Introduction to Artificial Intelligence

Chapter 2: Solving Problems by Searching (5)

Local Search Algorithms & Optimization Problems

cuu duong than cong . com

Nguyễn Hải Minh, Ph.D

nhminh@fit.hcmus.edu.vn

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Outline

1. Optimization Problems
2. Hill-climbing search
3. Simulated Annealing search
4. Local beam search
5. Genetic algorithm

cuu duong than cong . com

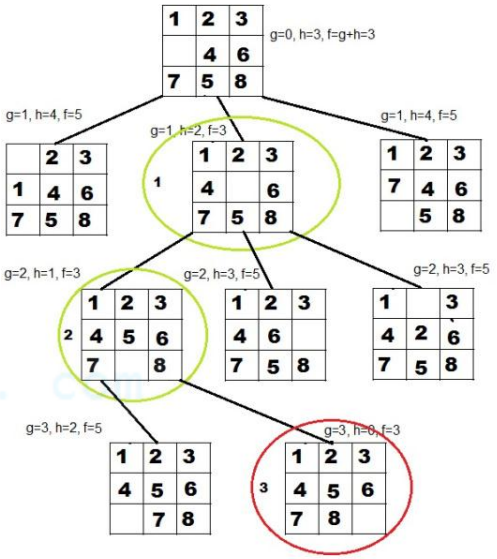
cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Local Search Algorithms & Optimization Problems

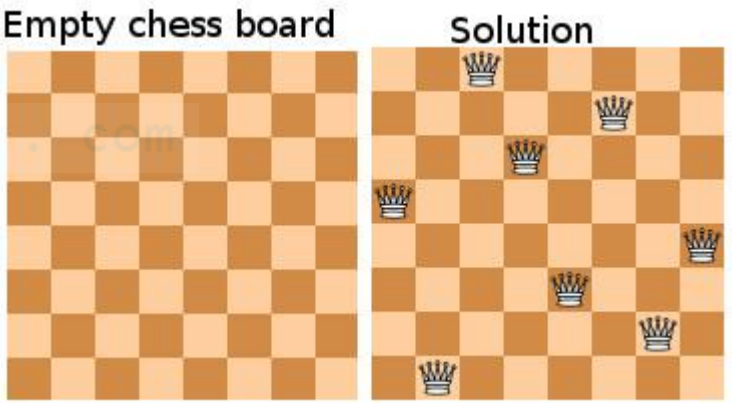
- Previous lecture:
  - ***Path*** to Goal is solution to problem

  → systematic exploration of search space: Global Search



- This lecture:
  - A ***State*** is solution to problem (path is irrelevant)
  - E.g., 8-queens

  → Different algorithms can be used: Local Search



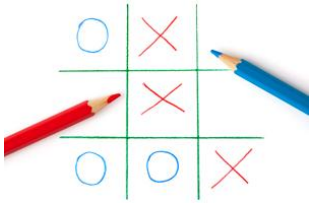
cuu duong than cong . com

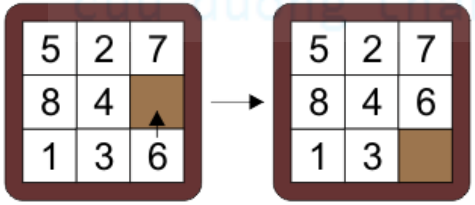
CuuDuongThanCong.com https://fb.com/tailieudientucntt

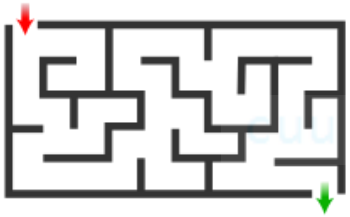
Two types of Problems

Goal Satisfaction

reach the goal node
Constraint Satisfaction



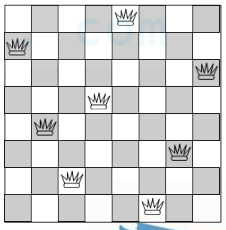


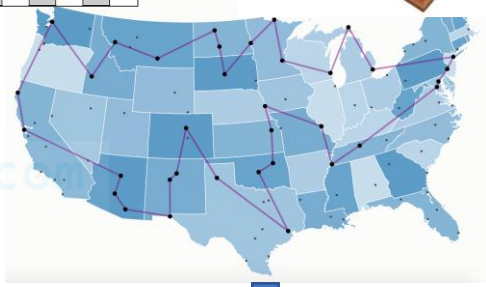


Global Search Algorithms

Optimization

Optimize objective $f(n)$
Constraint Optimization





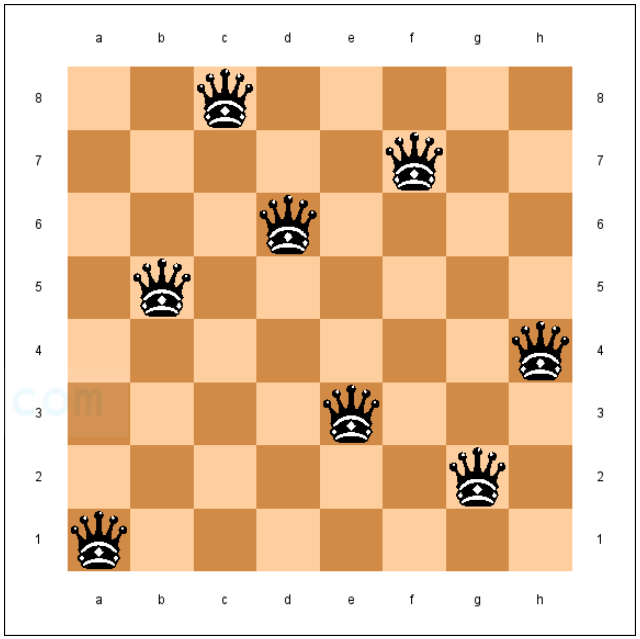
Local Search Algorithms

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Local Search Algorithms & Optimization Problems

- Global search:
  - Can solve n-queen for n = 200
  - Algorithm: ?

