Travelling salesman problem is an example of a combinatorial optimization problem. It is classified as *NP-hard* as no quick solution exists and increasing the number of cities exponentially increases the complexity of the problem. TSP can be described as finding the shortest path to visit a number of locations with set distances between each other and returning to the same location.

Combinatorial optimization is a subfield of mathematical optimization that consists of finding an optimal object from a finite set of objects, where the set of feasible solutions is discrete or can be reduced to a discrete set.

Computational intractability is a problem in which the only exact solution is one that takes too many resources (time, memory, etc.). In other words, a problem in which no efficient solution (other than the worst-case complexity) exists.

There are two ways of solving travelling salesman problems: linear algorithms and heuristic algorithms. Linear algorithms test all permutations and check which one is the shortest using **brute force search**. While this approach always gives a *perfect* solution, it runs on the factorial time complexity of *O(n!)*. Thus, even a TSP with 15 cities needs all 15! = 1.3076744e+12 possibilities to be checked which makes the traditional way impractical.

Hill Climbing is a heuristic search used for mathematical optimization problems.
Given a large set of inputs and a good heuristic function, it tries to find a sufficiently good solution to the problem. While the **heuristic methods** do not always output a perfect solution unlike the traditional algorithms, they make solving the travelling salesman *practically possible* by greatly *reducing the execution time* in cases where there are very large amounts of data. ***Accuracy, precision and completeness are sacrificed for execution time (speed).***

Genetic Algorithms:

The advantages of genetic algorithms are thought to be their ability to simultaneously sample vast fitness landscapes while escaping the local extrema that might trap more traditional hill-climbing approaches.

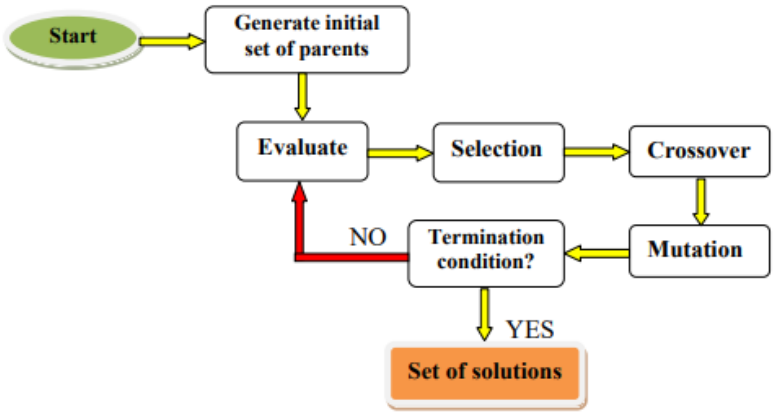


Fig.1 Process of genetic algorithm

Initialization of the first population is usually done *randomly* in genetic algorithms in order to create diversity and prevent premature convergence.

Fitness is a metric of how efficiently a solution solves the travelling salesman problem. A **fitness function** is a particular type of function that determines the fitness of solutions. **Fitness landscapes** are used to visualise the relationship between genotypes and reproductive success.

Mating pool is a concept which refers to a family of algorithms that are used to solve optimization and search problems. It is formed by candidate solutions that **selection operators** deem to have the highest fitness in the current population. The solutions are called **parents**. **Crossover operators** and **mutation operators** are applied to the parents. The resulting solutions are called **children**. The children form the new **population** that will go into mating pool selection unless the **termination condition** is met. The termination condition may consist of a distance for the shortest path, number of generations or execution time.

	How It Works	Advantages	Disadvantages
Roulette Wheel Selection	Random, but higher levels of fitness more likely. Imagine a wheel where the sizes of the slices are determined according to the fitness of the solutions and the wheel is turned in order to choose a member of the mating pool, the chosen solution is then removed from the wheel.	It gives a chance to be selected to all fitness values, thus preserving diversity.	If one of the fitness values is way higher than the others, diversity will be reduced and it may lead to premature convergence. Slower than stochastic universal sampling.
Stochastic Universal Sampling	Random, but higher levels of fitness more likely. Imagine a wheel where the sizes of the slices are determined according to the fitness of the solutions and the wheel is turned once in order to choose all of the members of the mating pool.	It gives a chance to be selected to all fitness values, thus preserving diversity. Faster than roulette wheel selection.	
Tournament Selection	Selected randomly and then compared according to fitness levels. The "winners" are added to the mating pool.	Easy to implement as no sorting is required. As weaker solutions can also be selected to be compared with each other, it preserves diversity by giving a chance to the weaker solutions.	If the tournament size is large, the same individual can always win which will reduce diversity leading to premature convergence.
Ranking Selection	Sorted according to fitness levels. The highest ranked ones have a higher chance of being selected.	All individuals have a chance to be selected which preserves diversity.	Computationally expensive as it has to sort all solutions. Leads to slower convergence because all of the highest ranked solutions are similar.
Truncation Selection	Sorted according to a special trait, the ones above a certain threshold are selected as parents.	Fastest and easiest to implement.	Doesn’t give opportunity to weaker solutions, thus not preserving diversity.

Crossover in genetic algorithms refers to the combination of the genetic information of two parents to generate new offspring that may potentially be more efficient than the parents. One point and two point crossovers can not be used in genetic algorithms as they may result in the repetition or omission of the cities in the offspring.

Partially mapped crossover (PMX)

Choose a random sub-sequence from P1 and copy it to F1:

```
P1: J B F C A D H G I E
P2: F A G D H C E B J I

F1: * * F C A D H * * *
```

Set up elementwise mappings between P1 and P2 for cities in P1 that are not already in F1. If the corresponding city C from P2 is already in F1 then resume mapping from the location of C in P1, repeating until a city not in F1 is found:

J ↔ G B ↔ E G ↔ B I ↔ J E ↔ I

Add the remaining cities from P1 to F1 and change them according to the mappings:

```
P1: J B F C A D H G I E
P2: F A G D H C E B J I

F1: * * F C A D H * * *
      ↓ ↓           ↓ ↓ ↓
F1: J B F C A D H G I E
      ↓ ↓           ↓ ↓ ↓
F1: G E F C A D H B J I
```

Order crossover (OX)

Choose a sub-sequence from one parent and preserve the relative order of the remaining cities from the other.

Take a random sub-sequence S from P1 and copy it to F1. Starting with the empty element just after S in F1, copy all cities that are not already in F1 from P2 in the order they appear in P2.

```
P1: J B F C A D H G I E
P2: F A G D H C E B J I

F1: * B F C A D * * * *
    H B F C A D E J I G
```

Cycle crossover (CX)

In F1, every city maintains the position it had in at least one of its parents.

P1: J B F C A D H G I E
P2: F A G D H C E B J I

Choose the first city from P1 and copy it to F1. Check the corresponding city in P2 (here, city F) and copy it to F1, in the same position it occurs in P1. Repeat.

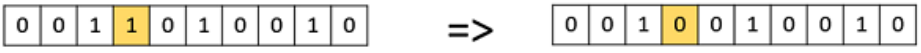
F1: **J** B F * A * H G I E

When you encounter a city that is already in F1 the cycle is complete. Now fill in the remaining cities from P2.

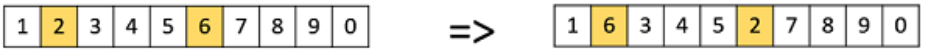
F1: J B F **D** A **C** H G I E

Mutation is a genetic operator used to maintain genetic diversity from one generation to the next. Mutation changes one or more gene values of a chromosome which changes the properties of the solution. Unlike crossover, mutation is essential for genetic algorithms. **Mutation rate** is the rate at which mutations happen. If the mutation rate is too low, the progress is slow. If the mutation rate is too high, the genetic algorithm is reduced to a random search.

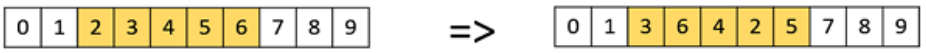
- *Bit Flip Mutation*: One or more random bits on a chromosome are selected and then flipped



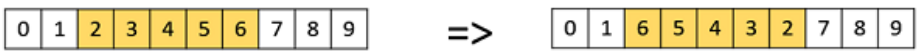
- *Swap Mutation*: Two random positions on a chromosome are selected and then swapped



- *Scramble Mutation*: A subset of positions are selected on a chromosome and then they are shuffled randomly



- *Inversion Mutation*: A subset of positions are selected on a chromosome and then they are inverted



Convergence in genetic algorithms is a situation that occurs when the children' fitness values can not pass the parents'. If the algorithm gets stuck in a **local extremum** before reaching the global extremum (optimum solution), it is called premature convergence. **Premature convergence** can be prevented using selection operators that award diversity as well as fitness and having mutation rates that aren't too low.

The idea of **elitism** is to preserve the best individual of a population. Due to crossover and mutation operators the best individuals are lost. The aim of elitism is to not only preserve the best individual but at the same time allowing it to participate in reproduction so as to propagate the best characteristics to the next generations.

Exploration refers to the discovery of new solutions in order to find the global optimum rather than being stuck in a local one. **Novelty Search** is an exploration algorithm driven by the novelty of a behaviour.

Exploitation refers to the refinement of existing products, resources, knowledge and competencies, and is associated with incremental changes and learning through local search in order to have a better fitness.

Successful implementations of genetic algorithms strike a *natural balance between exploration and exploitation*, and techniques such as **simulated annealing** (which is an optimization method which mimics the slow cooling of metals) can fine-tune that balance as the algorithm progresses towards convergence

Problem space is all of the various components that go into creating a resolution for a problem.