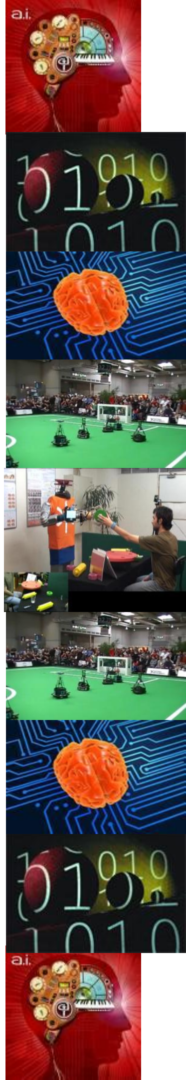
Search and Metaheuristics

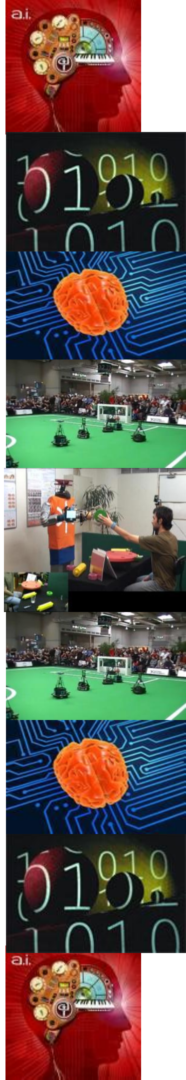
Introduction to Metaheuristics

- When the search space is too large and we cannot apply exact methods we use Metaheuristics.
- Exact methods will take too long to return a solution.



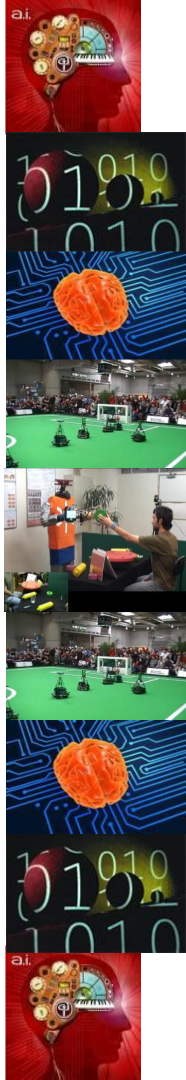
Introduction to Metaheuristics

- A good enough solution within a reasonable amount of time would be acceptable.
- A (near) optimal solution.
- Metaheuristics provide such solutions.



Introduction to Metaheuristics

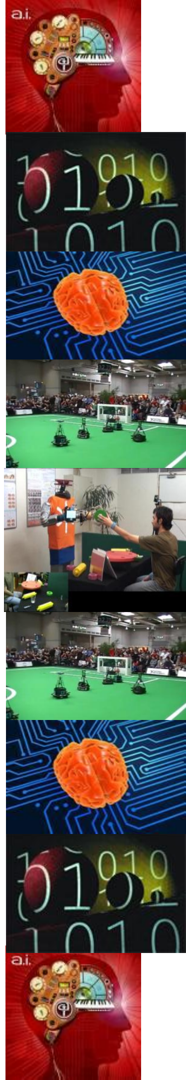
- Heuristic - to find (greek)
- Meta - in an upper layer (eg meta-data = data about data)
- Metaheuristics = Heuristics of heuristics.



Optimization Problems

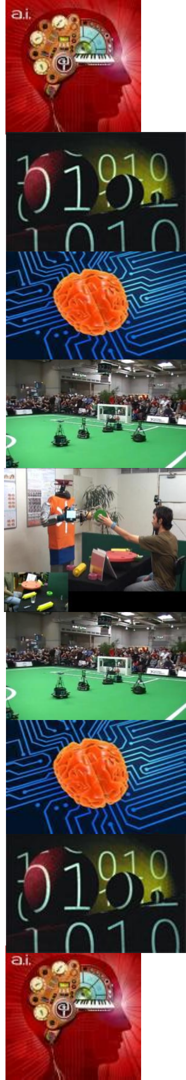
<u>Two Types</u>

- Continuous - if the variables are continuous.
- Discrete - if the variables are discrete.



