BEYOND CLASSICAL SEARCH: LOCAL SEARCH

CONTENTS

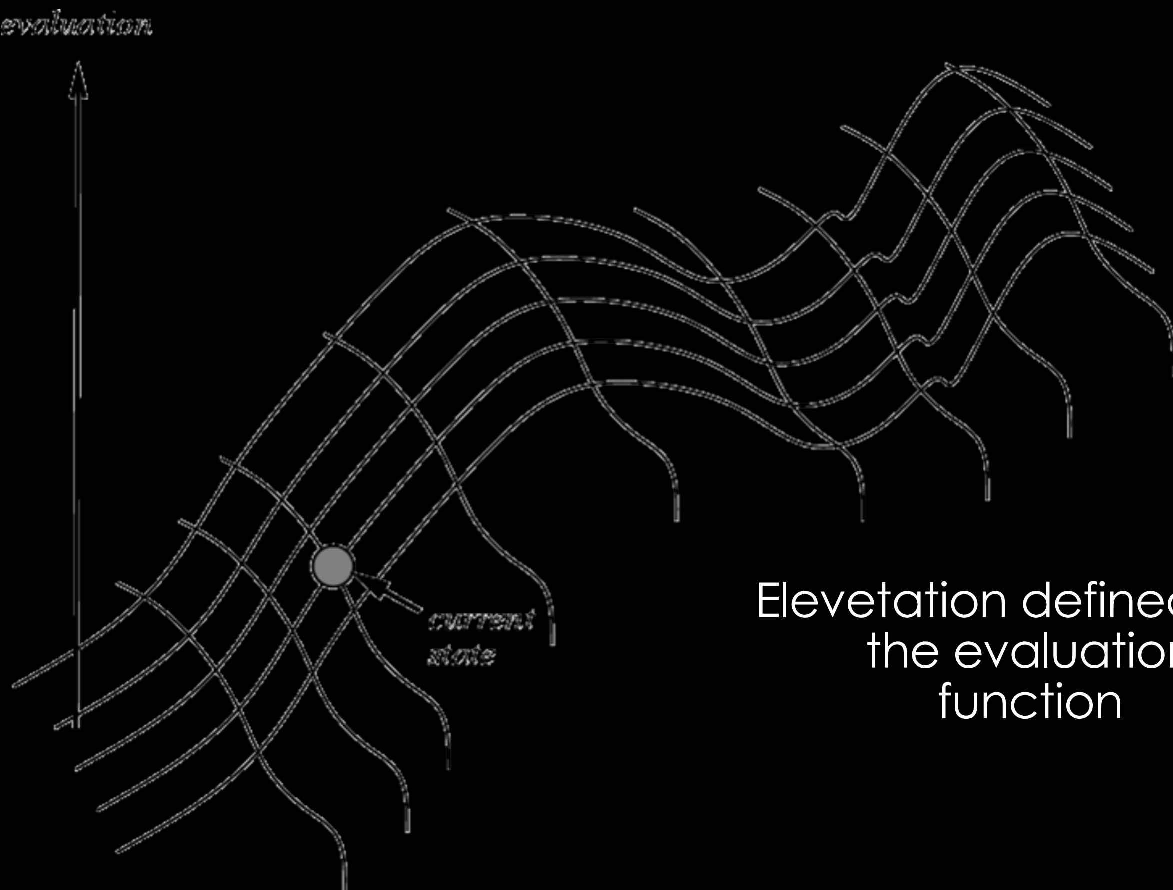
- Local search and optimization
- Local search algorithms:
  - Hill-climbing search
  - Local beam search
  - Simulated annealing search
  - Genetics algorithms

LOCAL SEARCH AND OPTIMIZATION

- Local search:
  - It only uses the current state and moves only to its succesors.
  - It does not remember the past path!
  - The path to the goal is irrelevant, to reach the goal is the objective.
- Idea:
  - It starts with an approximate initial idea for reaching the goal.
- Advantages:
  - It uses few memory, because it does not remember the path.
  - Frecuently it reaches reasonable solutions in very big or infinite state spaces.
- This local search algorithms are also useful when:
  - We want to find the solution with an objective function.
  - For example when we use Genetic Algorithms.

