AI Assignment - 3

SHOBHIT RAJ
2022482

Q1.

a) DIRECT SAMPLING - In involves generating samples directly from the distribution given.

Strengths: It is very efficient & easy to do if distribution is known & works well with simple datasets or when probabilities are given.

Weakness: Inefficient for high dimensional distributions where difficult to sample from joint probability.

REJECTION SAMPLING - Samples are drawn from a proposal distribution, then accepted or rejected based on a criteria involving target distribution.

Strengths: more complex distributions can be handled.

Weakness: If skewed or multi-modal distributions, then inefficient if acceptance rate is low.

GIBBS SAMPLING - A Markov Chain method that samples each variable conditional on the current values of the others.

Strengths: suitable for complex, joint probability distributions.

Weakness: More iterations are required for convergence & is sensitive to initial starting point.

Comparison in context of dataset:

Direct Sampling: Practical for this straightforward dataset as air travel/stress levels, etc. explicitly given.

Rejection Sampling: Less practical here due to limited complexity of the dataset but useful if need to estimate probabilities not given.

Gibbs Sampling: lead to overhead, & not justified for this small & simple dataset, not much valuable/useful here.

b) $P(\text{Leisure} \mid \text{Train}) = 0.4$

Total sample = 100

N.o. of people train prefer = 30

Expected no. of people traveling for leisure = $P(\text{leisure} \mid \text{Train}) \times$ N.o. of train travelers.

$= 0.4 \times 30 = \underline{12}$.

c) $P(\text{air}) = 0.8$

$P(\text{business} \mid \text{air}) = 0.2$

$P(\text{business} \mid \text{air}) = \dfrac{P(\text{business} \wedge \text{air})}{P(\text{air})}$

$\therefore P(\text{business} \wedge \text{air}) = 0.2 \times 0.8 = \underline{0.160}$ or $\underline{16\%}$.

d) Accuracy → refers to how close the estimate is to the true value. Increasing sample size helps reduce bias due to random sampling errors, improving accuracy.

Precision → refers to consistency of the estimate. Larger sample sizes decrease variability, leading to more precise estimates.

Implications for the dataset: Probabilities like $P(\text{Leisure} \mid \text{Train})$ are relatively small. With small sample size, random fluctuations may lead to over/under-estimation. Increasing sample size would provide more reliable estimate & overall improve confidence in the results when generalizing findings to the entire population.

Q2.

a) Random Variable:

B : Person reads books.
J : Person accesses academic journals.
C : Person participates in book clubs.

Statements:-

1. $P(B \vee J) = 0.910$
2. $P(J|B) = 0.400$ and $P(\neg J|B) = 0.600$
3. $P(C|B) = 0.320$
4. $P(J \wedge \neg B) = 0.227$
5. $P(\neg B \wedge \neg J) = 0.090$
6. $P(J|\neg B) = 0.716$
7. $P(C \wedge J) = 0.088$
8. $P(C \vee J) = 0.631$
9. $P(J|C) = 0.400$
10. $P(J) = 0.500$
11. $P(C|\neg B) = 0.004$

b) Probability axioms that are satisfied:

1. Non negativity → All probabilities ≥ 0.
2. Total Probability → Sum of all disjoint events = 1.

   Eg.- $P(B \vee J) + P(\neg B \wedge \neg J) = 0.91 + 0.09 = 1$.
3. All individual probabilities b/t 0 & 1.

c) Joint Probability Distribution table :-

B	J	C	Probability
0	0	0	0.0892
0	0	1	0.0008
0	1	0	0.2264
0	1	1	0.0006
1	0	0	0.2788
1	0	1	0.1312
1	1	0	0.1856
1	1	1	0.0874

P.T.O.

Given, $P(\neg B \wedge J) = 0.227$, $P(J|B) = 0.400$, $P(C \wedge J) = 0.088$, $P(C|B) = .32$

$P(\neg B \wedge \neg J) = 0.090$, $P(\neg J|B) = 0.600$, $P(C|\neg B) = .0044$

As, $P(\neg B) = P(\neg B \wedge J) + P(\neg B \wedge \neg J) = 0.317$.

As $P(B) + P(\neg B) = 1 \Rightarrow P(B) = 0.683$.

As $P(B \wedge \neg J) = P(\neg J|B) \times P(B) = 0.6 \times 0.683 = 0.4098 \approx 0.410$

As $P(B \wedge J) = P(J|B) \times P(B) = 0.4 \times 0.683 = 0.2732$

As $P(C \wedge B) = P(C|B) \times P(B) = 0.32 \times 0.683 = 0.219$

As $P(C \wedge \neg B) = P(C|\neg B) \times P(\neg B) = 0.0044 \times 0.317 = 0.001$

1. $P(\neg C \wedge \neg J \wedge \neg B) = P(\neg B \wedge \neg J) - P(C \wedge \neg J \wedge \neg B)$
   $= 0.090 - 0.0008 = 0.0892$

2. $P(\neg B \wedge \neg J \wedge C) = P(C \wedge \neg B) - P(C \wedge J \wedge \neg B)$
   $= 0.0014 - 0.0006 = 0.0008$

3. $P(\neg B \wedge J \wedge \neg C) = P(\neg B \wedge J) - P(C \wedge J \wedge \neg B)$
   $= 0.227 - 0.0008 = 0.2264$

4. $P(\neg B \wedge J \wedge C) = P(C \wedge J) - P(C \wedge J \wedge B)$
   $= 0.088 - 0.0874 = 0.0006$

5. $P(B \wedge \neg J \wedge \neg C) = P(B \wedge \neg J) - P(C \wedge \neg J \wedge B)$
   $= 0.41 - 0.1312 = 0.2788$

6. $P(C \wedge \neg J \wedge B) = P(C|B) \times P(B \wedge \neg J) = 0.32 \times 0.41 = 0.1312$

7. $P(B \wedge J \wedge \neg C) = P(B \wedge J) - P(C \wedge J \wedge B)$
   $= 0.273 - 0.0874 = 0.1856$

8. $P(B \wedge J \wedge C) = P(C|B \wedge J) \cdot P(B \wedge J) = P(C|B) \times P(B \wedge J)$
   $= 0.32 \times 0.273$
   $= 0.0874.$

d)

1. B & J → if $P(B \wedge J) = P(B) \cdot P(J)$

$P(B \wedge J)$ = 0.273 (from $P(B \wedge J \wedge C) + P(B \wedge J \wedge \neg C) + P(B \wedge \neg J \wedge \neg C)$)

$P(B)$ = 0.683 (from $P(B \wedge J \wedge C) + P(B \wedge J \wedge \neg C) + P(B \wedge \neg J \wedge \neg C)$)

$P(J)$ = 0.5

$\therefore P(B) \cdot P(J) = 0.3415 \neq P(B \wedge J)$

Hence, B & J are not independent.

