Local Search

10S3001 – Artificial Intelligence

Samuel I. G. Situmeang

Faculty of Informatics and Electrical Engineering

Objectives

Students are able:

- to explain the local search background,
- to contrast hill-climbing search and genetic algorithm, and
- to apply genetic algorithms to solve simple problems.

Siswa mampu:

- untuk menjelaskan latar belakang pencarian lokal,
- untuk membedakan *hill-climbing search* dan *genetic algorithm*, serta
- untuk menerapkan *genetic algorithm* untuk menyelesaikan persoalan sederhana.

3

Local Search

What is local search?
What are the local search strategies?
What is state space landscape for local search?

Local Search

- Search algorithms seen so far are designed to **explore search spaces systematically**.
- Problems: observable, deterministic, known environments where the solution is a sequence of actions.
- Real-World problems are more complex.
- When a goal is found, the path to that goal constitutes a solution to the problem. But, depending on the applications, the path may or may not matter.
- If the path does not matter/systematic search is not possible, then consider another class of algorithms.

Local Search

- In such cases, we can use iterative improvement algorithms, **Local Search**.
- Also useful in pure **optimization problems** where the goal is to find the best state according to an **optimization function**.
- **Examples**:
  - Integrated circuit design, telecommunications network optimization, etc.
  - N-puzzle or 8-queen: what matters is the final configuration of the puzzle, not the intermediary steps to reach it.

Local Search

- **Idea**: keep a single “current” state, and try to improve it.
- Move only to neighbors of that node.
- **Advantages**:
  1. No need to maintain a search tree.
  2. Use very little memory.
  3. Can often find good enough solutions in continuous or large state spaces.

Local Search Strategies

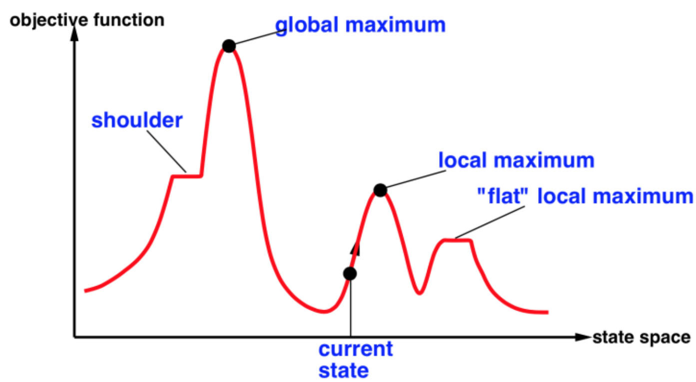
- **Hill-climbing search**

  A local search strategy which continuously moves in the direction of increasing elevation/value to find the peak of the mountain or best solution to the problem.

- **Genetic algorithm**

  Inspired from evaluation theory and aimed to find optimum solution by searching problem space randomly.

State Space Landscape

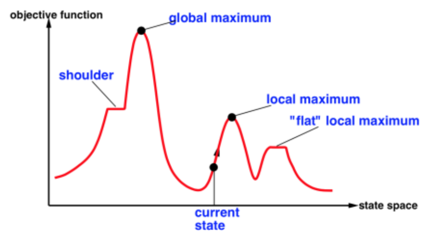


State space landscape

To understand local search, we find it useful to consider the **state-space landscape**. A landscape has both “location” (defined by the state) and “elevation” (defined by the value of the heuristic cost function or objective function). Local search algorithms explore this landscape. A **complete** local search algorithm always finds a goal if one exists; an **optimal** algorithm always finds a global minimum/maximum.

State Space Landscape

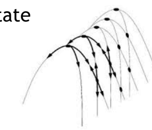
- State space landscape is a graphical representation of the set of states our search algorithm can reach and the value of our objective function (the function which we wish to maximize).
- X-axis : denotes the state space, ie. states or configuration our algorithm may reach.
- Y-axis : denotes the values of **objective function** corresponding to a particular state.
- The best solution will be that state space where objective function has maximum value (global maximum).

