**Course 6: Deep Learning**

In this course, we will explore different informed search strategies, starting with best-first search and its use, heuristic functions, and the greedy approach. We will then dive into the A\* algorithm, its workings, examples, code implementation, and a brief overview of its variations and applications. After that, we'll move on to local search strategies, specifically simple local search and beam local search. We'll use the example of constructing a stable tour of rectangular pieces to demonstrate how these strategies work and iterate. Finally, we will cover simulated annealing and tabu search, discussing their basic elements, and applying them to the examples of the 8-queens problem and the coin payment problem, respectively. Let's begin!

**Best-First Search:**

Best-first search is an informed search algorithm that explores the most promising paths first. It uses a heuristic function to estimate the cost or value of each potential path. The heuristic function guides the search by assigning scores to states, indicating their desirability. The greedy approach is a specific type of best-first search that selects the most promising path at each step, without considering future consequences. This approach is efficient but may not always yield the optimal solution.

**A\* Algorithm:**

The A\* algorithm is an informed search algorithm that combines the benefits of both breadth-first search and best-first search. It uses both the cost to reach a state and the estimated cost to reach the goal from that state. The algorithm maintains two functions for each state: g(n) - the cost to reach the state, and h(n) - the heuristic estimate of the cost to reach the goal. The A\* algorithm explores the states with the lowest value of g(n) + h(n) first, ensuring an optimal solution if the heuristic function is admissible and consistent.

To illustrate the A\* algorithm, let's consider the example of finding the shortest path in a graph. We have a graph with nodes and edges, where each edge has a weight. The goal is to find the shortest path from the start node to the target node. The A\* algorithm uses the sum of the cost to reach a node and the estimated cost to reach the target node as the priority for exploration. By choosing the node with the lowest priority, it gradually expands the search space until it reaches the target node.

Here's a simple implementation of the A\* algorithm in Python:



In this implementation, the heuristic function provides an estimate of the remaining cost from a given node to the goal. The distance function calculates the cost of moving from one node to its neighbor. The get\_neighbors function retrieves the neighboring nodes of a given node. The reconstruct\_path function reconstructs the path from the came\_from dictionary.

**Local Search Strategies:**

**a) Simple Local Search:**

Simple local search is a strategy that explores the neighborhood of a given solution by making small modifications to it. It iteratively moves from one solution to a better neighboring solution until it reaches a local optimum. In the provided example of constructing a stable tour of rectangular pieces, simple local search would involve swapping adjacent pieces to improve stability.

**b) Beam Local Search:**

Beam local search is another local search strategy that focuses on exploring multiple paths simultaneously. It maintains a set of the best solutions, known as the beam, instead of just considering one solution. By selecting the most promising solutions at each step, beam local search can escape local optima and explore a larger portion of the search space.

**Simulated Annealing:**

Simulated annealing is a probabilistic metaheuristic inspired by the annealing process in metallurgy. It allows the algorithm to explore solutions that are worse than the current solution, providing a chance to escape local optima. Simulated annealing gradually reduces the probability of accepting worse solutions over time, mimicking the cooling process in annealing. It is particularly useful in problems with rugged search spaces and multiple local optima.

Let's apply simulated annealing to the 8-queens problem. The problem is to place eight queens on an 8x8 chessboard such that no two queens threaten each other. Simulated annealing starts with an initial random configuration of queens and iteratively makes random moves. It accepts moves that improve the solution or even those that worsen it with a certain probability based on the temperature parameter. As the temperature decreases, the algorithm becomes more selective and eventually converges to a solution.

**Tabu Search:**

Tabu search is a metaheuristic that uses memory structures called tabu lists to guide the search. It prevents revisiting recently explored solutions, encouraging the exploration of new regions of the search space. Tabu search maintains a short-term memory of recently visited solutions and forbids their selection. This mechanism helps the algorithm avoid cycles and escape local optima.

For the example of paying a sum S using n coins of values vi, tabu search could be applied by considering different combinations of coins and avoiding solutions that have been recently explored. The algorithm iteratively explores the search space by making moves that improve the solution while respecting the constraints imposed by the tabu list.