2. J & C → if $P(J \wedge C) = P(J) \cdot P(C)$

$P(J \wedge C)$ = 0.088

$P(J)$ = 0.5

$P(C)$ = $P(C \wedge J \wedge B) + P(B \wedge \neg J \wedge C) + P(\neg B \wedge \neg J \wedge C)$
= 0.22

$\therefore P(J) \cdot P(C) = 0.11 \neq P(J \wedge C)$

Hence, J & C are not independent.

3. C & B → if $P(C \wedge B) = P(C) \cdot P(B)$

$P(C \wedge B)$ = 0.2186

$P(C)$ = 0.22

$P(B)$ = 0.683

0.22 X 0.683 = 0.15

$P(C \wedge B) \neq P(C) \cdot P(B)$

Hence, C & B are not independent.

Q3.

a) Let A - Adversial perturbation caused misclassification.
B - Backdoor attack caused misclasification
M - Misclassification alarm observed.

Initially, A & B are considered independent, so $P(A \cap B) = P(A) \cdot P(B)$.
However, M is a common effect caused by either A or B.
Using law of total probability:

$$P(M) = P(M|A)\,P(A) + P(M|B) \cdot P(B)$$

When we update our beliefs based on the new information about prevalence of backdoor attacks, we apply Bayesian Inference:

$$P(A|M) = \frac{P(M|A) \cdot P(A)}{P(M)}$$

b) 1. Prior:

$P(A)$ → prior probability of adversarial perturbations causing misclassification.

$P(B)$ → prior probability of backdoor attacks causing misclassification.

2. Likelihood:

$P(M|A)$ → probability of a misclassification alarm being raised given adversarial perturbations.

$P(M|B)$ → probability of a misclassification alarm being raised given a backdoor attack.

3. Posteriors:

$P(A|M)$ → Updated probability that adversarial perturbations caused the misclassification, given observation of alarm.

$P(B|M)$ → Updated probability that backdoor attacks caused the misclassification, given the observation of alarm.

c) Initially, A & B are independent, but after observing M, they become conditionally dependent.

The prior $P(B)$ increases, reflecting higher likelihood of backdoor attacks, and affected the posterior $P(A|M)$.

As $P(M) = P(M|B) \cdot P(B) \rightsquigarrow P(B) \uparrow \Rightarrow P(M) \uparrow$

and $P(A|M) = \dfrac{P(M|A) \cdot P(A)}{P(M)} \rightsquigarrow P(M) \uparrow \Rightarrow P(A|M) \downarrow$

This means that increased prevalance of backdoor attacks reduces the likelihood of adversarial perturbations being the cause of the misclassification. ~ "explains away" phenomena

Coding Questions

Q4. Bayesian Network for fare classification

Task 1 :

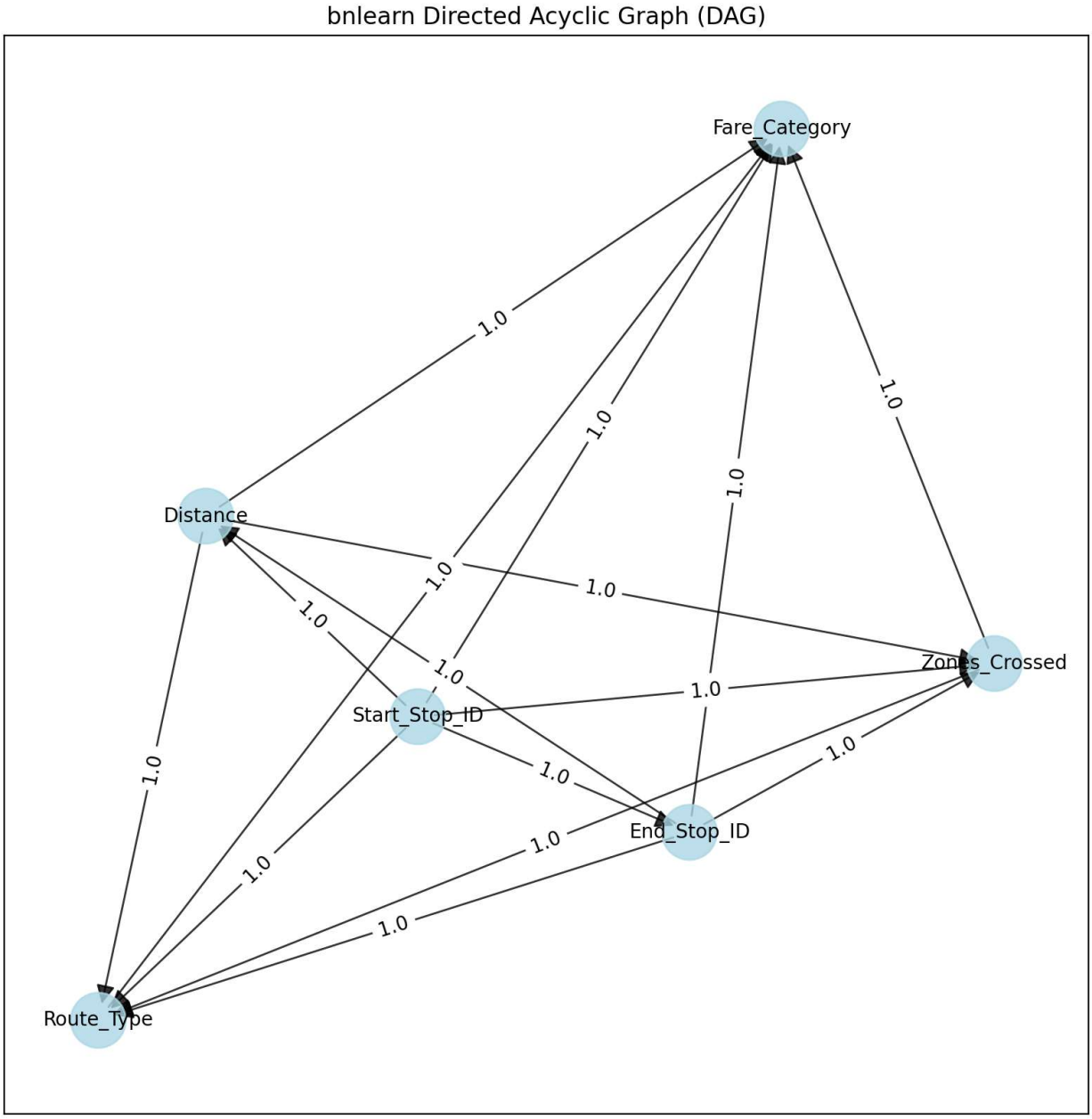
For Task 1, I constructed an initial Bayesian Network to predict fare categories using the provided features. I defined a Directed Acyclic Graph (DAG) where each node represents a feature or the target (Fare_Category), and the edges represent the conditional dependencies between them. The network includes all possible feature pairs (6C2 = 15 pairs), ensuring that each feature is connected to others that logically influence it. Specifically, Start_Stop_ID and End_Stop_ID influence various other features, including Distance, Zones_Crossed, Route_Type, and ultimately, Fare_Category. This structure allows the network to capture the relationships between the features and predict fare categories based on journey details.