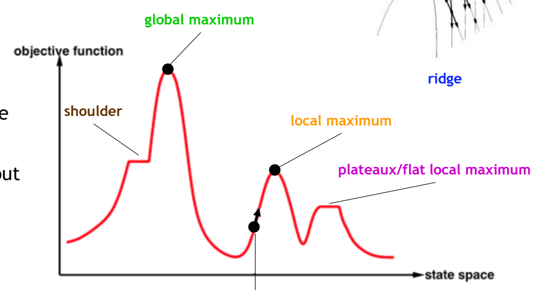


Source: https://www.geeksforgeeks.org/introduction-hill-climbing-artificial-intelligence/

Different Regions in the State Space Landscape

- Local maximum: It is a state which is better than its neighboring state however there exists a state which is better than it (global maximum). This state is better because here the value of the objective function is higher than its neighbors.
- Global maximum: It is the best possible state in the state space diagram. This because at this state, objective function has the highest value.
- Plateaux/flat local maximum: It is a flat region of state space where neighboring states have the same value.
- Ridge: It is a region which is higher than its neighbors but itself has a slope. It is a special kind of local maximum.
- Current state: The region of state space diagram where we are currently present during the search.
- Shoulder: It is a plateau that has an uphill edge.





Source: https://www.geeksforgeeks.org/introduction-hill-climbing-artificial-intelligence/

11

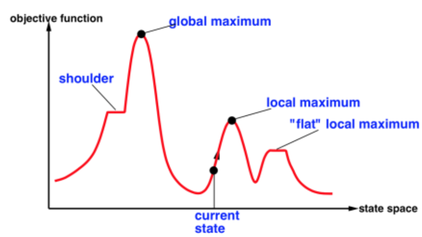
Hill Climbing

What is Hill Climbing?

How Hill Climbing 's Algorithm Works?

Hill-Climbing Search

- Also called **greedy local search**. Looks only to immediate good neighbors and not beyond.
- Search moves uphill: moves in the direction of increasing elevation/value to find the top of the mountain.
- Terminates when it reaches a **peak**.
- Can terminate with a local maximum, global maximum or can get stuck and no progress is possible.
- A **node is a state and a value**.



Hill-Climbing Search

function HILL-CLIMBING(initialState)
    returns State that is a local maximum

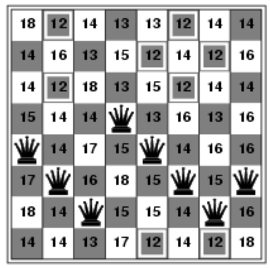
    initialize current **with** initialState

    loop do
        neighbor = a highest-valued successor of current

        if neighbor.value $\leq$ current.value:
            return **current.state**

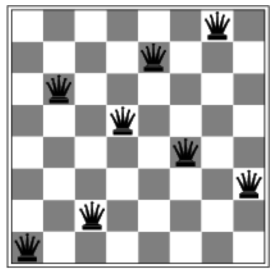
        current = neighbor

Hill-Climbing Search: 8-Queens Problem



- h = number of pairs of queens that are attacking each other, either directly or indirectly
- $h = 17$ for the above state and the goal state has $h = 0$

Hill-Climbing Search: 8-Queens Problem



- A local minimum with $h = 1$

Problems in Different Regions in Hill climbing

Hill climbing cannot reach the optimal/best state (global maximum) if it enters any of the following regions:

- **Local maximum**: At a local maximum all neighboring states have a values which is worse than the current state. Since hill-climbing uses a greedy approach, it will not move to the worse state and terminate itself. The process will end even though a better solution may exist.
- **Plateaux**: On plateaux all neighbors have same value . Hence, it is not possible to select the best direction.
- **Ridge**: Any point on a ridge can look like peak because movement in all possible directions is downward. Hence the algorithm stops when it reaches this state.

Source: https://www.geeksforgeeks.org/introduction-hill-climbing-artificial-intelligence/

Sideways moves in which you want to escape a plateau such as local maxima or shoulder. And the idea is, if we move sideways, there is a hope that we're going to escape from the flat area and reach a shoulder. We can put a limit on the number of sideways moves as we avoid infinite loops.

Another variant is called a random-restart in which we do hill climbing several times to overcome local maxima. This means keep trying until you get a better solution. So it depends on the problems. If the problem requires to find the goal, then we could do that several times until we find the goal. And if the problem is to optimize some objective function, we could do the local search several times until we reach the maximum value for this function.

And finally, stochastic hill climbing chosen at random among the uphill moves in order to find a better solution than just looking just around the current state.

Problems in Different Regions in Hill climbing

How to overcome the problems:

- To overcome local maximum problem: Utilize backtracking technique. Maintain a list of visited states. If the search reaches an undesirable state, it can backtrack to the previous configuration and explore a new path.
- To overcome plateaux: Make a big jump. Randomly select a state far away from the current state. Chances are that we will land at a non-plateaux region.
- To overcome ridge: In this kind of obstacle, use two or more rules before testing. It implies moving in several directions at once.

