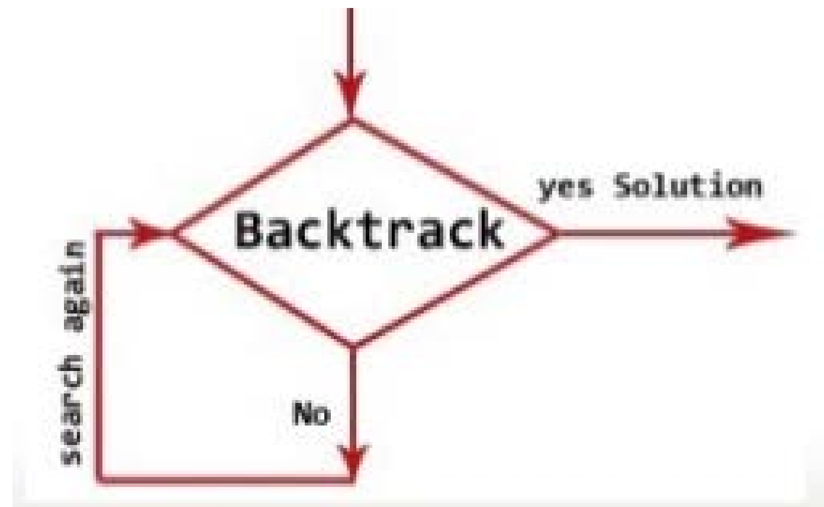
Binary Search Tree




```
BinarySearch(list[], min, max, key)
if max <min then
   return  false
else
   mid = (max+min) / 2
   if list[mid] >key then
      return  BinarySearch(list[], min, mid-1, key)
   else if list[mid] <key then
      return  BinarySearch(list[], mid+1, max, key)
   else
      return  mid
   end if
end if
```

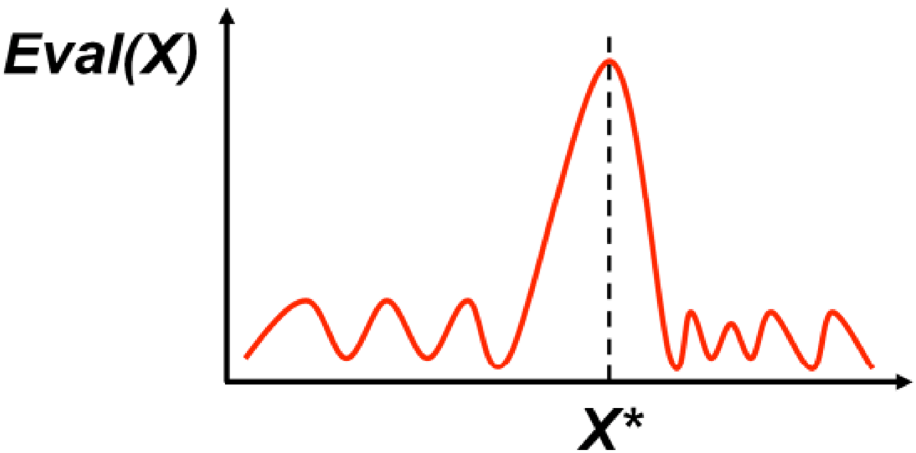
Backtracking Search Algorithm

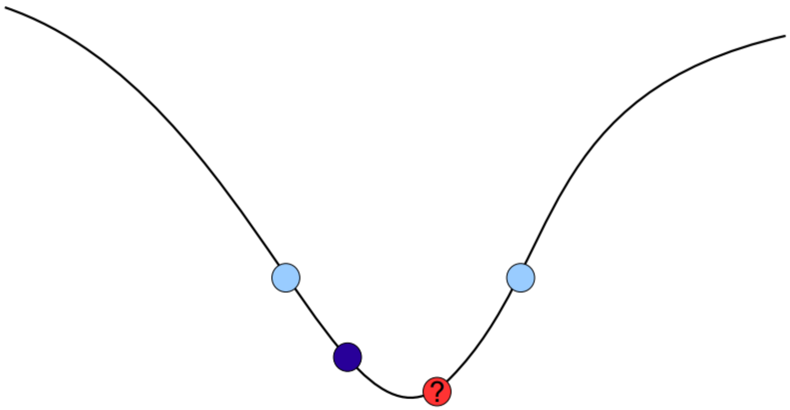




Backtracking search *is used for a* ***depth-first search*** *that chooses values for one variable at a time and* ***backtracks*** *when a variable has no legal values left to assign.*