The DAG was visualized using bnlearn's plotting functions, which helps in understanding how the different features are interrelated. After constructing the network, I applied parameter learning to fit the model to the data, allowing the Bayesian Network to learn the conditional probabilities and prepare for predictions.

The base model required 524.62 seconds to fit the data, which is set as the baseline for comparison.

The base model achieved an accuracy of 100% on the validation dataset, successfully predicting all 350 test cases correctly.

The visualization of the Bayesian Network is given below :-


bnlearn Directed Acyclic Graph (DAG)
Fare_Category
Distance
Zones_Crossed
Start_Stop_ID
End_Stop_ID
Route_Type
1.0

Task 2 :

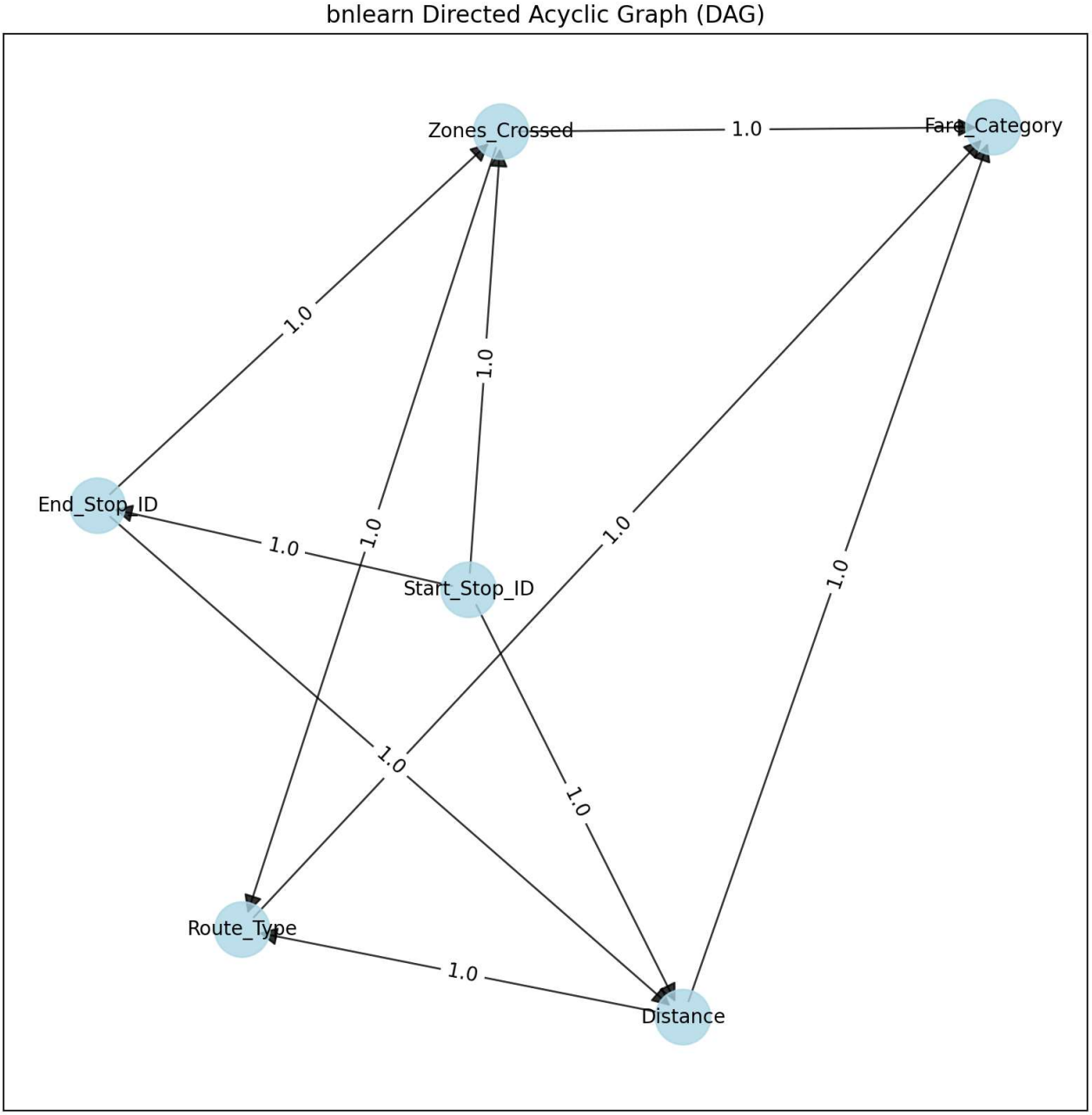
For Task 2, I applied pruning techniques to simplify the initial Bayesian Network and improve both efficiency and prediction accuracy. I started by identifying redundant edges in the network. Specifically, the edges from Start_Stop_ID to End_Stop_ID, from Start_Stop_ID to Fare_Category, and from Start_Stop_ID to Route_Type were removed, as they didn't contribute significantly to predicting the fare category. Similarly, the edges from End_Stop_ID to Route_Type, End_Stop_ID to Fare_Category, and Distance to Zones_Crossed were also pruned, as they were less relevant in terms of their influence on the target variable. This edge pruning process helped reduce the model's complexity and improved its computational efficiency.

Additionally, node pruning was performed on Fare_Category by simplifying its dependencies, ensuring that only the most relevant features remained connected to it. These changes resulted in a more efficient and faster model by reducing the number of edges and simplifying the conditional probability tables.

The pruned model required 265.81 seconds, nearly halving the time compared to the base model, demonstrating improved computational efficiency while maintaining accuracy.

The pruned model also achieved an accuracy of 100%, maintaining the predictive performance of the base model.

