Introduction to Artificial Intelligence– CII2M3

Genetic Algorithm

ADF

Outline

- What is a Solution?
- Metaheuristic Search
  - Evolutionary Algorithms

Outline

- Genetic Algorithm
  - Chromosome
  - Fitness
  - Parent Selection
  - Crossover and Mutation
  - Survivor Selection

Solution in Genetic Algorithm

Solution in Genetic Algorithm

- Genetic Algorithm models its solution as a string/array
- Default form is 1-dimensional Array
  - But not limited to it

Solution in Genetic Algorithm

- Also called a Chromosome
- Also called an Individual Solution
- Engineering skills are needed in order to Design a Solution
- Occam’s Razor: The simpler the better

Solution Design Example

Travelling Salesman Problem

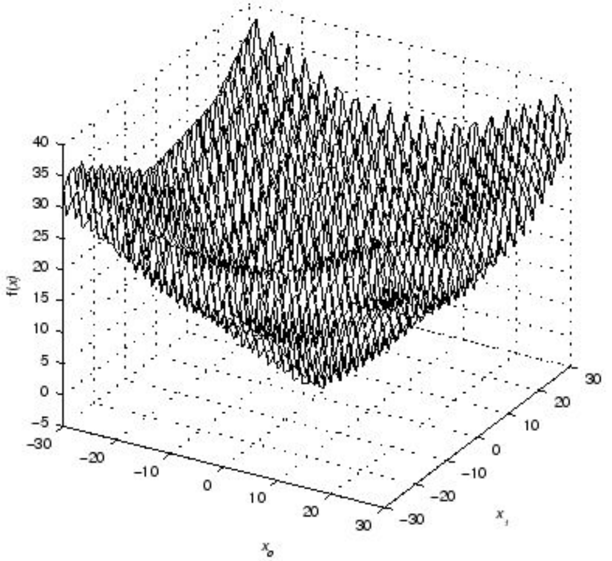
Travelling Salesman Problem

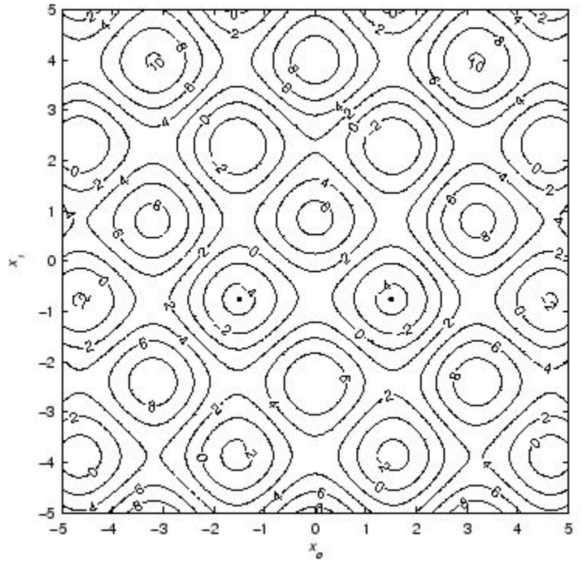
- If we have N city to visit
  - City ID: 1, 2, …, N
  - Assume All city is fully connected

- Then a solution is:
  - Array $[1 \times N]$ of Integer
  - Value: the order of visit

Mathematical optimization

- $$f(\vec{x}) = \sum_{i=0}^{D-1} \left(e^{-0.2} \sqrt{x_i^2 + x_{i+1}^2} + 3(\cos(2x_i) + \sin(2x_{i+1})) \right)$$





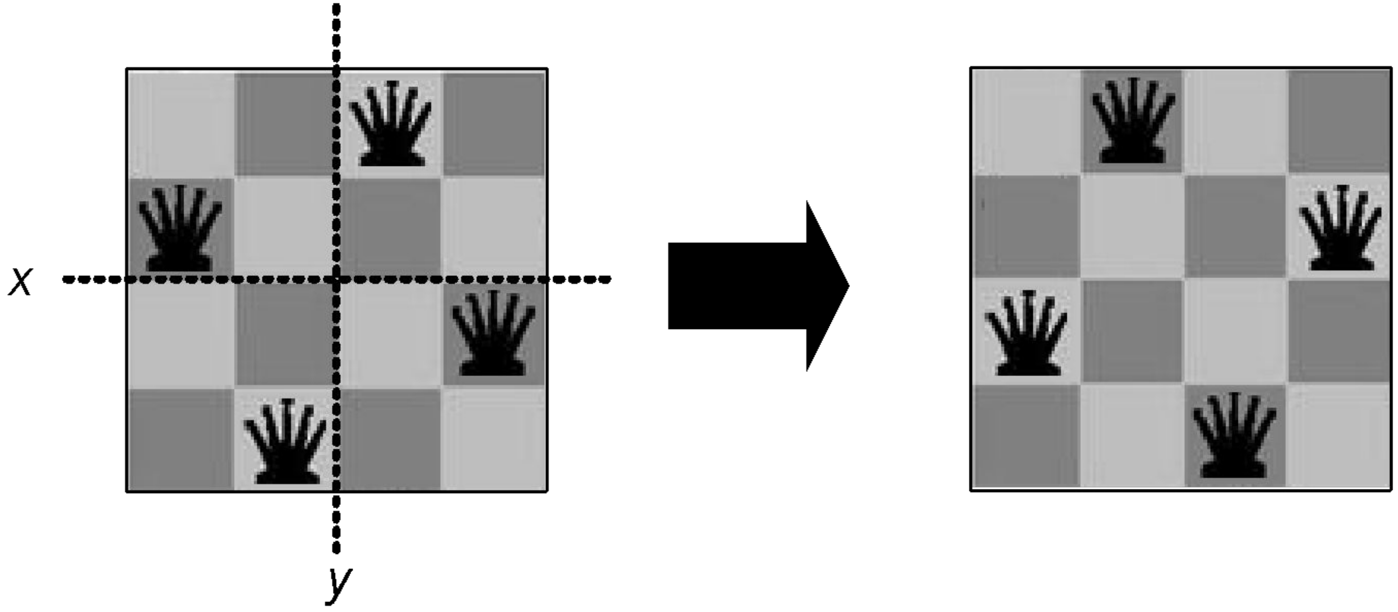
Mathematical optimization

- If we have a function to optimize
  - $f(x_1, x_2, \ldots, x_N)$
- Then a solution is:
  - Array $[1 \times N]$ of Float
  - Value: the order of visit

Knapsack Problem

The Water Jug Problem

4-Queens Problem



Solusi 1

Solusi 2

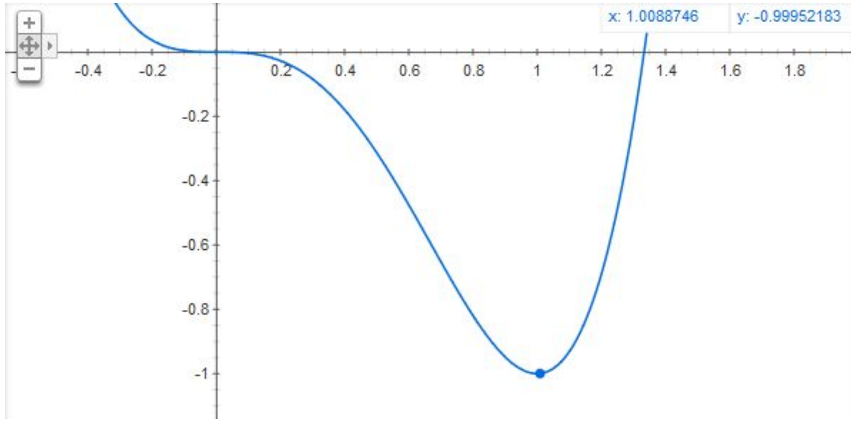
Metaheuristic Search

Metaheuristic Search

- A higher-level procedure or heuristic designed to find, generate, or select a heuristic (partial search algorithm) that may provide a sufficiently good solution to an optimization problem,
  - especially with incomplete or imperfect information or limited computation capacity
- Sample a set of solutions which is too large to be completely sampled

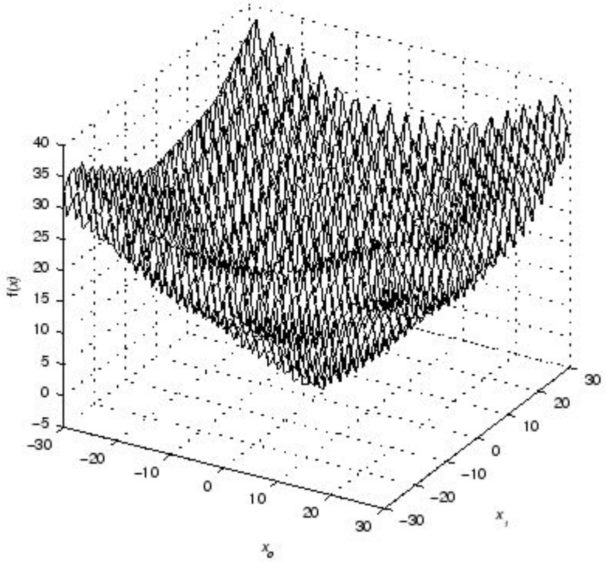
Mathematical optimization

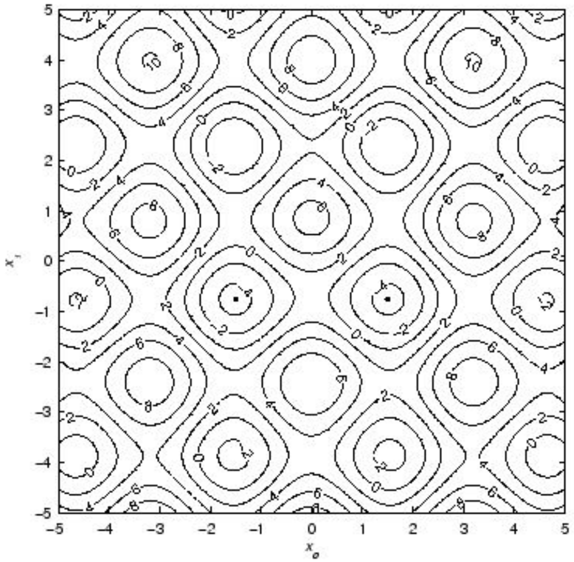
- Minimizing a function $y = 3x^4 - 4x^3$
- Searching approach:
  - Find an x that minimize y
- Mathematical approach:
  - First derivative
  - $\frac{dy}{dx} = 12x^3 - 12x^2$, yield $x = 1$