Line Search Methods

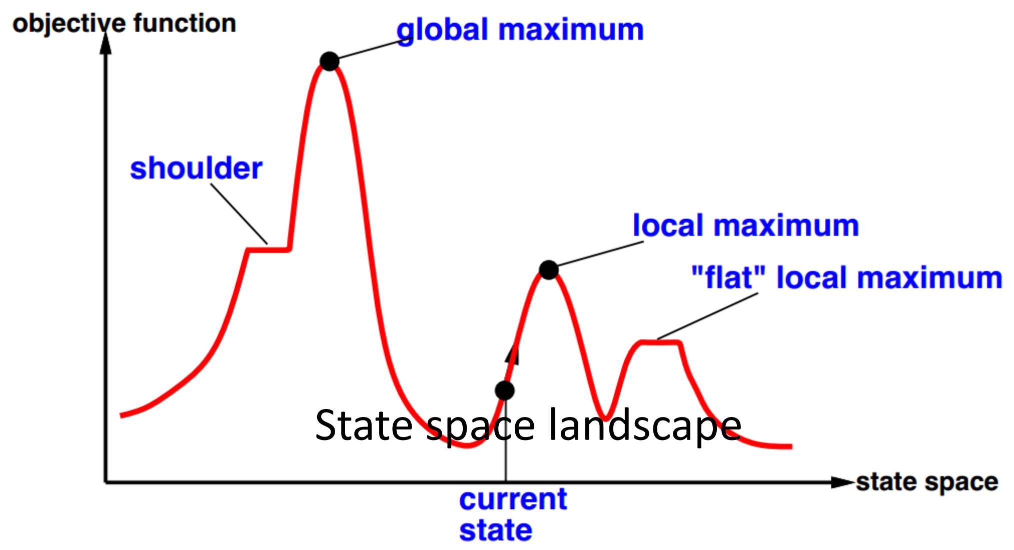


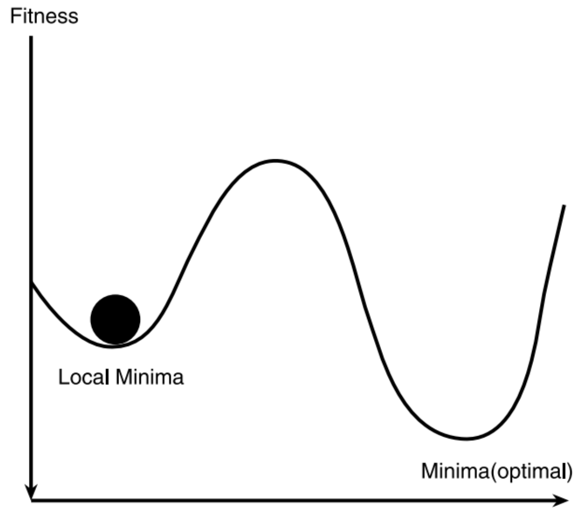


$$\min_{\alpha>0} f(x_k + \alpha p_k).$$

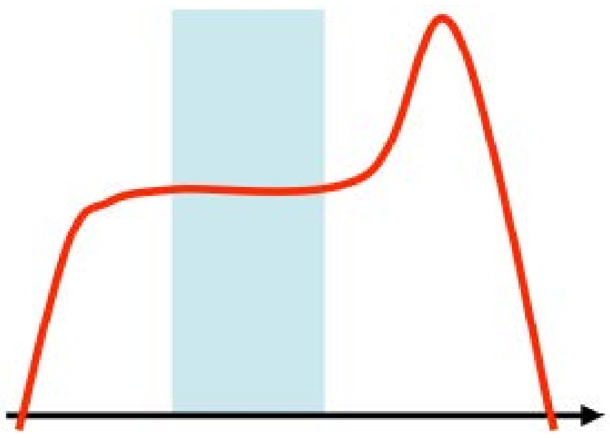
Line search method *seeks the minimum of a defined nonlinear function by selecting a* ***reasonable direction vector***

Local vs Global Maxima/Minima





Can get stuck in a local maximum



Can get stuck on a plateau

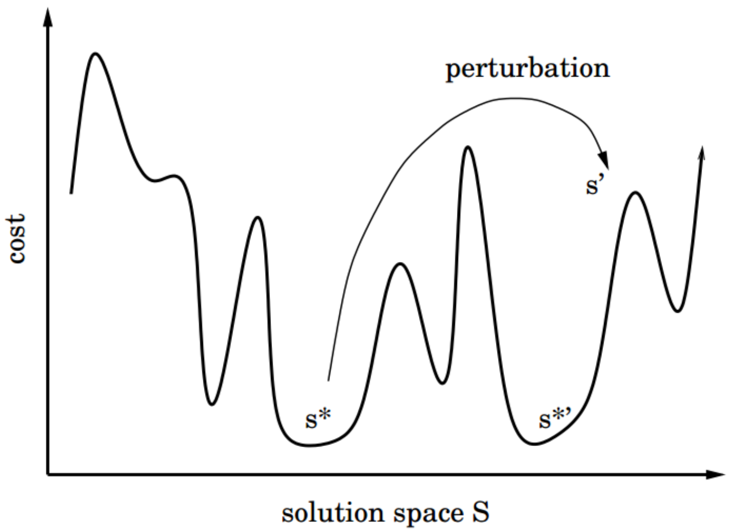
Greedy Search

```
Algorithm Greedy (a,n)
//a[1:n]contains the n inputs.
{
    solution:=0;//initialize the solution.
    for i:=1 to n do
    {
        x:=Select(a);
        if Feasible( solution, x) then
                solution:=Union(solution,x);
    }
    return solution;
}
```

A ***greedy* search** follows a **heuristic** by making **the locally optimal** choice at each stage with the hope of **finding a global optimum**

Iterated Local Search

procedure *Iterated Local Search*
$s_0 \leftarrow$ GenerateInitialSolution
$s^* \leftarrow$ LocalSearch(s_0)
repeat
$s' \leftarrow$ Perturbation$(s^*, history)$
$s^{*\prime} \leftarrow$ LocalSearch(s')
$s^* \leftarrow$ AcceptanceCriterion$(s^*, s^{*\prime}, history)$
until termination condition met
end

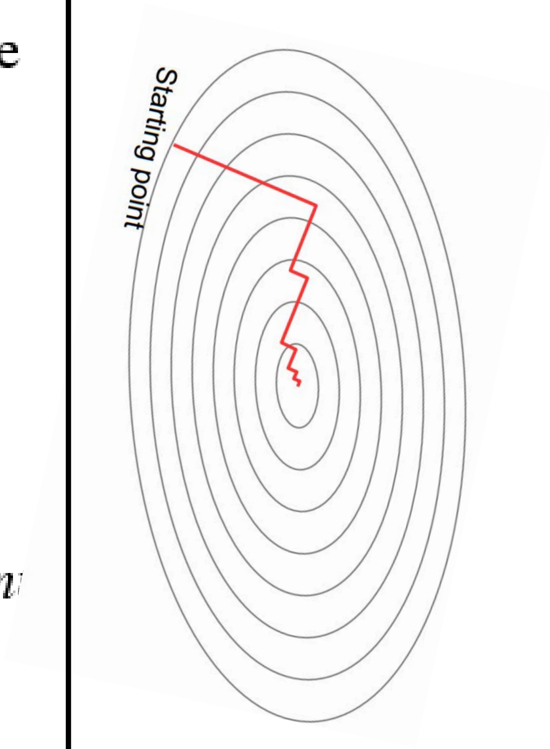


- **Iterated Local Search** = a sequence of **locally optimal** solutions
- The **modified solution** = **Perturbing** the current local minimum
- The perturbation strength has to be sufficient to **lead the trajectory**

Hill Climbing

```
function HILL-CLIMBING(problem) returns a solution state
   inputs: problem, a problem
   static: current, a node
           next, a node

   current ← MAKE-NODE(INITIAL-STATE[problem])
   loop do
       next ← a highest-valued successor of current
       if VALUE[next] < VALUE[current] then return current
       current ← next
   end
```



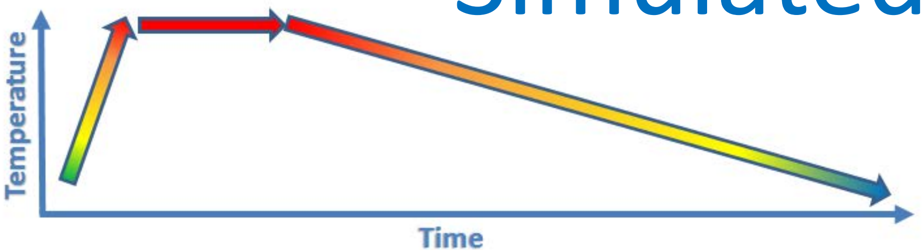
- **Hill climbing** is a **local search technique.**
- Hill Climbing **gets stuck in local minima**
- When **local minima exist**, Hill Climbing is **suboptimal.**