Source: https://www.geeksforgeeks.org/introduction-hill-climbing-artificial-intelligence/

Sideways moves in which you want to escape a plateau such as local maxima or shoulder. And the idea is, if we move sideways, there is a hope that we're going to escape from the flat area and reach a shoulder. We can put a limit on the number of sideways moves as we avoid infinite loops.

Another variant is called a random-restart in which we do hill climbing several times to overcome local maxima. This means keep trying until you get a better solution. So it depends on the problems. If the problem requires to find the goal, then we could do that several times until we find the goal. And if the problem is to optimize some objective function, we could do the local search several times until we reach the maximum value for this function.

And finally, stochastic hill climbing chosen at random among the uphill moves in order to find a better solution than just looking just around the current state.

Hill-Climbing Search

- **Hill climbing** effective in general but depends on shape of the landscape.
  - Successful in many real-problems after a reasonable number of restarts.
- **Local beam search** maintains k states instead of one state.
  - Select the k best successor, and useful information is passed among the states.
- **Stochastic beam search** choose k successors are random.
  - Helps alleviate the problem of the states agglomerating around the same part of the state space.

19 Genetic Algorithm

What is Genetic Algorithm?

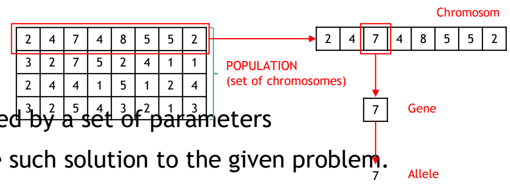
How GA's Algorithm Works?

Genetic Algorithm

- **Genetic algorithm (GA)** is a variant of stochastic beam search.
  - Successor states are generated by combining two parents rather by modifying a single state.
- The process is inspired by **natural selection** to find the best solution to a problem.
- In nature, only the strong one survive, the process of eliminating the weak is called **natural selection**.
- Genetic algorithm use that same principle to **eliminate the “weak” solutions** and finally **produce the best solution**.

Basic Terminology

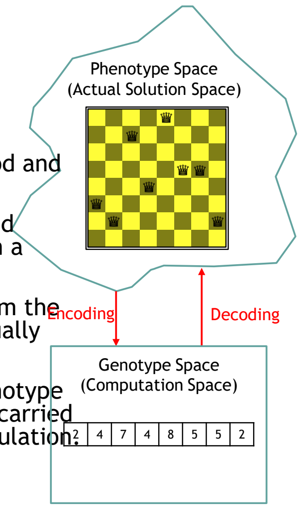
- **Population** – It is a subset of all the possible (encoded) solutions to the given problem. is characterized by a set of parameters



- **Chromosom** – or individual is one such solution to the given problem.
- **Gene** – A gene is one element position of a chromosome.
- **Allele** – It is the value a gene takes for a particular chromosome.

Basic Terminology

- **Genotype** – The population in the computation space. In the computation space, the solutions are represented in a way which can be easily understood and manipulated using a computing system.
- **Phenotype** – The population in the actual real world solution space in which solutions are represented in a way they are represented in real world situations.
- **Decoding** – A process of transforming a solution from the genotype to the phenotype space. Solutions are usually encoded as a bit strings or integers.
- **Encoding** - A process of transforming from the phenotype to genotype space. Decoding should be fast as it is carried out repeatedly in a GA during the fitness value calculation.

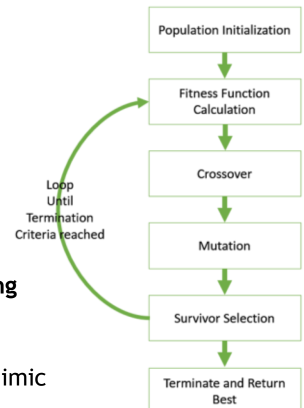


Basic Terminology

- **Fitness Function** – A function which takes a candidate solution to the problem as input and produces as output how "fit" or how "good" the solution is with respect to the problem in consideration. In some cases, the fitness function and the objective function may be the same, while in others it might be different based on the problem.
- **Genetic Operators** – These alter the genetic composition of the offspring. These include **crossover**, **mutation**, **selection**, etc.

Basic Structure

- Start with an **initial population** (which may be generated at random or seeded by other heuristics).
- **Select** parents from this population for mating.
- Apply **crossover** and **mutation** operators on the parents to generate new off-springs.
- And finally these off-springs **replace the existing individuals** in the population and the process repeats.
- In this way, genetic algorithms actually try to mimic the human evolution to some extent.