Mathematical optimization

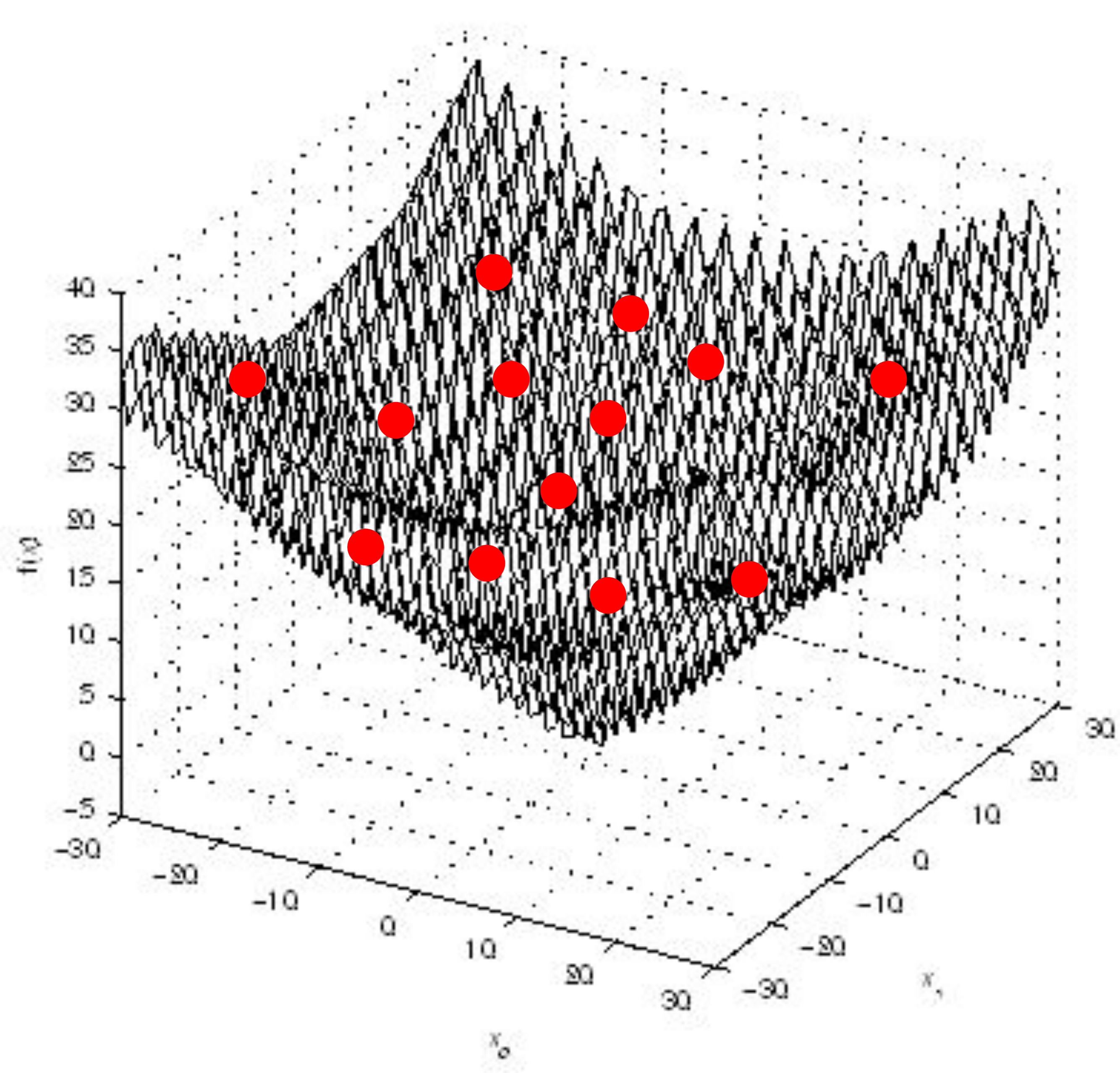
- $$f(\vec{x}) = \sum_{i=0}^{D-1} \left(e^{-0.2} \sqrt{x_i^2 + x_{i+1}^2} + 3(\cos(2x_i) + \sin(2x_{i+1})) \right)$$



