# CSE 571: Artificial Intelligence - A Hands-On Experience

Pavan Kumar Raja
MS in Computer Science, School of Computing and Augmented Intelligence, ASU.
praja3@asu.edu

## ABSTRACT

A semester-long project aimed at giving a detailed hands-on experience in the algorithms used in Artificial Intelligence. This project included 7 in-class activities along with 3 programming projects aimed at getting experience in building software for 3 major learning milestones in AI (Search, Planning, and Q-learning algorithms). All the activities and projects were completed in an individual capacity.

## 1 INTRODUCTION

This project gives a detailed hands-on experience on all the aspects of AI learning in the course CSE 571. The activity included 7 hands-on sessions to help students get familiarized with AI algorithms. Topics for these sessions included search algorithms to find an optimal path from an initial state to a goal state, Planning Domain Definition Language along with planning graph technique to make plan search optimal, Policy evaluation algorithm to evaluate a given policy, Value Iteration algorithm to come up with an optimal policy to traverse a state graph, Q-learning for policy generation in a stochastic environment, Probabilistic inference, Maximum Likelihood Approximation, Maximum A Posteriori approximation and prediction using Bayesian inference. Furthermore, I successfully completed project which had 3 parts aimed at critical junctions in the AI course. These projects helped me in gaining experience in developing AI algorithms in a controlled environment (simulations). These programming projects included concepts such as search paradigm in AI, Planning using PDDL and its optimization, and Reinforcement learning[2].

## 2 HANDS-ON SESSIONS AND PROJECTS

The following sub-sections go over detailed learning, implementation, and developments involved in the project.

### 2.1 Hands-On session

(1) ***First Hands-On session:*** This activity concentrated on search algorithms in AI namely Breadth-First Search, Greedy Best First Search, Uniform Cost Search, and A* search[2]. Here, given a search tree and heuristics (if applicable), I was able to learn how the above algorithms behaved in terms of node expansion and search path returned from source to target.

(2) ***Second Hands-On session:*** This activity concentrated on the Planning graph algorithm used to come up with heuristics for action plan generation. In this activity, given a Planning Domain and its setup, I was able to find the next states and mutex links between each action of the domain using planning graph logic[2].

(3) ***Third Hands-On session:*** This activity concentrated on the Policy evaluation of an MDP problem. Here, given a policy, I was tasked to use the policy iteration formula to find the value of each state using the given policy for multiple iterations. Using this formula we'll eventually converge on the maximum possible values for each state for the policy given[2].
Policy evaluation[2]: $V^{\pi}_{k+1}(s) = \sum_{s'} T(s, \pi(s), s')[R(s) + \gamma V^{\pi}_{k}(s')]$

(4) ***Fourth Hands-On session:*** This activity concentrated on the Value Iteration of an MDP problem. Here, given an MDP I was tasked to use the Value iteration formula to find the policy and value for a few iterations. Using this formula we'll eventually converge on the optimal policy and value for it.[2]
Value Iteration[2]: $V_{k+1}(s) = max_a \sum_{s'} T(s, a, s')[R(s) + \gamma V_k(s')]$

(5) ***Fifth Hands-On session:*** This activity concentrated on the SARSA update algorithm (To find maximum Q value, given the policy) and Q-learning algorithm (To find optimal policy) of MDP. In this activity, I was able to successfully use these algorithms/formulas to get a good understanding of reinforcement learning in AI.[2]
SARSA[2]: $Q^{\pi}(s, a) = (1 - \alpha)Q^{\pi}(s, a) + \alpha[R(s) + \gamma Q^{\pi}(s', a')]$
Q-learning[2][1]: $Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha[R(s) + \gamma max_{a'} Q(s', a')]$

(6) ***Sixth Hands-On session:*** This session involved the evaluation of Bayesian networks and the use of marginalization and d-separation techniques to reduce the complexity of calculating joint probabilities of a random variable of the network.[2]

(7) ***Seventh Hands-On session:*** This session provided hands-on experience in working with probability inference techniques in AI. This covered topics such as Maximum Likelihood Approximation, Maximum A-Posteriori approximation, and prediction using Bayesian inference.
Maximum Likelihood[2]: $h_{ML} = argmax_{h_i} P(d|h_i)$
Maximum A-Posteriori[2]: $h_{MAP} = argmax_{h_i} P(h_i|d)$
Bayesian inference[2]: $P(X|d) = \sum_{h_i} P(X|d)P(h_i|d)$

### 2.2 Programming projects

Programming projects were designed on ros melodic framework and python programming language along with Gazebo environment to visualize simulation of the action plans generated as part of the projects. Gazebo AI simulation environments that I worked on were canWorld, bookWorld, and cafeWorld[6]. The 3 projects undertaken in this environment can be described as:

1. ***Project 1 - Implementation of Search algorithms[3]***: Here given a canWorld environment with a start location, target location, and multiple cans blocking the path, we have to find an optimal path from source to target. The algorithms that I implemented are,

*Code base:*
*1. AI algorithms*

(1) BFS (breath first search[2]): A level search algorithm where the goal (or target in this case) was searched level-by-level in the graph.
(2) UCS (Uniform Cost search or Dijkstra's algorithm[2]): In this search algorithm, the next node to expand is selected such that it has the minimum cumulative cost from source to node in fringe.
(3) GBFS (Greedy Best First Search algorithm[2]): This search algorithm requires a heuristic for each node which gives an estimate of how far the goal is from that particular node. Heuristics are admissible - *gives optimal cost path* if it doesn't overestimate the cost to the goal (heuristic of goal node is 0). The node selected to expand next has the minimum heuristic value among the nodes in the fringe.
(4) Astar[2]: A start search is a combination of UCS and GBFS, where the node selected has a minimum value of a function between the cost of source-node and the heuristic of the node.
The default heuristic used by me was the Manhattan heuristic (because it is always admissible), where the total number of squares traveled horizontally and vertically from the current square to reach the destination square is considered.
(5) Custom astar: Similar to A star but I had the choice of coming up with a heuristic that will outperform Manhattan.
I used a factor(because 2 move actions needed to be performed to move to the next cell) of Euclidean distance, and observed it indeed outperformed the Manhattan heuristic.
heuristic: $h(s) = 2 * \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$

Action plans generated using the above algorithms were executed Gazebo canWorld environment to verify if they had generated the correct plans to reach the destination location.

2. ***Project 2 - Planning[5]***: This project gave an overview of PDDl problem definition development, action design (including precondition and effect), domain-independent goal design, and refinement of higher-level planning to lower-level bot commands. For this project, Gazebo was used exclusively to verify our planning domain definition and refinement of the high-level plan to lower-level gazebo bot understanding commands. The planning domain/environment was bookWorld and cafeWorld and is defined with following attributes[6]:

(1) *Objects and Goals*: books - bins, food - table.
(2) *Types of Objects*: book types and food types
(3) *Sizes of Objects and Goals*: Sizes of books, bins, food, and table. Each book/food is compatible with bins/table only when their sizes match.
(4) *Number of Object Types*: The number of distinct object types.
(5) *Number of Objects*: This is the total number of objects of each type and each size in an environment.
(6) *Number of Goals*: The total number of goals generated is the number of object types * the number of sizes of the objects.
(7) *Load Locations*: Load locations are the locations of turtlebot from which it can pick up an object.

The extra thing I did with this project was a. Using universal and existential quantifiers to write the goal formula, this makes the goal definition domain-independent and defines the goal for both bookWorld and cafeWorld[6]. b. Improved high-level refiner (One that converts plans to actions) and lower-level search algorithm by using a-star heuristic search for both[6].

3. ***Project 3 - Reinforcement learning (Q-learning)[4]***: This project was aimed at introducing myself to reinforcement learning using the Q-learning formula for the stochastic world (MDP). The environments used for this project were the same as Project 3 and had the same definitions. Development of this project mainly included computing Q-values using the formula[2][1]:
$Q(s,a) = (1-\alpha)Q(s,a) + \alpha[R(s,a,s') + \gamma max_{a'} Q(s',a')]$
Computing cumulative reward for each step by using the formula[2]:
$R_{cumulative} = R_{cumulative} + \gamma^{step} R(s,a,s')$
And 1% learning rate decay using formula[2]:
$\epsilon = max(0.99\epsilon, \epsilon)$
Both the environment is defined as an MDP. So, each of the actions had success and failure rewards, and I could only evaluate a state once the robot visits it. Each of the environments was trained on 500 episodes with 500 max steps so that we could derive policies to reach the goal and converge on optimal Q-values of each observed state.

## 3 RESULTS

The main idea of the programming projects was to get familiarized with the algorithm implementation and see a real-time comparison of many algorithms used in AI. The following subsections give such evaluation of each project.

### 3.1 Project 1 - Implementation of Search algorithms

The search algorithms can be evaluated based on parameters like nodes expanded and time taken to generate the source-destination path. Nodes expanded and Time taken graphs shows a comparison of the performance of these algorithms. From the results we can see that A-star with a good heuristic will perform better than other algorithms in terms of nodes expanded (Fig.1) and time taken(Fig.2) while being optimal (In terms of cost)[2].

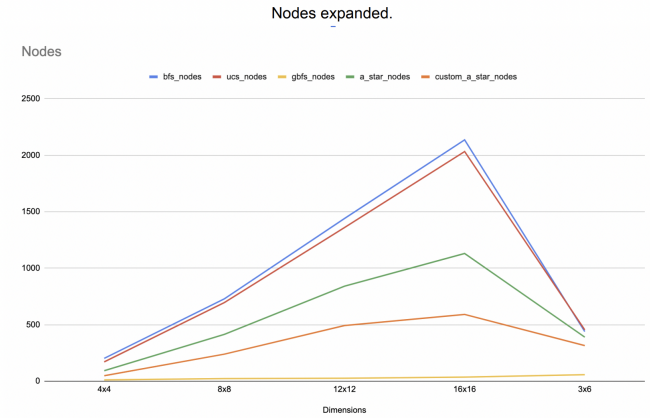


**Figure 1:** Nodes expanded by each algorithm

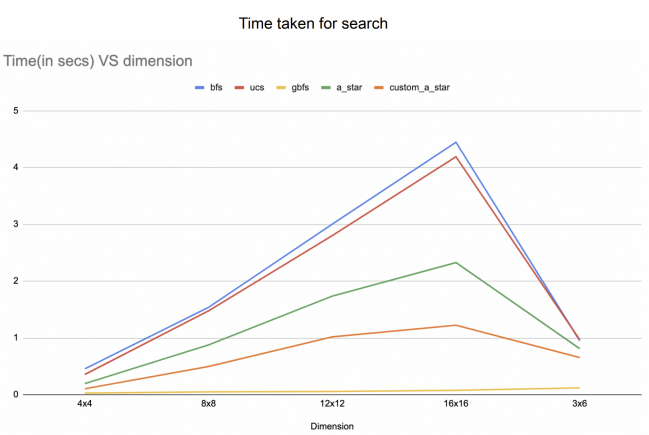


**Figure 2:** Time taken by each algorithm

## 3.2 Project 2 - Planning

The evaluation of the project was done based on if the robot generates an optimal action plan to reach the goal state[2] (Fig.4). From the results in Fig.3, we can see that plans are not in the higher level but instead all the higher-level actions are refined to lower-level actions that are understood by the gazebo robot[6]. This gives an interesting problem where we not only need to have an optimal action plan but also optimal refinements.

```
Object_types; object_count; seed; exception; refined_plan; environment
1; 1; 1; None; [('move', ['MoveF', 'TurnCCW', 'MoveF', 'MoveF', 'MoveF',
'MoveF', 'MoveF', 'MoveF', 'MoveF', 'MoveF', 'MoveF', 'TurnCW', 'MoveF',
'TurnCCW', 'MoveF', 'TurnCW', 'MoveF', 'TurnCCW', 'MoveF', 'MoveF',
'TurnCCW', 'MoveF', 'MoveF', 'MoveF', 'TurnCW', 'MoveF', 'MoveF', 'TurnCW'], [0.5,
7.0, 'EAST']), ('pick', 'hamburger_1'), ('move', [], [0.5, 7.0, 'EAST']),
('move', ['TurnCW', 'MoveF', 'MoveF', 'MoveF', 'TurnCCW', 'MoveF', 'MoveF',
'TurnCW', 'MoveF', 'MoveF', 'MoveF', 'MoveF', 'TurnCW', 'MoveF', 'MoveF',
'TurnCCW', 'MoveF', 'MoveF', 'TurnCCW'], [0.5, 3.0, 'EAST']), ('place',
'hamburger_1', 'table_1'), ('move', ['TurnCCW', 'MoveF', 'MoveF', 'MoveF',
'TurnCW', 'MoveF', 'TurnCCW', 'MoveF', 'TurnCW', 'MoveF', 'TurnCCW',
'MoveF', 'MoveF', 'TurnCCW', 'MoveF', 'MoveF', 'TurnCW', 'MoveF', 'MoveF',
'TurnCW'], [0.5, 7.0, 'EAST']), ('pick', 'hamburger_2'), ('move',
['TurnCW', 'MoveF', 'MoveF', 'TurnCCW', 'MoveF', 'MoveF', 'TurnCW',
'MoveF', 'MoveF', 'MoveF', 'MoveF', 'TurnCW', 'MoveF', 'MoveF', 'TurnCCW',
'MoveF', 'MoveF', 'MoveF', 'MoveF', 'TurnCCW'], [0.5, 2.0, 'EAST']),
('place', 'hamburger_2', 'table_2')]; cafeWorld
```

**Figure 3:** Plan generated for cafeWorld (1x1)

```
(:goal
        (and
        (forall (?fd - food)
        (exists (?tb - table ?lc - location ?sz - size ?ty - food_type)
        (and
                (Food_At ?fd ?lc)
                (Table_At ?tb ?lc)
                (Food_Type ?fd ?ty)
                (Ordered ?tb ?ty)
                (Portion_Size ?fd ?sz)
                (Ordered_Portion ?tb ?sz)
        )
))))
```

**Figure 4:** Goal definition for cafeWorld

## 3.3 Project 3 - Reinforcement learning

The evaluation of project 3 was just to observe the convergence of cumulative reward generated by our action plans per episode for each of the world. Results in Fig.5 shows that a good implementation of Q-learning is very effective in finding the optimal policy and Q-value(weighted cumulative rewards) of MDPs, we can also verify that as Q-learning visits each state repeatedly it slowly converges to the optimal policy leading to convergence of Q-values at each state[1].

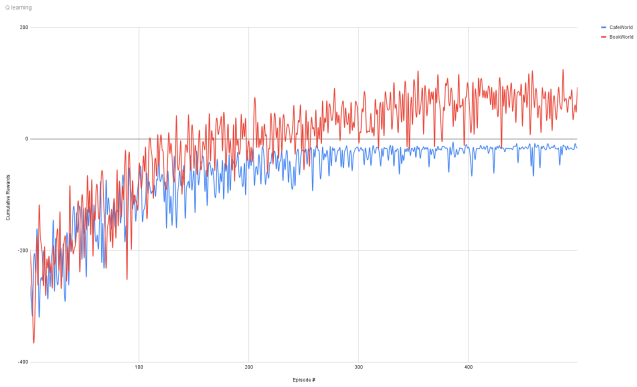

**Figure 5:** Cumulative reward across episodes

## 4 OUTCOME

Combining my learning from both hands-on sessions and programming projects, I have acquired in-depth knowledge of many AI algorithms in the domain of state search, planning, and learning. The hands-on sessions prepared me well by giving me experience in the iterative nature of it for the programming projects. By completing these programming projects I have acquired a strong understanding of the ros melodic framework and gazebo simulation environment. I found that these tools and frameworks are useful tools in running and evaluating AI algorithms.

Overall, I found this course had a huge learning curve, and the projects gave me a glimpse into implementation of AI algorithms.

## REFERENCES


[1] Maei Hamid, Szepesvári Csaba, Bhatnagar Shalabh, and Sutton Richard. 2010. Toward off-policy learning control with function approximation in Proceedings of the 27th International Conference on Machine Learning. (2010). https://web.archive.org/web/20120908050052/http://webdocs.cs.ualberta.ca/~sutton/papers/MSBS-10.pdf

[2] Stuart Russell and Peter Norvig. 2021. Artificial Intelligence - A Modern Approach, 4th Edition. (2021). arXiv:0-13-461099-7

[3] Naman Shah, Rushang Karia, Rashmeet Kaur Nayyar, Pulkit Verma, and Siddharth Srivastava. 2022. Programming assignment 1 template. (2022).

[4] Abhyudaya Srinet and Siddharth Srivastava. 2019. Programming assignment 2 template. (2019).

[5] Abhyudaya Srinet, Pulkit Verma, Rushang Karia, and Siddharth Srivastava. 2022. Programming assignment 2 template. (2022).

[6] Stanford Artificial Intelligence Laboratory et al. Robotic Operating System. (????). https://www.ros.org