Stochastic Hill Climbing

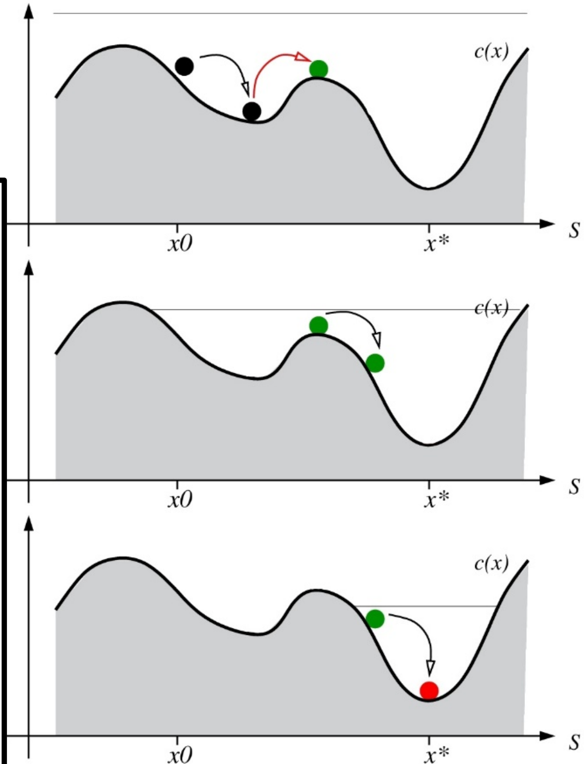
```
Input: Iter_max, ProblemSize
Output: Current
Current ⟵ RandomSolution(ProblemSize)
For (iter_i ∈ Iter_max)
   Candidate ⟵ RandomNeighbor(Current)
   If (Cost(Candidate) ≥ Cost(Current))
      Current ⟵ Candidate
   End
End
Return (Current)
```

- **Random selection overcomes local minima**
- The **selection probability** can vary with the steepness.
- Stochastic Hill Climbing is a **Local Optimization** algorithm
- Does **not require derivatives** of the search space.

Simulated Annealing




```
S0 = GenerateSolution()
Temp = START_TEMP
K = 0 // iteration count
while(!stopping_condition(S0, Temp, K))
   S1 = Neighbor(S0) // make a neighbor of S0
   if(Fitness(S1) < Fitness(S0))
       S0 = S1 // if S1 is better than S0 take it
   else if(rand() < tempFunc(S0, S1, Temp, K))
       S0 = S1 // if S1 is worse than S0
               // conditionally take it
   end if
   AnnealingSchedule(S0, Temp, K) // anneal the temp
   K = K + 1 // increase the iteration count
end while
```

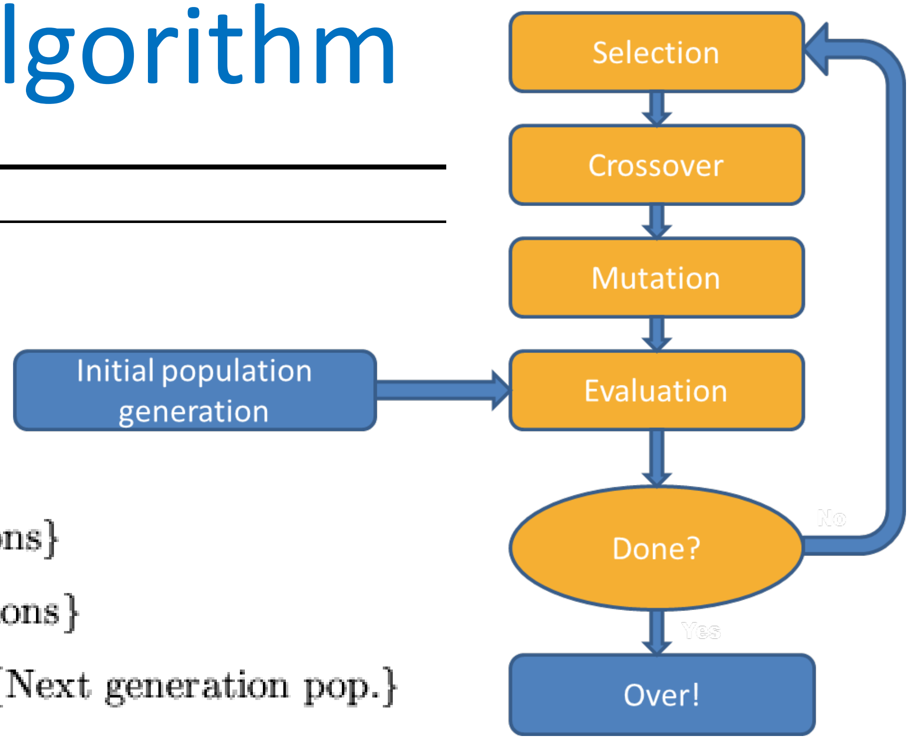


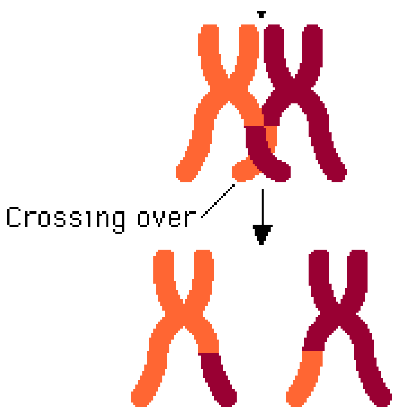
- **Simulated Annealing** = physics inspired twist on random walk
- Escape **local maxima** by **allowing some bad moves**
- Instead of picking the best move, **pick one randomly**

Genetic Algorithm

Algorithm 1 Pseudocode for a Genetic Algorithm

$t \leftarrow 0$;
InitPopulation$[P(t)]$; {Initializes the population}
EvalPopulation$[P(t)]$; {Evaluates the population}
while not termination **do**
    $P'(t) \leftarrow$ Variation$[P(t)]$; {Creation of new solutions}
    EvalPopulation$[P'(t)]$; {Evaluates the new solutions}
    $P(t+1) \leftarrow$ ApplyGeneticOperators$[P'(t) \cup Q]$; {Next generation pop.}
    $t \leftarrow t+1$;
end while