The visualization of the pruned Bayesian Network is given below :-


bnlearn Directed Acyclic Graph (DAG)
Zones_Crossed
1.0
Fare_Category
1.0
1.0
End_Stop_ID
1.0
1.0
1.0
Start_Stop_ID
1.0
1.0
1.0
1.0
Route_Type
1.0
Distance

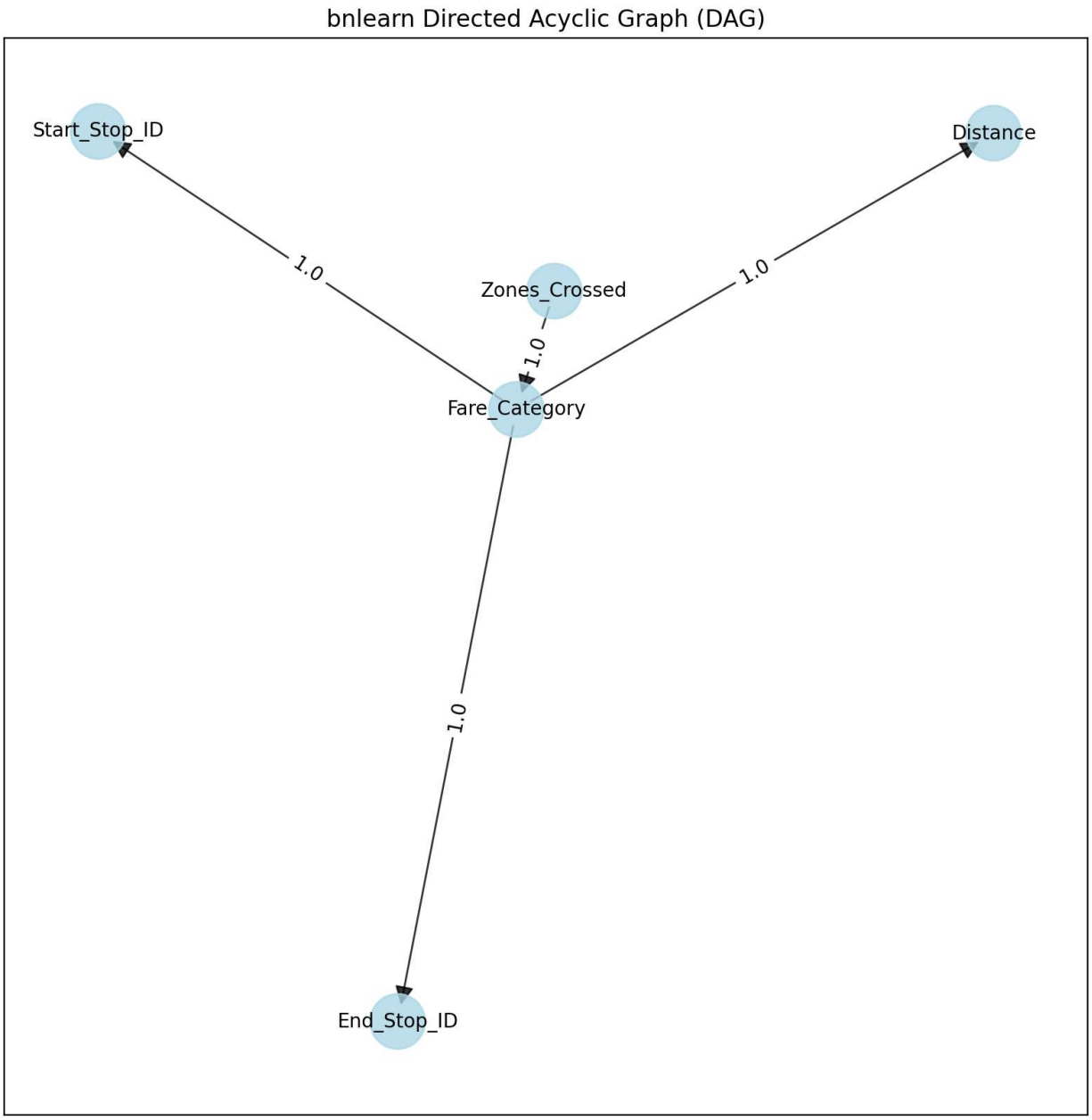
Task 3 :

For Task 3, the goal was to optimize the structure of the Bayesian Network by applying structure learning techniques. I used the Hill Climbing (HC) method from the bnlearn library to refine the network. Hill Climbing iteratively modifies the structure by adding, removing, or reversing edges, selecting the configuration that maximizes a chosen score (in this case, log-likelihood or Bayesian score). The initial network (from Task 1) served as the starting point, and the Hill Climbing method was applied to improve the network structure. This optimization aims to capture more accurate dependencies between features, leading to improved prediction accuracy.

The optimized model completed the fitting process in just 3.02 seconds, a dramatic improvement in efficiency over the base model, reducing computation time by more than 99% without sacrificing predictive performance.

The optimized model achieved an accuracy of 100%, successfully predicting all 350 test cases correctly.

The visualization of the optimized Bayesian Network is given below :-


bnlearn Directed Acyclic Graph (DAG)
Start_Stop_ID
Distance
Zones_Crossed
Fare_Category
End_Stop_ID
1.0
1.0
1.0
1.0

Q5.

(a) Hidden Markov Model (HMM) Formulation

The problem was modeled as a Hidden Markov Model (HMM) with:

- **State Space**: Each state represents a combination of a grid position (x,y)(x, y)(x,y) and a heading direction (N,S,E,WN, S, E, WN,S,E,W).
- **Transition Probabilities**: These depend on the movement policy. For the "random walk" policy, transitions are uniformly distributed among valid moves. For the "straight until obstacle" policy, the Roomba maintains its heading until it encounters a boundary.
- **Emission Probabilities**: The observed position is modeled as Gaussian noise centered around the true position, with the variance determined by σ2\sigma^2σ2.

This framework allows estimating the most likely sequence of states given a sequence of noisy observations.

(b) Implementation of the Viterbi Algorithm

The Viterbi algorithm was implemented to determine the most probable path of the Roomba:

- The **initial state** was set to the Roomba's starting position and heading.
- A **dynamic programming table** was built to store probabilities and backtrack pointers for each state and time step.
- Using **backtracking**, the most likely path was reconstructed.

This algorithm efficiently finds the optimal state sequence, given the noisy observations.

(c)

