Artificial Intelligence (CS303)

Lecture 2: Beyond Classical Search

Hints for this lecture

- **“Classical” search: use a tree representation**
- **“Beyond”: generalize to other types of representation**

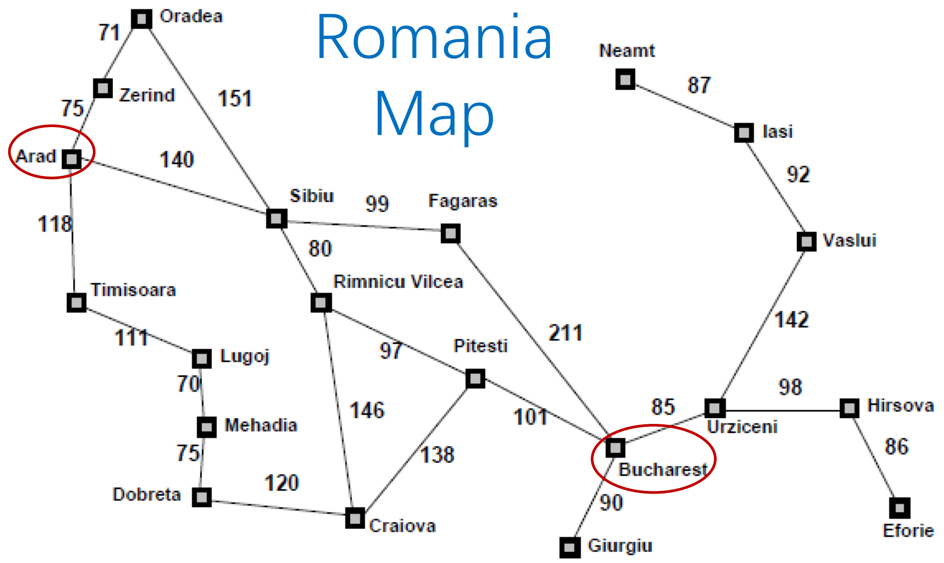
Outline of this lecture

- More Representations
- General Search Methods
- More complex real-world search scenarios

I. More representations

More representations

- Route planning on Romania map again…
- In lecture 1, we formulated the problem with tree representation.
- Any different way to formulate it?
- In other words, what is a “state”?
- Alternative: permutations of cities
- Which one is better?



More representations

- **In essence, as long as the state space is finite (finite set), tree representation can be used to formulate any problem.**

- **In essence, any problem solved on modern computing hardware can be viewed as having a finite state space.**

- **But we are not only concerned about efficacy, but also efficiency, for which the tree representation is not always the best choice.**

More representations

- **Advantage of tree search: we understand how a final solution is obtained step by step.**

- **But this advantage is usually unnecessary, e.g., for optimization problems.**

Optimization problem

$$\begin{aligned} &\text{maximize} \quad f(x) \\ &\text{subject to: } g_i(x) \pounds 0, \quad i = 1 \ldots m \\ &\qquad\qquad\quad\; h_j(x) = 0, \quad j = 1 \ldots p \end{aligned}$$

More representations

- **Optimization is ubiquitous.**

Transportation: Railway timetabling

Energy: Wind turbine design

Architecture: Truss design (Birds nest)

Finance: Portfolio optimization

More representations

- **When tackling an optimization problem, the “agent” search for a solution in the solution space.**
- **Each element of the solution space is a complete solution.**
- **We care much more about the quality of the obtained solution, rather than the interpretability of the search process.**
- **Of course the “action” still needs to be defined.**

More representations

- Representations of a solution space can be roughly categorized as:
    - Continuous
    - Discrete
        - Binary
        - Integer
        - Permutation
        - …

- Different representations may favor different search methods (operators), but most of them share a common framework.

II. General Search methods

General Search Methods (hint)

A. M. Turing

We have thus divided our problem into two parts. The child programme and the education process. These two remain very closely connected. We cannot expect to find a good child machine at the first attempt. One must experiment with teaching one such machine and see how well it learns. One can then try another and see if it is better or worse. There is an obvious connection between this process and evolution, by the identifications