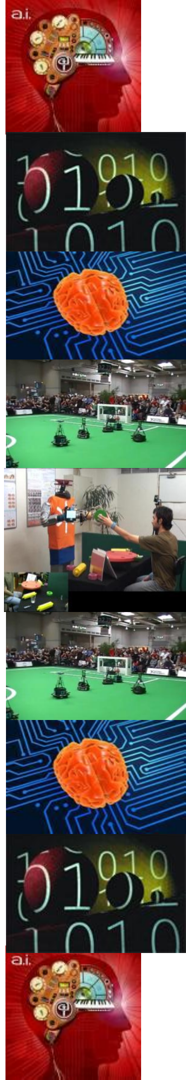
Solving Optimization Problems

- Branch and Bound - exact solutions
- Approximation - bounded
- Heuristic methods - deterministic
- Metaheuristics - deterministic +randomization



Introduction to Metaheuristics

- Combinatorial Optimisation Problems (COP) are discrete.
- Most COPs are NP-Hard
- Most CO problems involve searching for a combination of variables to solve a problem.

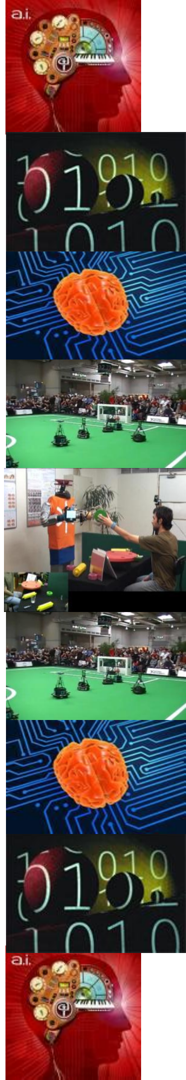


Combinatorial Optimization

A combinatorial optimization problem (COP) may be defined as

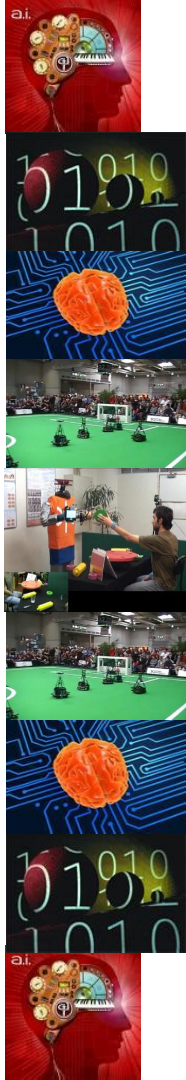
$$P = (S, f)$$

S is the search (solution) space



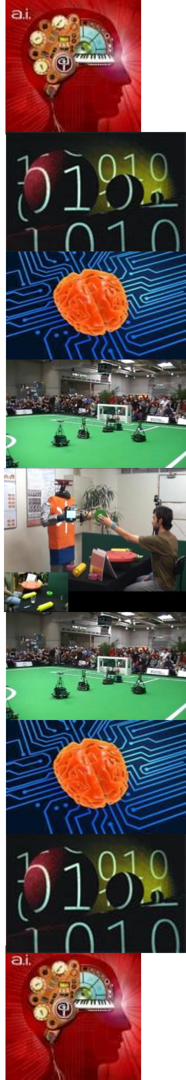
Formal Definition Metaheuristics

"A metaheuristic is formally defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for **exploring** and **exploiting** the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions."[Osman, I. H., & Laporte, G. (1996)]



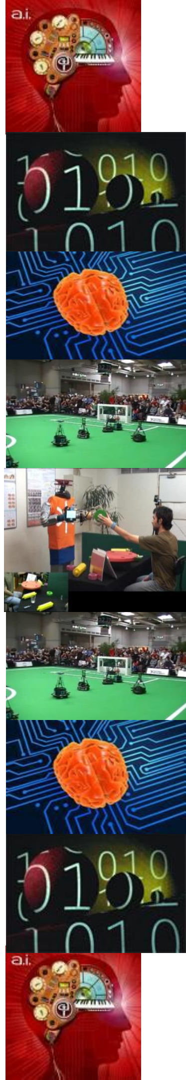
Characteristics of Metaheuristics

- Metaheuristics guide the search.
- Search for an (near) optimal solution
- Are not problem specific
- Are stochastic.