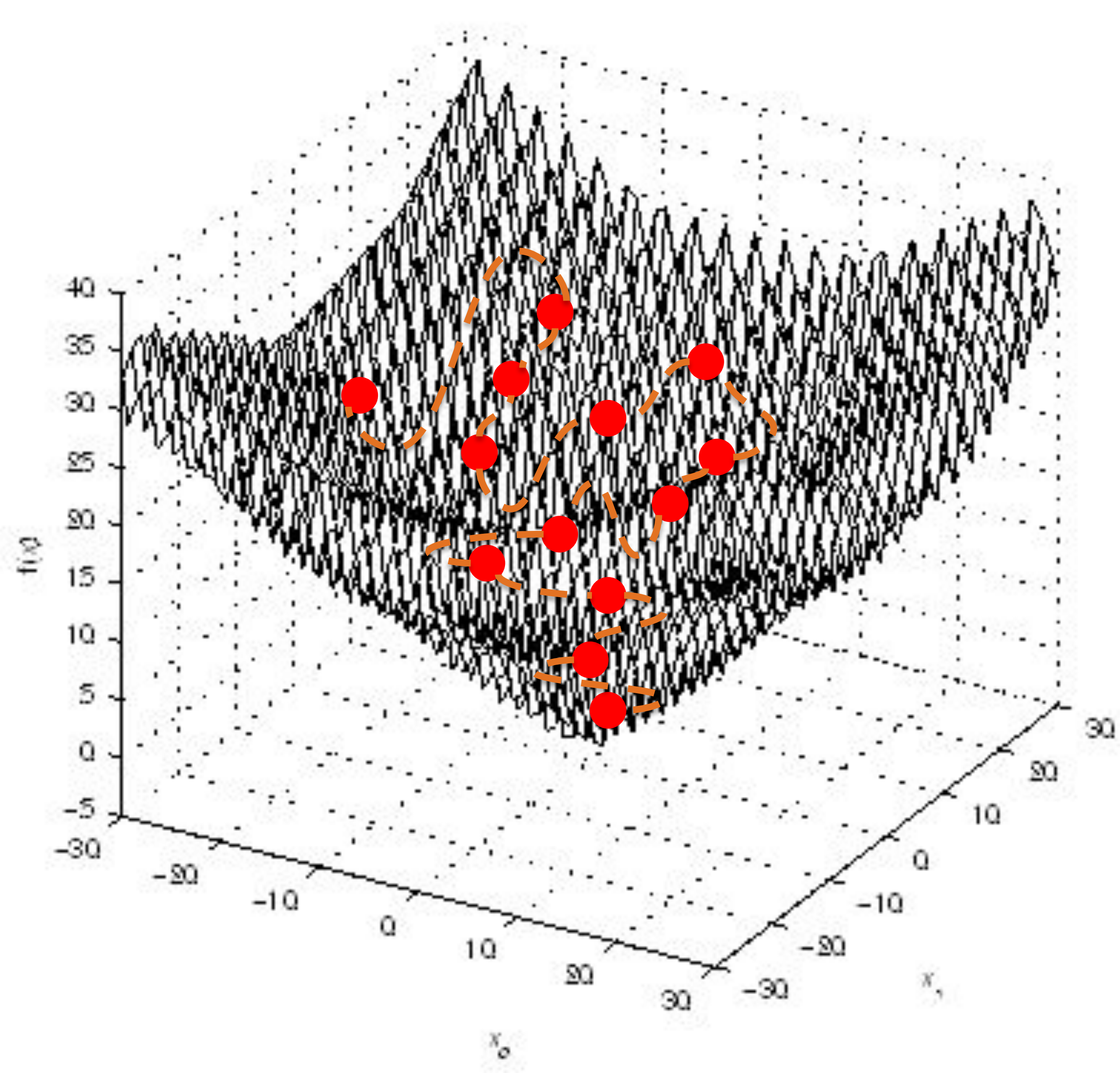


First derivative? **Really Difficult**

Random Search

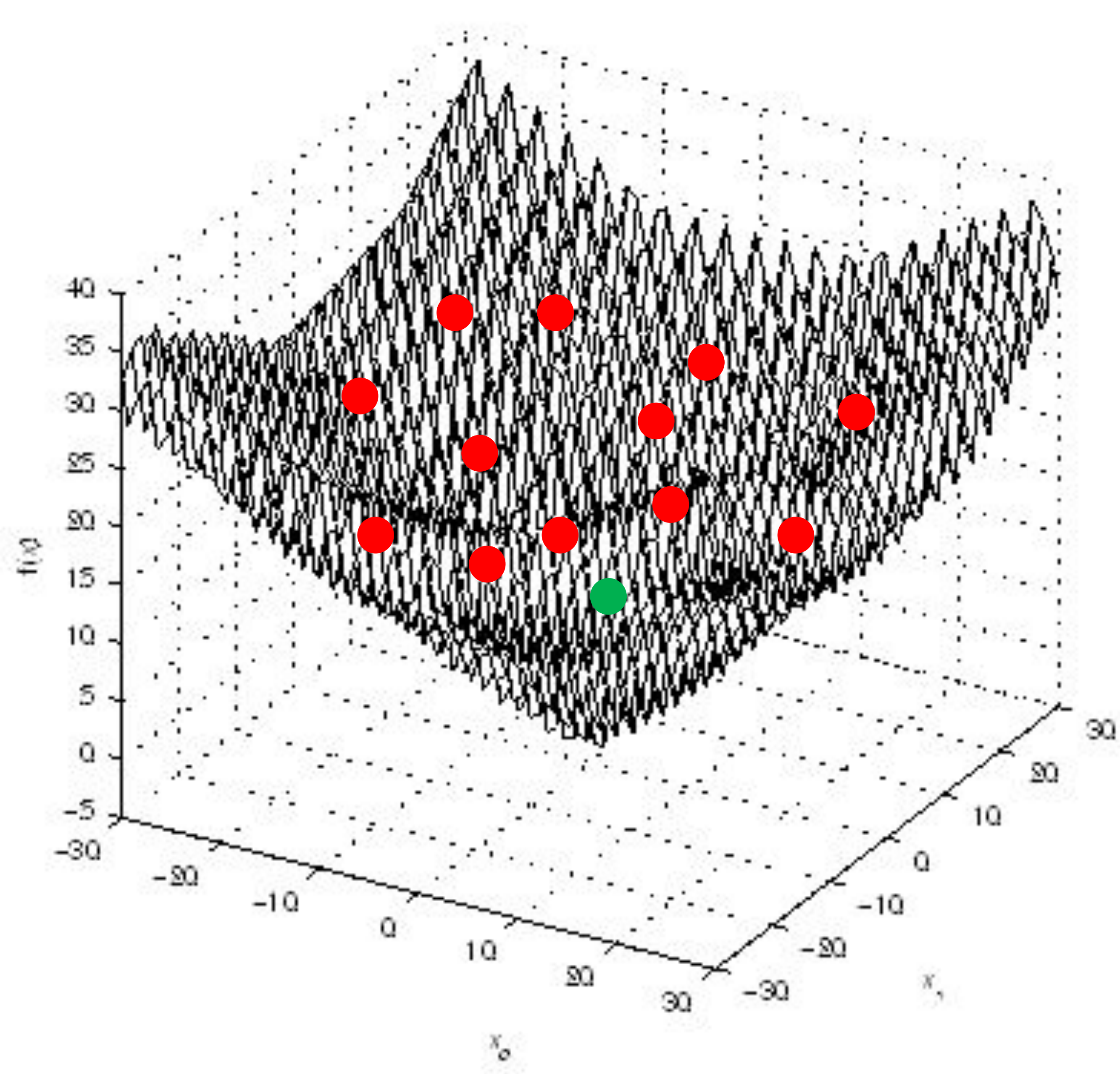


Simulated Annealing



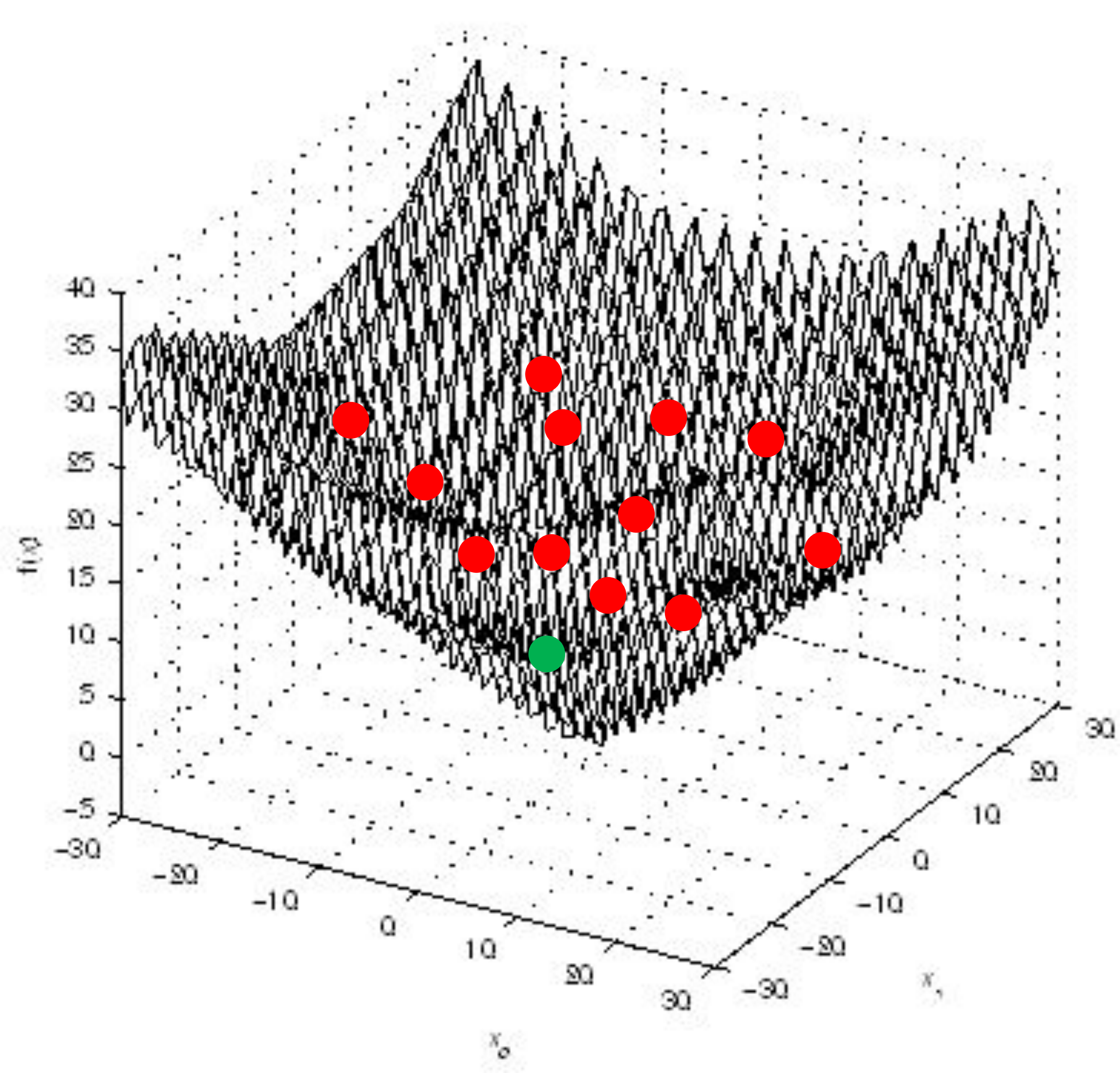
Genetic Algorithm

Generation=1



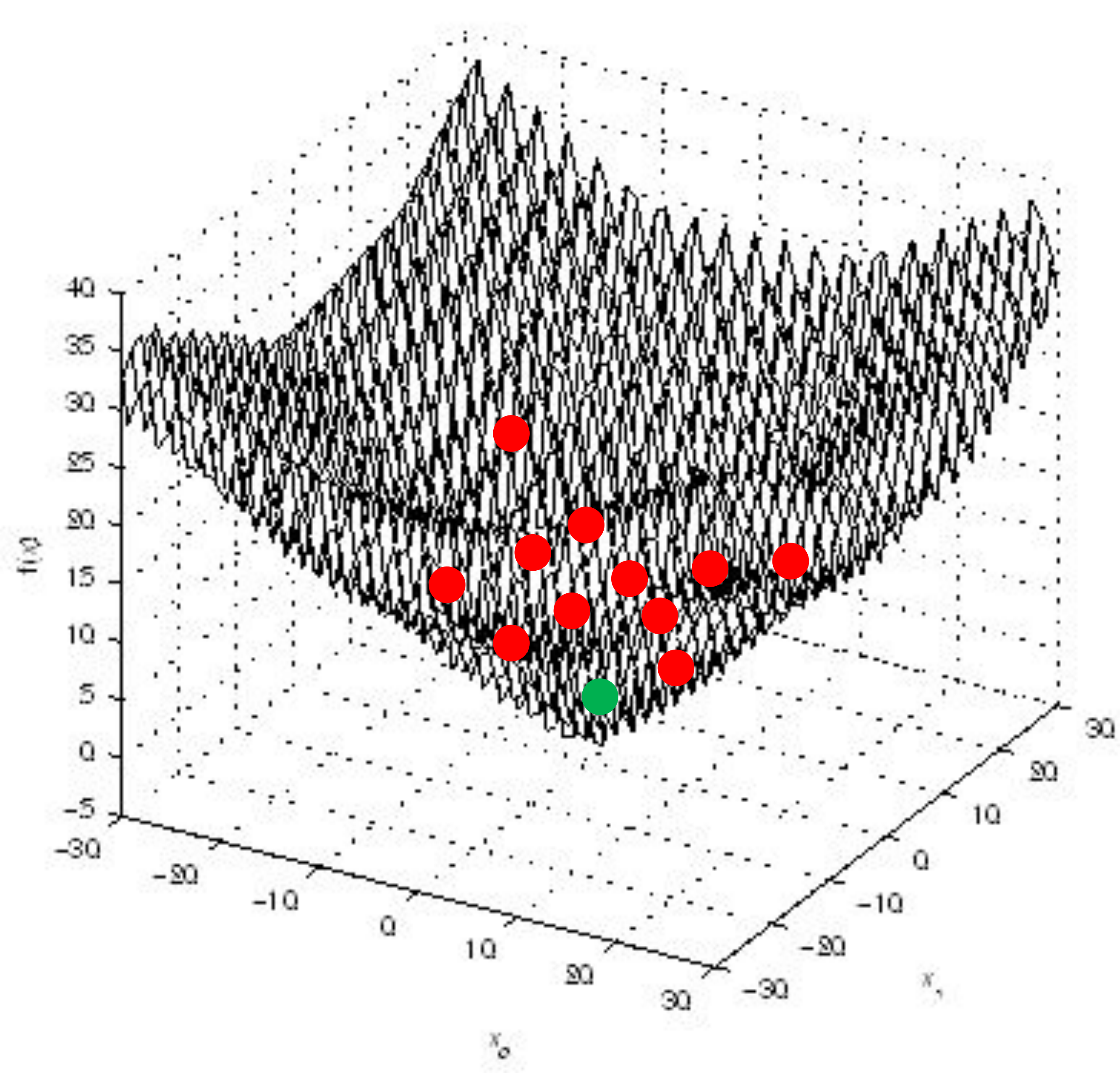
Genetic Algorithm

Generation=2



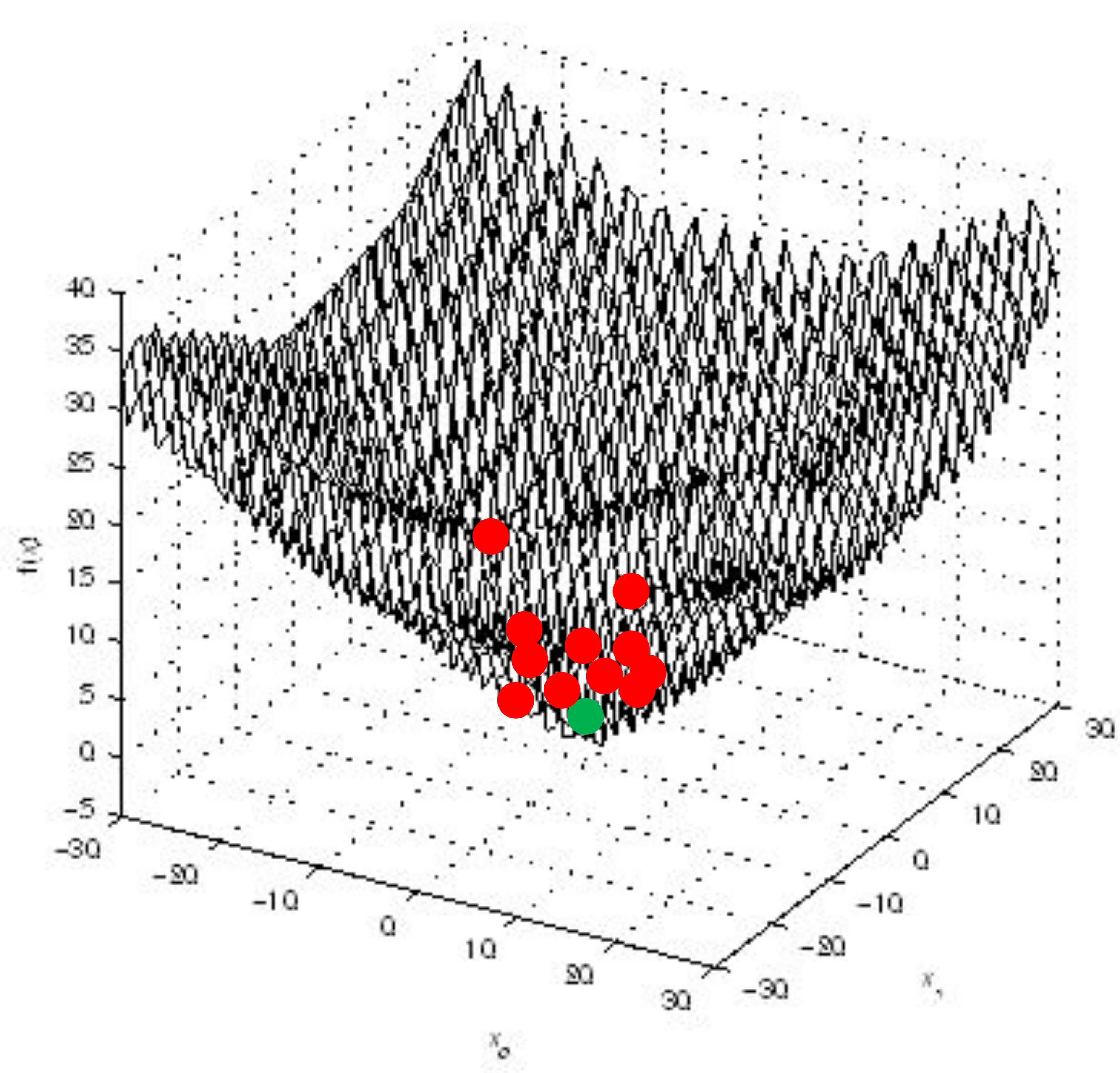
Genetic Algorithm

Generation=5



Genetic Algorithm

Generation=10

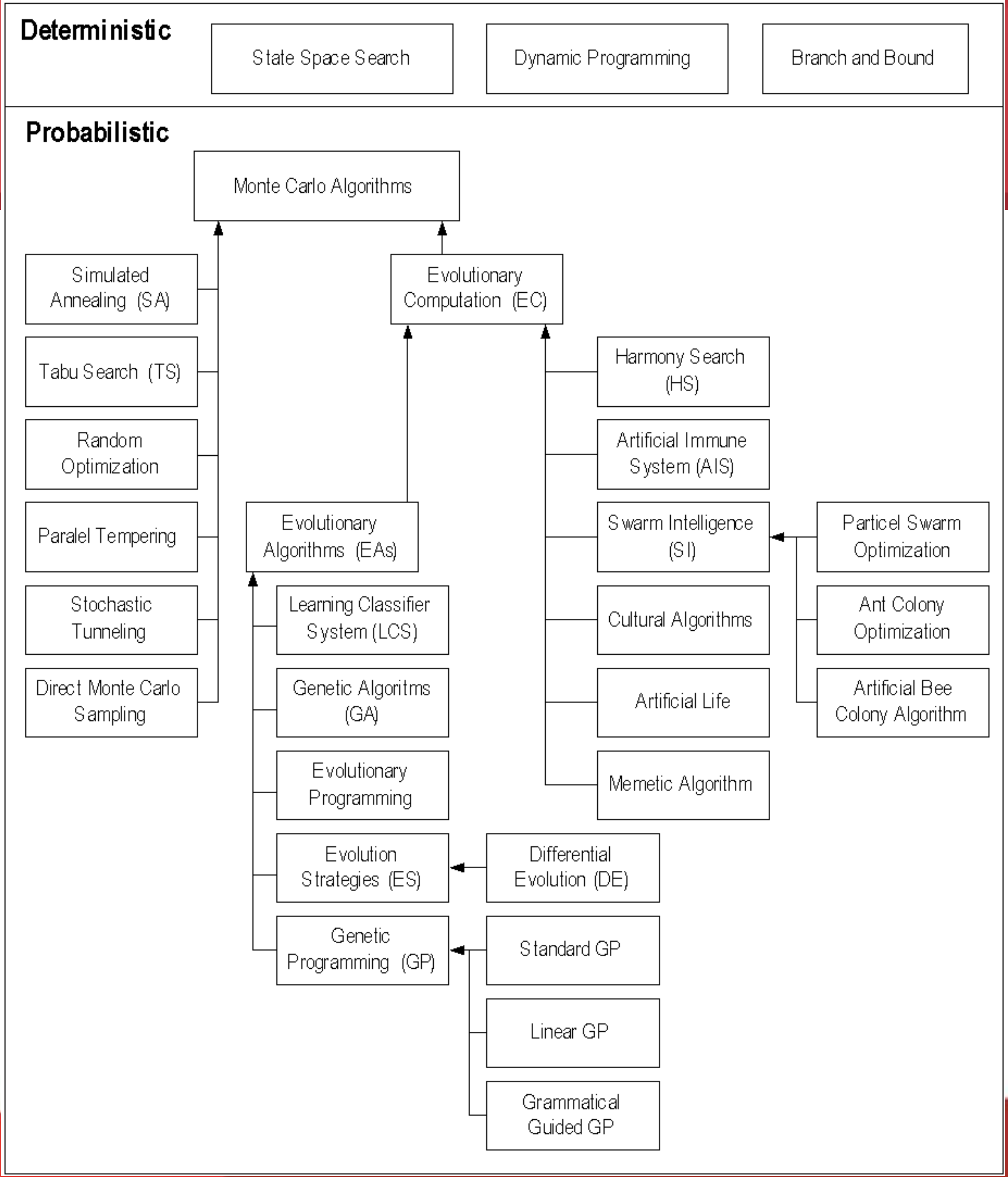


Time to Complete

- Example: TSP 100 locations, 8 hours work day

	Manual thinking	**Software A (Dijkstra)**	**Software B (Genetic Algorithm)**
Running time (algorithm)	0	2 hours	10 minutes
