Search Algo

Heuristic And MetaHeuristics

A heuristic is a problem-solving strategy that uses an educated guess or a simple, fast rule to quickly come up with a solution that is likely to be optimal. Heuristics are often used in optimization problems where finding the optimal solution is too time-consuming, but finding a good solution is still desirable. In these cases, heuristics can provide a balance between speed and accuracy.

Probabilistic metaheuristics, on the other hand, are optimization algorithms that use randomness as a way to escape from local optima and search for a globally optimal solution. These algorithms can be seen as a combination of deterministic and stochastic algorithms, as they use both systematic and random procedures to explore the solution space. Examples of probabilistic metaheuristics include Simulated Annealing, Genetic Algorithms, and Particle Swarm Optimization.

Both heuristics and probabilistic metaheuristics can be useful in solving complex optimization problems where finding the optimal solution is challenging. However, probabilistic metaheuristics are typically more computationally intensive and may require more fine-tuning, while heuristics are generally faster but may not always produce the best results.

Hill Climbing

Hill climbing is a heuristic optimization algorithm used for finding the maximum or minimum of a function. It's a type of greedy algorithm, which means it takes the best option available at each step, without considering the global optimum.

The hill climbing algorithm works by starting from an initial solution, then making small, random changes to the solution to see if it improves the outcome. If it does, the new solution becomes the current solution, and the process repeats. If the new solution is worse, the algorithm rejects the change and tries a different change. This continues until no further improvements can be found, and the algorithm returns the best solution it has found so far.

The algorithm is called "hill climbing" because it's as if the algorithm is climbing up a hill, trying to reach the top. The top of the hill represents the optimal solution. The algorithm starts at the bottom and takes small steps to get closer to the top. If it reaches a peak, it has to backtrack to find a better path.

The hill climbing algorithm can be applied to a wide range of optimization problems, including problems in machine learning, artificial intelligence, and operations research. It's a simple and fast algorithm, but it's also prone to getting stuck in local optima, which are suboptimal solutions that are close to the optimal solution but not actually the optimal solution. To avoid this issue, some variations of the hill climbing algorithm use a randomization element, such as simulated annealing or genetic algorithms, to help escape local optima and find the global optimum.

Simulated Annealing

Simulated Annealing is a probabilistic metaheuristic optimization algorithm inspired by annealing process in metallurgy. It is used to find an optimal solution to a optimization problem by minimizing an objective function.

The algorithm starts with a random solution and iteratively changes it by randomly selecting a neighbor solution and accepting or rejecting the change based on the objective function and a probability defined by a temperature parameter. The temperature parameter is gradually decreased over time to reduce the number of random changes and to converge towards a solution that is close to the optimal solution.

The idea behind the Simulated Annealing algorithm is to balance exploration and exploitation, so that the algorithm can find the optimal solution even if it is located in a different part of the search space. The temperature parameter ensures that the algorithm will not get stuck in a local minimum by allowing it to escape from it with a certain probability.

The Simulated Annealing algorithm has been applied to a wide range of optimization problems, including the traveling salesman problem, the knapsack problem, and the satisfiability problem. It has several advantages over other optimization algorithms, including its ability to find the global optimum and its ability to handle large search spaces with many local minima.

Beam Search

Beam search is a heuristic search algorithm that is commonly used in artificial intelligence and computational linguistics. It is a best-first search algorithm that explores all possible solutions by expanding the most promising ones at each step, while keeping only a fixed number of the best solutions, or "beams," at each step.

The algorithm works by keeping track of a set of k candidate solutions, where k is the width of the beam, and at each step, expanding the best k solutions by generating their children. The best k children are then selected to form the next set of candidate solutions, and the process repeats until a satisfactory solution is found or the search space has been exhausted.

One of the key features of beam search is its ability to prune the search space by only keeping the most promising solutions at each step. This reduces the amount of memory required and the computational time needed, making it a more efficient alternative to other search algorithms such as breadth-first search and depth-first search.