Characteristics of Metaheuristics

- Provide no guarantee of global or local optimality.
- May take a relatively long time so a stopping criteria needs to be defined.

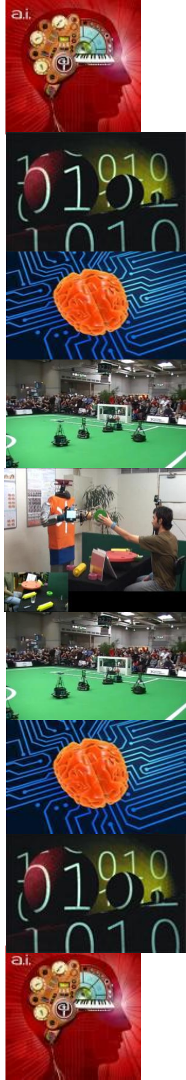


Classification of Metaheuristics

<u>Single point search vs Multipoint(population)</u>

- **Single point search** - work on a single solution iteratively improving it.

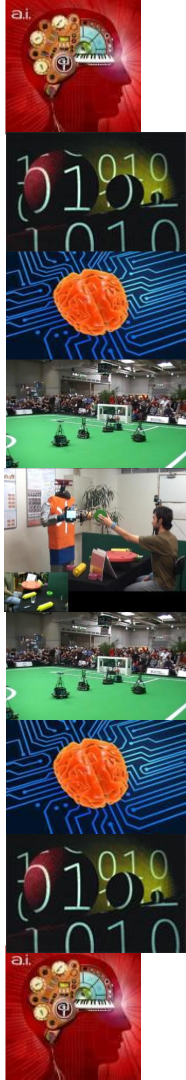
- **Population based** - work on a population of individuals each representing a solution



Single Point Searches

Examples

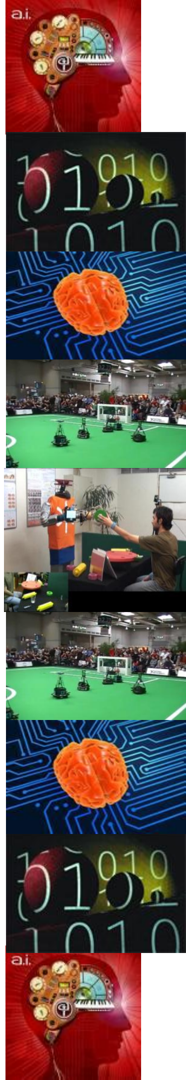
1. Tabu Search (TS).
2. Hill-climbing (HC).
3. Simulated annealing (SA).
4. Iterated Local Search (ILS).
5. Variable Neighborhood Search (VNS).



Population (Multipoint) Search

Examples

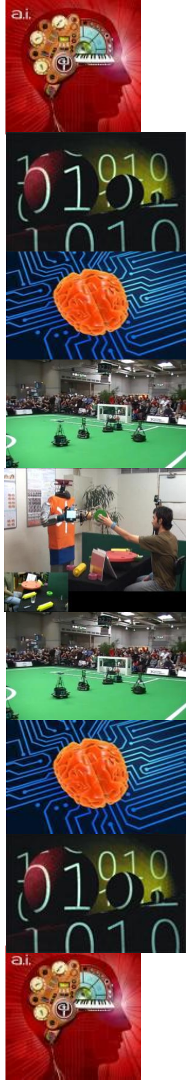
1. Genetic Algorithm(GA).
2. Genetic Programming(GP).
3. Grammatical Evolution (GE).
4. Particle Swarm Optimization(PSO).
5. Ant Colony Optimization.