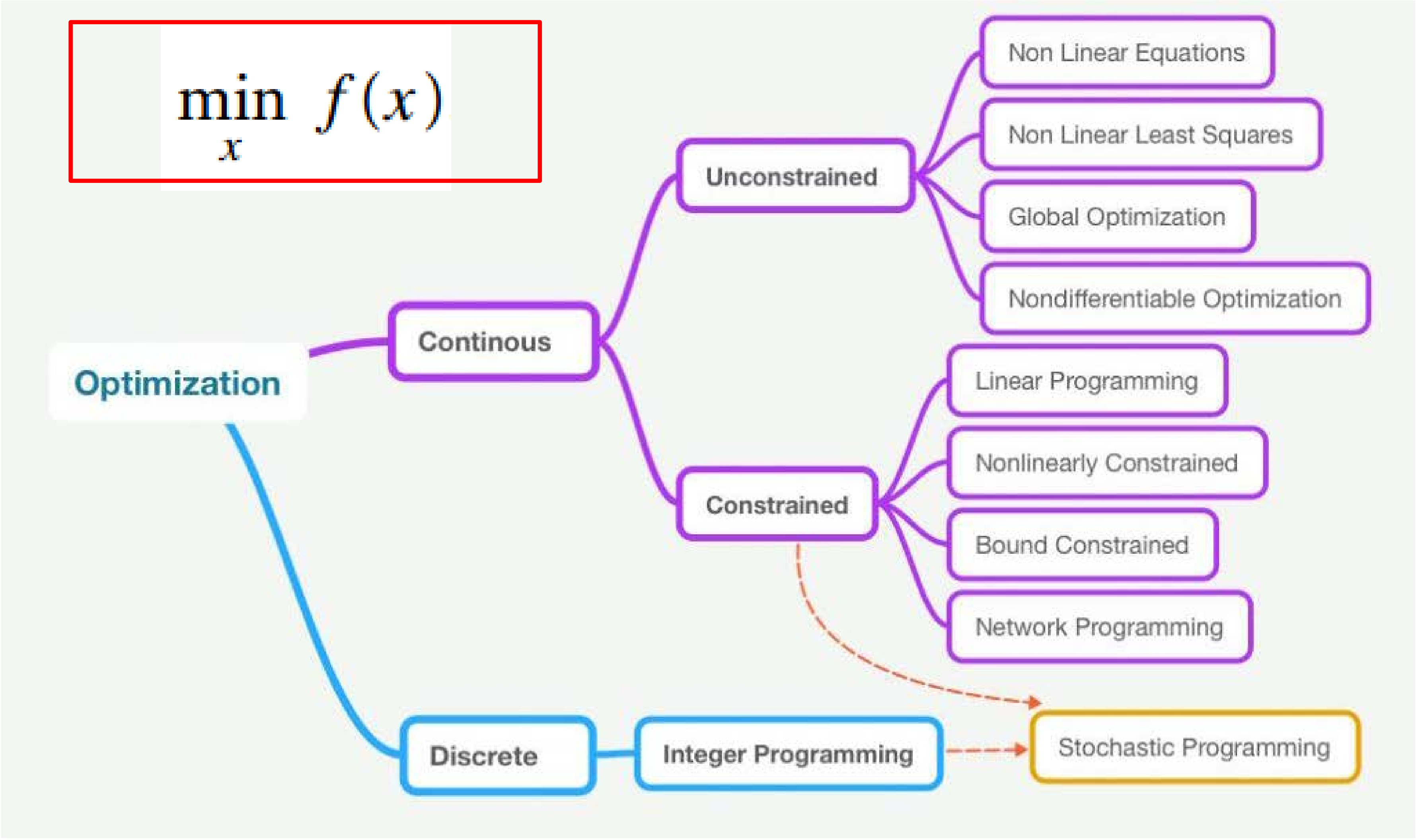




- **Twist** on **Local Search**
- **Successor** is generated by **combining parent states**
- A **state** is represented as a **string over an alphabet**
- **Randomly generated states** (population)

Part V – Optimizing

Numerical Optimization



Optimization Approaches

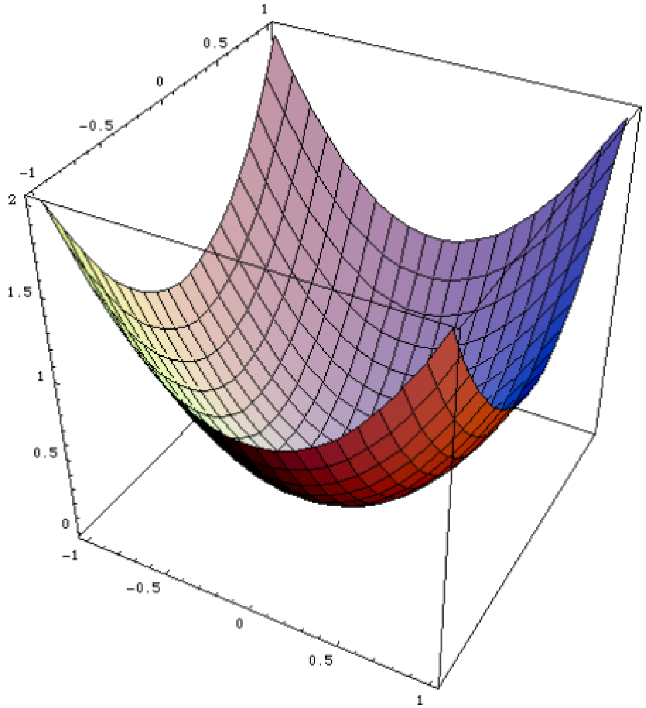
- **Derivative free optimization** refers to problems for which derivative information is unavailable or impractical to obtain.

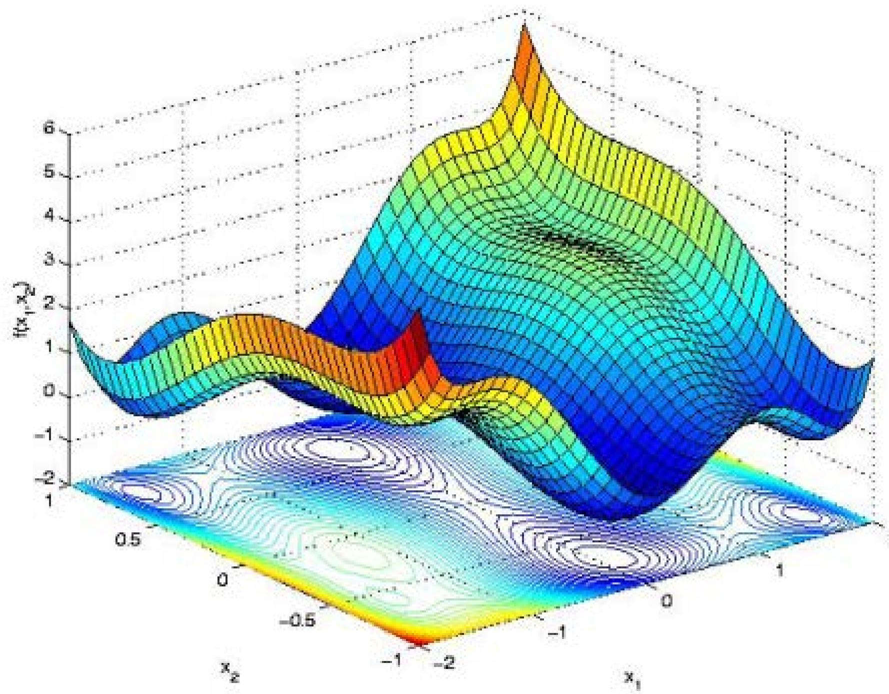
- **Gradient-based optimization** : method with the search directions defined by the *gradient* of the function at the current point.

- **Stochastic optimization** generate and use random variables or which involve random objective functions or random constraints.

- **Constrained optimization** optimize an objective function with respect to some variables in the presence of constraints.