Source: https://www.tutorialspoint.com/

Genotype Representation

1. **Binary Representation**
   - The solutions consists of bit string, i.e., for the 0/1 Knapsack Problem, if there are n items, we can represent a solution by a binary string of n elements, where the x^{th} element tells whether the item x is picked (1) or not (0).

0	0	1	0	1	1	1	0	0	1

2. **Integer Representation**
   - The solutions consists of discrete valued genes, i.e., we want to encode distances with four cardinal directions – North, South, East and West, we can encode them as $\{0,1,2,3\}$.

1	2	3	4	3	2	4	1	2	1

Genotype Representation

3. **Real Valued Representation**
   - The solutions consists of continuous valued genes.

0.5	0.2	0.6	0.8	0.7	0.4	0.3	0.2	0.1	0.9

4. **Permutation Representation**
   - The solutions is represented by an order of elements, i.e. the travelling salesman problem (TSP). The salesman has to take a tour of all the cities, visiting each city exactly once and come back to the starting city. The total distance of the tour has to be minimized. The solution to this TSP is naturally an ordering or permutation of all the cities and therefore using a permutation representation makes sense for this problem.

1	5	9	8	7	4	2	3	6	0

Population

- The population is usually defined as a 2D array of – **size population: size** × **chromosome size**.
- Two primary methods to initialize a population in a GA.
  - **Random Initialization** – Populate the initial population with completely random solutions.
  - **Heuristic initialization** – Populate the initial population using a known heuristic for the problem.

Population Models

- There are two population models widely in use.
  - **Steady State**
    - In steady state GA, we generate one or two off-springs in each iteration and they replace one or two individuals from the population. A steady state GA is also known as Incremental GA.
  - **Generational**
    - In a generational model, we 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.

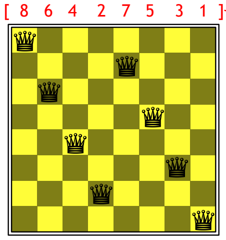
Fitness Function

- A fitness function is used to evaluate individuals.
- In most cases the fitness function and the objective function are the same as the objective is to either maximize or minimize the given objective function.
- However, for more complex problems with multiple objectives and constraints, an **Algorithm Designer** might choose to have a different fitness function.
- A fitness function should possess the following characteristics −
  - The fitness function should be sufficiently fast to compute.
  - It must quantitatively measure how fit a given solution is or how fit individuals can be produced from the given solution.

Fitness Function

- In the 8-queen problem, an individual can be represented by a string digits 1 to 8, that represents the position of the 8 queens in the 8 columns.
- Possible **fitness function** is the **number of non-attacking pairs of queens**.

[8 6 4 2 7 5 3 1]



the encoding of this state in the search space.

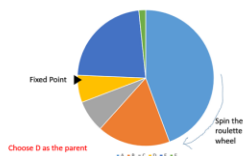
- **Fitness value** of the solution: 28.

Selection

- Selection is the process of selecting parents which mate and recombine to create off-springs for the next generation.
- **Maintaining good diversity** in the population is extremely crucial for the success of a GA.
- This taking up of the entire population by one extremely fit solution is known as **premature convergence** and is an undesirable condition in a GA.
- Selection Methods:
  - Fitness Proportionate Selection
  - Tournament Selection
  - Rank Selection
  - Random Selection

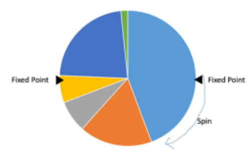
Selection Method: Fitness Proportionate Selection

- In this every individual can become a parent with a probability which is proportional to its fitness.
- Such a selection strategy applies a **selection pressure** to the more fit individuals in the population, evolving better individuals over time.
- Consider a circular wheel. The wheel is divided into n pies, where n is the number of individuals in the population. Each individual gets a portion of the circle which is proportional to its fitness value.
- Two implementations of fitness proportionate selection are possible: Roulette Wheel Selection and Stochastic Universal Sampling (SUS).