- Local search:
  - Can solve n-queen for n = 1,000,000
  - Algorithm:
    - Hill-climbing



cuu duong than cong . com

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Local Search Algorithms & Optimization Problems

- Local search
  - Keep track of single current state
  - Move only to neighboring states
  - Ignore paths

cuu duong than cong . com

- Advantages:
  1. Use very little memory
  2. Can often find reasonable solutions in large or infinite (continuous) state spaces.

CuuDuongThanCong.com https://fb.com/tailieudientucntt

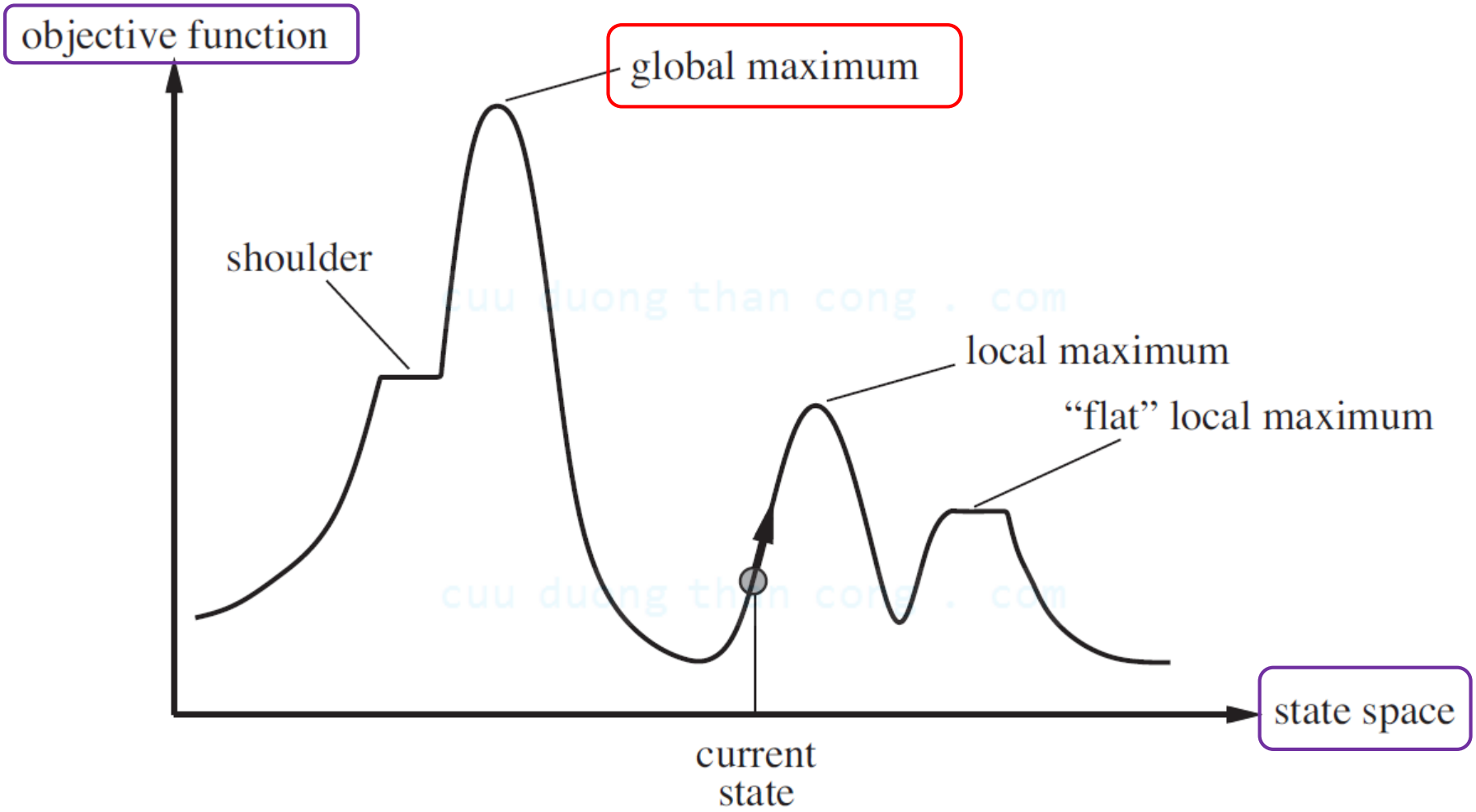
Local Search Algorithms & Optimization Problems

- "*Pure optimization*" problems
  - All states have an objective function
  - Goal is to find state with max (or min) objective value
  - Does not quite fit into path-cost/goal-state formulation
  - Local search can do quite well on these problems.
- Examples:
  - n-queens
  - Machine Allocation
  - Office Assignment
  - Travelling Sale-person Problem
  - Integrated-circuit design…

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

State-space Landscape of Searching for Max



CuuDuongThanCong.com https://fb.com/tailieudientucntt

Hill-climbing search

- A **loop** that continuously moves in the direction of increasing value → uphill
  - terminates when a peak is reached

  → *greedy local search*
- Value can be either:
  - Objective function value (maximized)
  - Heuristic function value (minimized)
- Characteristics:
  - Does not look ahead of the immediate neighbors of the current state.
  - Can randomly choose among the set of best successors, if multiple have the best value

  → *trying to find the top of Mount Everest while in a thick fog*

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Hill-climbing search

```
function HILL-CLIMBING(problem) returns a state that is a local maximum

  current ← MAKE-NODE(problem.INITIAL-STATE)
  loop do
    neighbor ← a highest-valued successor of current
    if neighbor.VALUE ≤ current.VALUE then return current.STATE
    current ← neighbor
```

Locality: move to ***best*** node that is next to current state

Termination: stop when local neighbors are ***no better*** than current state

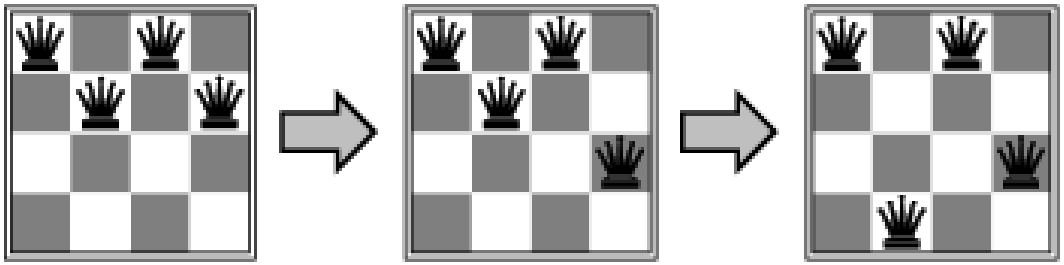
This version of HILL-CLIMBING found local maximum.