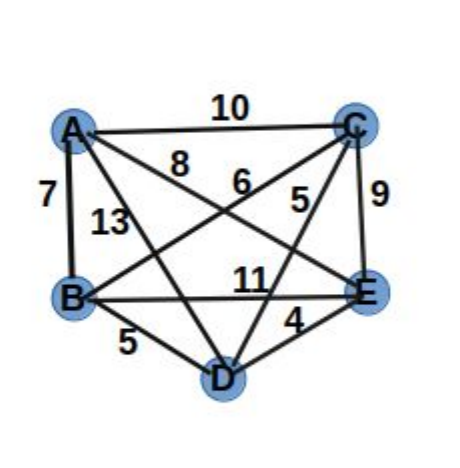
Encoding and Evaluation

Encoding i.e representation of a problem.

- Problem dependent.
- Must be able to evolve feasible solutions.



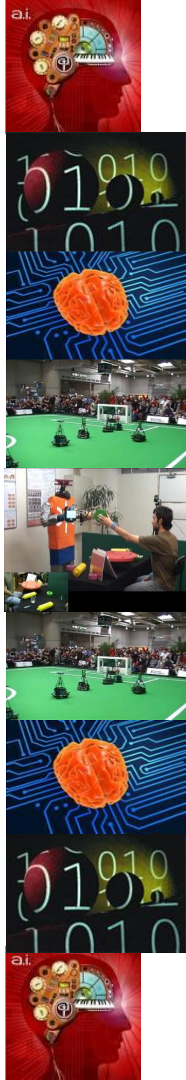
Encoding (Representation)



Assuming the D is the initial position in this TSP

1 = {D,B,A,C,E,D}
2 = {DB,BA,AC,CE,ED}
3 = {13,7,6,9,4} *

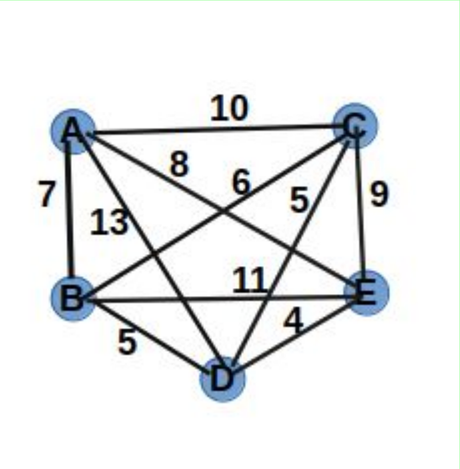
*Flawed representation

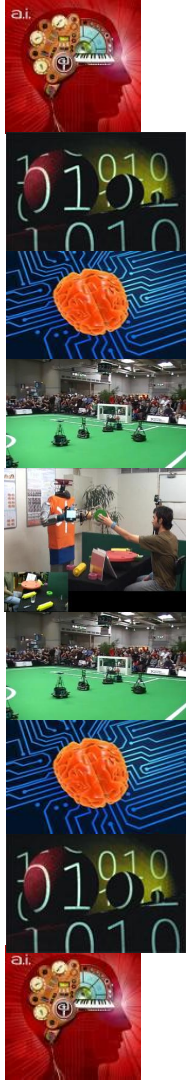


Evaluation

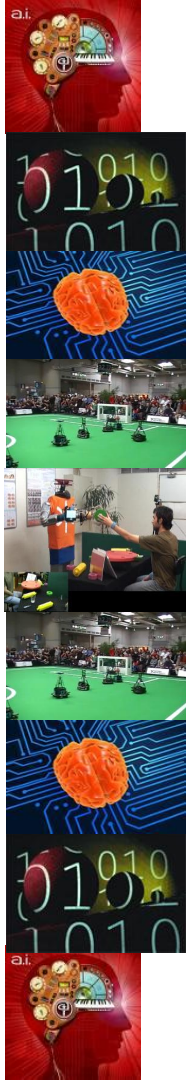
Objective(cost) function must be able to evaluate a solution with no ambiguity.