Numerical Optimization

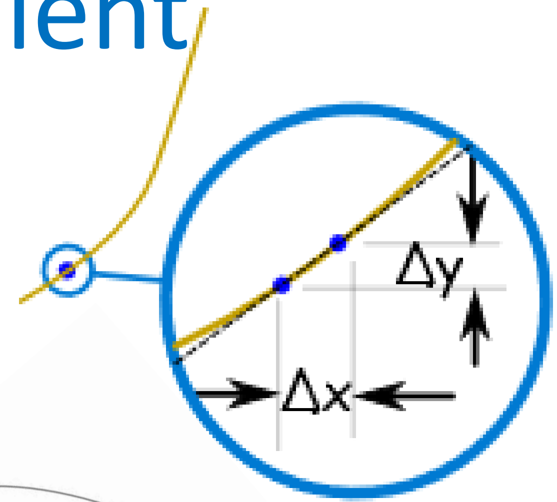




- **Convex** : any local minimum is a global minimum
- **Non-Convex** : bad local minima, stay stuck in a local minima that is not a global minima.

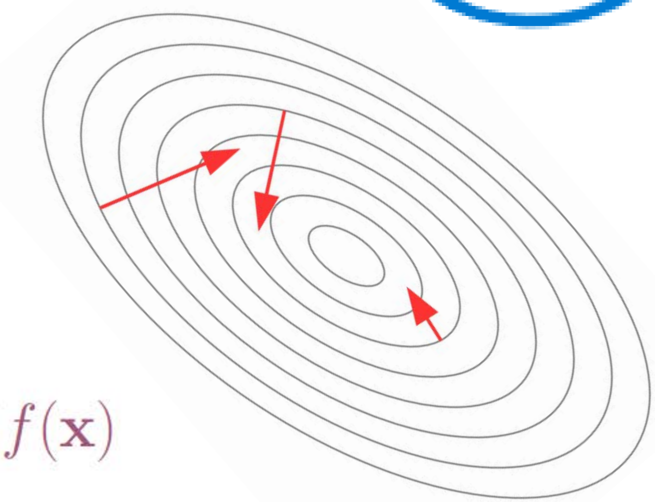
Derivatives and Gradient

- Central **Finite Differencing**



$$\frac{\partial f}{\partial x_i}(x) \approx \frac{f(x+\epsilon e_i) - f(x-\epsilon e_i)}{2\epsilon}.$$

- **Gradient** methods:

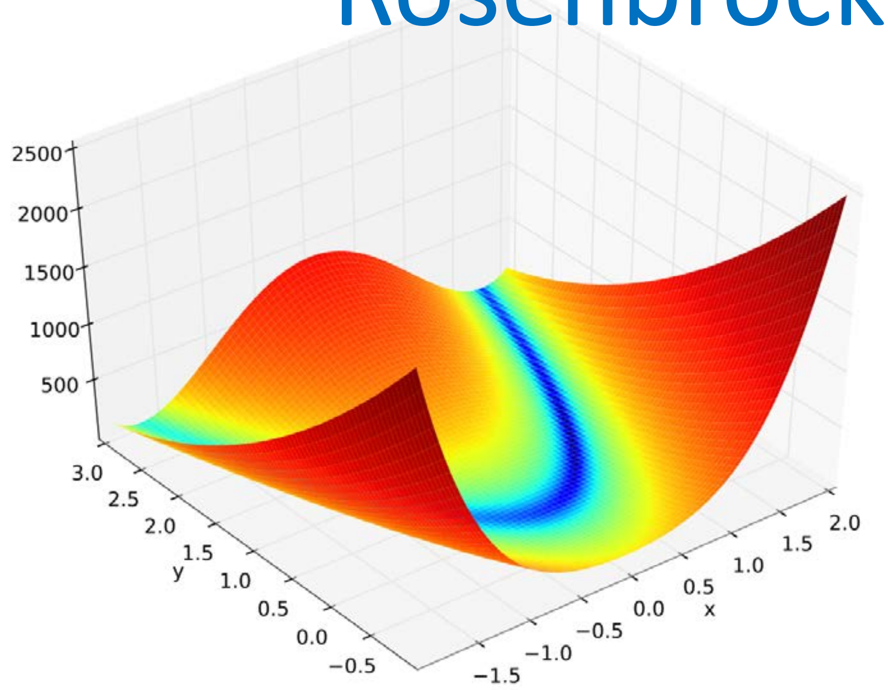


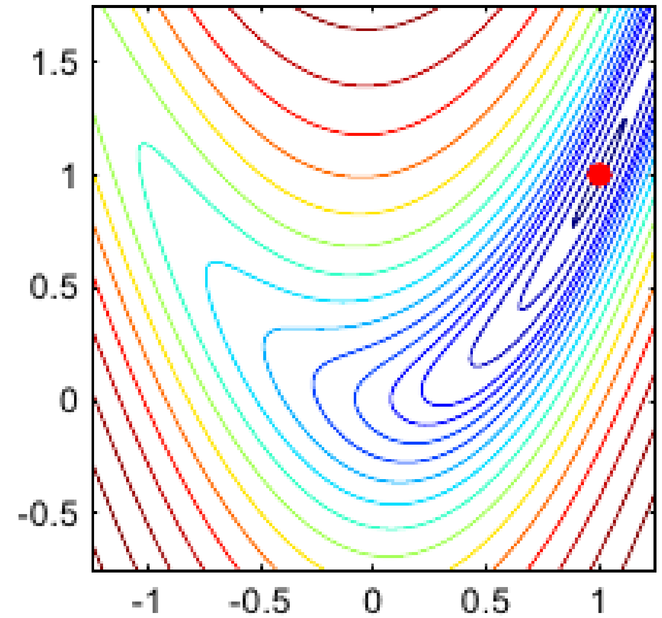
$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial y_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial y_2}, \frac{\partial f}{\partial x_3}, \frac{\partial f}{\partial y_3}\right)$$

to increase/reduce f, e.g., by $\mathbf{x} \leftarrow \mathbf{x} + \alpha \nabla f(\mathbf{x})$

- The gradient gives the steepest descent direction.
- Derivatives may not exist **or too costly to compute.**