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Hill-climbing example: n-queens

- n-queens problem:
  - complete-state formulation:
    - All n queens on the board, 1 per column
  - Successor function:
    - move a single queen to another square in the same column.

    →Each state has ? sucessors

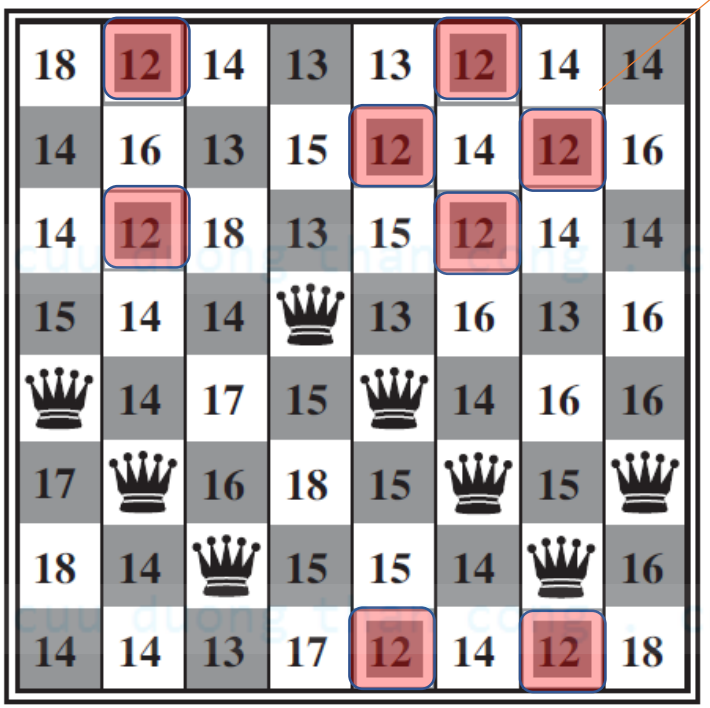


- Example of a heuristic function *h(n)*:
  - the number of pairs of queens that are attacking each other (directly or indirectly)
  - We want to reach *h* = 0 (global minimum)

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Hill-climbing example: 8-queens

- $(c_1\ c_2\ c_3\ c_4\ c_5\ c_6\ c_7\ c_8) = (5\ 6\ 7\ 4\ 5\ 6\ 7\ 6)$



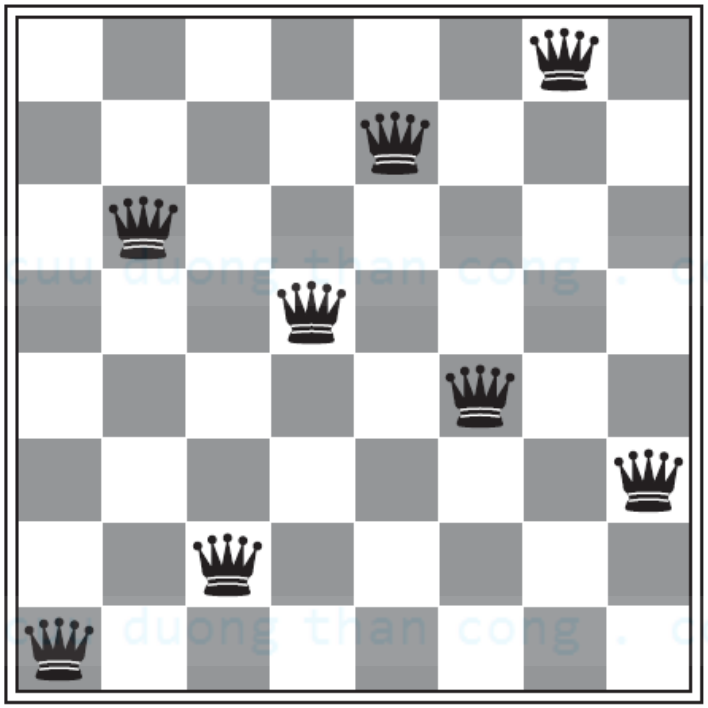
The best moves

- An 8-queens state with heuristic cost estimate *h*=17, showing the value of *h* for each possible successor obtained by moving a queen within its column.

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Hill-climbing example: 8-queens

- $(c_1\ c_2\ c_3\ c_4\ c_5\ c_6\ c_7\ c_8) = (8\ 3\ 7\ 4\ 2\ 5\ 1\ 6)$



- A local minimum in the 8-queens state space; the state has h=1 but every successor has a higher cost.