Chromosome	Fitness Value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

Roulette Wheel Selection



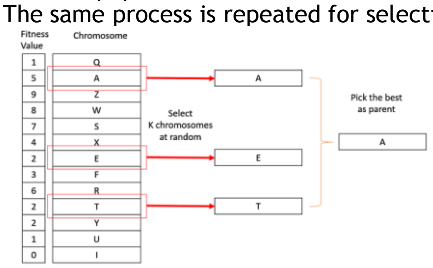
Chromosome	Fitness Value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

Stochastic Universal Sampling (SUS)

Source: https://www.tutorialspoint.com/

Selection Method: Tournament Selection

- Select K individuals from the population at random and select the best out of these to become a parent. The same process is repeated for selecting the next parent.

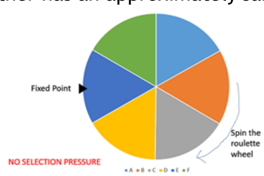


Source: https://www.tutorialspoint.com/

- Extremely popular in literature as it can even work with negative fitness values.

Selection Method: Rank Selection

- Also works with negative fitness values and is mostly used when the individuals in the population have very close fitness values (this happens usually at the end of the run).
- This leads to each individual having an almost equal share of the pie (like in case of fitness proportionate selection) as shown in the following image and hence each individual no matter how fit relative to each other has an approximately same probability of getting selected as a parent.



Chromosome	Fitness Value	Rank
A	8.1	1
B	8.0	4
C	8.05	2
D	7.95	6
E	8.02	3
F	7.99	5

Source: https://www.tutorialspoint.com/

- The selection of the parents depends on the rank of each individual and not the fitness. The higher ranked individuals are preferred more than the lower ranked ones.

Selection Method: Random Selection

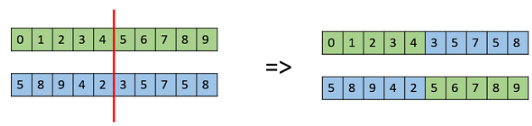
- In this strategy, we randomly select parents from the existing population.
- There is no selection pressure towards fitter individuals and therefore this strategy is usually avoided.

Crossover

- In this more than one parent is selected and one or more off-springs are produced using the genetic material of the parents.
- Crossover is usually applied in a GA with a high probability – p_c.
- Basic Crossover Operators
  - One Point Crossover
  - Multi Point Crossover
  - Uniform Crossover
  - Whole Arithmetic Recombination

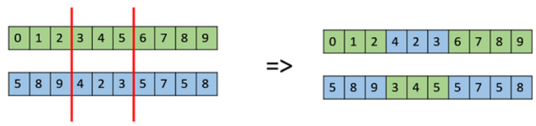
Crossover Operator: One Point Crossover

- A random crossover point is selected and the tails of its two parents are swapped to get new off-springs.



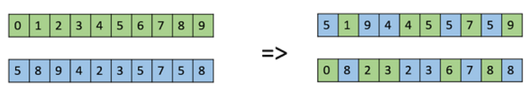
Crossover Operator: Multi Point Crossover

- Multi point crossover is a generalization of the one-point crossover wherein alternating segments are swapped to get new off-springs.



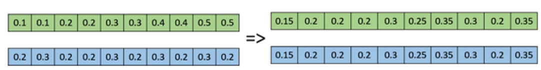
Crossover Operator: Uniform Crossover

- We treat each gene separately.
- In this, we essentially flip a coin for each chromosome to decide whether or not it'll be included in the off-spring.
- We can also bias the coin to one parent, to have more genetic material in the child from that parent.



Crossover Operator: Whole Arithmetic Recombination

- This is commonly used for integer representations and works by taking the weighted average of the two parents by using the following formulae:
  - Child1 = $\alpha.x + (1-\alpha).y$
  - Child2 = $\alpha.x + (1-\alpha).y$
- Obviously, if $\alpha = 0.5$, then both the children will be identical as shown in the following image.



Mutation

- Mutation may be defined as a small random tweak in the chromosome, to get a new solution.
- It is used to maintain and introduce diversity in the genetic population and is usually applied with a low probability – p_m.
- If the probability is very high, the GA gets reduced to a random search.
- Basic Mutation Operators
  - Bit Flip Mutation
  - Swap Mutation
  - Scramble Mutation
  - Inversion Mutation

Mutation Operators: Bit Flip Mutation

- We select one or more random bits and flip them.
- This is used for binary encoded GAs.

0	0	1	**1**	0	1	0	0	1	0

=>

0	0	1	**0**	0	1	0	0	1	0

Mutation Operators: Swap Mutation

- Select two positions on the chromosome at random, and interchange the values.
- This is common in permutation based encodings.