HILL CLIMBING SEARCH

HILL CLIMBING …



Elevetation defined by
the evaluation
function

IDEA OF HILL-CLIMBING

I. Choose the succesor with the highest heuristic evaluation.

II. If there is a succeser *s* from current state *n:*

- **h(s) > h(n)**
- **h(s) >= h(t)** for all succesors *t* of *n*:

- Then move from state *n* to state *s*.
- Else, stop in state *n*.

- Properties:
  - It ends with it reach a maximum.
  - It does not take into accout other states, only the next succesors.
  - It chooses randomly among the set of best succesors (random tiebreaker).
  - It does not use backtraking because it does nor remember the past states where it has visited.
- It is also known as: *Local Gready Search*

It is like climbing the Everest mountain with dense fog and with big amnesia!

HILL-CLIMBING ALGORITHM

function Hill-Climbing(*problem*) **returns** local maximum state

imput: *problem*, a problem

static variables: *current*, **a node.**

next, **a node.**

current ← CREATE-NODE(INITIAL-STATE[*problem*])

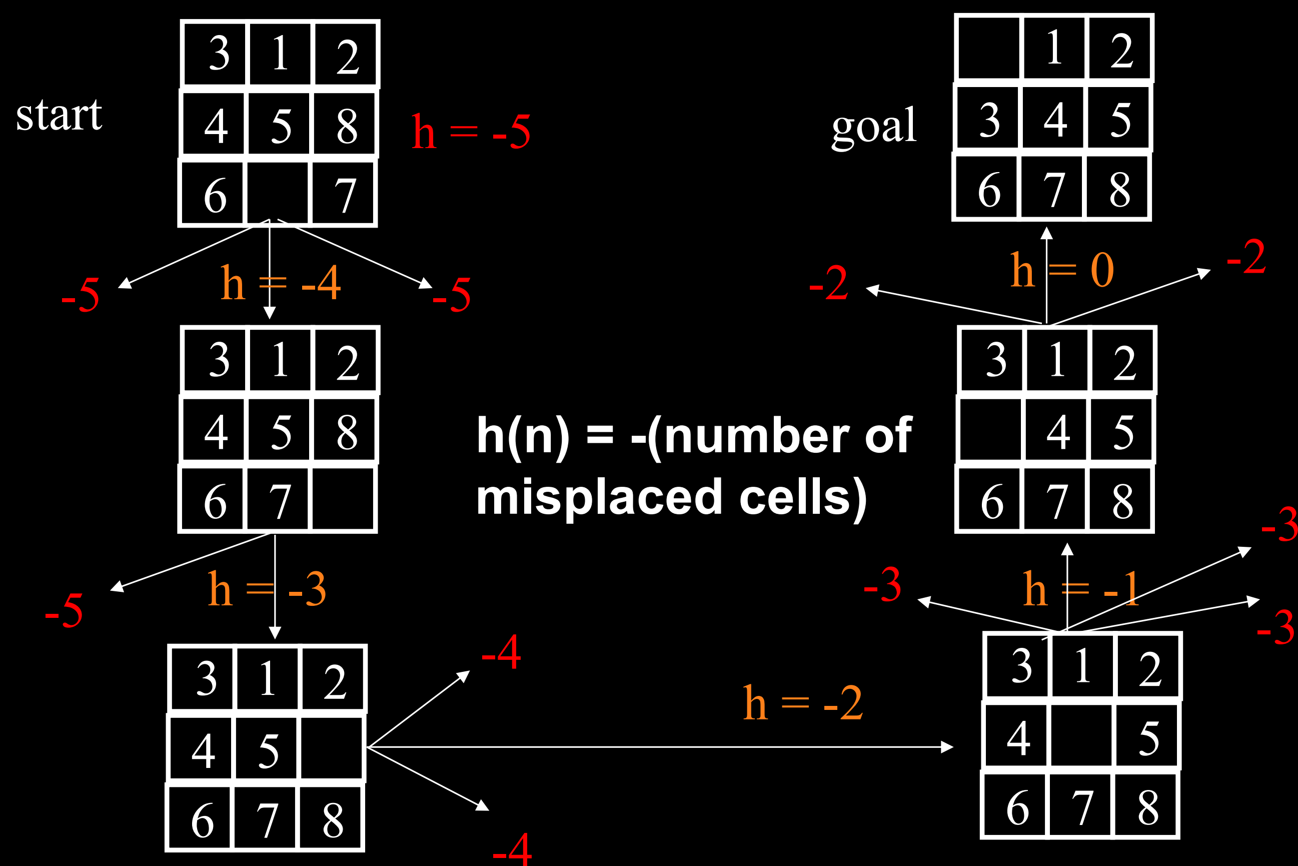
repeat

next ← succesor of *current* with the highest value

if VALUE [*next*] ≤ VALUE[*current*] **then return** STATE[*current*]

current ← *next*

EXAMPLE I



h(n) = -(number of misplaced cells)