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Performance of hill-climbing on 8-queens

- Randomly generated 8-queens starting states
  - 14% the time it solves the problem
  - 86% of the time it get stuck at a local minimum

- However…
  - Takes only 4 steps on average when it succeeds
  - And 3 on average when it gets stuck

  (for a state space with ~17 million states)

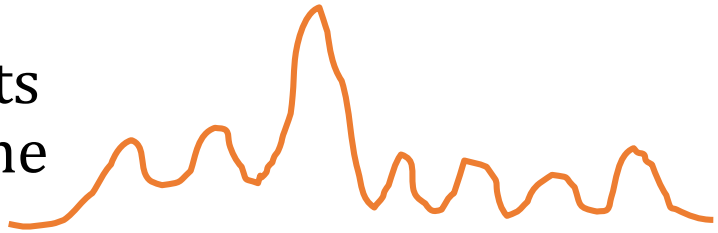
cuu duong than cong . com

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Hill-climbing drawbacks

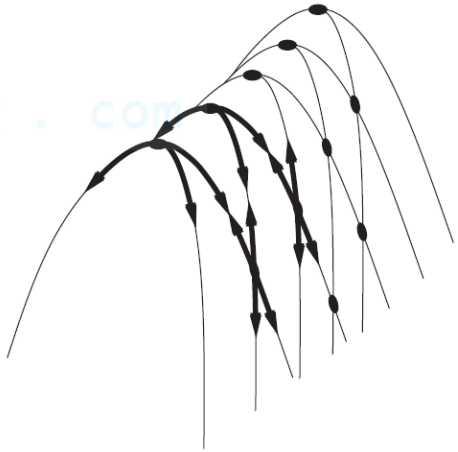
- Local Maxima: a peak higher than its neighboring states but lower than the global maximum



→ *Hill-climbing is suboptimal*

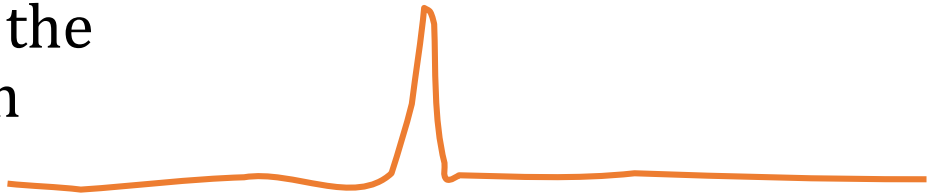
cuu duong than cong . com

- Ridge: sequence of local maxima difficult for greedy algorithms to navigate



cuu duong than cong . com

- Plateau: (Shoulders) an area of the state space where the evaluation function is flat.



Escaping Shoulders: Sideways Moves

- If no downhill (uphill) moves, allow *sideways* moves in hope that algorithm can escape
  - Need to place a **limit** on the possible number of sideways moves to avoid infinite loops
- For 8-queens
  - Now allow sideways moves with a limit of 100
  - Raises percentage of problem instances solved from 14 to 94%
  - However….
    - 21 steps for every successful solution
    - 64 for each failure

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Hill-climbing variations

1. Stochastic hill-climbing
   - Random selection among the uphill moves.
   - The selection probability can vary with the steepness of the uphill move.

   → *converges more slowly than steepest ascent, but in some state landscapes, it finds better solutions*
2. First-choice hill-climbing
   - Generating successors randomly until a better one is found.

   → *Useful when there are a very large number of successors.*
3. Random-restart hill-climbing

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Random Restarts Hill-climbing

- Tries to avoid getting stuck in **local maxima.**

- Different variations
  - For each restart: run until termination vs run for a fixed time
  - Run a fixed number of restarts or run indefinitely

- Analysis
  - Say each search has probability p of success
    - E.g., for 8-queens, $p = 0.14$ with no sideways moves
  - Expected number of restarts?
  - Expected number of steps taken?

cuu duong than cong . com

Search using Simulated Annealing

❑Idea:

Escape local maxima by allowing some "***bad***" moves (downhill) but *gradually decrease* their size and frequency

cuu duong than cong . com

- **Probability** of taking downhill move decreases with number of iterations, steepness of downhill move
- Controlled by **annealing schedule**

cuu duong than cong . com

→ *Inspired by tempering of glass, metal*

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Physical Interpretation of Simulated Annealing

- Annealing = physical process of cooling a liquid or metal until particles achieve a certain frozen crystal state.
  - Simulated Annealing:
    - free variables are like particles
    - seek "low energy" (high quality) configuration
    - get this by slowly reducing temperature T, which particles move around randomly

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Search using Simulated Annealing

function SIMULATED-ANNEALING(*problem*, *schedule*) **returns** a solution state
  inputs: *problem*, a problem
    schedule, a mapping from time to "temperature"

  current ← MAKE-NODE(*problem*.INITIAL-STATE)
  for $t = 1$ **to** ∞ **do**
    $T \leftarrow schedule(t)$
    if $T = 0$ **then return** *current*
    next ← a randomly selected successor of *current*
    $\Delta E \leftarrow next.\text{VALUE} - current.\text{VALUE}$
    if $\Delta E > 0$ **then** $current \leftarrow next$
    else *current* ← *next* only with probability $e^{\Delta E/T}$