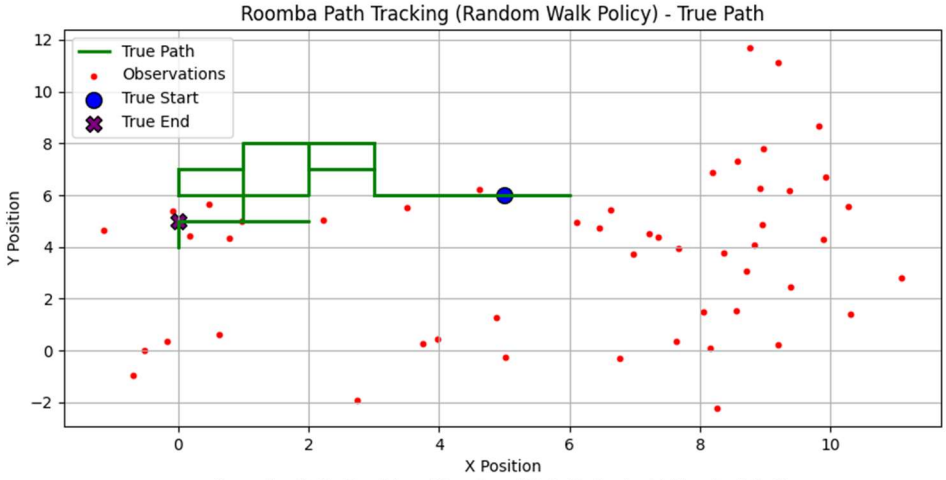
Seed Value : 111

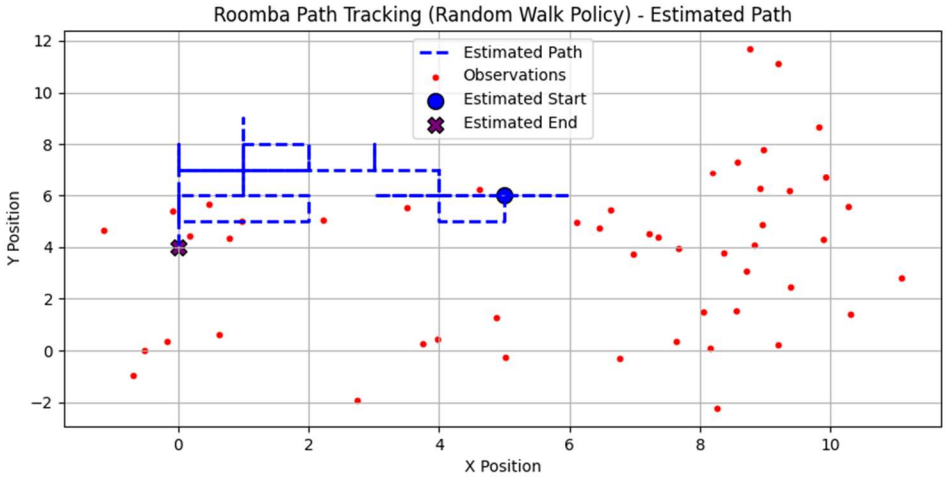
Processing policy: random_walk

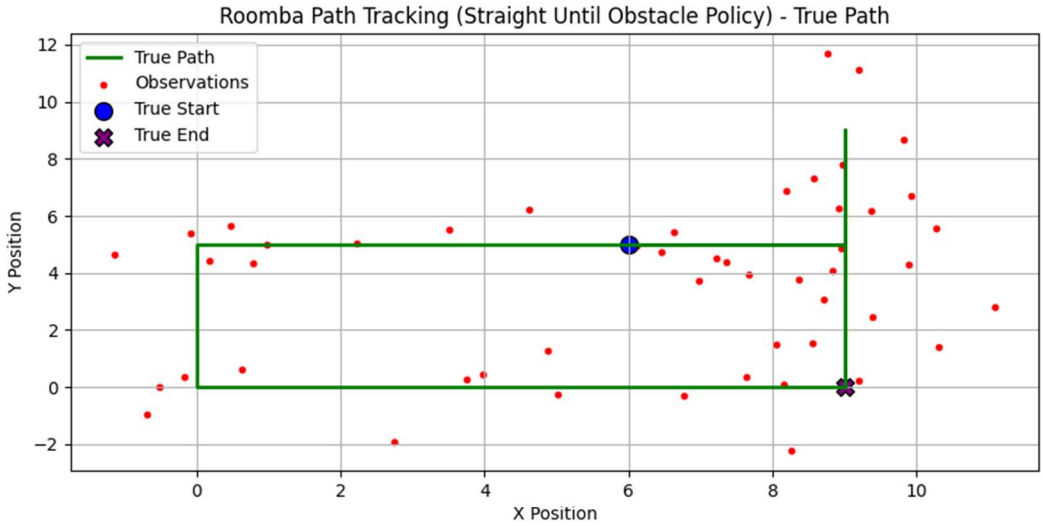
Tracking accuracy for random walk policy: 42.00%

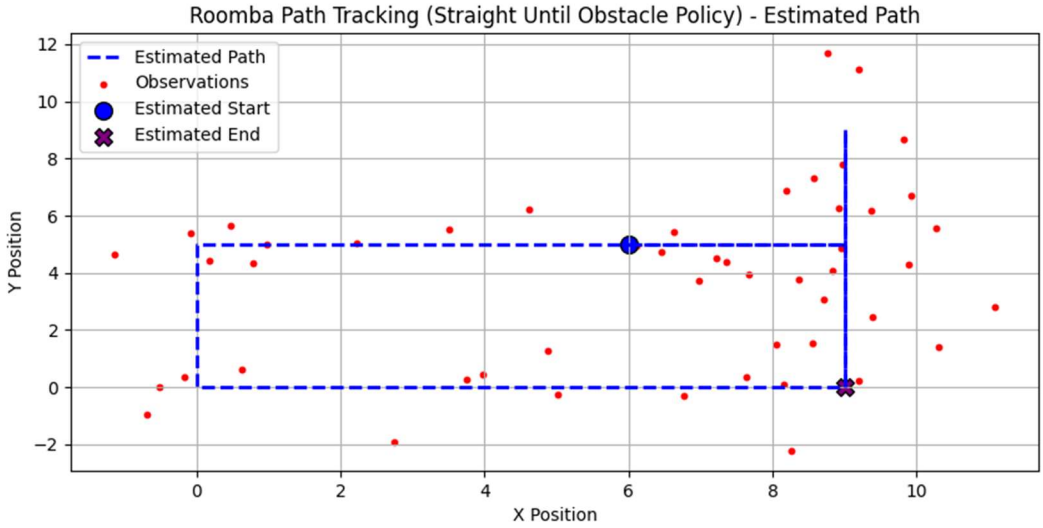
Processing policy: straight_until_obstacle

Tracking accuracy for straight until obstacle policy: 100.00%






Roomba Path Tracking (Straight Until Obstacle Policy) - True Path
True Path
Observations
True Start
True End
Y Position
X Position


Roomba Path Tracking (Straight Until Obstacle Policy) - Estimated Path
Estimated Path
Observations
Estimated Start
Estimated End
Y Position
X Position