Rosenbrock Function

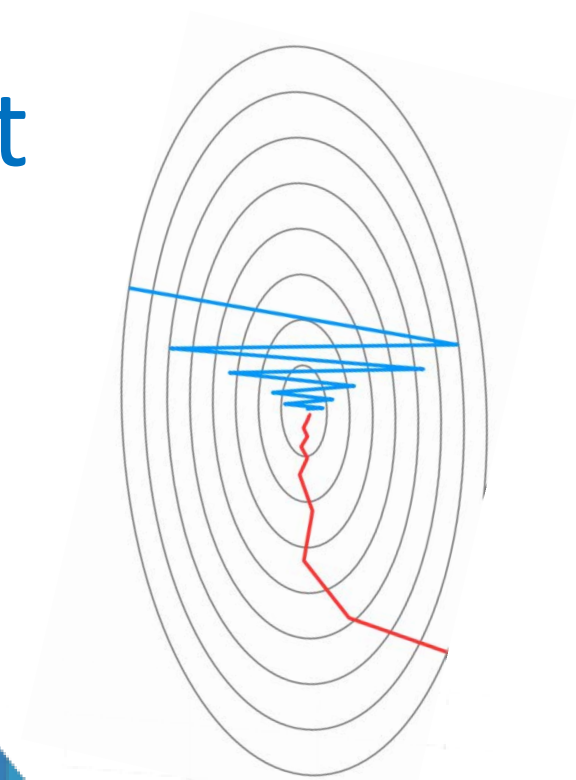




$$f(x,y) = (1-x)^2 + 100(y-x^2)^2$$

*The **Rosenbrock function** is a **non-convex function**, introduced by Howard H. Rosenbrock in 1960, which is used as a performance test problem for **optimization algorithms**.*

Gradient Descent



Performing a line search along the direction of the gradient

Algorithm: Gradient Descent

Input: $Y, \Theta, \mathbf{X}, \alpha$, tolerance, max iterations

Output: $\mathbf{\Theta}$

for $i = 0; i < max\ iterations; i++$ **do**

    current cost $= Cost(Y, \mathbf{X}, \mathbf{\Theta})$

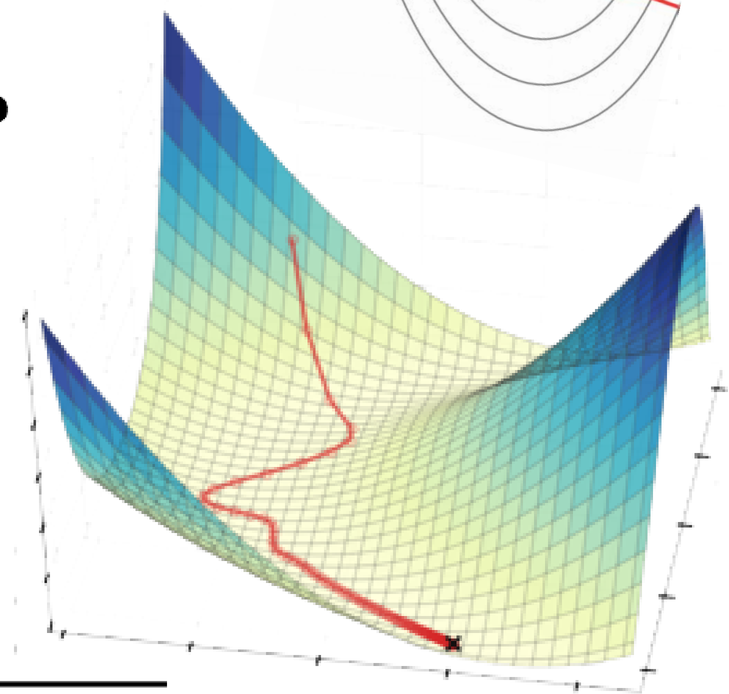
    if *current cost* $<$ *tolerance* **then**

        break

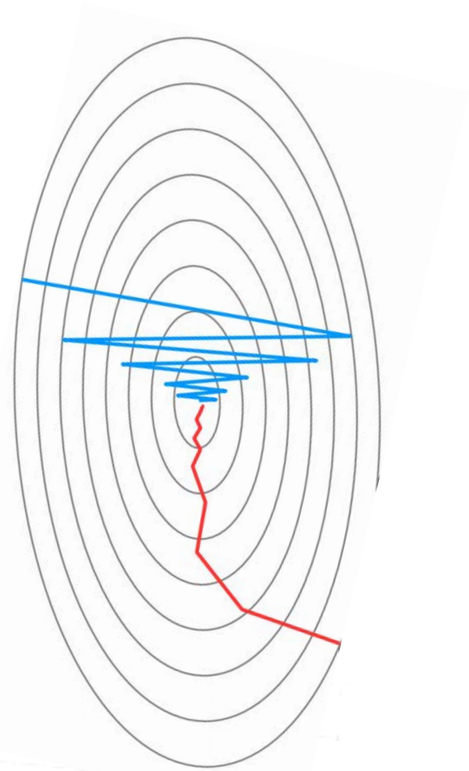
    else

        gradient $= Gradient(Y, \mathbf{X}, \mathbf{\Theta})$

        $\theta_j \leftarrow \theta_j - \alpha \cdot$ gradient

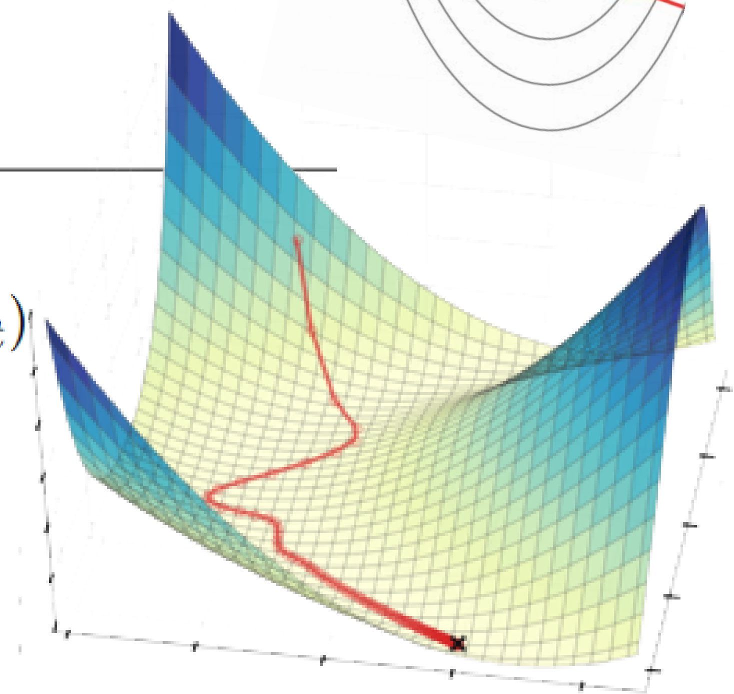


Gradient Descent



Algorithm 3.2 Gradient Descent

Input: Initial point w_0, gradient norm tolerance ϵ
Set $t = 0$
while $\|\nabla J(w_t)\| \geq \epsilon$ **do**
  $w_{t+1} = w_t - \eta_t \nabla J(w_t)$
  $t = t + 1$
end while
Return: w_t