Good neighbors: always accept ***better*** local moves

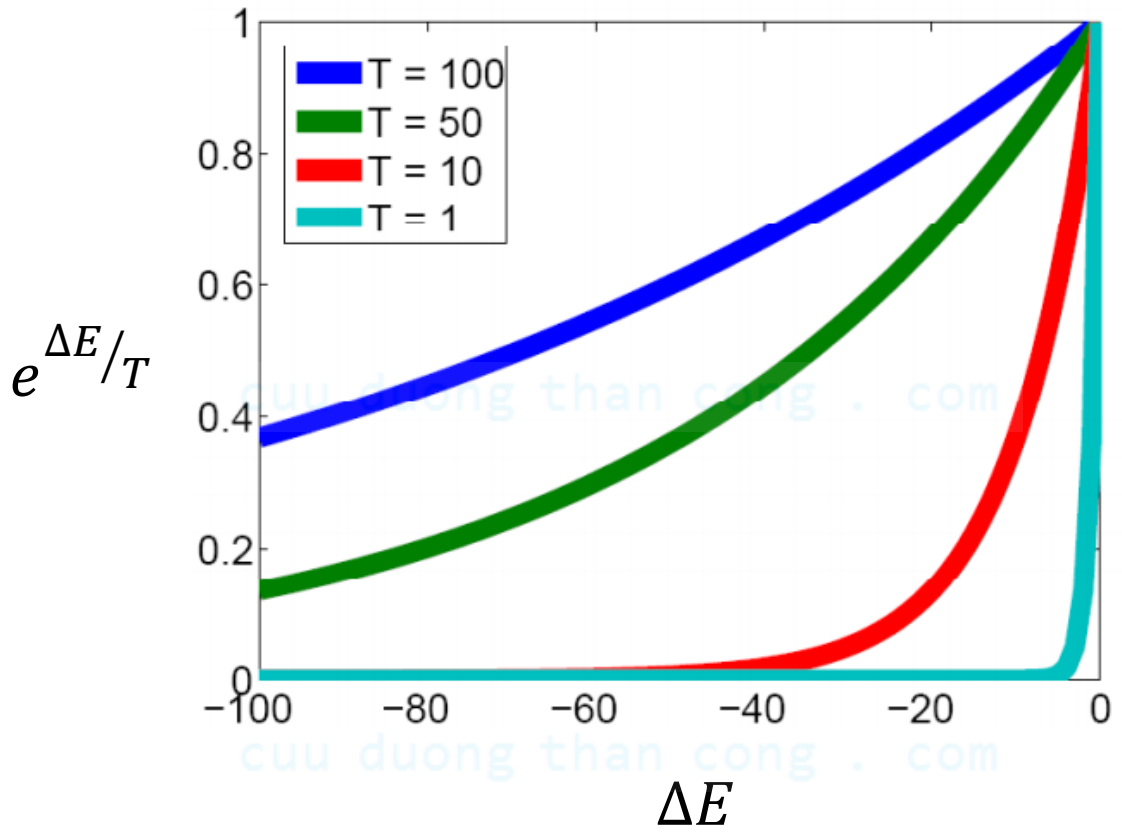
Bad neighbors: accept in proportion to "***badness***"

Temperature reduction: slowly decrease T over time

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Effect of Temperature

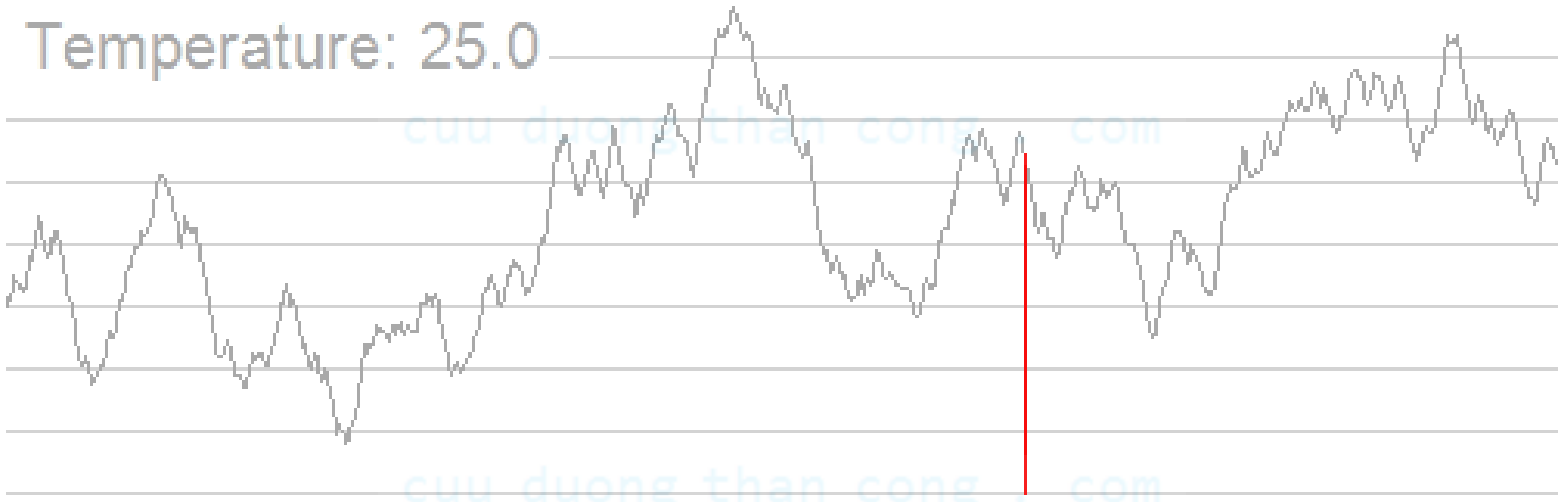


If temperature decreases slowly enough, the algorithm will find a global optimum with probability approaching 1.

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Search using Simulated Annealing

- Despite the many local maxima in this graph, the global maximum can still be found using **simulated annealing**



CuuDuongThanCong.com https://fb.com/tailieudientucntt

Example on Simulated Annealing

- Lets say there are 3 moves available, with changes in the objective function of
  - $\Delta E_1 = -0.1$
  - $\Delta E_2 = 0.5$ (good move)
  - $\Delta E_3 = -5$

- Let T=1, pick a move randomly:
  - if ΔE_2 is picked, move there.
  - if ΔE_1 ΔE_3 are picked:
    - move 1: prob1 = $e^{\Delta E/T} = e^{-0.1} = 0.9$,
    - move 3: prob3 = $e^{\Delta E/T} = e^{-5} = 0.05$

90% of the time we will accept this move

5% of the time we will accept this move

- T = "temperature" parameter
  - high T => probability of "locally bad" move is higher
  - low T => probability of "locally bad" move is lower
  - Typically, T is ***decreased*** as the algorithm runs longer
    - i.e., there is a "temperature schedule"

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Simulated Annealing in Practice

- Simulated annealing was first used extensively to solve VLSI layout problems in the early 1980s.

- Other applications:
  - Traveling Salesman Problem
  - Factory Scheduling
  - Timetable Problem
  - Image Processing
  - …

- Useful for some problems, but can be very slow

→*Because T must be decreased very gradually to retain optimality*

- How do we decide the rate at which to decrease T?

→*This is a practical problem with this method*

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Local beam search

- Idea: Keeping only one node in memory is an extreme reaction to memory problems.