Beam search is used in a variety of applications, including speech recognition, machine translation, and game playing, where the algorithm is used to find the best move in a game by searching through all possible game states.

Genetic Algo

Genetic algorithms are optimization algorithms that are inspired by the principles of evolution and natural selection. The basic idea is to use the concepts of reproduction, mutation, and selection to generate a population of candidate solutions to a problem, and then to iteratively improve this population over time by applying these same principles.

At each iteration of the algorithm, the current population is evaluated to determine its fitness with respect to a problem-specific objective function. The best-performing individuals are then selected to form a new population, which is created by recombining the selected individuals and applying mutations to introduce new genetic material. This process is repeated for multiple generations until a satisfactory solution is found, or until a stopping criterion is met.

One of the strengths of genetic algorithms is their ability to find global solutions to complex problems, even when the search space is large and the objective function is non-linear and multimodal. This makes them useful for a wide range of optimization problems, including problems in areas such as engineering, finance, and machine learning.

In terms of implementation, a genetic algorithm typically involves several components, including a representation of the candidate solutions, a method for evaluating the fitness of the solutions, a method for selecting individuals to form the next population, and a method for recombining and mutating individuals. The specific details of these components will depend on the particular problem and the desired trade-off between accuracy, speed, and complexity.

And-Or Search Tree

The AND-OR search tree is a type of search tree used in artificial intelligence and computer science to represent the search space of a problem. It is used to model search problems where the goal is to find a sequence of actions that lead to a solution. The tree is composed of nodes, where each node represents a state in the search space. Each node has two branches, an AND branch and an OR branch.

The AND branch represents a sequence of actions that must be performed in order to reach the next state. The OR branch represents a choice between several alternatives, each of which leads to a different next state. The OR branch is used to model problems where there are multiple possible actions that can be taken in a state. The AND-OR search tree is used to model problems where there are both sequential actions that must be performed and choices between alternatives that must be made in order to reach a solution.

The AND-OR search tree can be used in conjunction with various search algorithms, such as depth-first search, breadth-first search, or best-first search, to solve a problem. The tree is used to represent the search space and the search algorithm is used to explore the tree and find the best solution.

TSP

The traveling salesman problem (TSP) is a well-known optimization problem in computer science and mathematics. The problem is to find the shortest possible route that visits a given set of cities and returns to the starting city, visiting each city exactly once.

In mathematical terms, the TSP is formulated as an instance of the graph theory problem known as the Hamiltonian cycle problem. The goal is to find a simple cycle (a cycle that does not intersect itself) that passes through all nodes of a given graph.

The TSP is a classic optimization problem that is often used as a benchmark for testing the performance of optimization algorithms. It is considered a NP-hard problem, meaning that there is no known algorithm that can solve it optimally in polynomial time for all instances of the problem. As a result, approximate solutions are often used in practice, such as those produced by heuristics or metaheuristics like genetic algorithms or simulated annealing.

In real-world applications, the TSP can be used to model problems such as scheduling delivery trucks, optimizing airline routes, or finding the most efficient way to tour a set of attractions.

N- Queen

The n-queens problem is a classic problem in computer science and mathematics, where the goal is to place n queens on an n x n chessboard such that no two queens threaten each other, i.e., no two queens are in the same row, column, or diagonal. The problem can be generalized to any size chessboard, but it is often considered for an 8×8 chessboard, where the goal is to place 8 queens.

The n-queens problem is a classic example of a constraint satisfaction problem, where the goal is to find a solution that satisfies a set of constraints. In this case, the constraints are that no two queens can occupy the same row, column, or diagonal on the chessboard. The n-queens problem is a difficult problem, as the number of possible configurations grows rapidly as the size of the chessboard increases. Nevertheless, it can be solved using a variety of algorithms, including backtracking, branch and bound, and heuristics.

More info

These notes are written in a way that is easy to understand, but again I may not have organised them as the same order of the slides because everyone has it's own way of writing especially in writing notes, so if you still have some doubts. Feel free to report it to me :)