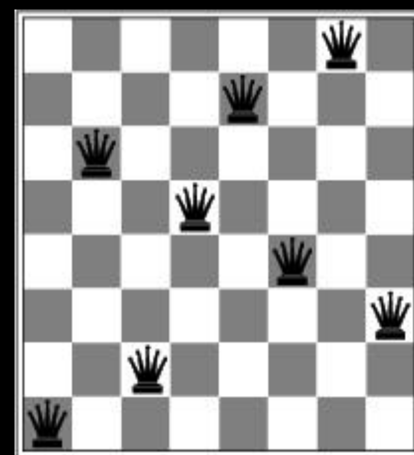
EXAMPLE II: N-QUEENS

- Place n queens in a chees board of $n \times n$ cells such that no queen attacks any other.

LOCAL MINIMUM: 8-QUEENS

a)

b)

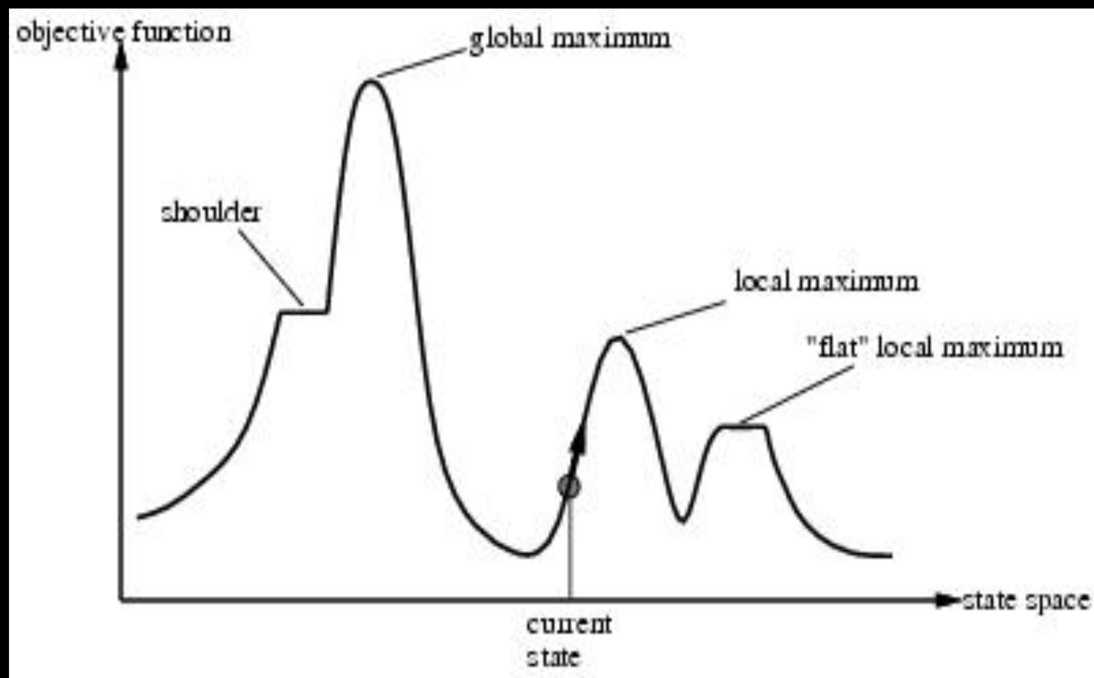


h = the number of pairs of queens that attack each other.

a) A state with **h**=17 and the **h** value of each possible successor.

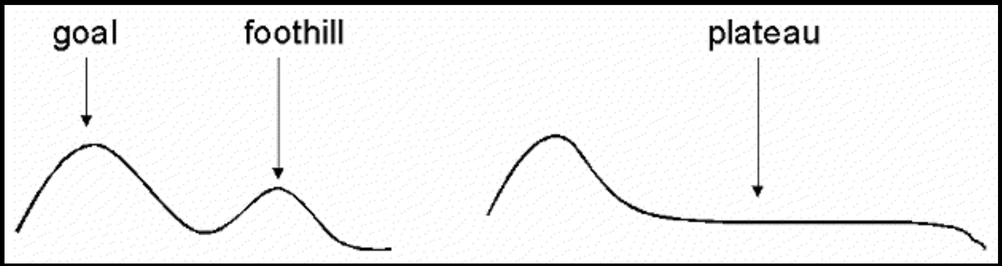
b) A local minimum in the state space (***h***=1).