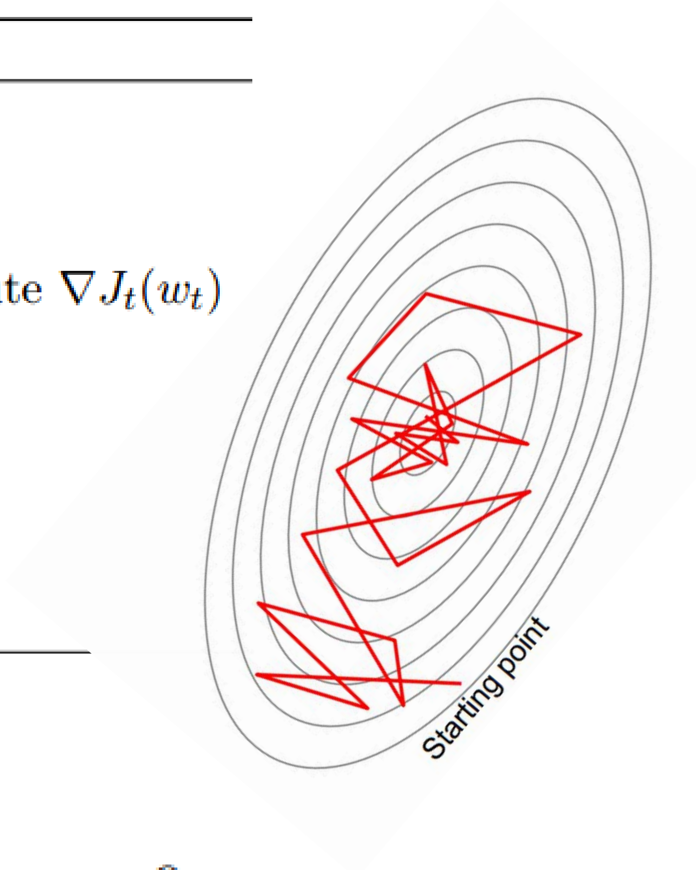
- **Gradient** of the input function: $\nabla J(w_t)$
- Decaying **Stepsize**: $\eta_t = 1/\sqrt{t}$.
- Gradient Descent **Update**:

$$w_{t+1} = w_t - \eta_t \nabla J(w_t)$$

Stochastic Gradient Descent

Algorithm 3.8 Stochastic Gradient Descent

Input: Maximum iterations T, batch size k, and τ
Set $t = 0$ and $w_0 = 0$
while $t < T$ **do**
    Choose a subset of k data points (x_i^t, y_i^t) and compute $\nabla J_t(w_t)$
    Compute stepsize $\eta_t = \sqrt{\frac{\tau}{\tau+t}}$
    $w_{t+1} = w_t - \eta_t \nabla J_t(w_t)$
    $t = t + 1$
end while
Return: w_T



- **Very noisy estimation of the gradient**
- **Decaying Stepsize:**

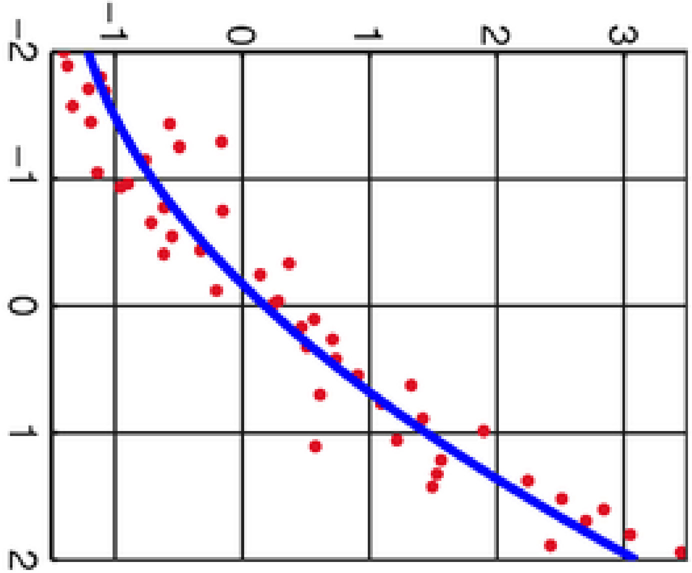
$$\eta_t = \frac{\tau}{\tau + t} \qquad \tau > 0$$

- **Gradient Descent Updates:** $$w_{t+1} = w_t - \eta_t \nabla J(w_t)$$

Linear Least Squares

$$\mathbf{Ax} = \mathbf{b}$$

$$\underbrace{\begin{bmatrix} 1 & 2 & -2 & -6 \\ -3 & -1 & -2 & \alpha \\ -4 & 3 & 9 & 16 \\ 5 & 7 & -6 & -15 \end{bmatrix}}_{\boldsymbol{A}} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}}_{\boldsymbol{x}} = \underbrace{\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}}_{\boldsymbol{b}}$$



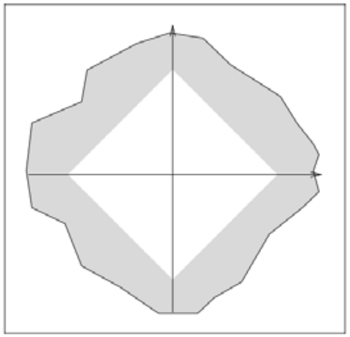
$$f(\mathbf{x}) = \frac{1}{2}\|\mathbf{Ax} - \mathbf{b}\|^2$$

$$\min_x \|Ax - b\|_2$$

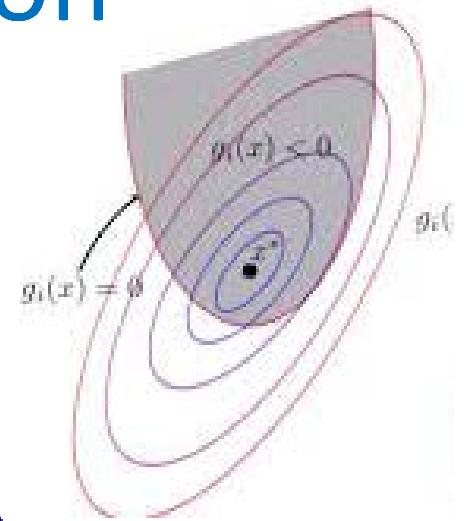
Data fitting ***estimated parameters*** *to optimize the fitting. The solution is obtained by solving the* ***normal equation****.*