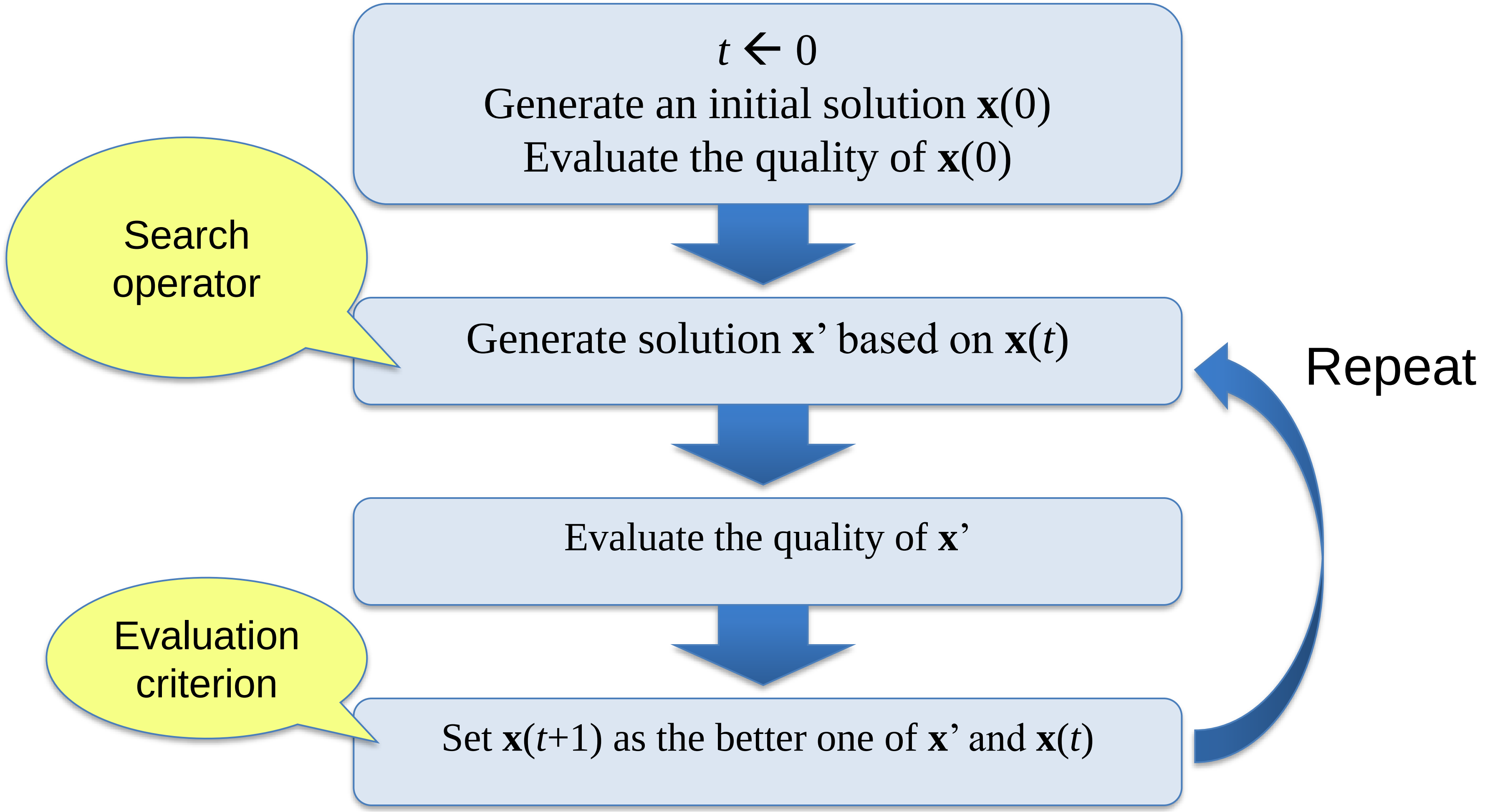
Structure of the child machine = hereditary material

Changes of the child machine = mutation,

Natural selection = judgment of the experimenter

One may hope, however, that this process will be more expeditious than evolution. The survival of the fittest is a slow method for measuring advantages. The experimenter, by the exercise of intelligence, should he able to speed it up. Equally important is the fact that he is not restricted to random mutations. If he can trace a cause for some weakness he can probably think of the kind of mutation which will improve it.

General Search Methods (framework)



$t \leftarrow 0$
Generate an initial solution $\mathbf{x}(0)$
Evaluate the quality of $\mathbf{x}(0)$
Search operator
Generate solution $\mathbf{x}'$ based on $\mathbf{x}(t)$
Repeat
Evaluate the quality of $\mathbf{x}'$
Evaluation criterion
Set $\mathbf{x}(t+1)$ as the better one of $\mathbf{x}'$ and $\mathbf{x}(t)$

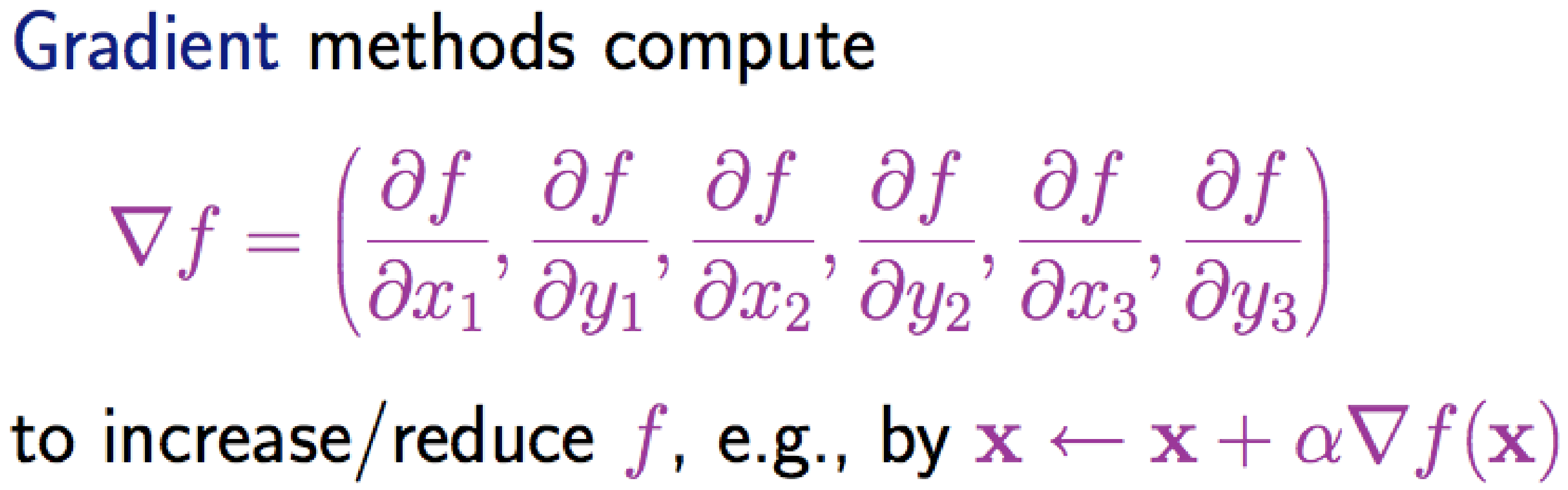
General Search Methods (framework)