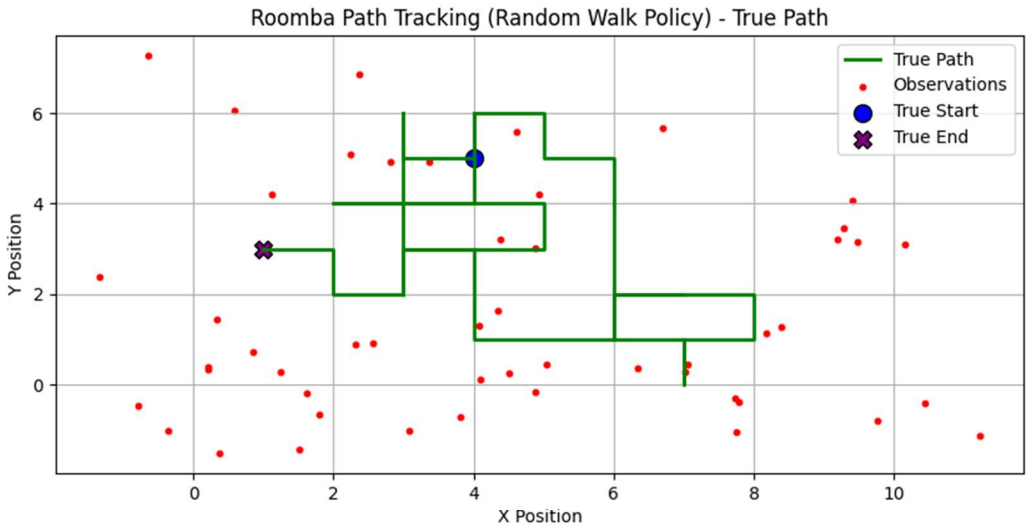
Seed Value : 666

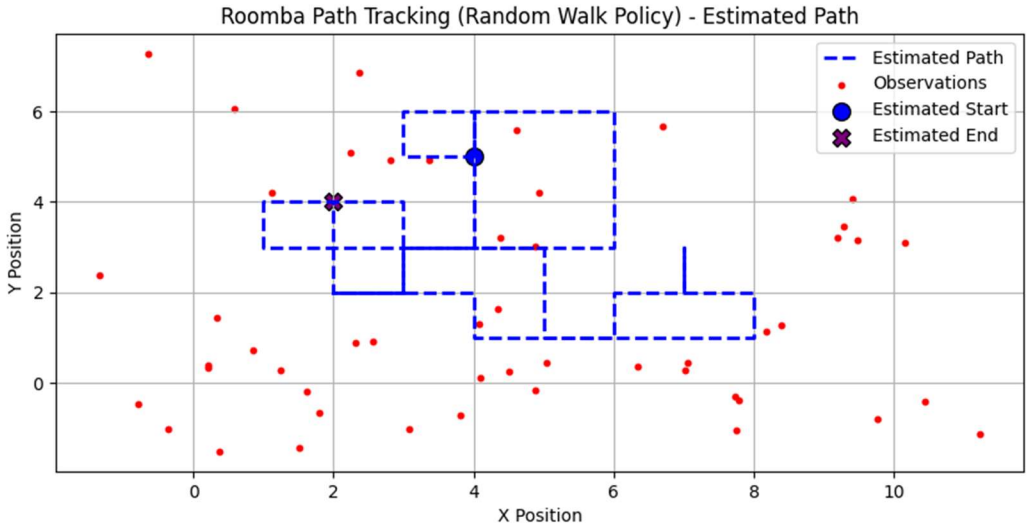
Processing policy: random_walk

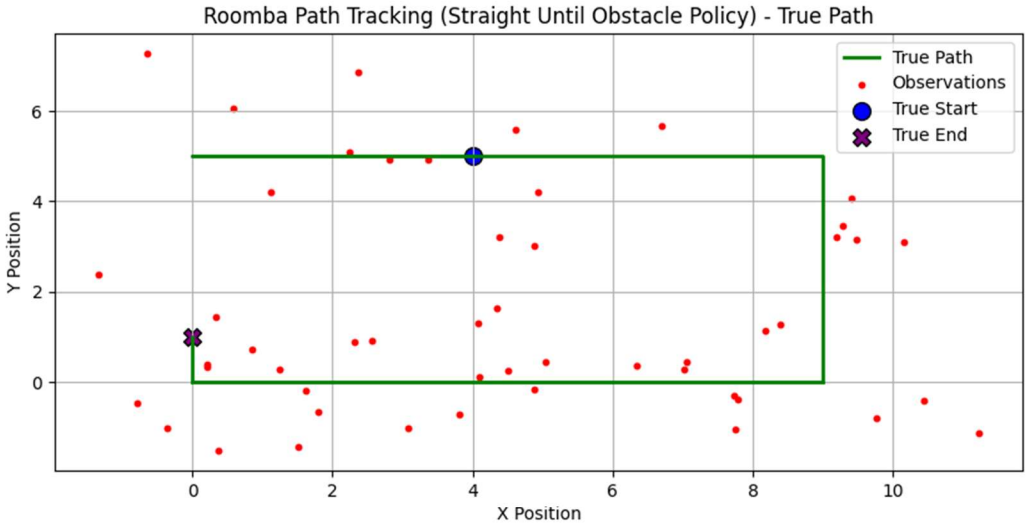
Tracking accuracy for random walk policy: 60.00%

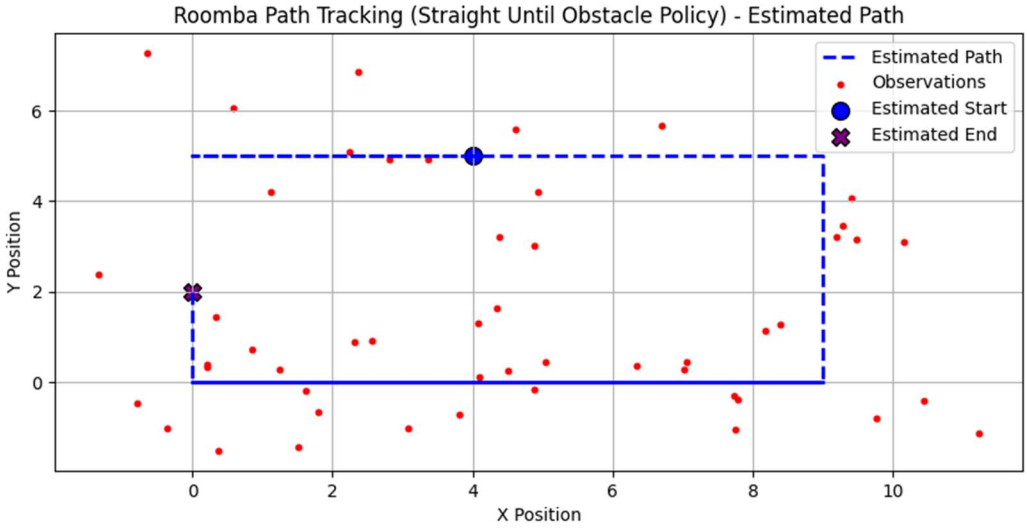
Processing policy: straight_until_obstacle

Tracking accuracy for straight until obstacle policy: 96.00%






Roomba Path Tracking (Straight Until Obstacle Policy) - True Path
True Path
Observations
True Start
True End
Y Position
X Position
0
2
4
6
8
10