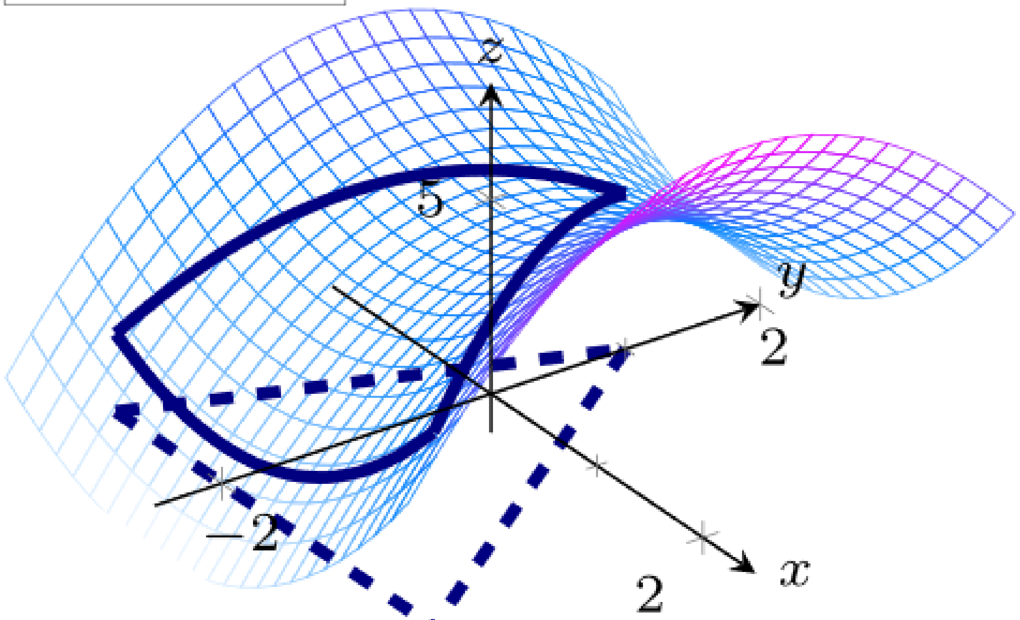
$$\boxed{\nabla f(x) = \mathbf{A}^T\mathbf{Ax} - \mathbf{A}^T\mathbf{b}}$$

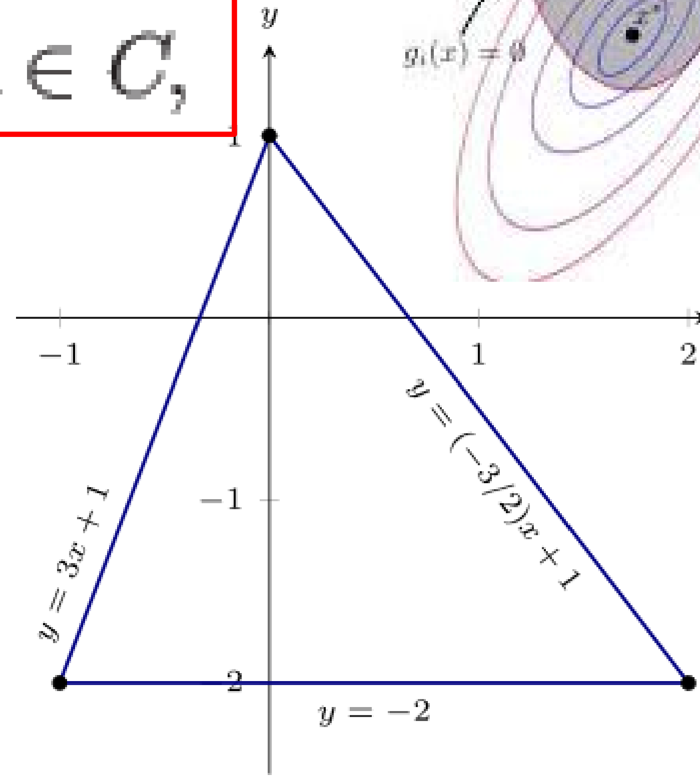
Constrained Optimization



$$\begin{array}{ll} \min_{\mathbf{x}} & f(\mathbf{x}) \\ \text{subject to} & \mathbf{x} \in C, \end{array}$$







Constrained optimization *is the process of optimizing an objective function in the presence of* ***constraints*** *on variables.*

Part VI – Conclusions

Game vs Search

- **Search:**
  - Solution is **heuristic** method for finding goal
  - Find **(near-)optimal** solution
  - **Evaluation** of a **cost function**

- **Game:**
  - Solution is **strategy.**
  - **Approximate** solution.
  - **Unpredictable** opponent.
  - Kind of **uncertainty.**

Limitation of Game Theory

- Some assumptions are not practical.
- Increasing complexity with the increase of the players.
- **Infinite number of strategy.**
- **Outcome: the gain of on person is the loss of another person.**
- **Knowledge about strategy: players as the knowledge of strategies.**
- **Assumption of maximin and minimax**
- Assumption that players are equally wise and behave rationally.
- Risk and Certainty of Pay Offs / **Rules of game**

Lab Activities

- Activity 1: Payoff Matrix and Equilibrium (30 min)
- Activity 2: Stochastic Hill Climbing (30 min)

- Break (no break)

- Activity 3: Backtracking and Search (30 min)
- Activity 4: Gradient Decent (30 min)

References

[1] - Problem Solving, search and Control strategy - myreaders.info
[2] - Heuristic Search - Andrea Torsello
[3] - Problem Solving and Search - Russell and Norvig
[4] - Artificial Intelligence - Nancy E. Reed
[5] - Artificial Intelligence - Tabu Search - Fred Glover
[6] - Numerical Optimization Techniques - Leon Bottou
[7] - Search Techniques for Artificial Intelligence - Ulle Endriss
[8] - Artificial Intelligence Programming in Prolog - Tim Smith
[9] - Recursive Backtracking - Marty Stepp and Helene Martin
[10] - Computational Game Theory - University of Oxford
[11] - Introduction to Artificial Intelligence State Space Search Gregory Adam
[12] - AI Techniques for Game Programming - Team LRN
[13] - Uninformed search methods - Doina Precup
[14] - Algorithmic Game theory - Noam Nisan
[15] - A tutorial on Tabu Search - Hertz et al
[16] - Local search algorithms - Peter Ljunglof
[17] - Problem Solving and Search in Artificial Intelligence Basic Concepts - Nysret Musliu
[18] - Artificial Intelligence - Game Theory
[19] - Solving Problems by Searching - Marco Chiarandini - Stuart Russell and Peter Norvig
[20] - Graduate AI Computational Game Theory - Michael Benisch

References

[21] - Artifical Intelligence For Games - Ian Millington and John Funge

[22] - Introduction to AI Techniques: Game Search, Minimax, and Alpha Beta Pruning

[23] - Game Theory - University of Notre Dame

[24] - Introduction to informed search The A* search algorithm - Penn University

[25] - Least Squares with Examples in Signal Processing Ivan Selesnick

[26] - Linear Least Squares Optimization - Kurt Bryan

[27] - Nonlinear Optimization Least Squares Problems - Nicolas Borlin

[28] - Adversarial Search and GamePlaying - Oliver Schulte

[29] - Numerical Optimization - Nocedal and Wright

[30] - Introduction to non-linear optimization - Ross A. Lippert

[31] - State Space Representation and Search - Prof. Hugh Murrell

[32] - Fundamentals of Artificial Intelligence - Alan Smaill

[33] - Iterated Local Search Variable Neighborhood Search - Thomas Stutzle

[34] - Backtracking Search Algorithms - Peter van Beek

[35] - Tabu Search - Fred Glover and Rafael Marti

[36] - Based on slides of Padhraic Smyth, Stuart Russell, Rao Kambhampati, Raj Rao, Dan Weld…