- **Two basic issues (differs over concrete search methods):**
  - **search operator**
  - **evaluation criterion (or replacement strategy)**

Greedy idea in continuous space

Gradient descent

- Suppose the objective function $f(x_1, y_1, x_2, y_2, x_3, y_3)$ is differentiable



Gradient methods compute

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

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

Limitations of Gradient Descent

- **GD is great, why we need other approaches?**

Discrete Search Space

Railway timetabling

Turbine Design

Non-differentiable constraints + mixed integer search space

Non-differentiable Objective function

Truss design

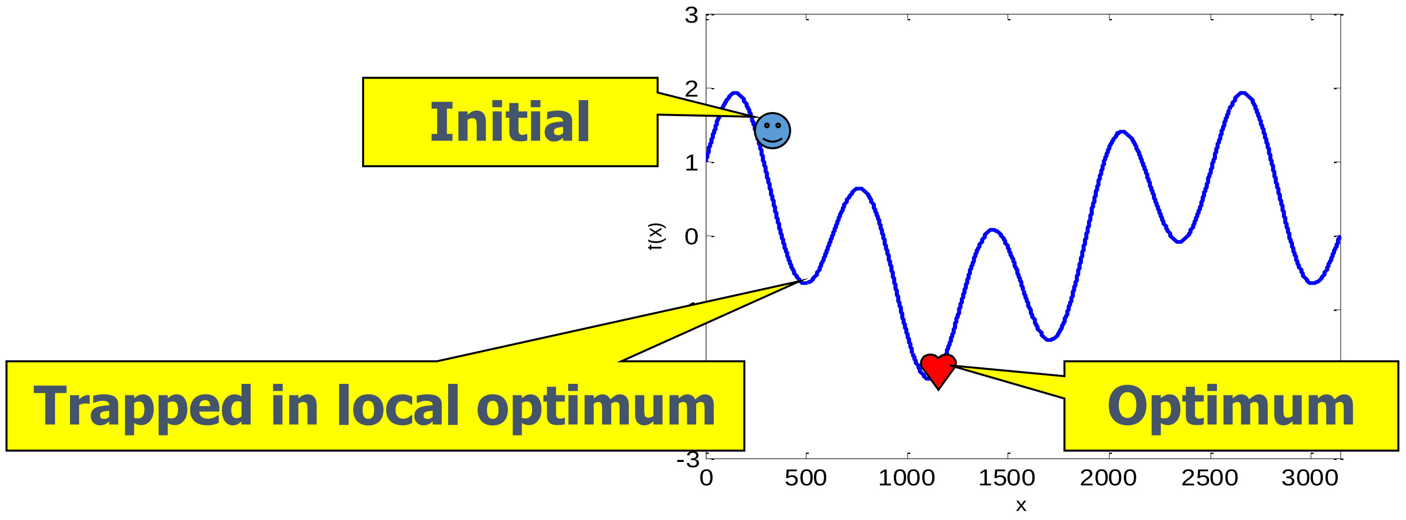
Portfolio optimization

Discrete Search space

Greedy idea in discrete space

- Idea: heuristic + sampling to “approximate” the behavior of GD.
- Modify the solution with a predefined search operator
  - Flip 1 (or n) bit of a binary string
  - Exchange the positions of two elements in a permutation
  - Use a probability distribution to generate a new solution
- But greedy idea may fail on multimodal problems