- Keep track of k states instead of one
  - Initially: k randomly selected states
  - Next: determine all successors of k states
  - If any of successors is goal → finished
  - Else select k best from successors and repeat

cuu duong than cong . com

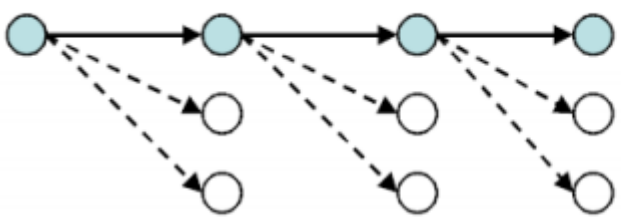
CuuDuongThanCong.com https://fb.com/tailieudientucntt

Local beam search

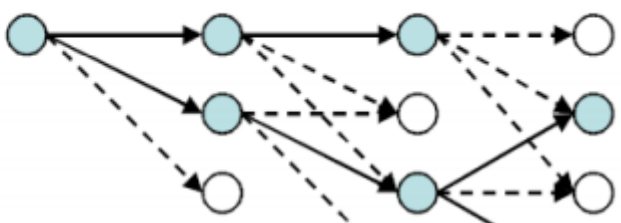
- Major difference with random-restart search
  - Information is shared among *k* search threads.

  →Searches that find good states recruit other searches to join them

cuu duong than cong . com



Greedy search



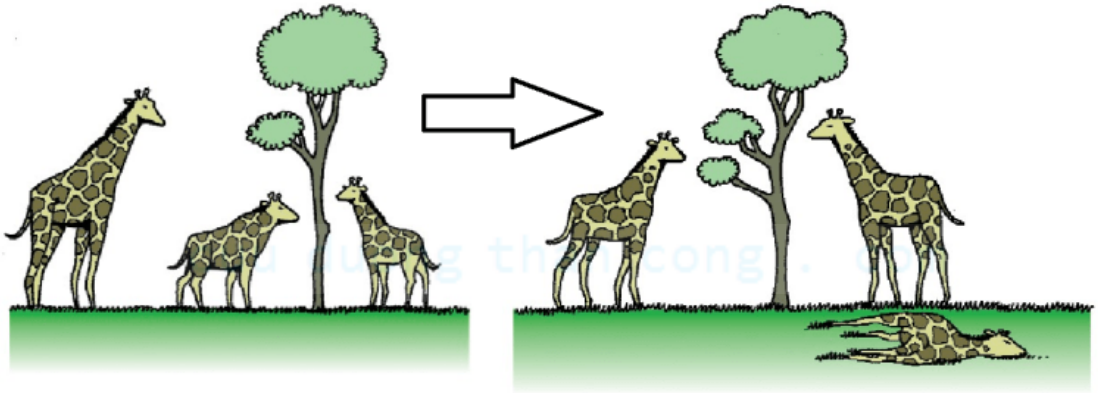
Beam search

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Local beam search

- Problem: quite often, all *k states end up on same local hill*
  - → *Stochastic beam search:* choose *k* successors randomly, biased towards good ones.
  - → Resemblance to the process of natural selection
    - "successors" (offspring) of a "state" (organism) populate the next generation according to its "value" (fitness).



Natural Selection in action

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Genetic algorithms

- Twist on Local Search:
  - successor is generated by combining two parent states

- A state is represented as a string over a finite alphabet (e.g. binary)
  - 8-queens
    - State = position of 8 queens each in a column
      => 8 x log(8) bits = 24 bits (for binary representation)

cuu duong than cong . com

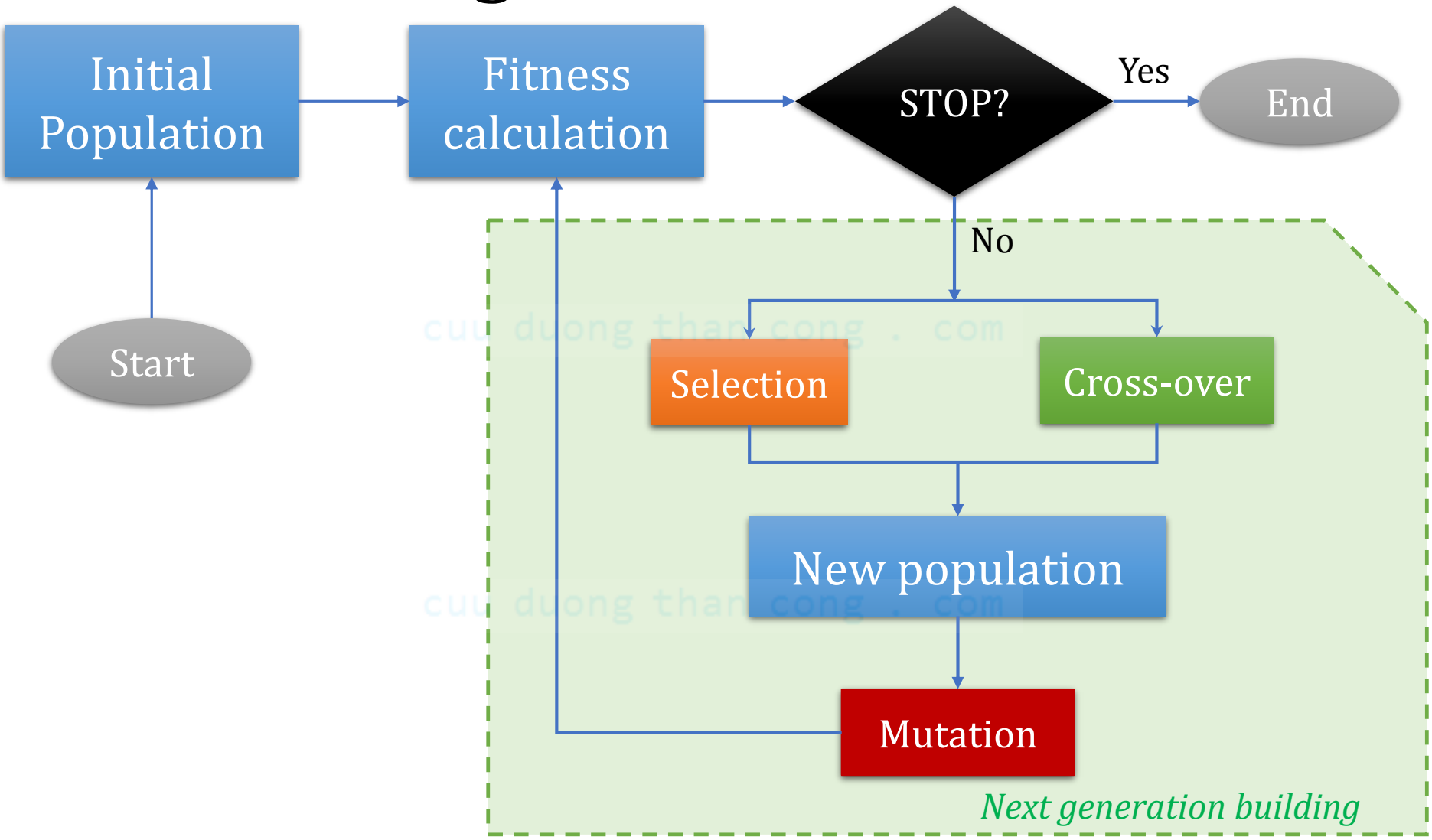
CuuDuongThanCong.com https://fb.com/tailieudientucntt

