INTELLIGENT CONTROL OF ROBOTIC SYSTEMS

Prof. M. Felix Orlando

Department of Electrical Engineering

Indian Institute of Technology Roorkee

Lecture 17: Introduction to Search Methods

Good morning, everyone. Today, we are going to see the introduction to search methods applied to robotic systems. Before getting into the search methods, we will see the outline of what we are going to study in this course or in this lecture. First, we are going to see the introduction to search methods and their importance, their evolution, and then the major two types of search methods, which are breadth search method and depth search method, which are precisely breadth-first search method and depth-first search method, which involve first-in-first-served and last-in-first-served approaches.

Okay, so let us see the search methods involved in robotics. Search algorithm. Search algorithms are algorithms that enable robotic systems to navigate or locate paths effectively in their work volume. The importance or the significance of the search methods is for tasks such as obstacle avoidance, exploration, and autonomous navigation. It directly impacts a robot's efficiency, safety, and reliability in dynamic environments, where both the obstacles and the targets vary with respect to time.

Now, coming to the evolution in search methods, the early influence started in the year 1959, when breadth-first search method and depth-first search method were developed and formalized as part of early AI research, building on graph theory concepts by researchers like Edward and Claudette. Then we see the heuristic search that was developed in the year 1968, which is a popular technique called the A* algorithm by Peter Hart, Nils Nilsson, and Bertram Raphael. at SRI International, marking a significant advancement in informed search. Then the modern integration started in the year 1990s onwards, where the integration with AI led to adaptive search algorithms, and the contributions made by Stuart Russell and Peter Norvig in their book Artificial Intelligence: A Modern Approach show significant growth.

The integration of search algorithms with AI is shown very well in that book. Now, the key milestones associated with search methods are the development of SLAM, which is simultaneous localization and mapping, incorporating search methods with contributions

by researchers like Sebastian and Huff. Next, we come to the fundamental formulation of a search or planning problem. Before entering into the formulation of the search problem, we need to define certain parameters associated with this formulation of the search algorithm. First, we define a state.

We know that it is a distinct situation for the world, which is represented by a vector x or by a term x. The state space is formed by the possible states, forming the state vector space. Actions are means of transforming the world, denoted by U. That means through this action, the system will transform from one state to another. For example, from state X1 to state X2. This transformation is

represented by a state transition function, say f, which provides the new state by taking the current state and current action as its input. Thus, the state transition functional equation is given by $x' = f(x, u)$

Now, having the action, there is an action space, which is the set of all possible actions over all states. Thus, the action space is represented by U, which is equal to $U = \bigcup_{x \in X} U(x)$

where X is a state belonging to the state space X. Here, U(X) is the action space specifically for the state X. Now, let us talk about goal states, which are the desired states for which we are applying actions through search and planning algorithms. It is denoted by X_G, $X_G \subset X$. Now, let us see an example of formulating a search or planning problem. So, the problem statement is as follows.

A robot moves on a grid with integer coordinates. We have a grid for the robotic system, and let us say the robot is located in the center, which means let us say it is at (0, 0), the location of the robotic system. Then it has to move either up, down, right, or left, depending on the grid locations. The grid locations are nothing but the states. The robot is moving to different states, which are nothing but the locations represented in this 2D grid.

So here, in this schematic, we see a state transition graph. It represents the movements as transitions between grid points. So the mathematical formulation goes by this. The state space is a set of all grid points. Which are nothing but the integer pairs. $X = \{(i, j) : i, j \in \mathbb{Z}\}$ The action space here is nothing but the set of all possible movements.

Obtained through the action U, where U (capital U) is the set of points. Up. Go up from the current location. Go down from the current location. Up.

Go right and go left. These are the actions which come under the set U, called the action set U or action space, formed by various actions to navigate and reach particular grid points. So, the state transition equation is now given by $f(x, u) = x + u$

where X belongs to the state space and U belongs to the action space. The initial state for the robotic system on the grid, here taken as the origin (0, 0), is the initial state, meaning the initial position of the robotic system.

The goal state can be given as a grid point, which can be a single point or a set of target points serially, which can be reached first, second, third, fourth, and fifth, or a single target point. Now, understanding systematic search: we must understand the term 'systematic' in search algorithms. Systematic search ensures all reachable states are explored, which is critical for determining the existence of solutions in finite or infinite state spaces. Here, we mostly use the graph search algorithm, where the state transition graph is revealed incrementally as actions are applied.

Not as a complete one. The complete map will not be revealed in the beginning. The map is getting explored incrementally. Incrementally as the robotic system moves from one state to the another state and it gets incremented. The systematic search in finite graphs we will see that now.