Hill Climbing



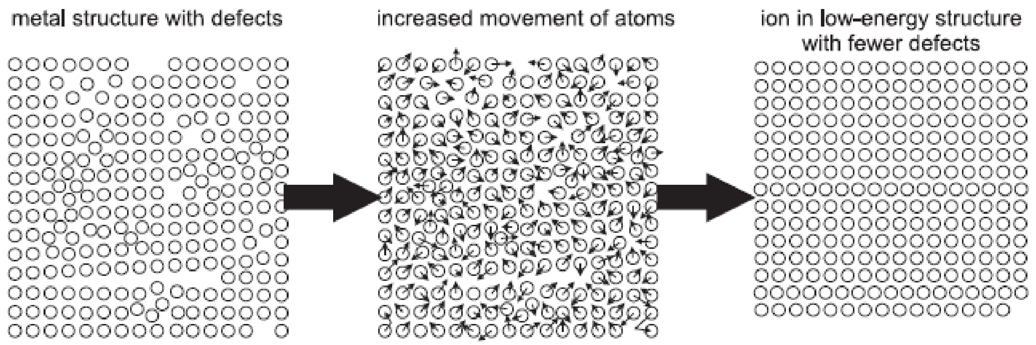
Non-greedy ideas

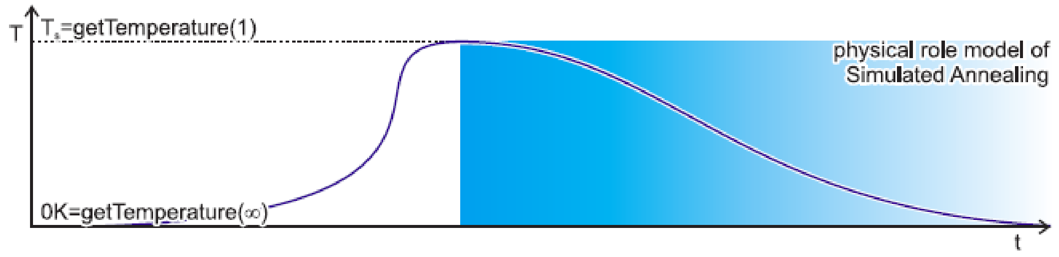
- Typical ideas:
  - Simulated Annealing
  - Tabu Search
  - Population-based search

- Usually referred to as metaheuristics.

Simulated Annealing

Rainer Halama, Radreifenproduktion im Bochumer Verein, ExtraSchicht


metal structure with defects
increased movement of atoms
ion in low-energy structure with fewer defects



temperature from high to low
when high temperature, form the shape
when low temperature, polish the detail

Simulated Annealing

- **Hill Climbing → Simulated Annealing**

```
Do while  (halt condition is not satisfied)
• Generate solution x' based on x(t)
• Evaluate the f (x')
• If  f(x')>f(x_i),
• Then x(t+1) = x',  otherwise x(t+1) = x
• i=i+1
```

Replace this line with a probability *p*

$$p = \begin{cases} 1 & if \quad f(x_i) < f(x_i^{'}) \\ \exp(-\frac{f(x_i) - f(x_i^{'})}{T}) & if \quad f(x_i) \geq f(x_i^{'}) \end{cases}$$

Simulated Annealing

```
• Generate an initial $\mathbf{x}(0)$
• Set the initial temperature $T(0)$;
• Do while (halt condition is not satisfied)
    Get current temperature $T(i)$;
    Generate solution x’ based on $\mathbf{x}(i)$
    Calculate $f(x')$以及$\Delta f = f(\mathbf{x}(i)) - f(x')$
    If $\Delta f < 0$ (for maximization)， then $\mathbf{x}(i+1)= x'$
    Otherwise, $p = \exp(-\Delta f / T(i))$;
        If $c$ = random[0,1] $< p$, $\mathbf{x}(i+1)= x'$;
        Otherwise $\mathbf{x}(i+1)= \mathbf{x}(i)$;
    $i=i+1$
    Update $T(i)$;
• End Do
```

Population-based search (local beam search)

Idea: keep k states instead of 1; choose top k of all their successors

Not the same as k searches run in parallel!
Searches that find good states recruit other searches to join them

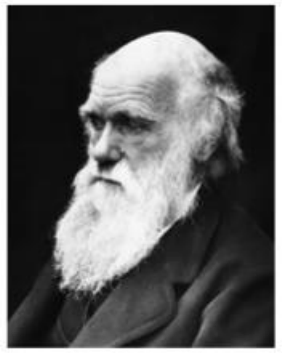
Problem: quite often, all k states end up on same local hill

Idea: choose k successors randomly, biased towards good ones

Observe the close analogy to natural selection!