Genetic algorithms

- Start with *k* randomly generated states (population)
- Evaluation function (fitness function).
  - Higher values for better states.
  - Opposite to heuristic function, e.g., *#non-attacking pairs in 8-queens*
- Produce the next generation of states by "simulated evolution"
  - Random selection
  - Crossover
  - Random mutation

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Genetic algorithms



CuuDuongThanCong.com https://fb.com/tailieudientucntt

Genetic algorithms

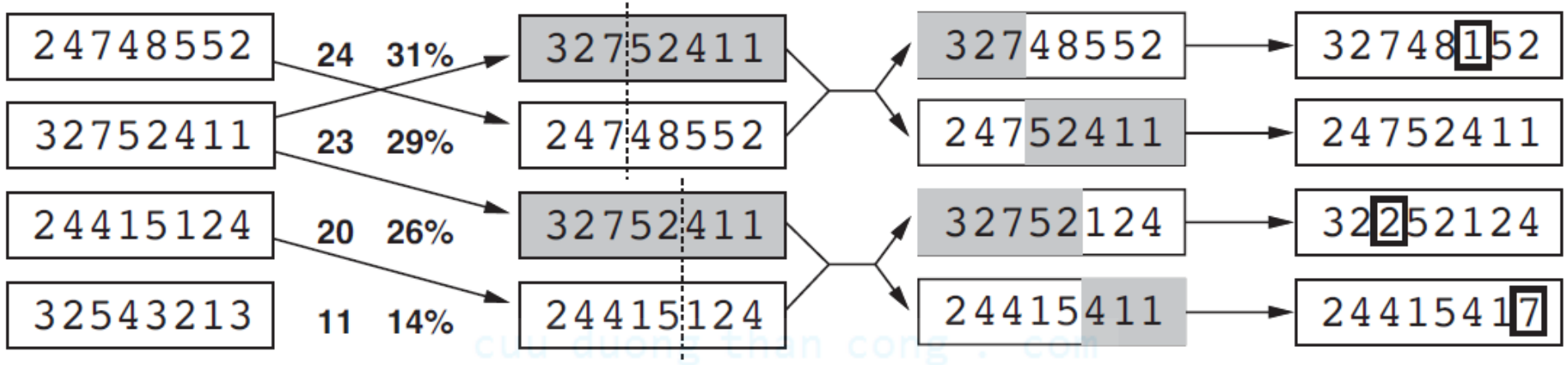
- Genetic representation:
  - Use integers
  - Use bit string

- Fitness function: number of non-attacking pairs of queens (min = 0, max = 8×7/2 = 28)
  - 24/(24+23+20+11) = 31%
  - 23/(24+23+20+11) = 29% etc

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Genetic algorithms



(a) Initial Population | (b) Fitness Function | (c) Selection | (d) Crossover | (e) Mutation

4 states for 8-queens problem

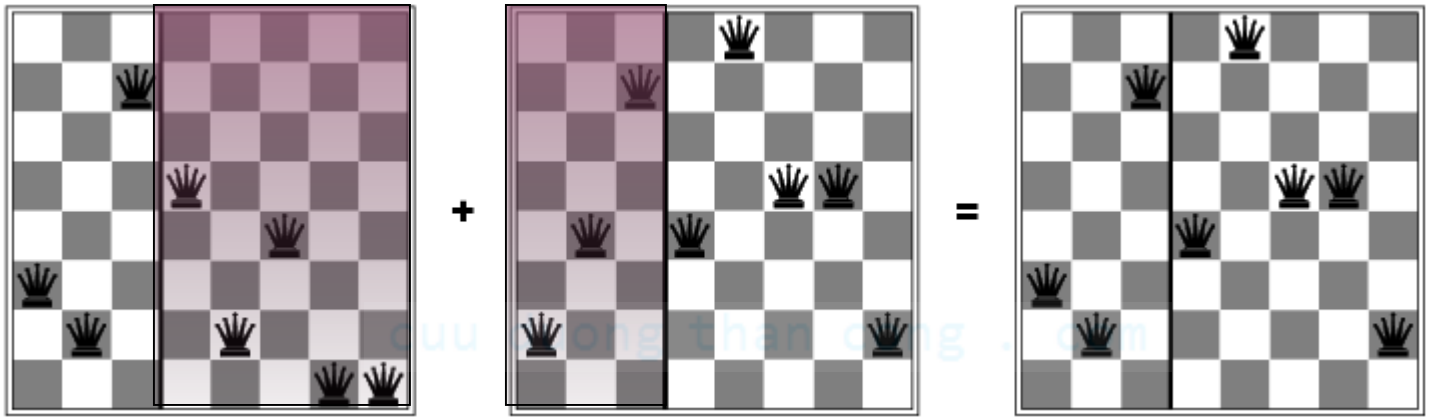
2 pairs of 2 states randomly selected based on fitness. Random crossover points selected