And it is important to ensure that the algorithm avoids infinite loops and redundant exploration. That is in the case of finite graphs because we must make sure that finite graphs based systematic search ensures the algorithm is avoiding infinite loops and redundant or excessive exploration of this same state point or the location. And it should also guarantee that the algorithm explores every reachable state in the state space. And this way, this algorithm can determine in the finite time whether a solution exists or not. Now, let us see the systematic search in the graphs involving infinite graphs.

Here, systematicity requires finding a solution in finite time if it exists. If the solution exists then we need to find the solution in finite time. That is the systematicity expected in the infinite graph based search algorithm. So we relax our requirements for systematicity by allowing the algorithm to search indefinitely if no solution exists. Here

One requirement is that every reachable vertex must eventually be explored as the number of iterations increases. This algorithm continues with the assumption that the number of vertices is countably infinite. Even though we say infinite, it could be countably infinite in this infinite graph search algorithm. So let us see this schematic

figure: A indicates that from the initial state, it goes toward the solution in one direction. Because of this, most of the space around this state or the reaching direction remains undiscovered.

Hence, the systematicity is compromised in Figure A-based search, a unidirectional search. Whereas in Figure B, we can see that from the initial state, the search happens in all directions through wavefronts, which could be circular or square wavefronts. It spreads so that in all directions, the goal or the solution is searched from the initial state. So this type of search through wavefronts ensures broader coverage, and no state is revisited in this case; hence, this type of search through wavefronts is systematic.

Now let us talk about the general forward search. At any point during a search, there are three kinds of states. During the search, there are three kinds of states. One is unvisited, as the name indicates. These are the states that have not been visited yet.

Initially it is every state except the initial state XI. Now there is another state called Dead state. Which are the visitor states. But visiting and recording them.

Or exploring them. Will not provide you any information. And there are alive states. Which are nothing but the encountered state. But possibly has unvisited.

Next neighboring state. These alive states are. Stored in priority queue. P. Using a priority function. Again.

These are alive. States. Are stored in a priority queue. Called. P, which is an array.

Using a priority function. The only significant difference. Between various search algorithms. Is the choice of this priority. Function.

How we can save or arrange the queues or queues of the states. First in, first out is one basic option. FIFO, first in, first out is one basic option here to have this choice function priority. In first in, first out, States are processed in the order they are added into the Q or the priority queue P. Now, coming to the general overview of this algorithm.

First, we start with the initial state. That means P starts with the initial state X_i, and there is a termination condition that we must also set, where the while loop continues until P is empty or a solution is found. Next, there is also a failure scenario where P becomes empty without finding a goal state. P array where all the elements of this array get exploded, but we could not find a solution.

So, it is a failure scenario. The failure scenario in infinite graphs is such that the algorithm may not terminate if a reachable portion of X is infinite. That is also something we must consider. Now, let us have the pseudocode or the algorithm. First, record and then remove the highest-ranked state X from the QP.

That is a priority QP. Check if X is a goal state. If yes, report success and terminate the process of searching. Expand the state for each action that belongs to the action space U of X as the input. That is the state X as the input.

Compute the next state X new which is equal to the function of the current state and the current action. Then, determine if X is unvisited, insert X new into the P array. That is the priority array. Else, skip X new since it is either dead or already in P. Now, let us see the particular forward search methods. which are nothing but the breadth first search method or depth first search method.

There are two methods which we are going to see. The first one is breadth first search method. The concept here is it explores all the states at the present depth level before moving on to states at the next depth level. That means if it is initial state, It will search here and when it is coming here, it will search here again before moving to the next state level.

So, it will search here and then it takes the direction, go up, go next, then search exploring here fully, then go up, then search. So, this is the way, this is the concept utilized here. So, it explores all the states at the present depth level before moving on to states at the next depth level. So, the priority Q follows this scenario which is first in first out. So, the algorithm here is first initialize the priority vector or array P with initial state X i. While P is not empty X

Record and then remove the highest-ranked state, which is nothing but the oldest state. Check if it is a gold state or X belongs to the XG set, that is, the XG gold set. If yes, report success and terminate. If not, expand the state X and add unvisited states to P. I repeat, if not, expand the state X and add unvisited states to P. Now, let us see the example. So, we have the grid with the initial state being X 0, 0—that is, the point is the integer point 0, 0, which is the grid point initial state, and the goal state is 2, 2.

So, the exploration order here is: it starts from 0, 0 and then goes to 0, 1, then 0, 2—that is, it starts from here. So, this is x; this is y. It starts here, then goes here, then it goes 0, 1; 0, 2; then 0 Moves here. 1, 0. Then 1, 1.

Okay. So, accordingly. Right. Then it moves. So, like that.

Then it reaches the 2, 2. Alright. So, this is the exploration order in this case. That is the breadth-first search approach. Then let us talk about the depth-first search method.

Here the concept is to explore as far down a branch as possible before backtracking.