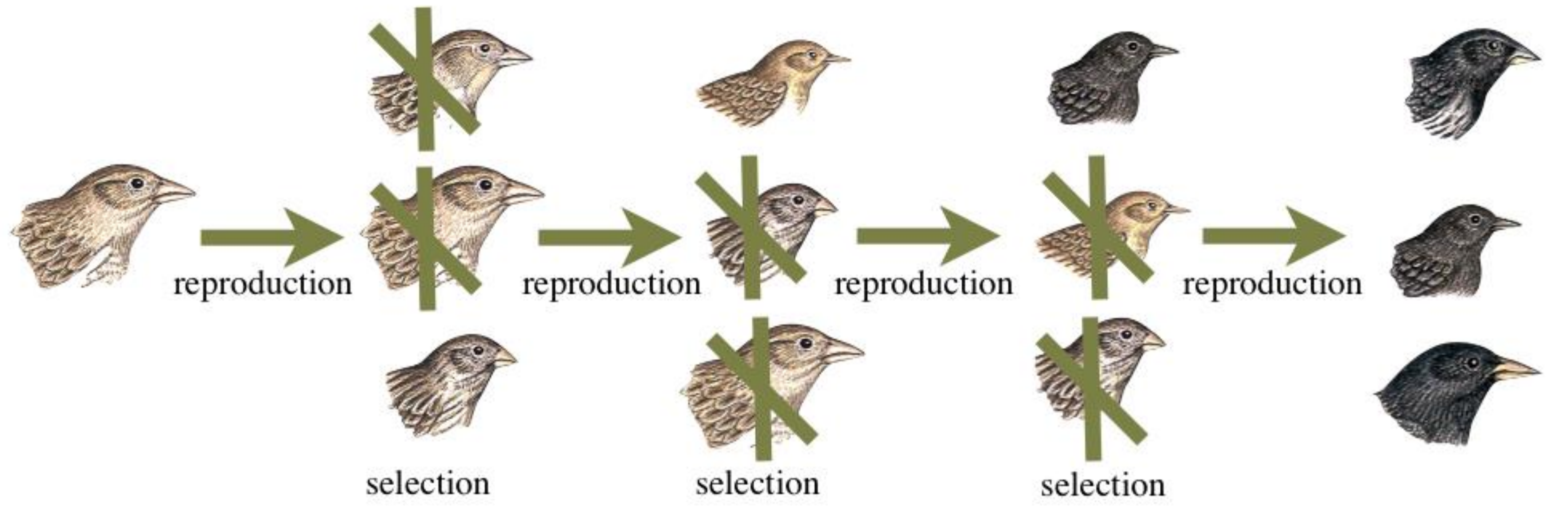
Population-based search (Evolutionary Algorithms)

Biological evolution


Charles Darwin
1809-1882

C. Darwin, after collecting abundant evidence, developed a theory about how species evolve.

reproduction with variation + nature selection



Population-based search (Evolutionary Algorithms)

```
1. Generate the initial **population** $P(0)$ at random, and set
   $i \leftarrow 0$;
2. REPEAT
   (a) Evaluate the fitness of each individual in $P(i)$;
   (b) **Select** parents from $P(i)$ based on their fitness in $P(i)$;
   (c) **Generate** offspring from the parents using *crossover* and
       *mutation* to form $P(i+1)$;
   (d) $i \leftarrow i + 1$;
3. UNTIL halting criteria are satisfied
```

Why “Population” could be Beneficial?

- **Suppose for a continuous optimization problem**
  - New solutions are generated by sampling a Gaussian distribution
  - For each iteration, the better solutions are preserved (elitism)

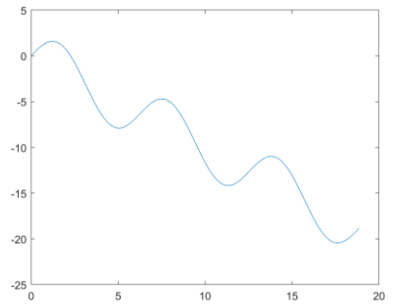
- **The above process is equivalent to optimizing a new objective function**

$$\mathcal{J} = \int f(\boldsymbol{x})p(\boldsymbol{x}|\boldsymbol{\theta}_1)\mathrm{d}\boldsymbol{x}$$

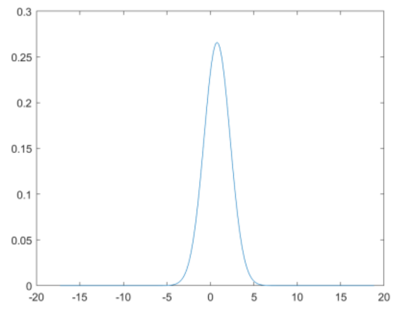
- **A major advantage is that the original objective function is “smoothed”.**

Why “Population” could be Beneficial?