1 = {D,B,A,C,E,D} = 35
2 = {DB,BA,AC,CE,ED} = 35
3 = {13,7,6,9,4} * = 39



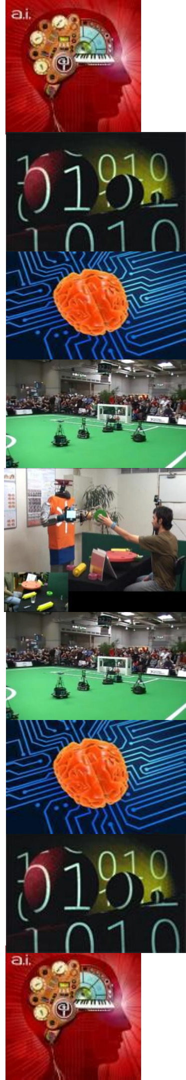


- Objective(cost) function and the fitness function may be the same or different depending on the problem(algorithm).
- TSP - obj function path , cost function path evaluation.



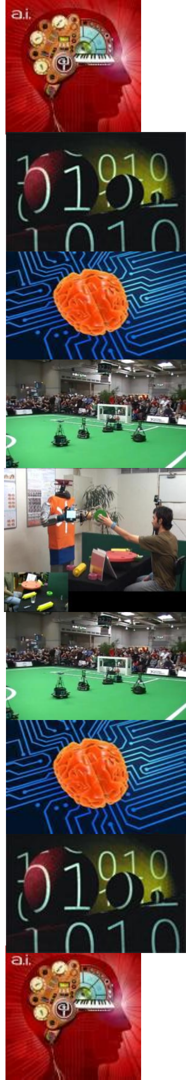
Simulated Annealing

- Imitates the annealing process in metal work.
- Slowly cool down a heated solid, so that all particles arrange in the ground energy state.
- At each temperature wait until the solid reaches its thermal equilibrium.



Simulated Annealing

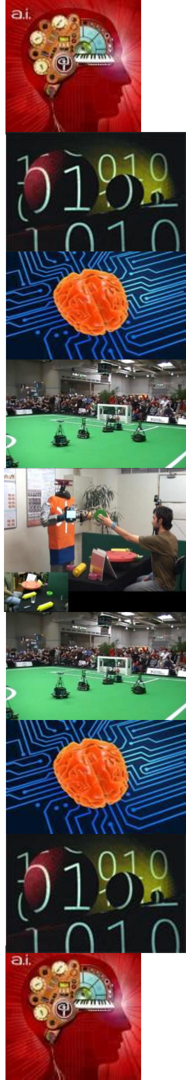
- The temperature T starts high and is lowered at each iteration.
- At each iteration a new candidate solution whose distance from the ideal is proportional to the temperature.
- The value of the temperature may influence acceptance of low values.



Simulated Annealing Algorithm

- **Algorithm 1** Simulated Annealing(Initial State s, Cost c)

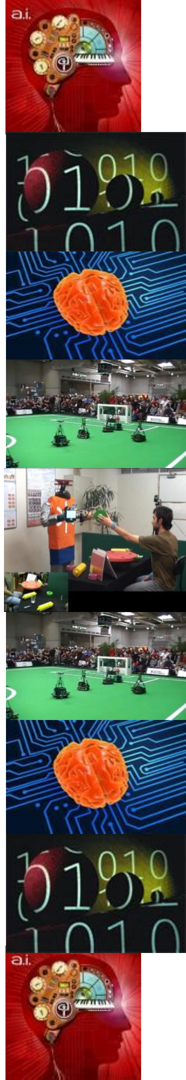
```
begin
CURRENT = {s}; BEST= {s}
Set initial value of T = T0 in the same magnitude as the typical differences
between adjacent losses(costs);
t = 1;
repeat
Set NEXT to randomly chosen adjacent state referred to as CURRENT;
Δcost = cost(NEXT) - cost(CURRENT)
Set CURRENT= NEXT with probability min {1,exp(-Δcost/T )};
if  cost(CURRENT) < cost(BEST)  then BEST=CURRENT;
    t ⇐ t + 1; T ⇐ T0 /log(t + 1);
    until no improvement in BEST for N iterations;
    return BEST;
    end
```