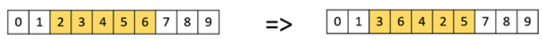
1	2	3	4	5	6	7	8	9	0

=>

1	6	3	4	5	2	7	8	9	0

Mutation Operators: Scramble Mutation

- From the entire chromosome, a subset of genes is chosen and their values are scrambled or shuffled randomly.



Mutation Operators: Inversion Mutation

- Select a subset of genes like in scramble mutation, but instead of shuffling the subset, we merely invert the entire string in the subset.

0	1	2	3	4	5	6	7	8	9

=>

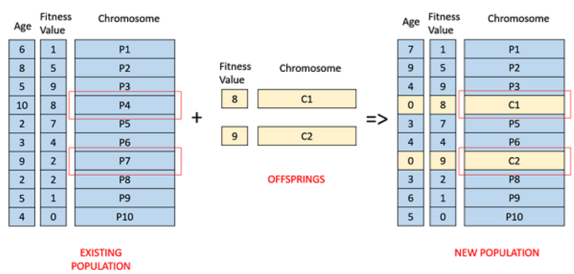
0	1	6	5	4	3	2	7	8	9

Survivor Selection

- The Survivor Selection Policy determines which individuals are to be kicked out and which are to be kept in the next generation.
- Crucial as it should ensure that the fitter individuals are not kicked out of the population, while at the same time diversity should be maintained in the population.
- Some GAs employ **Elitism**. In simple terms, it means the current fittest member of the population is always propagated to the next generation. Therefore, under no circumstance can the fittest member of the current population be replaced.
- The easiest policy is to kick random members out of the population, but such an approach frequently has convergence issues, therefore the following strategies are widely used:
  - Age Based Selection
  - Fitness Based Selection

Survivor Selection Strategy: Age Based Selection

- In Age-Based Selection, we don’t have a notion of a fitness.
- It is based on the premise that each individual is allowed in the population for a finite generation where it is allowed to reproduce, after that, it is kicked out of the population no matter how good its fitness is.

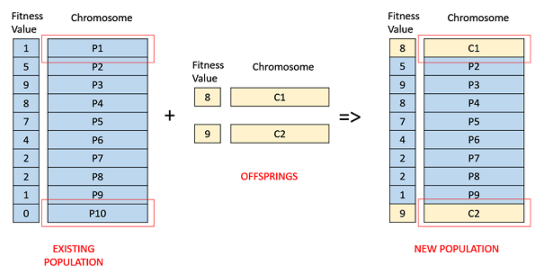


Source: *https://www.tutorialspoint.com/*

Survivor Selection Strategy:

Fitness Based Selection

- In this fitness based selection, the children tend to replace the least fit individuals in the population.
- The selection of the least fit individuals may be done using a variation of any of the selection policies described before – tournament selection, fitness proportionate selection, etc.



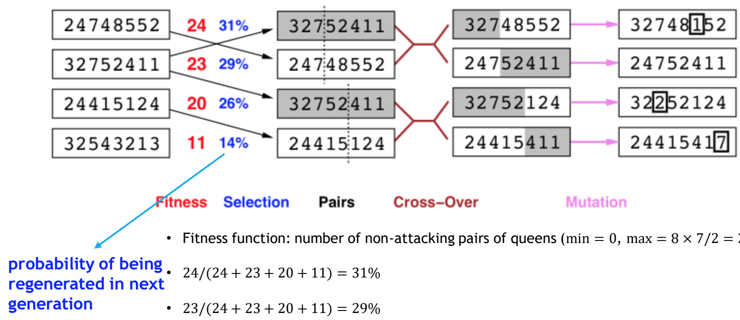
Source: https://www.tutorialspoint.com/

Termination Condition

- Usually, we keep one of the following termination conditions:
  - When there has been no improvement in the population for X iterations.
  - When we reach an absolute number of generations.
  - When the objective function value has reached a certain pre-defined value.

Genetic Algorithm: 8-Queens Problem

Generate successors from pairs of states.



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

Genetic Algorithm

function GENETIC-ALGORITHM(population, fitness-function)
returns an individual

repeat

initialize new-population **with** $\emptyset$

for i=1 to size(population) **do**

x = random-select(population,fitness-function)
x = random-select(population,fitness-function)
child = cross-over(x,y)
mutate (child) with a small random probability
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 w.r.t. fitness-function

References

- S. J. Russell and P. Borvig. (2020). *Artificial Intelligence: A Modern Approach (4th Edition)*, Prentice Hall International.
  - Chapter 4

eof