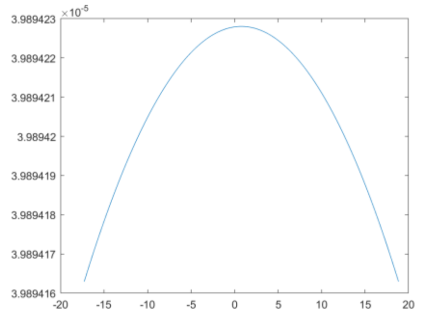
- **The smoothing effect**

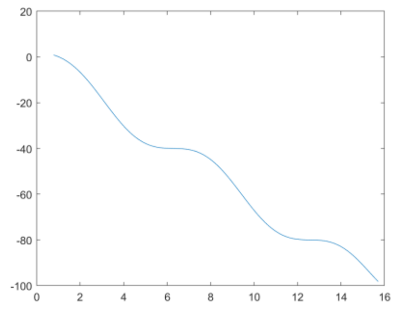


$f(\boldsymbol{x})$

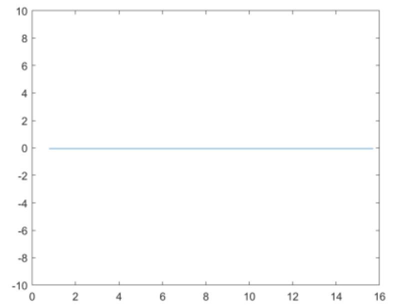


$p(\boldsymbol{x}|\boldsymbol{\theta}_i)$



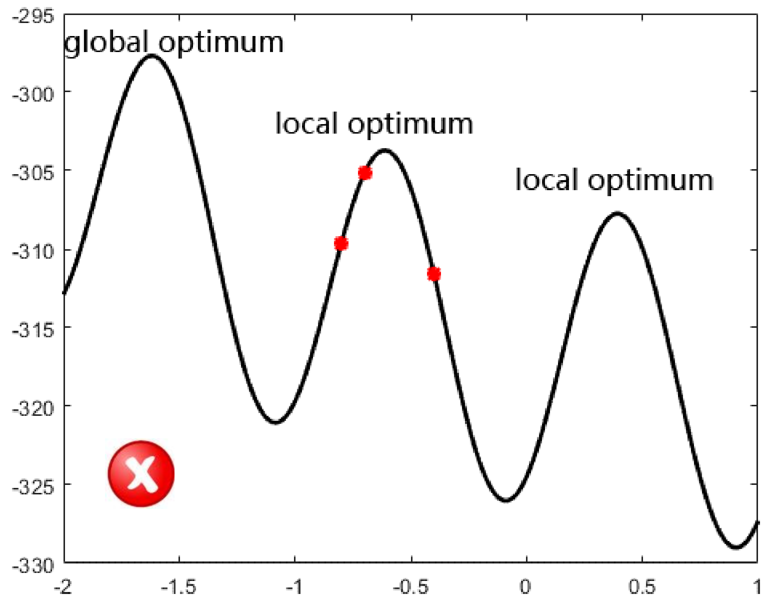


$$\mathcal{J} = \int f(\boldsymbol{x})p(\boldsymbol{x}|\boldsymbol{\theta}_1)\mathrm{d}\boldsymbol{x}$$



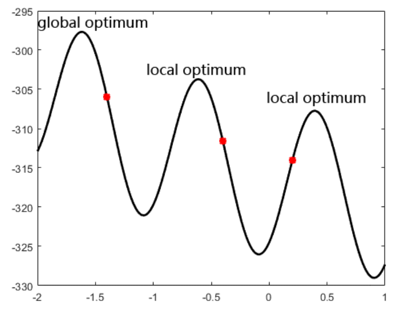
Why "Population" could be Beneficial?

- Smoothing cannot solve everything.
- Intuitively, population is not beneficial in the following case



Making Population Even More Beneficial?

- Suppose the algorithm has converged to some (global or local) optimum



- If run the algorithm again, we hope the algorithm (i.e., the distribution corresponding to the final population) converge to a different optimum.

$$\mathcal{J} = \int f(\boldsymbol{x})p(\boldsymbol{x}|\boldsymbol{\theta}_1)\mathrm{d}\boldsymbol{x} + \int f(\boldsymbol{x})p(\boldsymbol{x}|\boldsymbol{\theta}_2)\mathrm{d}\boldsymbol{x} + (-C(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) - C(\boldsymbol{\theta}_2, \boldsymbol{\theta}_1))$$

$$\mathcal{J} = \sum_{i=1}^{\lambda} \int f(\boldsymbol{x})p(\boldsymbol{x}|\boldsymbol{\theta}_i)\mathrm{d}\boldsymbol{x} + \sum_{i=1}^{\lambda}\sum_{j=1}^{\lambda}(-C(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j))$$

More Examples of Evolutionary Algorithms

Let's use the simple EA to maximise the function $f(x) = x^2$ with x in the *integer* interval $[0, 31]$, i.e., $x = 0, 1, \cdots, 30, 31$.

The first step of EA applications is *encoding* (i.e., the representation of chromosomes). We adopt binary representation for integers. Five bits are used to represent integers up to 31. Assume that the population size is 4.

1. Generate initial population at random, e.g., 01101, 11000, 01000, 10011. These are *chromosomes* or *genotypes*.
2. Calculate fitness value for each individual.
   (a) Decode the individual into an integer (called *phenotypes*),

$$01101 \rightarrow 13, 11000 \rightarrow 24, 01000 \rightarrow 8, 10011 \rightarrow 19;$$

More Examples of Evolutionary Algorithms

(b) Evaluate the fitness according to $f(x) = x^2$,

$$13 \longrightarrow 169, 24 \longrightarrow 576, 8 \longrightarrow 64, 19 \longrightarrow 361.$$

3. Select two individuals for crossover based on their fitness. If roulette-wheel selection is used, then

$$p_i = \frac{f_i}{\sum_j f_j}.$$

Two offspring are often produced and added to an intermediat population. Repeat this step until the intermediate population is filled. In our example,

$$p_1(13) = 169/1170 = 0.14 \quad p_2(24) = 576/1170 = 0.49$$
$$p_3(8) = 64/1170 = 0.06 \quad p_4(19) = 361.1170 = 0.31$$

Assume we have $crossover(01101, 11000)$ and $crossover(10011, 11000)$. We may obtain offspring 0110 0 and

More Examples of Evolutionary Algorithms

1100 1 from $crossover(01101, 11000)$ by choosing a random crossover point at 4, and obtain 10 000 and 11 011 from $crossover(10011, 11000)$ by choosing a random crossover point at 2. Now the intermediate population is