New states after crossover

Random mutation applied

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Genetic algorithms



Has the effect of "jumping" to a completely different new part of the search space (quite non-local)

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Genetic algorithm pseudocode

```
function GENETIC-ALGORITHM(population, FITNESS-FN) returns an individual
  inputs: population, a set of individuals
          FITNESS-FN, a function that 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 enough time has elapsed
  return the best individual in population, according to FITNESS-FN
```


CuuDuongThanCong.com https://fb.com/tailieudientucntt

Genetic algorithm pseudocode

function REPRODUCE(x, y) **returns** an individual
  inputs: x, y, parent individuals

  $n \leftarrow$ LENGTH(x); $c \leftarrow$ random number from 1 to n
  return APPEND(SUBSTRING(x, 1, c), SUBSTRING(y, $c+1$, n))

cuu duong than cong . com

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Comments on genetic algorithms

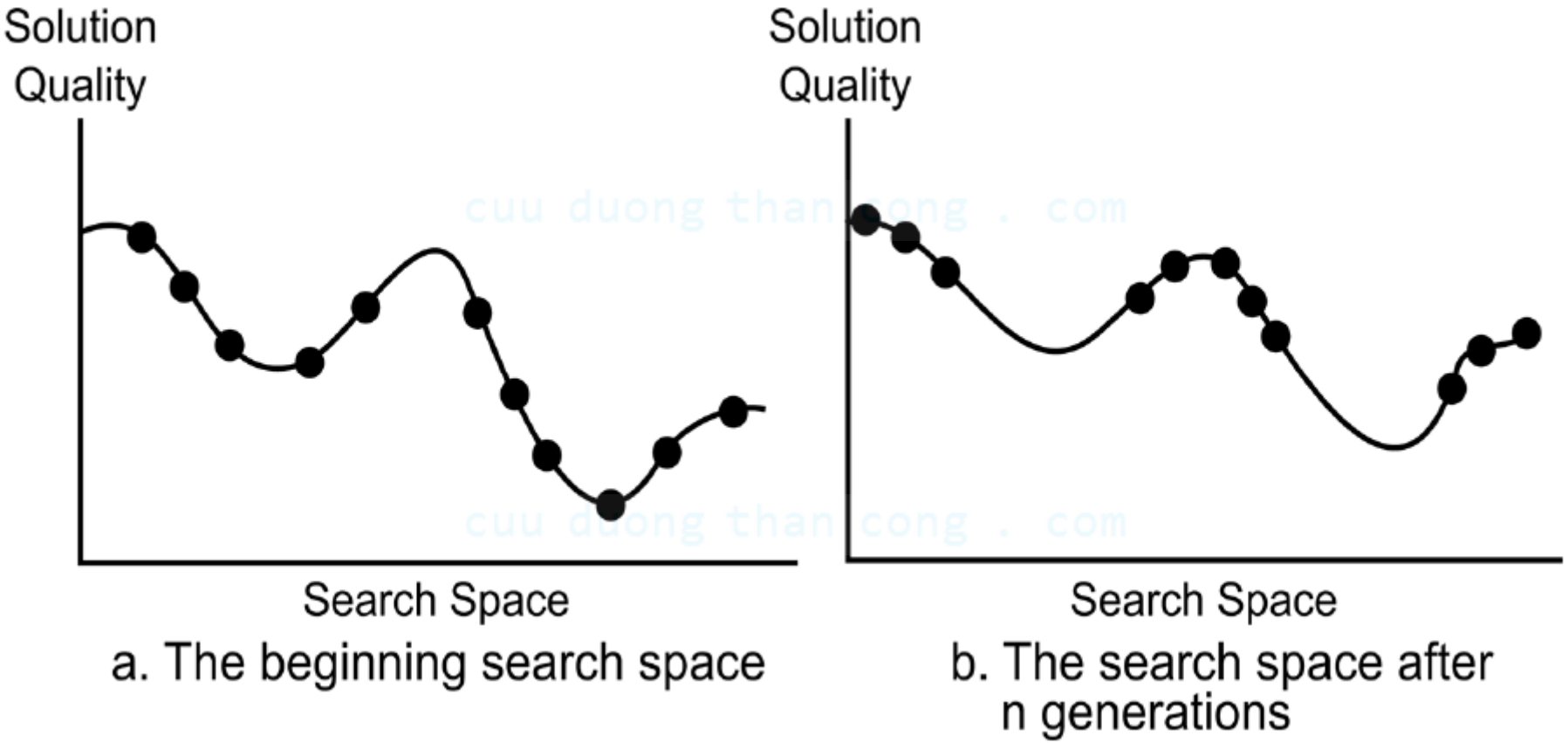
- Positive points
  - Random exploration can find solutions that local search can't
    - (via crossover primarily)
    - Can solve "hard" problem
  - Rely on very *little domain knowledge*
  - Appealing connection to human evolution
    - E.g., see related area of genetic programming

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Comments on genetic algorithms

- Positive points



CuuDuongThanCong.com https://fb.com/tailieudientucntt

Comments on genetic algorithms

- Negative points
  - Large number of "*tunable*" parameters
    - Difficult to replicate performance from one problem to another
  - Lack of good empirical studies comparing to simpler methods
  - Useful on some (small?) set of problems but no convincing evidence that GAs are better than hill-climbing w/random restarts in general

cuu duong than cong . com

- Application: Genetic Programming!

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Next class

- Chapter 2: Solving Problems by Searching (cont.)
  - Adversarial Search (Games)

cuu duong than cong . com

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt