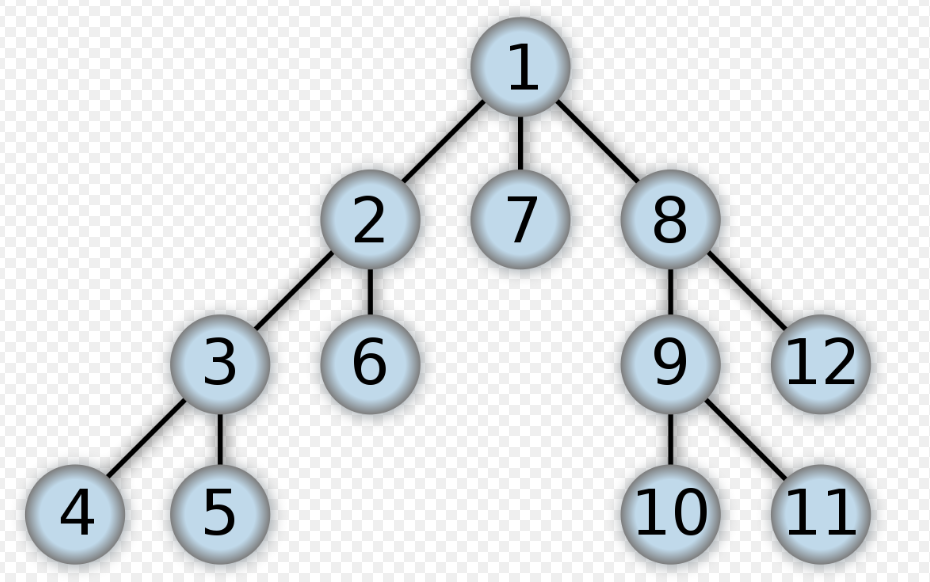
**Search Algorithms**

*Depth-First Search*

* Algorithm for traversing a tree or graph data structure.
* Search Method:

1. Starting from the root node, visit each parent node on one side
2. Reach the very end, then visit the child of each parent from the bottom up
3. When returning to the root node, work on the other side and repeat.

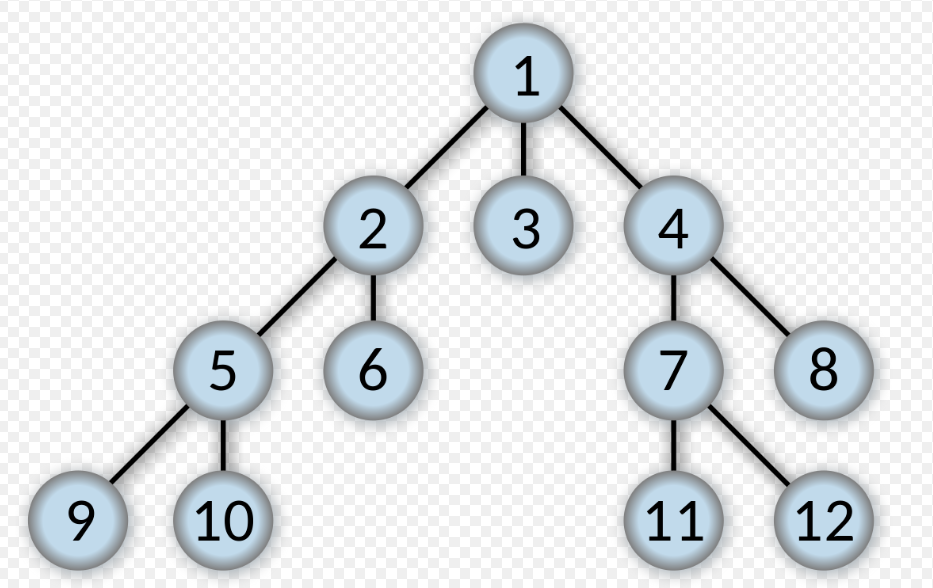


* Time Complexity: O(V + E), Depth- First Search visits every vertex in the graph and checks every edge.
* Advantages:
  1. Memory Requirement is linear with respect to the graph. The algorithm stores a stack of nodes from the root to the current node.
  2. Time Complexity is time-limited over space-limited.
  3. If Depth-First Search finds a solution without exploring much nodes the time and space is much smaller.
* Disadvantages:
  1. Not guaranteed to find a solution
  2. Cut-Off depth is smaller so time complexity is more
  3. May be trapped on left-most path forever
  4. Not guaranteed to find minimal solution

*Breadth-First Search*

* Algorithm for traversing a tree or graph data structure.
* Search Method:
  1. BFS starts at tree root and explores neighbour nodes first.
  2. Once neighbour nodes are explored, it moves to neighbour of neighbour nodes.

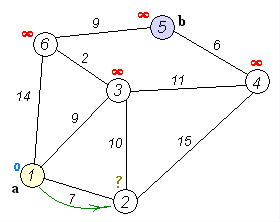
*.*



* Time Complexity: O(V + E), every vertex and every edge will be explored in the worst case.
* Advantages:
  1. Solution will definitely be found if there is a possible solution.
  2. Will never become trapped by unwanted nodes
  3. If there are more than one solution, it will find the solution with minimal steps.
* Disadvantages
  1. Memory Constraints for storing multiple stacks for each route
  2. If solution is deep then it consumes time.

*Djikstras Algorithm*

* Algorithm for finding the shortest path between nodes in a graph
* Search Method:
  1. Mark nodes unvisited in a set.
  2. Assign root value as Zero.
  3. Calculate distances to neighbour nodes and save distance of each node.
  4. When the node is added to distance, remove it from the unvisited set.
  5. Move to the next unvisited nodes and add the distance.
  6. If the destination node is moved from the unvisited set, then stop.
  7. The distance can be calculated.



* Time complexity is O(|V|^2)
* Advantages:
  1. Fastest Path search algorithm.
  2. Finding shortest path between two nodes
* Disadvantages:
  1. Blind Search operations

*Iterative Deepening DFS*

Iterative Deepening DFS is a state space/graph search strategy in which the depth is limited. The search is run with increasing depth limits, but uses less memory. On each iteration it visits nodes in the search tree in the same order as DFS but accumulates similar to BFS.

Advantages:

* Combines DFS space efficiency and BFS completeness. Also where branching factor is finite.
* Optimal when path cost is a non decreasing function of node depth
* IDDFS improves heuristics as it estimates the accurate branching and completes more quickly.
* Algorithm responsiveness is extremely quick. The earlier iterations supply some indication of a result quickly.

Time complexity is O (b^d) – b is branching factor, d is depth of the goal

Space complexity is O (d) where d is the depth of the goal

*A\* Algorithm*

A\* algorithm is used in pathfinding and graph traversal. It is a n informed search algorithm meaning that it solves problems by searching among all possible paths to the the goal for the one that incurs the smallest cost in time,distance etc.. It considers the best cost path to lead to the solution.

* A\* algorithm is complete and will always find a solution
* Its performance and accuracy make it powerful as a search algorithm
* Achieves better performance by using heuristics
* A\* is admissible in that it considers fewer nodes than other algorithms with the same heuristics.

*Best-First Search*

Best-First Search is an algorithm that explores a graph by expanding the most promising node chosen to a specified rule.

* Best-First estimates according to a heuristic evaluation function which depends on the description of the goal, the info gathered up to some node and any extra knowledge about the problem domain.
* The heuristic attempts a prediction of how close the end of a path is to a solution, so that paths which are judged closer to a solution are expanded first. A priority queue is implemented to select the best path

*Greedy BFS*

1. If the successor heuristic is better than the parent, the successor is moved to the front
2. Else the successor is inserted into queue in a location determined by heuristic value

*Genetic Algorithm*

The genetic algorithm is a metaheuristic inspired by natural selection in real world applications. Genetic algorithms are used to generate high-quality solutions to optimization and search problems by relying on the operators of mutation, crossover and selection.

*Five Phases*

1. *Initial population*

The process is provided with a population of individuals. An individual is characterized by a gene. Genes are joined to form a chromosome which is the solution.

1. *Fitness Function*

The fitness function determines how fit an individual is which is how well it can compete with other individuals. Each individual is assigned a fitness score. The probability that individual is selected for reproduction is based on the fitness score.

1. *Selection*

The idea of selection phase is to select the fittest individuals and let them pass their genes onto the next generation. Two parents are selected based on fitness score. Individuals with high fitness have more chance of reproduction.

1. *Crossover*

Crossover is the most important phase of the genetic algorithm. For each pair of parents to be mated, a crossover point is chosen at random from within the genes. Offspring are created by exchanging genes of parents among themselves until the crossover point is reached. New offspring are added to the population.

1. *Mutation*

In certain new offspring formed, some of their genes are subjected to mutation with a low random probability. This implies that some of the bits in the bit string can be flipped. Mutation occurs to maintain diversity and prevent premature convergence

1. *Termination*

The algorithm terminates if the population has converged where the offspring are too similar to their parent generation. Then it is given that the solution has been provided.

def genetic\_algorithm()

{

range\_of\_valid\_vals,

fitness\_func,

elite\_fraction = 1/5,

pop\_size = 50,

mutation\_rate = 0.6,

max\_step = 100,

max\_iterations = 100

}

**Range\_of\_valid\_vals**

Purpose: What values the gene will hold.

Increase: Reduction in performance but more complex, interesting data obtained

Decrease: Faster performance but premature convergence may occur

**Fitness\_func**

Purpose: To refine the elitist fraction according to some parameters

Increase: Takes longer for individuals of a population to mutate enough to become elite

Decrease: The standard of elite individuals will drop

**Elite\_Fraction**

Purpose: The elite fraction is the percentage of population which have the strongest genes

Increase: The overall elite percentage will be less strong and will correlate more to the overall population

Decrease: The elite percentage will be much stronger than the rest of population.

**Pop\_Size**

Purpose: The entire population to be put on the genetic algorithm

Increase: Premature convergence is less likely to occur

Decrease: Premature convergence is more likely to occur

**Mutation\_Rate**

Purpose: The rate at which individuals are likely to change their data

Increase: Individuals are less likely to look like their parents

Decrease: Individuals are more likely to look like their parents and premature convergence may occur quicker

**Max\_Steps**

Purpose: The max amount of looping through the main genetic algorithm steps before testing

Increase: More data is obtained but premature convergence may occur

Decrease: Less change in genetic algorithm data

**Max\_Iterations**

Purpose: Max iterations through the entire population

Increase: Increases the spread of data among population

Decrease: Decreases the spread of data among the population

*Additional Terms*

a. Genetic Heuristic

In addition to the main variants, various heuristics can be applied to make calculation faster.The speciation heuristic penalizes crossover between candidates that are too similar. This encourages population diversity and prevents premature convergence.

b. Building Block Hypothesis

Genetic algorithms are simple to implement but behaviourisms are difficult to understand. It is difficult to understand why these algorithms frequently succeed at generating solutions of high fitness when applied to practical problems. BBH consists of:

* + 1. A description heuristic that performs adaptation by identifying and recombining building blicks
    2. Hypothesis that a genetic algorithm performs adaptations by implementing this heuristic

c. Limitations

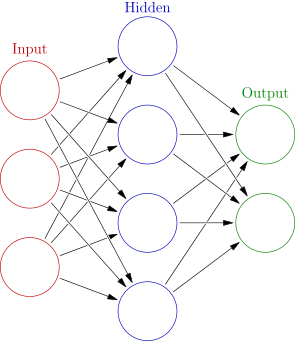
There are limitations to the usage of genetic algorithms to alternative optimization algorithms:

* Repeated fitness function evaluation for complex problems is the most prohibitive and limiting segment of artificial evolutionary algorithms. High-  
  dimensional multimodal problems often require very expensive fitness functions.
* Genetic Algorithms do not work well with complexity. Where the number of elements which are exposed to mutation is large, there is often exponential increase in search space size.
* The better solution is only in comparison to other solutions. Stop criterion is not clear
* Genetic Algorithms tend to converge on local optima or even arbitrary points rather than global optimum. This problem can be alleviated by altering the fitness function, increasing rate of mutation or using selection techniques that maintain diverse population of solutions.
* Operating on dynamic data sets is difficult.
* Genetic algorithms cannot effectively solve problems in which the only fitness measure is a single right/wrong measure.
* The suitability of genetic algorithms is dependent on the amount of knowledge of the problem, well known problems have better specialized approaches.

d. Elitism

A practical variant of constructing a new population is to allow the best organisms from current generation to carry over, unaltered. This strategy is elitist selection.

*Neural Network*

**

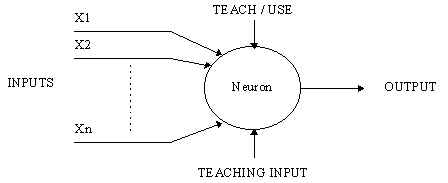
Neural networks are computing systems that are based off the human brain. These systems learn to perform tasks by having a supply of sample data. A large number of highly interconnected neurons work in unison to solve specific problems. Neural networks can be used to extract patterns and trends thar are too complex for humans.

Advantages:

* Adaptive learning – An ability to learn how to do tasks based on data given for training or initial experience
* Self Organisation – the ability to create its own hierarchy of information it recieves in learning
* Fault tolerance – Partial destruction of a neural network leads to degradation of performance, although some neural network capabilities may be retained.
* Complexity – Neural networks can solve abstract and complex problems approximately regardless of size or linearality

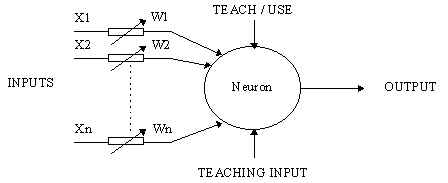
Disadvantages:

* Simplicity – Too complex for usage in simple problems
* Approximation – Increasing the accuracy of solutions can increase complexity by several magnitudes
* Training – Training a neural network requires the input of many test cases.
* Process – the process of a neural network is invisible and can only be manipulated by changing inputs and examining outputs.



A neural network consists of many neurons. Each neuron has multiple inputs which generate one output. Neurons have two modes of operation: training mode and firing mode. In training mode, the neuron can be trained to fire for particular patterns. In using mode, when a taught input is detected, its associated output becomes the current output. Neurons also follow the firing rule.

The firing rule is an important concept in neural networks. A firing rule can determine whether a neuron should fire an input pattern.

**

A more complex neuron is one with attached weighting to each input. The effect that each input has is dependent on the weight of the input. The weight of the input is a number which when multiplied with the input gives the weighted input. The weighted input are added together and if they exceed a pre-set threshold value, the neuron fires. The formula for the neuron firing from the input is:

**X1W1+X2W2+X3W3 > T**

**NN: Supervised v unsupervised learning**

**Supervised learning** a type of system in which both input and resulting output data are provided. Supervised learning provides training data for a neural network.

Advantages: -

* + - Make judgements which humans can relate to
    - Makes exact specifications about class definition for a perfect decision boundary
    - Determines how many classes there can be
    - After training, you can keep decision boundary as a mathematical formula for classifying future inputs

Disadvantages:-

* + - Decision boundary might be overtrained which result in object not being fit into their proper class
    - Training needs a lot of computation time
    - Each class must have a lot of training and in big datasets this can be a challenge

**Unsupervised learning** is where all observations are assumed to be caused by latent variables. Latent variables are undefined and when a neural net is unsupervised it judges incoming variables based on its training data.

Advantages:-

* + - The primary benefit is that a neural net with unsupervised learning can decide what data is placed where
    - The performance of the algorithm and training data increases the power of the neural network

Disadvantages:-

* + - If the training data is too small or too complex then unsupervised learning will not be as intuitive.

**NN: Transfer Function**

The transfer function is a mathematical function giving the corresponding output calue for each possible input to the device. The unit of the domain and range of the transfer function depend on the device.

**NN: Loss/Cost Function**

Loss or cost function is a function that maps an event or values of one or more variables onto a real number which represents some cost associated with the event. An optimization problem seeks to minimize this function Loss functions are used to capture negative aspects of a problem in order to improve them.

**NN: Back Propagation**

Back propagation is a method in neural networks to calculate the gradient that is needed in the calculation of weights to be used in the network. It is used to train deep neural nets which is a neuarl network with multiple hidden layers. Backpropagation requires the derivative of the loss function. It is considered to be supervised learning method. The goal of this algorithm is to find a function

**NN: Forward Propagation**

Forward propagation retrieves the output and compares it with the input in order to get the cost funtion error. To minimize this error, propagate backwards by finding the derivative of the error with respect to each weight and then subtract this value from the weight value. By propagating forward in a neural network you can see how well those weights are performing and then will back propagate to update the weights. This will continue until there is a minimum value for the cost function.