Roomba Path Tracking (Straight Until Obstacle Policy) - Estimated Path
Estimated Path
Observations
Estimated Start
Estimated End
Y Position
X Position
0
2
4
6
8
10

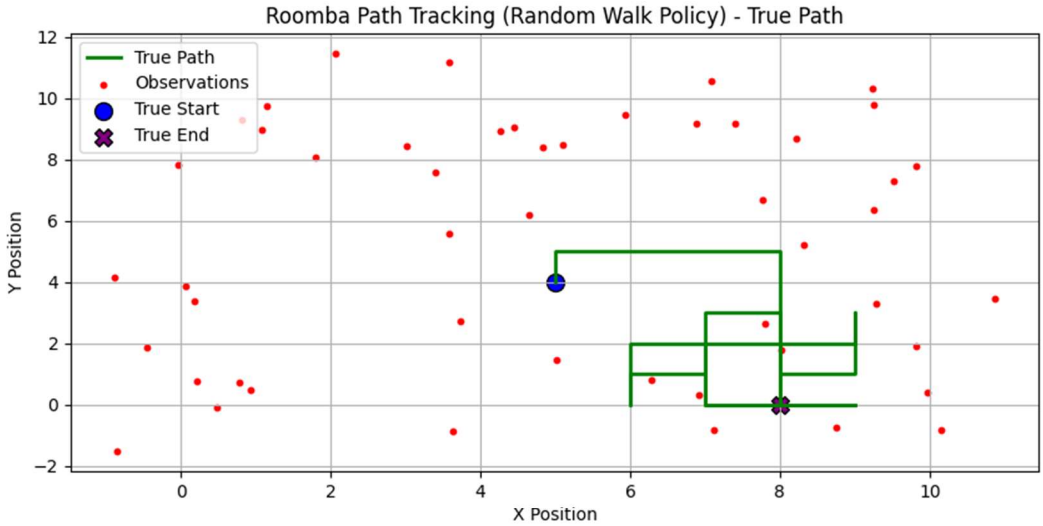
Seed Value: 42

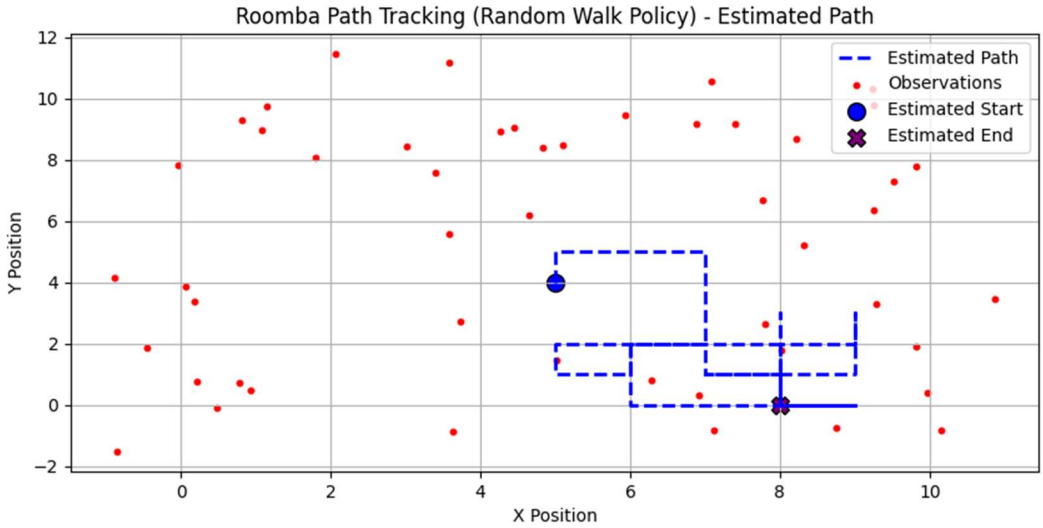
Processing policy: random_walk

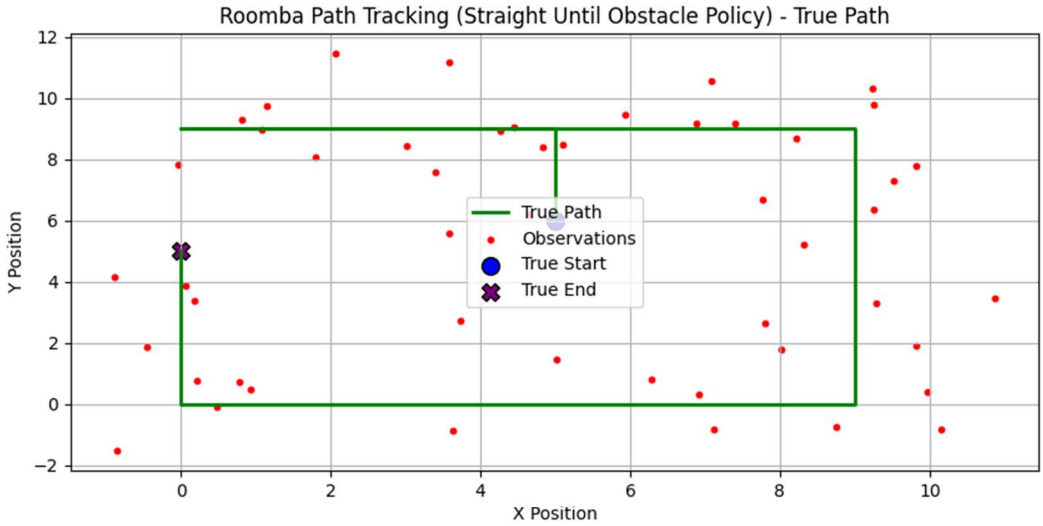
Tracking accuracy for random walk policy: 64.00%

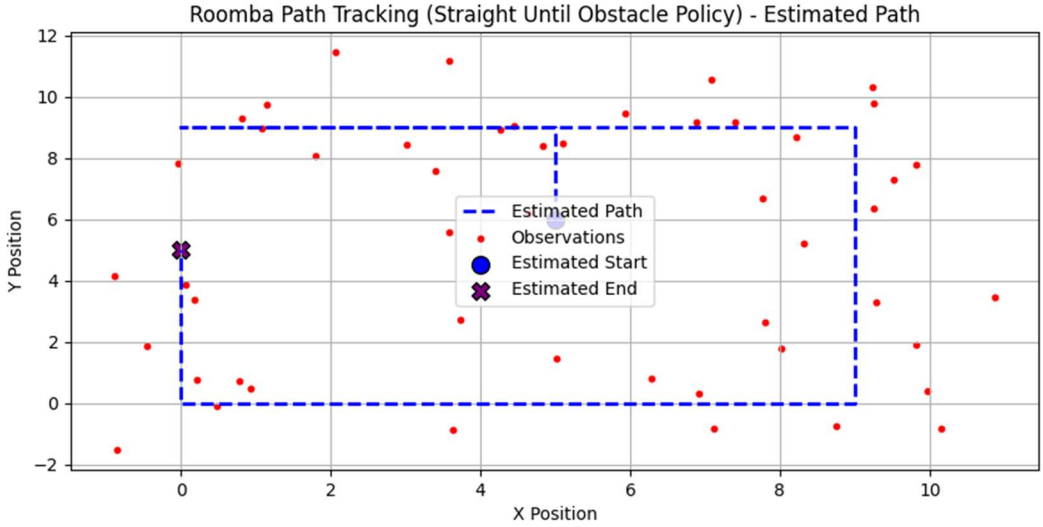
Processing policy: straight_until_obstacle