STATE SPACE POSSIBILITIES

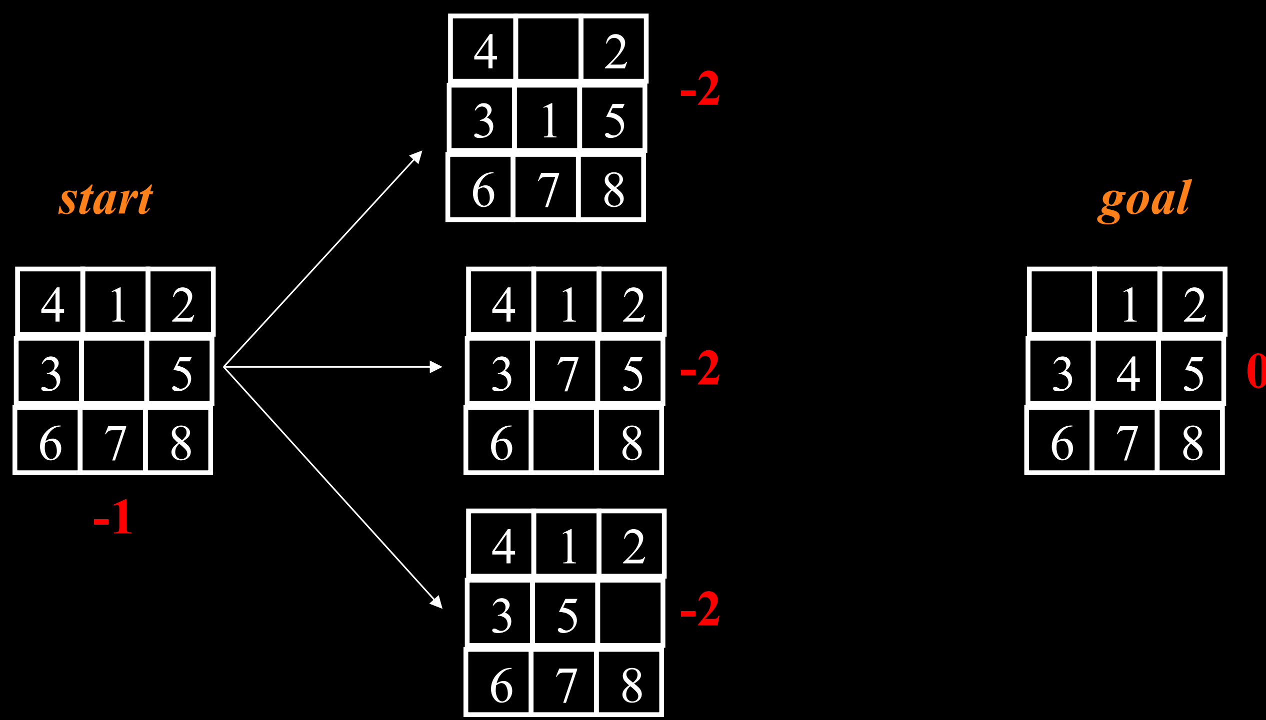


HILL-CLIMBING DISADVANTAGES

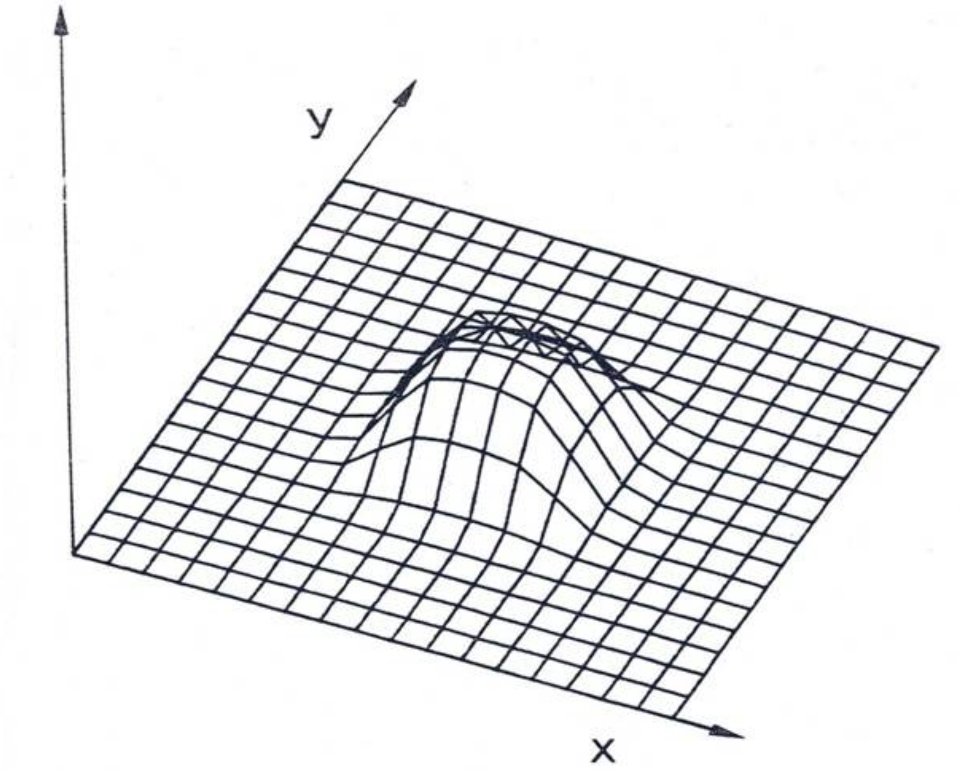
- Disadvantages:
  - **Local maxima (foothills).**
  - **Plateaus (shoulders and flat local maxima)**