Simulated Annealing

<u>The following are needed</u>

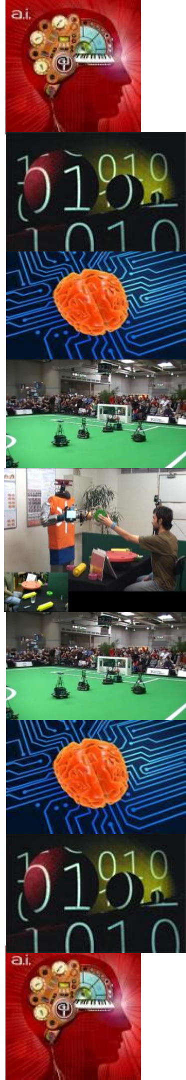
1. A method to generate an initial solution.
2. A generation function to find neighbours in order to select a NEXT candidate.
3. A cost function.
4. An evaluation criterion.
5. A stop criterion.



Simulated Annealing

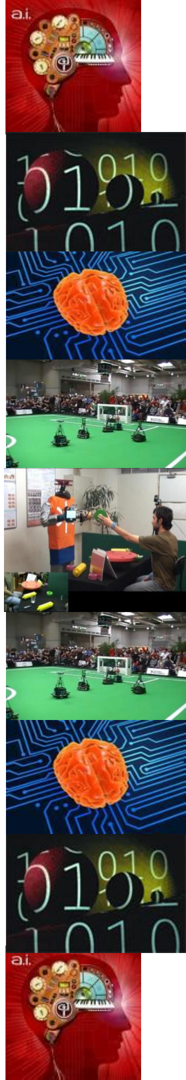
Example

If we were to use Simulated Annealing to search for the given binary string how would we go about it.



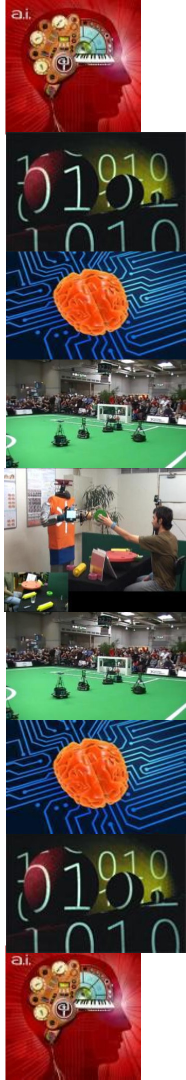
Simulated Annealing

- We need to address
  Representation (encoding).
  Evaluation.



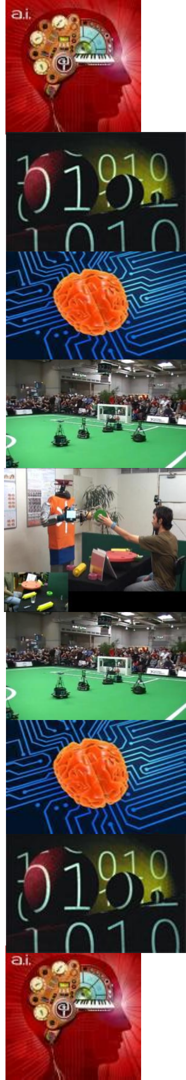
Simulated Annealing

- Encoding our solutions as binary strings 5 bit in length.
- Evaluation can be either the number of 1’s in place or the decimal values of the solution.