Time to complete the route resulted	11 hours	7 hours	7 hours and 20 minutes
Total time spent	11 hours	9 hours	7.5 hours
overtime	**3 hours**	**1 hours**	**-**

Optimization Algorithm



Evolutionary Computation

Evolutionary Computation

- an abstraction from the theory of biological evolution that is used to create optimization procedures or methodologies,
- usually implemented on computers, that are used to solve problems

Evolutionary Algorithms

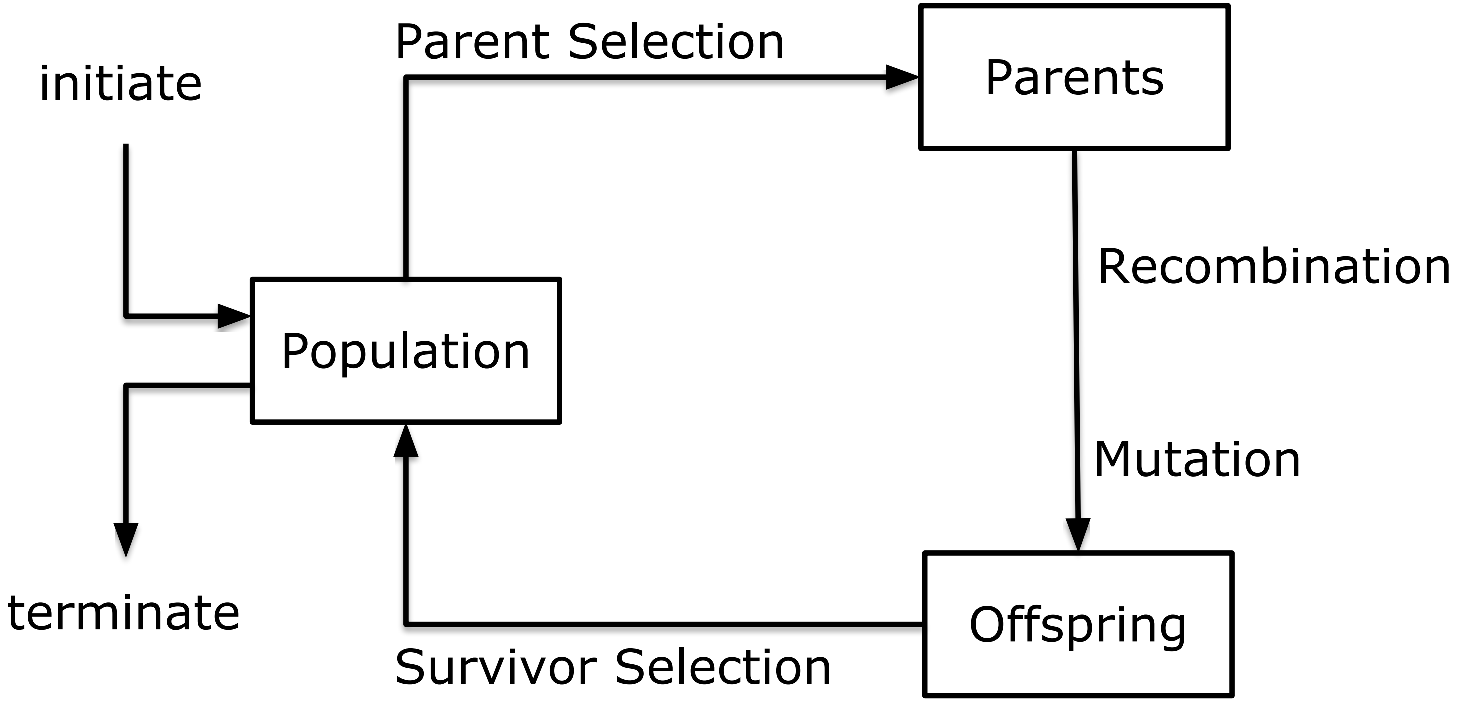
- generic, population-based meta-heuristic optimization algorithms that use biology-inspired mechanisms like mutation, crossover, natural selection and survival of the fittest.

- EAs are algorithms which implement the EC abstractions

Evolutionary Algorithms

- Genetic Algorithms (GA): binary strings
- Evolution Strategies (ES): real-valued vectors
- Evolutionary Programming (EP): finite state machines
- Genetic Programming (GP): LISP trees
- Differential Evolution (DE)
- Grammatical Evolution (GE)

EAs General Scheme

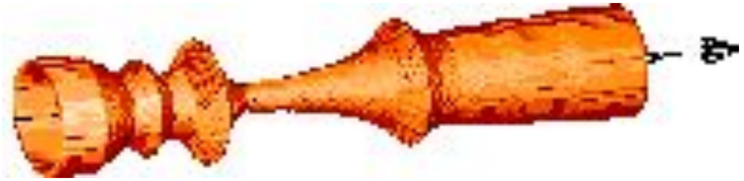


EC Applications: Optimization

- Scheduling
  - Course scheduling, company/project time table, hospital scheduling, etc.
- Knapsack Problem, Packaging,
- Cutting Stock Problem
- Design Simulation
- Etc.

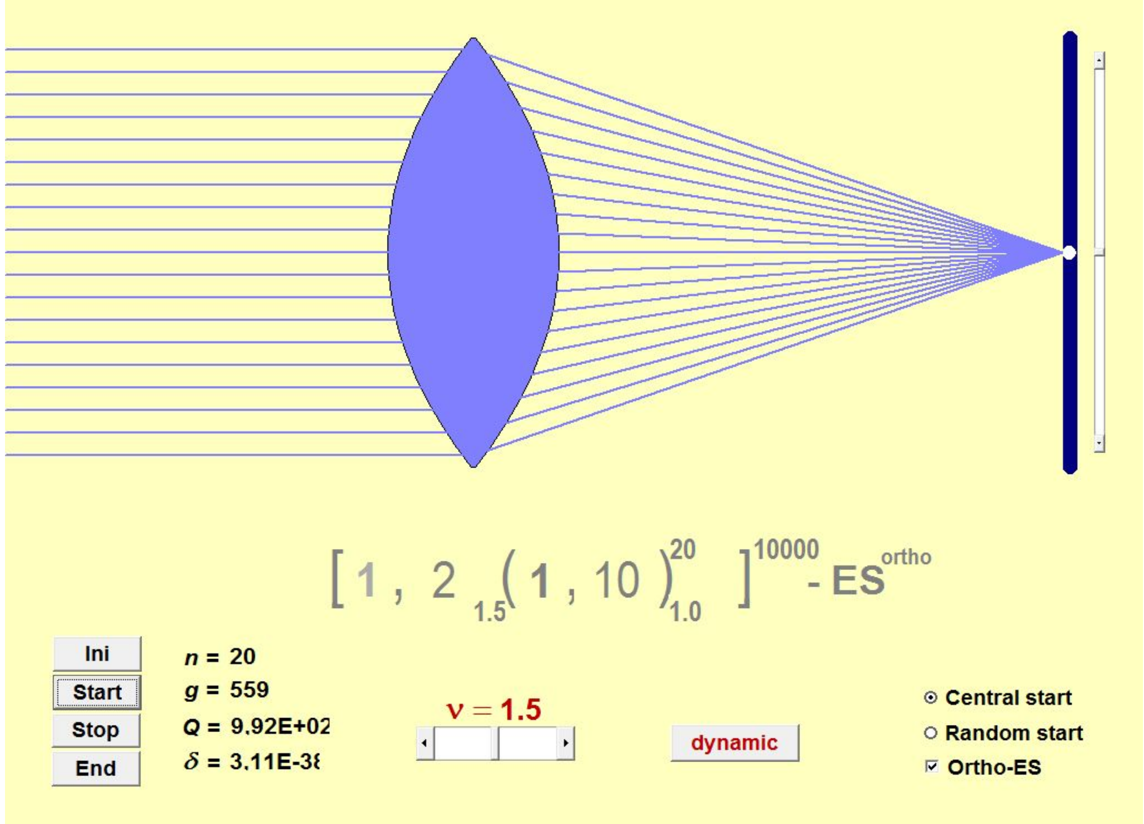
Jet Nozzle Design

Startin
g



Resultin
g

Optic Lenses Design



Swarm Intelligence

Genetic Algorithms

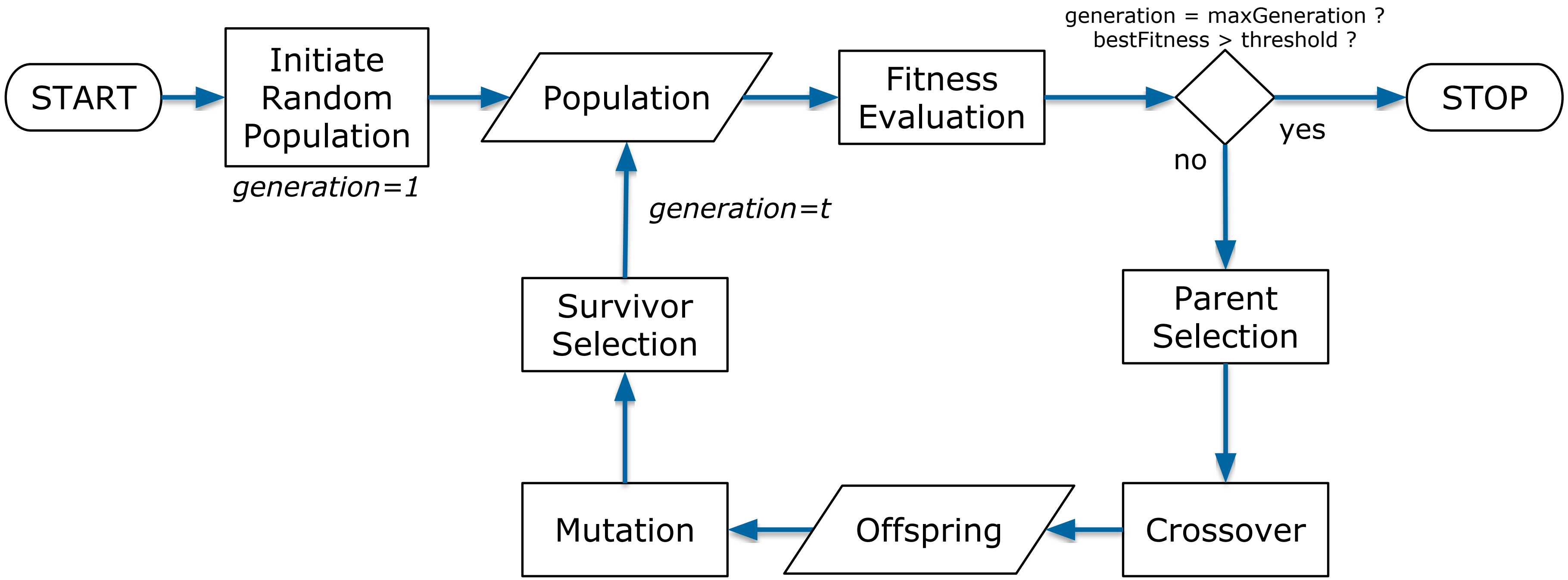
Genetic Algorithms

- a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover
- Originally developed by John Holland (1975)

Properties of Genetic Algorithms

- Individual - Any possible solution
- Population - Group of all individuals
- Search Space - All possible solutions to the problem
- Chromosome - Blueprint for an individual
- Fitness – quality of a solution
- Recombination - decomposes two distinct solutions and then randomly mixes their parts to form novel solutions
- Mutation - randomly perturbs a candidate solution

Genetic Algorithms



Chromosome

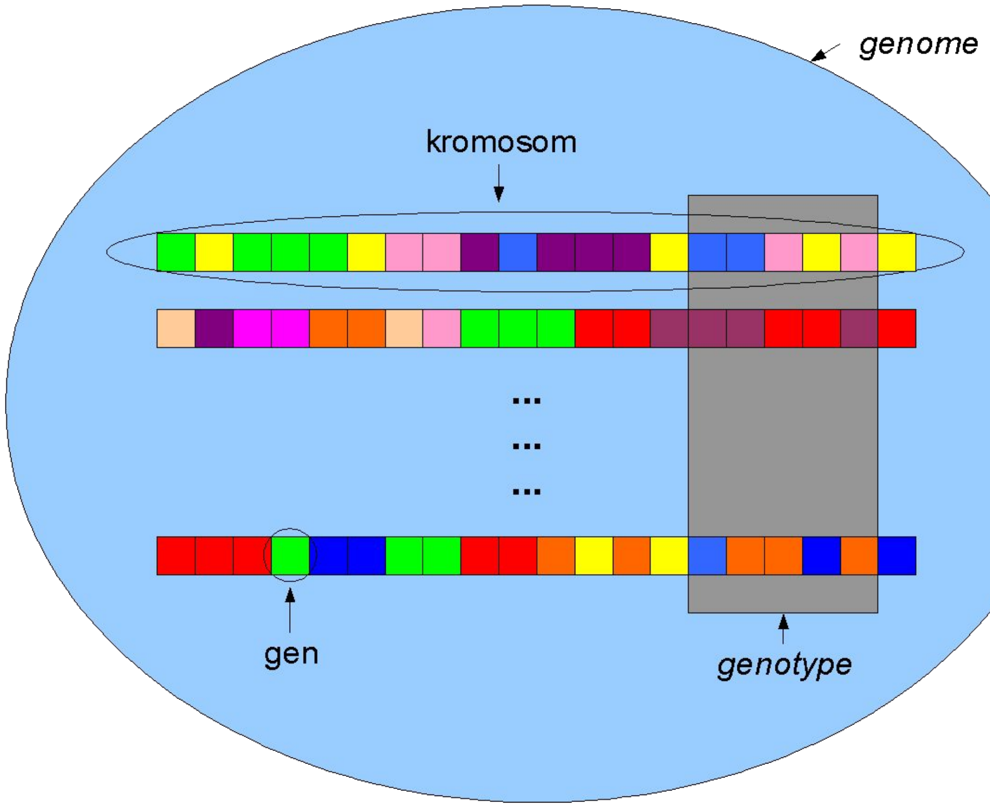
Representing Solution

- Design what is a solution, and how to measure its quality
- Example
  - TSP Problem
    - Solution: List of visited city
    - Quality: Cost to visit all city using the provided list
  - Knapsack Problem
    - Solution: list of selected goods
    - Quality: value of selected goods, overweight or not
  - Minimizing/Maximizing a Function
    - Solution: $x_1, x_2, \ldots x_n$

Chromosome

- Blueprint (a set of parameters) which define a proposed solution to the problem
- Represented in a string
  - Traditionally as binary, but other encodings such as integer or real are also possible
- The evolution usually starts from a population of randomly generated individuals

Chromosome



Chromosome Example

- Problem:

  Find x_1 and x_2 that minimize function $f(x_1, x_2) = x_1^3 + \frac{1}{3}x_2^2$ in range [-2, 3]

- Individual Representation
  - Integer Value, Real Value
  - Binary Encoding (to represents Integer)
  - Integer Encoding (to represents Real)
  - Real Encoding (to represents Real)
  - Etc.

Chromosome Example

Genotype	Phenotype
Integer Value | -1 | 2 |	$x_1 = -1$ $x_2 = 2$
Real Value | -1.15 | 1.89 |	$x_1 = -1.15$ $x_2 = 1.89$

Chromosome Example

Genotype	Phenotype
Binary Encoding using 3 bits (3 gens) | 0 | 1 | 0 | 1 | 1 | 0 | $x = r_{min} + \frac{r_{max} - r_{min}}{\Sigma_{i=1}^{N} 2^{-i}} (g_1 * 2^{-1} + g_1 * 2^{-2} + \cdots + g_N * 2^{-N})$ $x_1 = -2 + \frac{3-(-2)}{(2^{-1} + 2^{-2} + 2^{-3})} (0 * 2^{-1} + 1 * 2^{-2} + 0 * 2^{-3})$ $x_2 = -2 + \frac{3-(-2)}{(2^{-1} + 2^{-2} + 2^{-3})} (1 * 2^{-1} + 1 * 2^{-2} + 0 * 2^{-3})$	$x_1 = -0.57$ $x_2 = 2.29$

Chromosome Example

Genotype	Phenotype
Integer Encoding using 3 gens	
2 | 0 | 5 | 7 | 6 | 9	
$x = r_{min} + \frac{r_{max} - r_{min}}{\sum_{i=1}^{N} 9 * 10^{-i}}(g_1 * 10^{-1} + g_1 * 10^{-2} + \cdots + g_N * 10^{-N})$	$x_1 = -0.97$
$x_1 = -2 + \frac{3-(-2)}{9 * (10^{-1} + 10^{-2} + 10^{-3})}(2 * 10^{-1} + 0 * 10^{-2} + 5 * 10^{-3})$	$x_2 = 1.84$
$x_2 = -2 + \frac{3-(-2)}{9 * (10^{-1} + 10^{-2} + 10^{-3})}(7 * 10^{-1} + 6 * 10^{-2} + 9 * 10^{-3})$	

Chromosome Example

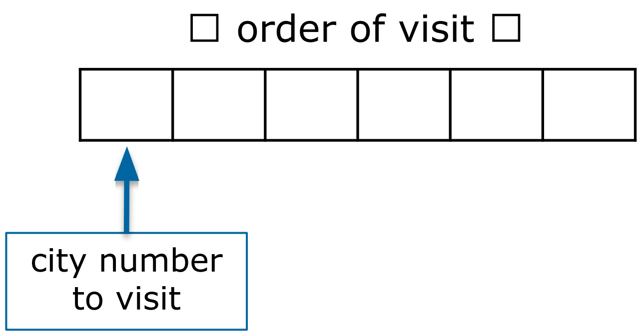
Genotype	Phenotype
Real Encoding using 3 gens 0.47 | 0.08 | 0.13 | 0.73 | 0.92 | 0.66 $x = r_{min} + \frac{r_{max} - r_{min}}{N}(g_1 + g_1 + \cdots + g_N)$ $x_1 = -2 + \frac{3-(-2)}{3}(0.47 + 0.08 + 0.13)$ $x_2 = -2 + \frac{3-(-2)}{3}(0.73 + 0.92 + 0.66)$	$x_1 = -0.87$ $x_2 = 1.85$

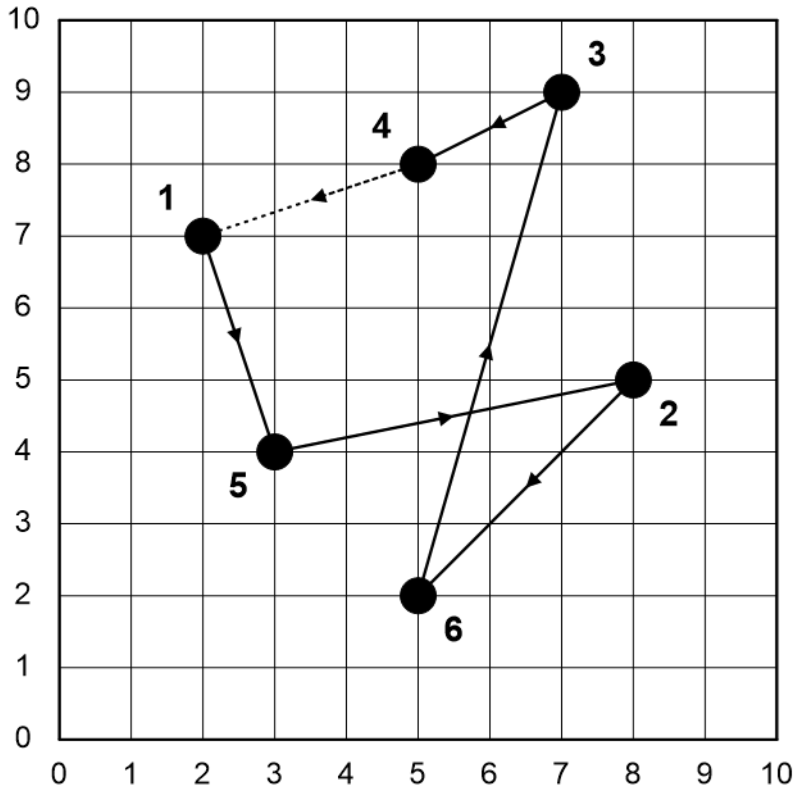
Chromosome Example

- For optimizing (real) values, the best practice is to encode the phenotypes into a much longer genotypes (chromosomes)

Chromosome Example

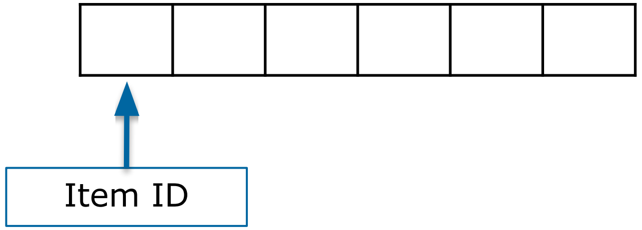
- Traveling Salesman Problem
- Individual Representation
  - Ordered chromosome
  - Permutation





Chromosome Example

- Knapsack Problem
  - Bag capacity: 25
- Individual Representation
  - Integer, non repeatable
  - Except 0



ID	Weight	Value
1	12	10
2	5	4
3	4	3
4	9	2
5	7	5
6	11	10
7	15	20
8	6	7
9	3	1
10	2	2
0	0	0

Fitness Function

Objective Function

- Evaluation of a chromosome
- Values/properties of a solution represented by a chromosome
- A solution may have multiple objective functions designed
- Should consider all the existing constraints

Fitness Function

- Quality of a chromosome
  - A value to indicate the quality of a solution
- Survival of the fittest
  - The higher the fitness means the better the solution
  - Higher fitness should survive
  - Low fitness individual will perish

Fitness Function

- Characteristic
  - Clearly defined (from the objective function)
  - Should be sufficiently fast to compute
  - Should generate intuitive results.
  - The best/worst candidates should have best/worst score values

Fitness Function

- Maximizing the objective function

$$f = h \qquad\qquad f = h_1 + h_2 + \cdots h_n$$

- Minimizing the objective function

$$f = \frac{1}{h + a} \qquad\qquad f = \frac{1}{h_1 + h_2 + \cdots + h_n + a}$$

$$f = -h \qquad\qquad f = \frac{1}{h_1+a} + \frac{1}{h_2+a} + \cdots + \frac{1}{h_n+a}$$

- Or its combination

Fitness Function Example

- Traveling Salesman Problem
  - Read each gen pair
  - If there is an edge (connection) for each pair of node in gen, calculate the cost, $f = 1/h$
  - If there is any pair of node that is not connected, $f = 0$
- Knapsack Problem
  - Calculate the total weight, if the total < max, f = total value
  - If total > max, $f = 0$
- Function Optimization
  - Decode chromosome, if values are in boundaries, calculate the objective
  - If the values are outside the boundaries, $f = 0$

Parent Selection

Parent Selection

- Process to select individuals as parents to generate new offspring for the next generation
  - Individual solutions are selected through a fitness-based process

- Popular and well-studied methods:
  - Roulette Wheel Selection
  - Tournament Selection
  - [New] Roulette Wheel via Stochastic Acceptance

Roulette Wheel Selection

- Individuals are given a probability of being selected that is directly proportionate to their fitness

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}$$

- Two individuals are then chosen randomly based on these probabilities and produce offspring.
  - Stochastic, Time complexity $O(n)$ or $O(\log n)$

Roulette Wheel Selection

```
function RouletteWheelSelection (fitness)
        total = 0
        for indv = 1 to n do
                total += fitness(indv)

        r = random_uniform()
        indv = 1
        while( r > 0 ) do
                r -= fitness(indv)/total
                indv ++
        return indv
```

Chromosome	Fitness	
C1	2	0.25
C2	1	0.125
C3	1	0.125
C4	4	0.5
total	8	

random= 0.32

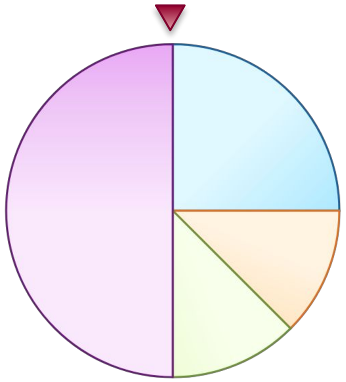


Illustration 1

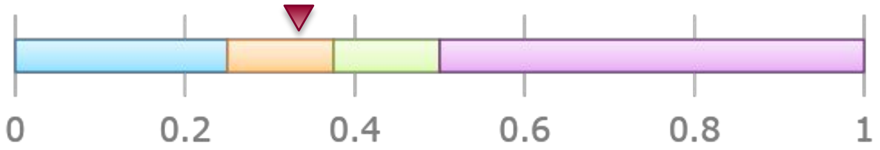


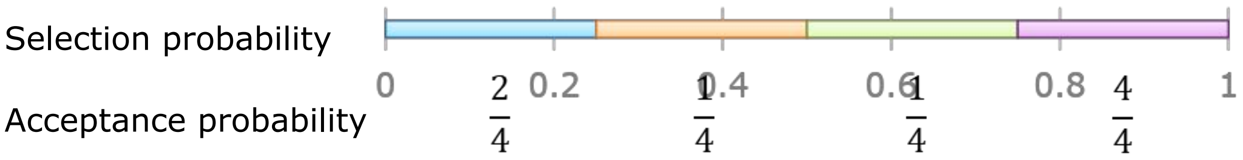
Illustration 2

Stochastic Roulette Wheel Selection

1. Select randomly one individual i with uniform probability $(1/N)$,
2. Accept selection with probability $\frac{fi}{fmax}$, where $fmax = \max\{f_i\}_{i=1}^{N}$ is the maximal fitness in the population
3. Otherwise, the procedure is repeated from step 1

```
function StochasticRouletteWheel(fitness)
        maxFitness = max(fitness)
        while(true)
                indv = random_uniform()*N
                r = random_uniform()
                if( r < fitness(indv)/maxFitness )
                        return indv
```

Chromosome	Fitness
C1	2
C2	1
C3	1
C4	4



Tournament Selection

- less stochastic noise,
- fast, easy to implement
- have a constant selection pressure

1. choose k (the tournament size) individuals from the population at random
2. choose the best individual from pool/tournament with probability p
3. choose the second-best individual with probability p*(1-p)
4. choose the third best individual with probability $p*((1-p)^2)$
5. and so on...

```
function tournamentSelection(pop, k):
   best = []
   for i=1 to k
      indv = pop[random(1, N)]
      if (best == []) or fitness(indv) > fitness(best)
         best = indv
   return best
```

Crossover and Mutation

Crossover or Recombination

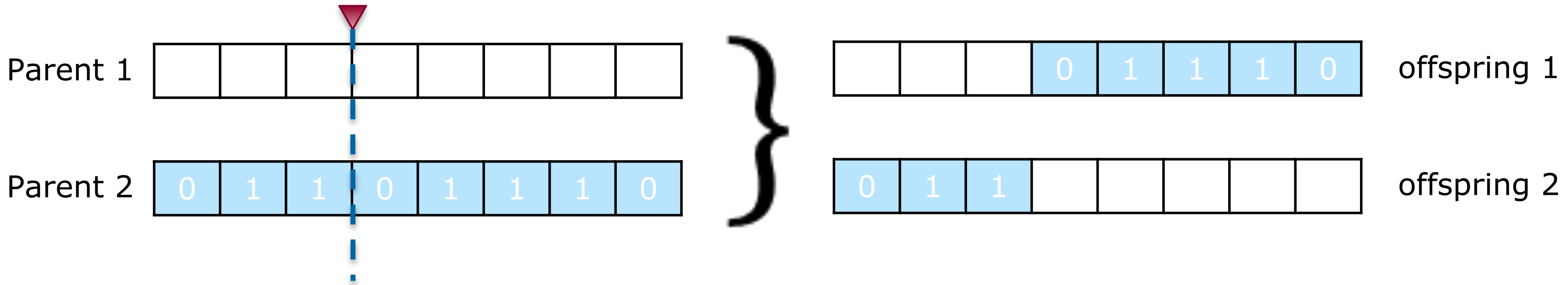
- Enables the evolutionary process to move toward promising regions of the search space
- Matches good parents' sub-solutions to construct better offspring
- Crossover does not always occur
  - based on a set probability
  - The probability of crossover is usually ~60% to 70%.

Crossover or Recombination

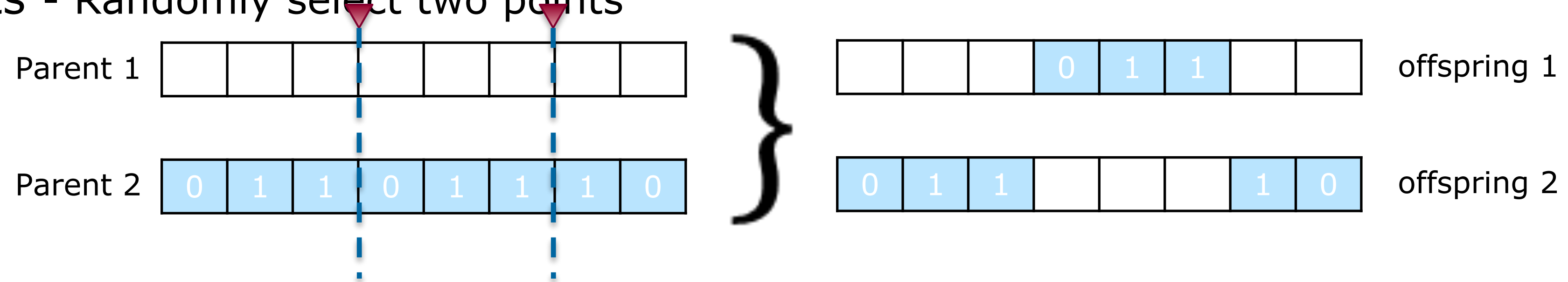
- General Schemes:
  - Single-Point, Two-Point, n-point crossover
  - Uniform crossover, Arithmetic crossover
  - etc.
- For ordered chromosome
  - Partially matched crossover
  - Cycle crossover
  - Order crossover
  - etc.

Crossover or Recombination

- Single-point - Randomly select a point



- Two-points - Randomly select two points



Mutation

- to simulate the effect of errors that happen with low probability during duplication

- small chance of mutation
  - loop through all the alleles
  - By a small probability (~1%), either change it by a small amount or replace it with a new value

Initial chromosome

Mutation process

1	0	1	1	0	0	0	1

Result:

Mutation

- General Schemes:
  - Bit-level mutation
  - Gene/allele-level mutation
  - Chromosome-level mutation
- For ordered chromosome
  - Swap mutation
  - Scramble mutation
  - etc.

Survivor Selection

Survivor Selection

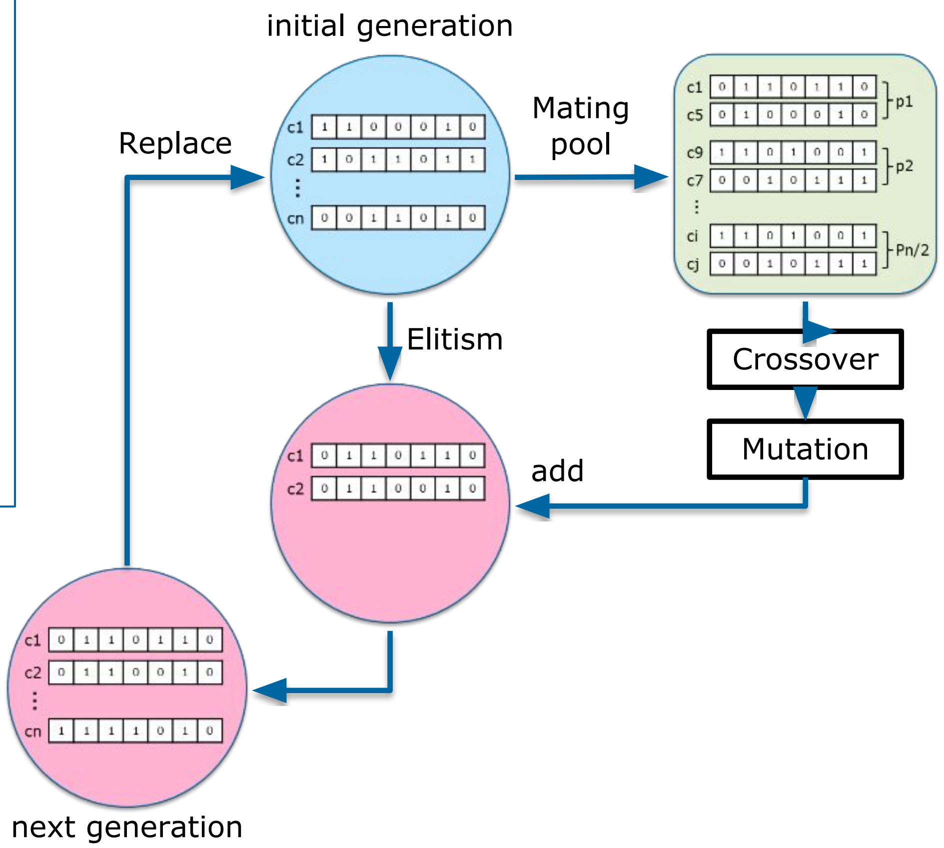
- Generational Replacement
  - generate n off-springs, where n is the population size, and the entire population is replaced by the new one at the end of the iteration
  - Must add a mechanism to ensure the best individual survives: **Elitism**
- Steady-State
  - generate one or two off-springs in each iteration and they replace one or two individuals from the population
  - Much simpler, but requires more generation to converge

Elitism

- Usually used in Generational Replacement type
- Copied the best chromosomes into the next generation once or twice
- Ensure the solution quality obtained will not decrease from one generation to the next

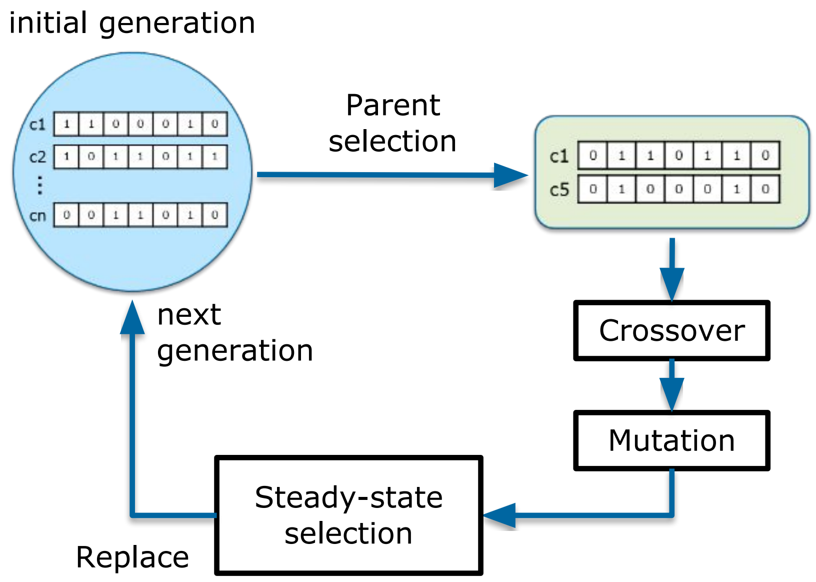
Generational Replacement

```
population = generatePopulation(N)
while stoppingCondition is not satisfied
        fitness = evaluate(population)
        newPopulation = elitism(population, fitness)
        while size(newPopulation) < N
                parent1, parent2 =
parentSelection(population)
                offspring = crossover(parent1, parent2,
pC)
                offspring = mutate(offspring, pM)
                newPopulation.add(offspring)
        end
        population = newPopulation
end
```



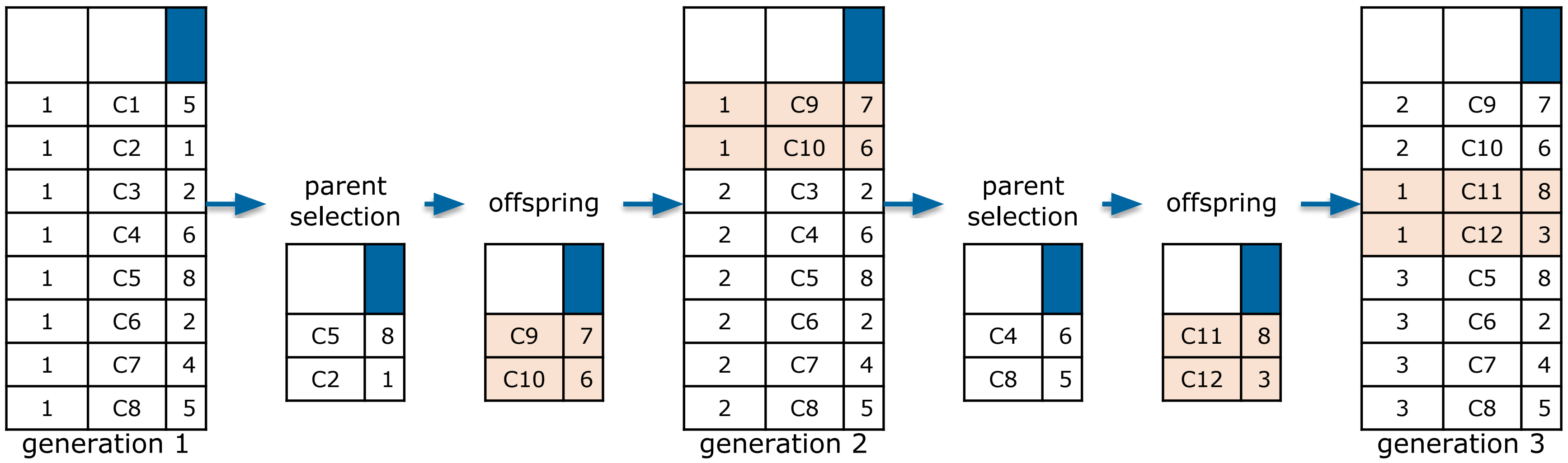
Steady-State

```
population = generatePopulation(N)
while stoppingCondition is not satisfied
        fitness = evaluate(population)
        parent1, parent2 = parentSelection(population)
        offspring = crossover(parent1, parent2, pC)
        offspring = mutate(offspring, pM)
        population = steadyState(population, offspring)
end
```



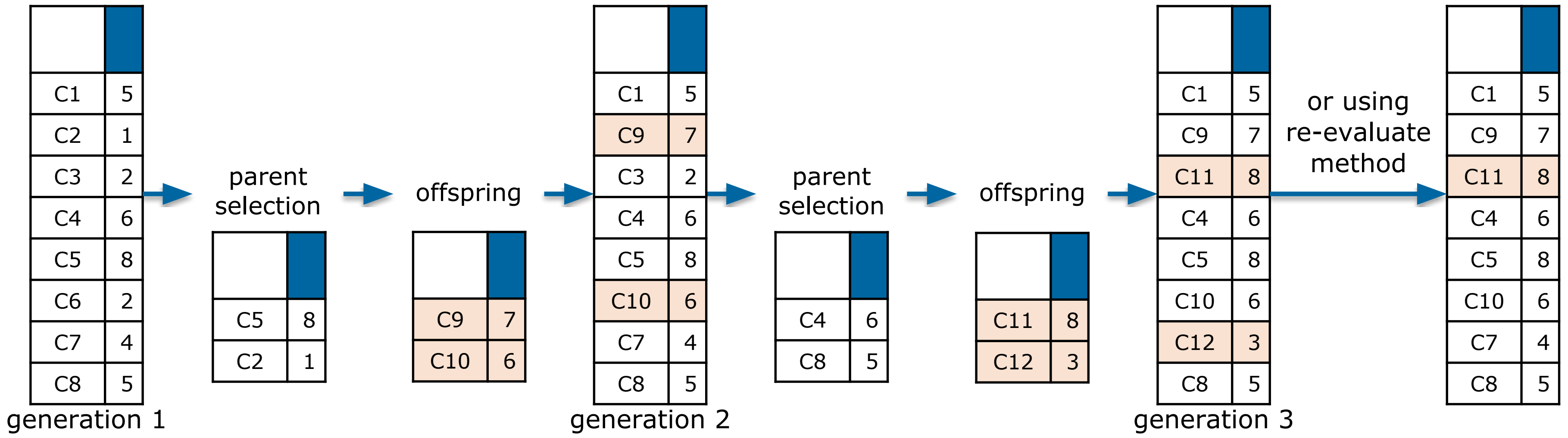
Steady-State procedures

- Age-based selection
  - Offspring replaces the oldest individual in population



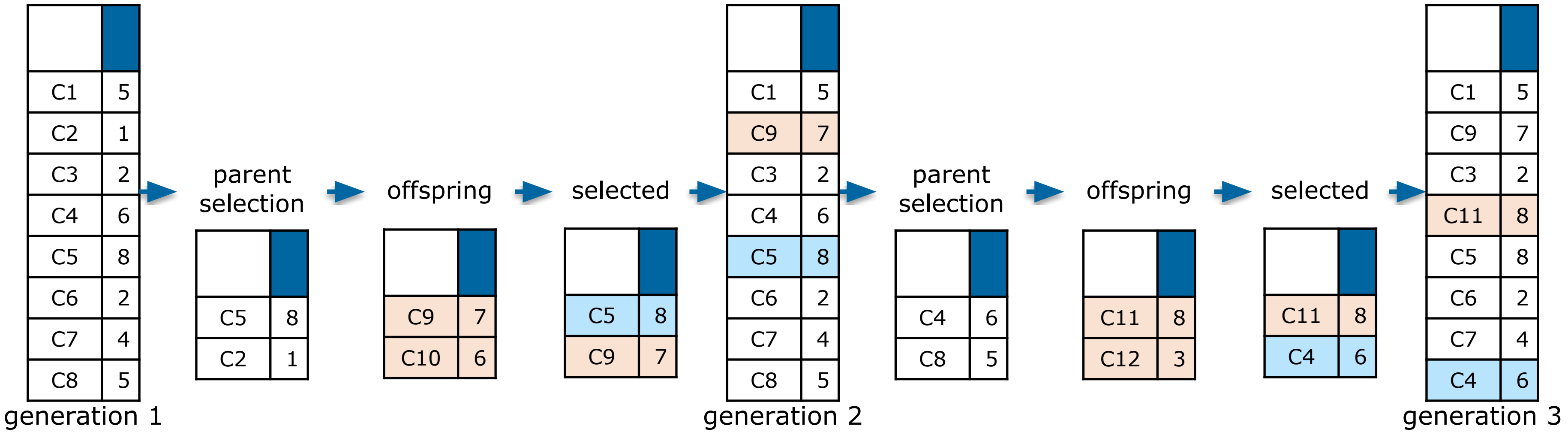
Steady-State procedures

- Fitness-based selection
  - Offspring replaces individual with the worst fitness value in population



Steady-State procedures

- Local Fitness-based selection
  - Select the best individuals form a pair of parents and its offspring to replace one or all the parents



Epilog

Stopping Criteria

- Max iteration (max generation)
- Time limit
- Fitness plateau
- Fitness threshold
- Population and Generational diversity

EAs are suitable for problems:

- Complex Problem
- Difficult to understand
- Can not use conventional methods
- Real time system
- The solution does not have to be the most optimal
- No prior knowledge
- No mathematical analysis is available

Problems Solved using EAs

- Traveling Salesman Problem
- Shortest Path
- Knapsack
- Cutting Stock Problem
- Scheduling
- Other Optimizing Problems

Choosing Searching Method

- How big is the problem space?
- What is the branching factor (b) and the depth of the solution (d)?
- How much processor and memory space available?
- Does the solution have to be optimal?
- Can the heuristic function be found/formulated?
- There is one goal or more?

Conclusion

- The methods included in the blind search require enormous memory to solve simple problems.
- With the current speed and limited computer memory, currently blind search is not yet possible to be implemented into the real world.
- The only method that might be used is Iterative Deepening Search (IDS) as it requires very little memory even though the processing time is very long.

Conclusion

- Among the search methods included in the heuristic search, A* is the best option when we can find a heuristic function for the problem to be solved.
- We can choose the A* variation that best fits the problem to be solved and the resources (time and memory) we have.
- When more than one type of heuristic function is found, choose the one closest to the actual cost.

Question?

Fakultas Informatika
School of Computing
Telkom University

THANK YOU