EXAMPLE OF A LOCAL MAXIMUM

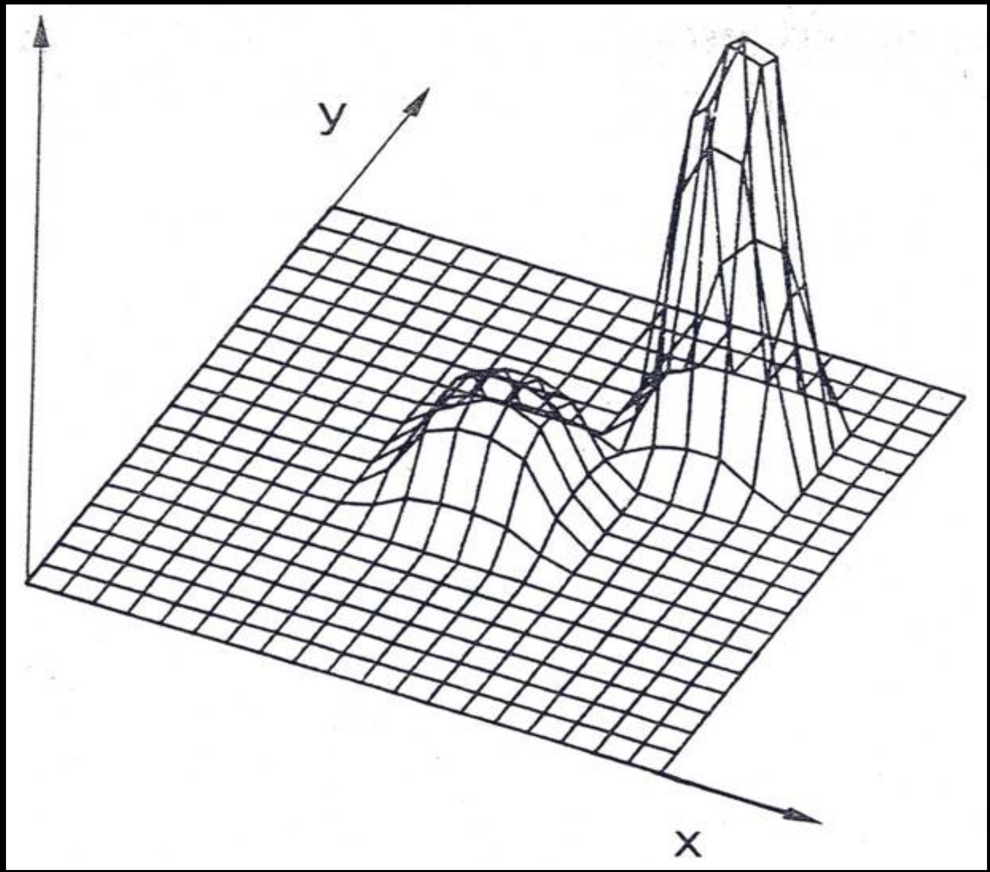


SIMPLE PROBLEM

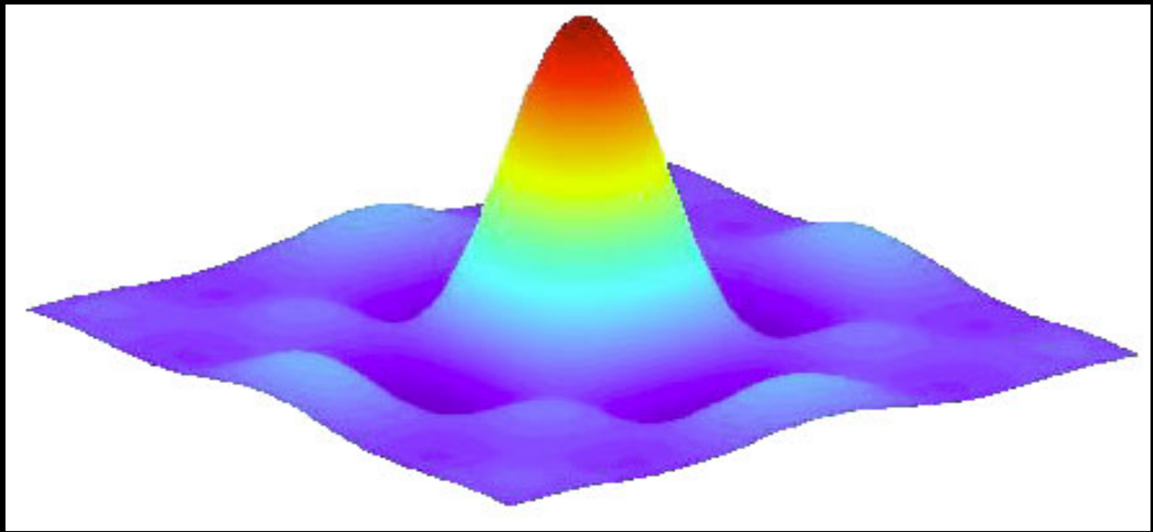


This and next several slides from Goldberg '89

MORE COMPLEX PROBLEM



MUCH MORE COMPLEX PROBLEM



SOLUTIONS FOR THESE PROBLEMS

- Stochastic hill climbing
- Random-restart hill climbing
- After all:
  - Many real problems have a landscape that looks very appropiate for this hill-climbing algorith!

LOCAL BEAM SEARCH

- It keeps track of *k* states rather than just one:
  - It begins with *k* randomly generated states.
  - At each step, all the successors of all *k* states are generated.
  - If any one is the goal, the algorithm halts.
  - Otherwise, it selects the *k* best successors from the complete list and repeats.
- Main difference to the k random re-starts:
  - In ramdom re-start search, each search process runs independenly of the others. Here useful info is passed among the parallel search threads.

SIMULATED ANNEALING SEARCH

SIMULATED ANNEALING

- *In metallurgy:*
  - *Annealing is the process used to temper or harden metals and glass by heating them to a very high temperature and then gradually colling them, thus allowing the material to reach a low-energy crystalline state.*
  - *This algorithm is inspired in this natural process!*
  - *And it has many succesful applications in real life!*
- .

SIMULATED ANNEALING SEARCH

- Idea:
  - To scape of local maxima, allowing ”bad” steps.
  - But decreasing gradually the size and frecuency of these “bad” steps.
- Ping-pong analogy:
  - Shake the surface hard enough (= high tempe
  - Then gradually reduce the intencity of the sho (= reduction of temperature).

- Parameter T
  - By analogy, this parameter is the ***“temperature”*** of the system.
  - T starts an initial high value and we gradually reduce to zero
  - If T is reduced slow enogh, we can reach the global optim

This algorithm has many succesful applications in real life!

SIMULATED ANNEALING ALGORITHM

function SIMULATED-ANNEALING(*problem, schedule*) **returns** a solution state

input: *problem*, a problem

schedule, a mapping from time to "temperature"

current ← MAKE-NODE(problem.INITIAL-STATE)

for t ← 1 to ∞ do

T ← *schedule(t)*

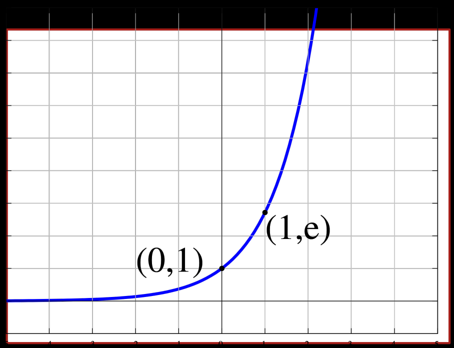
if *T = 0* **then return** *current*

next ← a randomly selected succsor of *current* state

Δ*E* ← *next.VALUE* – *current.VALUE*

if Δ*E* > 0 **then** *current* ← *next*

else *current* ← *next* only with probability $e^{\Delta E/T}$



For exmaple:

- **If temperature T is high, then "bad" moves are allowed with high probability, but they become more unlikely as T decreases.**

GENETIC ALGORITHMS

GENETIC ALGORITHM

- It is a variant of the beam search in which successors states are generated by combining two parent states rather than by modifying a single state.
- It is inspired by the Natural Selection (Charles Darwin) and Genetics (Gregor Mendel).

- Example of the selected and mutated butterfly …

GENETIC ALGORITHM IDEA

- Start with *k* random states (*initial population)*
- Generate new states by:
  1. **"*Crossover reproduction*" combining two *parent states***
  2. **"*Mutation*" of one state (with low probability)**

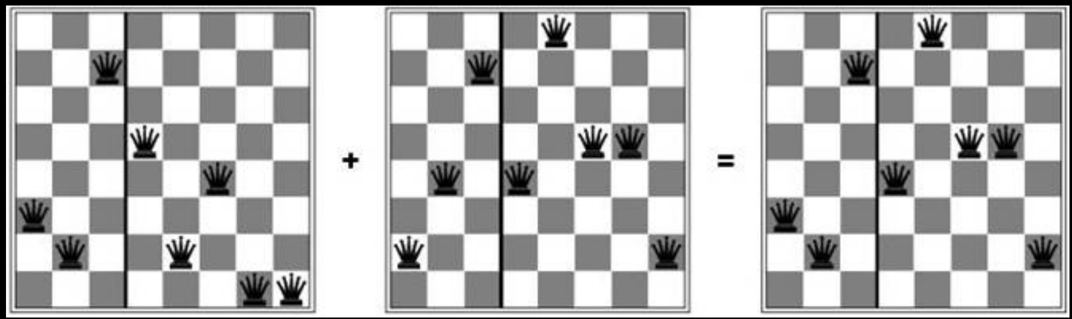
CROSSOVER REPRODUCTION

- If we have these two parents:
  Chromosome 1 1101100100110110
  Chromosome 2 1101111000011110

- Each bit or set of bits represent some characteristic of the solution.

EXAMPLE: 8-QUEENS CHROMOSOMES

101110001011…   110100010100…   101110001100…



REPRODUCTION

- For crossover reproduction, we need to randomly choose a “crossover point”.
- Example (| is the crossover point):

Parent 1	11001 | 00100110110
Parent 2	10011 | 11000011110
Child 1	11001 | 11000011110
Child 2	10011 | 00100110110

MUTATION

- Each subject is randomly selected for mutation.
- Each random mutation with a small independent probability.
- One digit can be mutated in the first, second and so on bit.

Original chromosome	110 1 1110000 1 1110
Mutated chromosome	110 0 1110000 0 1110

GENETIC ALGORITHM

function GENETIC-ALGORITHM(*population,* FITNESS-FN)
   returns an individual
   inputs: *population*, a set of individuals
      FITNESS-FN, a function tha measures the fitness of an individual
   repeat
      new_population ← empty set
      for i **=** 1 **to** SIZE(*population*) **do**
         x ← RANDOM-SELECTION (*population*, FITNESS-FN)
         y ← RANDOM-SELECTION (*population*, FITNESS-FN)
         child ← *REPRODUCE(x, y)*
         if (small random probability) **then** *child* ← MUTATE (*child*)
         add *child* to *new_population*
      population ← *new_population*
   until some individual is fit enough or time has elapsed
   return the best individual in population, according to FITNESS-FN