Simulated Annealing Pseudocode

```
begin
    t <- 0
    Randomly create a string  Vc (// this CURRENT)
    Repeat
      evaluate  Vc
      select   3 new strings  from the neighbourhood of Vc
       Let Vn be the best of the 3 strings
       if( f(Vc < f(Vn)  then  Vc <- Vn
       Else if (T > random())  then Vc <- Vn
       Update T according to the annealing schedule
       t <- t + 1;
     Until t <- N
  end
```

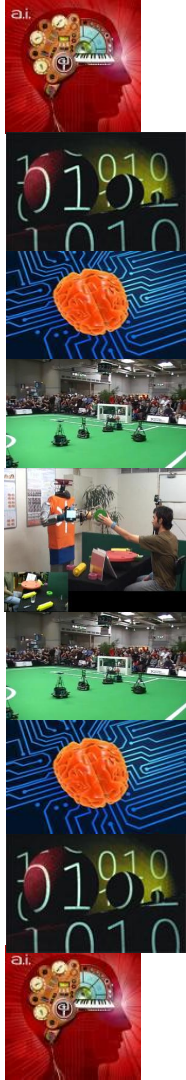


Simulated Annealing

Step 1

Randomly generate a current solution.

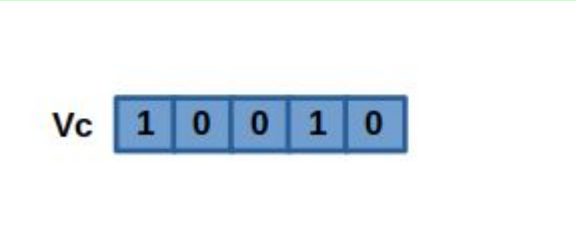
Vc	1	0	0	1	0

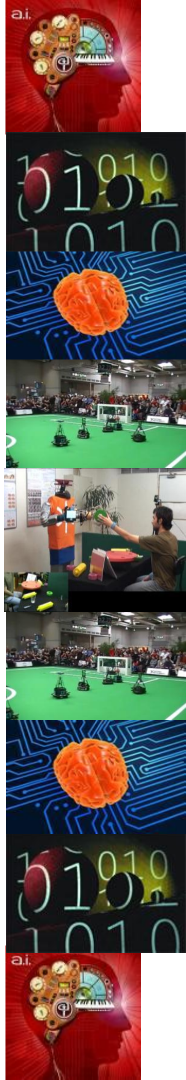


Simulated Annealing

Step 2

Evaluate Vc. cost of Vc = 18. (or 2 but ??)





Simulated Annealing

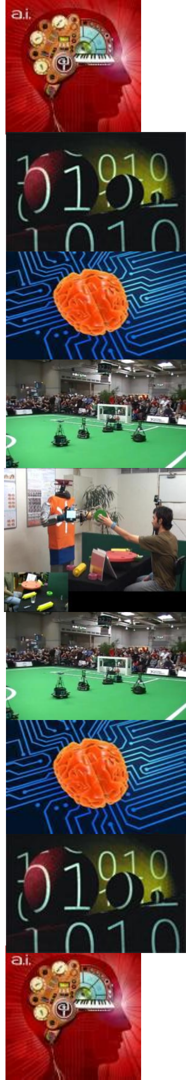
Step 3

Vc	1	0	0	1	0

Generate three solutions from the neighbourhood.

V_1 = 26 (or 3)
V_2 = 19 (or 3)
V_3 = 16 (or 1)

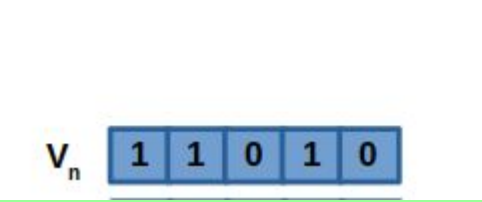
V_1	1	1	0	1	0
V_2	1	0	0	1	1
V_3	1	0	0	0	0