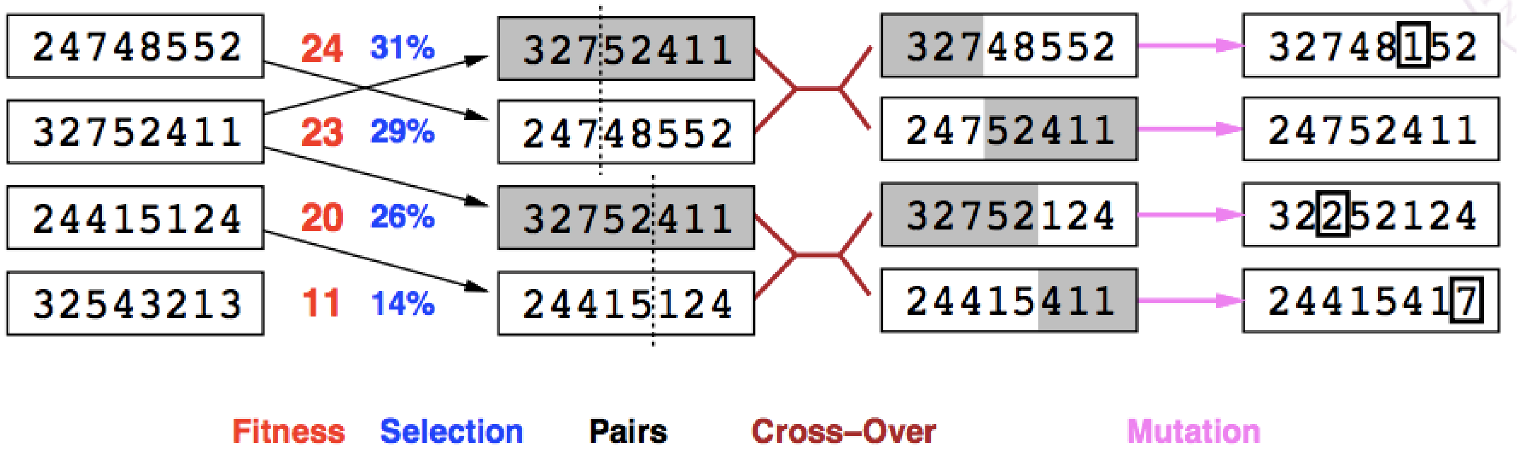
$$01100, 11001, 10000, 11011$$

4. Apply mutation to individuals in the intermediate population with a *small* probability. A simple mutation is bit-flipping. For example, we may have the following new population $P(1)$ after random mutation:

$$0110\mathbf{1}, 11001, \mathbf{0}0000, 11011$$

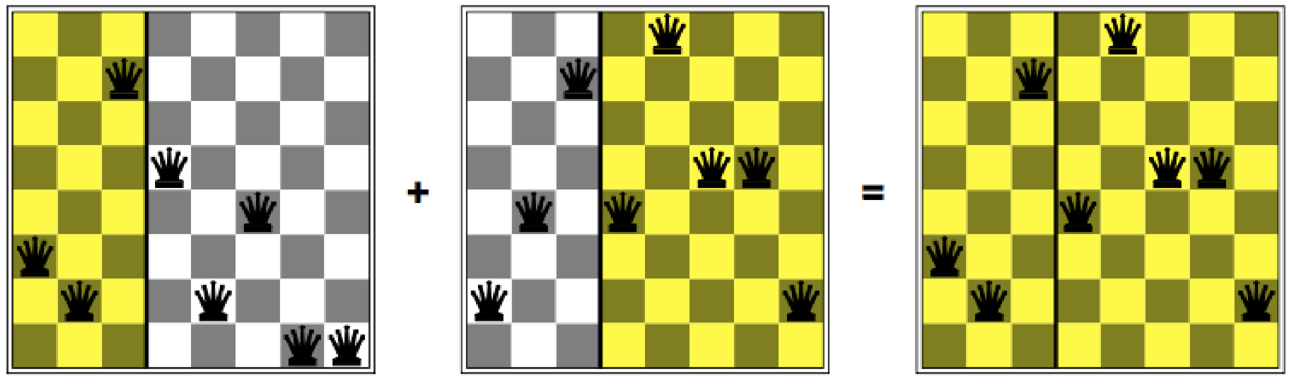
5. Goto Step 2 if not stop.

More Examples of Evolutionary Algorithms



GAs require states encoded as strings (GPs use programs)

Crossover helps **iff substrings are meaningful components**



Some applications of EAs

- High-speed train head design (Japan)

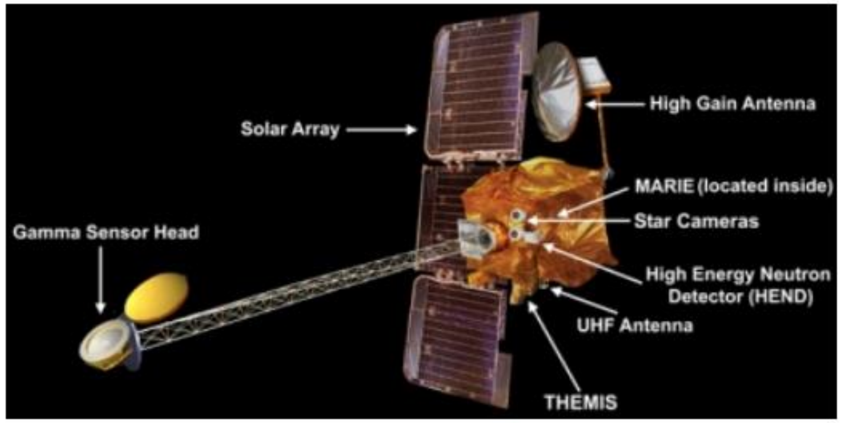
Series 700 Designed by human

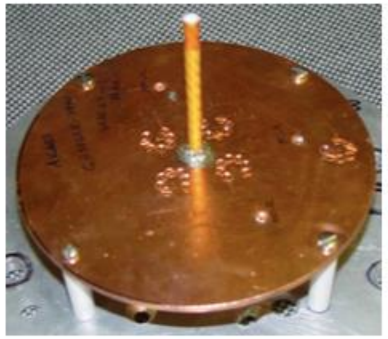
Series N700 Designed by EA

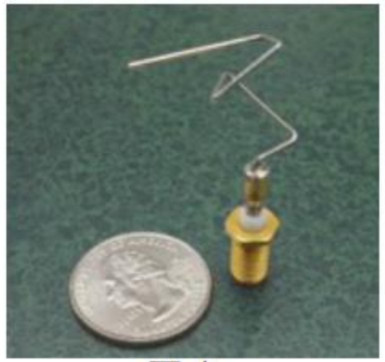
- Save 19% energy…30 increase in the output…

Some applications of EAs

- X-Band Antenna Design (NASA, US)



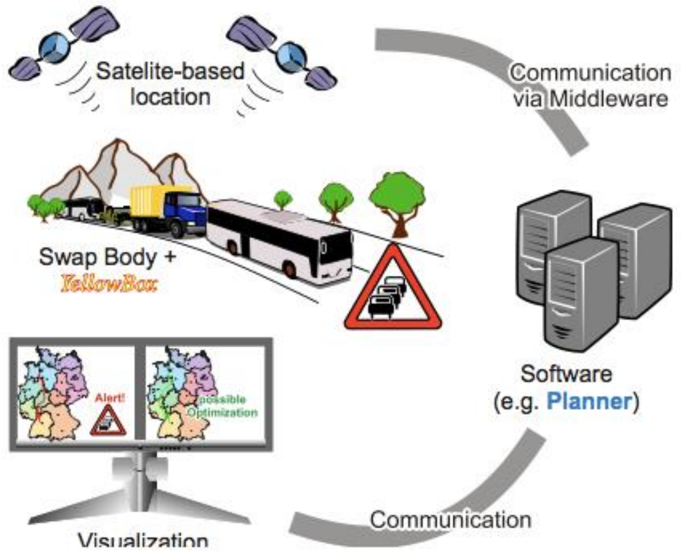

Human


EA

- Increase efficiency from 38% to 93%

Some applications of EAs

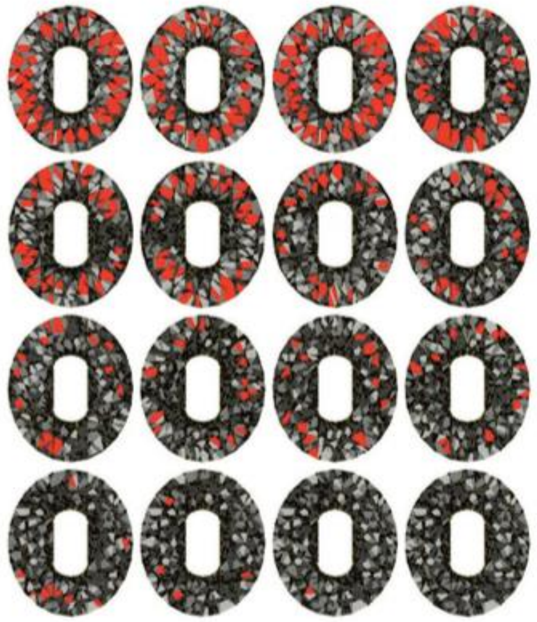
- Transportation Planning System (DHL, Germany)



- Save 9% of the transportation costs.

Some applications of EAs

- Birds Nest (China & Switzerland)



- The irregular ordering of the beams poses an insoluble problem for the then-current CAD tools.

Remarks

- SA, TS, and PBS are all search frameworks (like A*). For a specific problem, heuristic is incorporated in forms of search operators.

- For some special problem class, e.g., linear programming, quadratic programing or convex optimization problems in general, we have more mature tools (many textbooks on these issues).

III. More complex real-world search scenarios

More complex real-world search scenarios

- **Nondeterministic actions: agents do not act in a deterministic way.**

- **Partially observable states: You are not fully aware where you are.**

- **Online search: You can only learn the possible actions and states as moving.**

- **Approaches to the above scenarios are typically variants of the search methods we described so far.**

To be continued