So the priority Q here is last in, first out. That is the priority Q strategy. LIFO, that is last in, first out.

So, the algorithm. It states like this. Initialize P. Priority vector with initial state X, I. While P is not empty, perform this. Record and then remove the highest-ranked state.

Most recently added state. First do with that because it is a LIFO-based strategy. Check if it is a goal state. That is, X belongs to the goal set XG. If yes, report success.

Expand the state X and XG. Unvisited states to P. So, the example here is having the grid with the initial state 0,0 and the goal state 2,2. So, here you can go in this manner: start from 0,0, then move to 1,0, then 2,0, then 2,1, and 2,2. So, this is faster as compared to that of the Breadth-first search method.

Now coming to the A* search method. So the concept here is to combine the cost to reach a state with a heuristic estimate of the cost to reach the goal point. Cost plus heuristic estimate. of the cost to reach the goal—not only the cost, not only the heuristic approach, but a combination of both the cost as well as the heuristic estimates of the cost will help to reach. For example, here is a goal: we can reach this one, this one, this one, and then go here, go here, or go here, go here, then come here, this, this, and this to reach. So this approach tells us to reach in this fashion.

So instead of reaching this way to the goal or this way to the goal, the A* method tells us to reach in this fashion to reach the goal point, which means a combination of cost as well as the heuristic estimate. That is the motto or the idea of the A* search method. So the priority queue is states that are ordered by f(x), where f(x) is a function given by g(x) and h(x). It is a combination of g(x) and h(x). As I told, it is a combination of cost as well as the heuristic estimate of the cost. So g(x) is the cost from the start state to x.

And h(x) is a heuristic estimate of the cost from x to the goal. The algorithm for this A* approach is as follows. First, initialize the priority queue P with the initial state XI with f(XI) equal to h(XI). Because initially your g(x) is 0.

So, while p is not empty while the priority q array is not empty record and then remove the highest ranked state which is the lowest of f of x. Check if it is a gold state that means the current state belongs to the gold state vector. If S reports success, they expand state X and add unvisited states to the vector P with updated F of X. Now, again coming to the grid example. So, the example here is we have the initial state 0, 0 and the goal state again 2, 2. Here, we use the heuristic Manhattan distance. Which is similar to the Euclidean distance.

And here the exploration order is. From 0 to 0. 1 comma 1. 2 comma 2. So we have this grid.

0 0. So reach here. Then reach here. Because it is a combination of. okay that gives you a strategy which is both cost as well as the heuristic of the estimated cost will help you to have the action to move from one state to the other state and hence you can reach the goal state much effectively with less time now coming to the comparison of

Breadth-first search approach, depth-first search approach, and the A* search method. Now, coming to the breadth-first search method, the exploration strategy is level-order exploration, where all nodes at depth D are expanded before nodes at depth D plus 1. Here, the completeness is indicated in such a way that it is complete in finite state spaces, finds the shortest path in unweighted graphs, and the optimality is that it guarantees the shortest path in unweighted graphs, thereby ensuring optimality. The space complexity here is high as it stores all nodes in the

current depth, and due to space complexity, we have time complexity that is given by O(B^D), where B is the branching factor and D is the depth of the shallowest goal. Now, let us see with the depth-first search approach what happens there with these metrics. The exploration strategy here is deep exploration and goes as far as possible down one branch before backtracking. That is why it is named the depth-first search algorithm.

And the completeness, when we see the completeness of this algorithm, is not complete in infinite spaces or with infinite paths; it can get stuck in loops. Now, coming to the optimality metric, this algorithm does not guarantee finding the shortest path, and it depends on the order of exploration. Now, the other two metrics: space complexity is low here because it stores only the path from the root to the current node instead of storing so many states. And the time complexity is O(B^M), where B is the branching factor.

And M is the maximum depth of the search. Now coming to the A star search. The exploration strategy. In comparison to. The breadth first and depth first strategies.

So here it is. Cost plus heuristic based exploration. And thereby it balances exploration and exploitation. And coming to the completeness metric we can say that complete if the heuristic H of X is admissible that is never overestimates the true cost. And coming to the optimality metric it guarantees the shortest path if the heuristic H of X is

is admissible and consistent. And coming to the space and time complexities, the space complexity is very high because it stores all the nodes generated and the time complexity is given by O of BD in the worst case, but often more efficient than breadth first search approach due to heuristic guidance. Heuristic guidance makes this A star search algorithm effective as compared to the two other search approaches and with this I thank you all. By this I can say that we have concluded this lecture in a way that we have studied the introduction to search algorithms, the history, how it is evaluated and then we have seen the importance of search algorithms. And finally, we have seen three algorithms, which are nothing but the breadth-first search algorithm, depth-first search algorithm, and then A star algorithm.

And we have also compared. Thank you so much.