Simulated Annealing

Step 4

Let Vn be the best of the neighbours

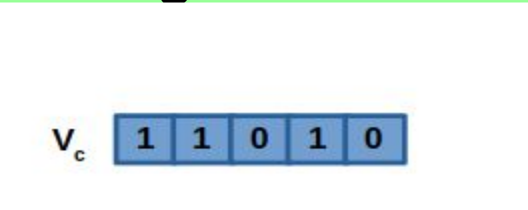


Simulated Annealing

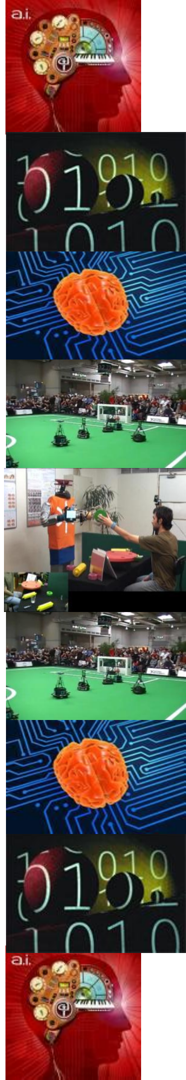
Step 5

If (f(Vc) < f(Vn)) then assign Vn to Vc

if (18 < 26) then



If f(Vc) >f(Vn) we will evaluate the Metropolis function given by
= $e^{((Vc-Vn)/Temperature)}$ **< random(0,1)** for acceptance.



SA Algorithm

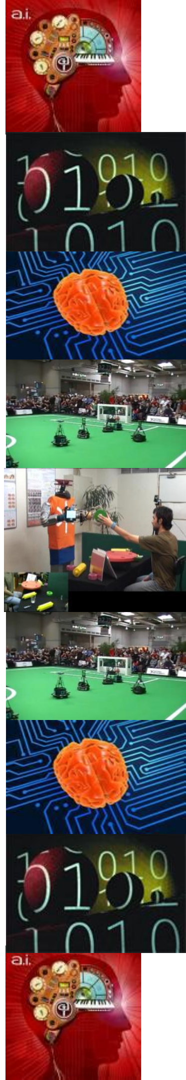
Step 6

Update T according to the annealing schedule.

The initial temp is set as hyperparameter at initialization.

Temperature = T_o/ (log t +1)

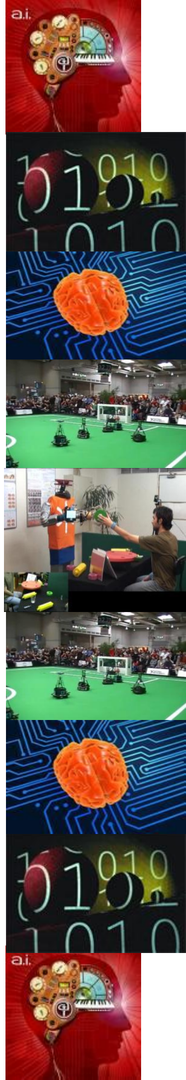
There are a number of way to reduce the temperature.



Simulated Annealing

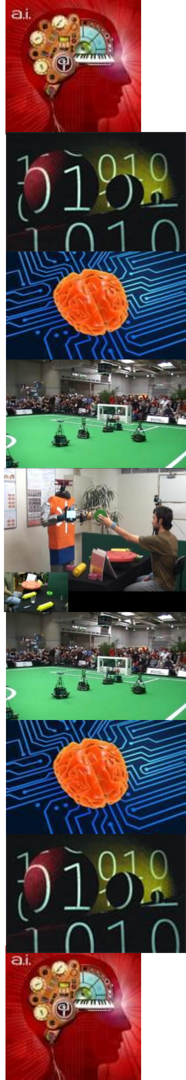
<u>**Repeat**</u>

Step 3 to Step 6



Simulated Annealing

- SA can be easily applied to a large number of problems.
- Tuning of the parameters is relatively easy.
- Generally the quality of the results of SA is good, although it can take a lot of time.
- Results are generally not reproducible another run can give a different result unless if the application is seeded.
- SA can leave an optimal solution and not find it keep track of the best solution.
- SA is capable of finding the best solution.



QUESTIONS ?????????

Next Lecture - Genetic Algorithms.