Tracking accuracy for straight until obstacle policy: 100.00%






Roomba Path Tracking (Straight Until Obstacle Policy) - True Path
True Path
Observations
True Start
True End
X Position
Y Position


Roomba Path Tracking (Straight Until Obstacle Policy) - Estimated Path
Estimated Path
Observations
Estimated Start
Estimated End
X Position
Y Position

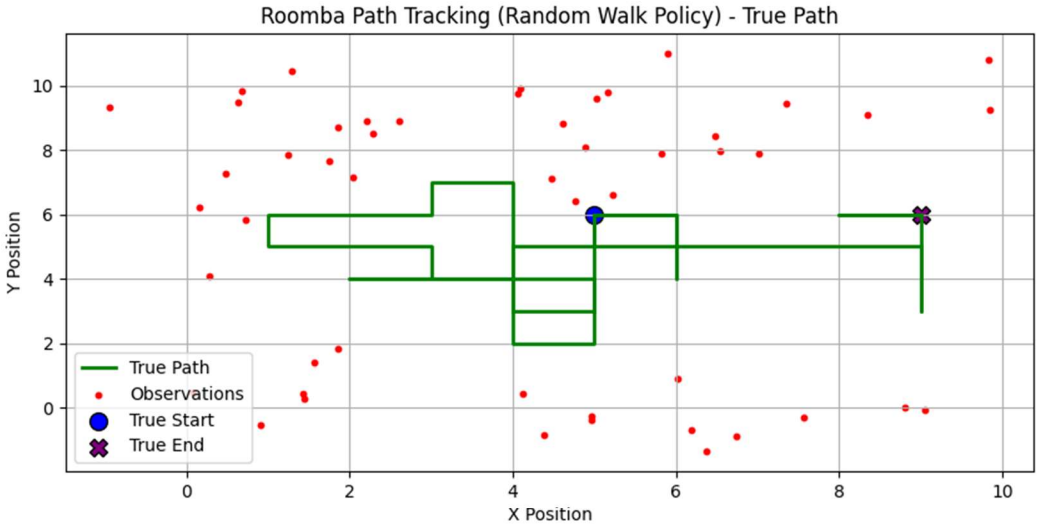
Seed Value: 20

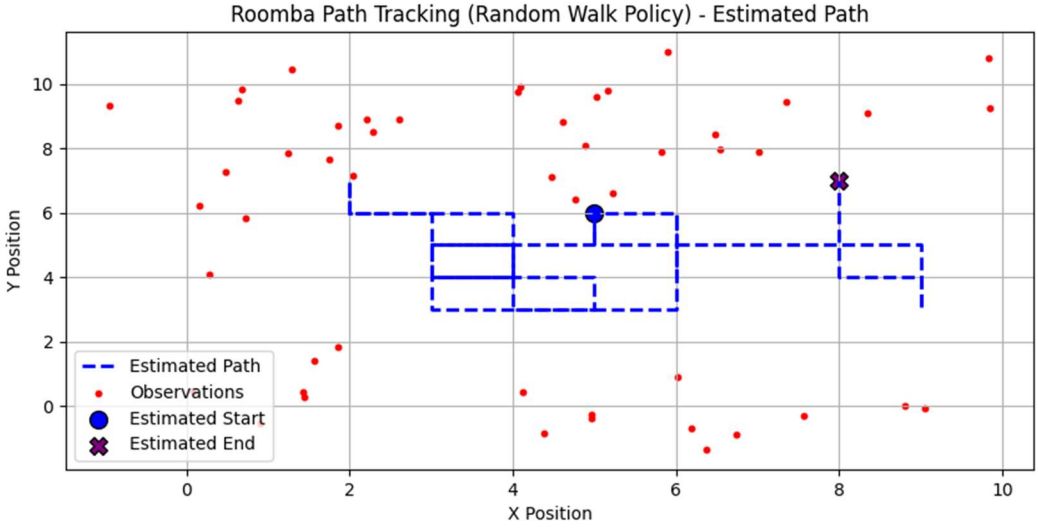
Processing policy: random_walk

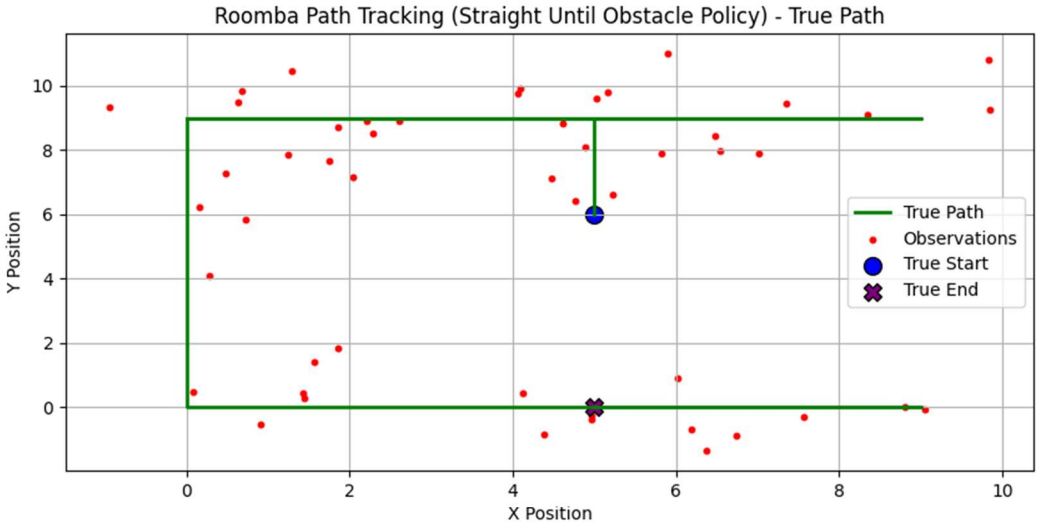
Tracking accuracy for random walk policy: 56.00%

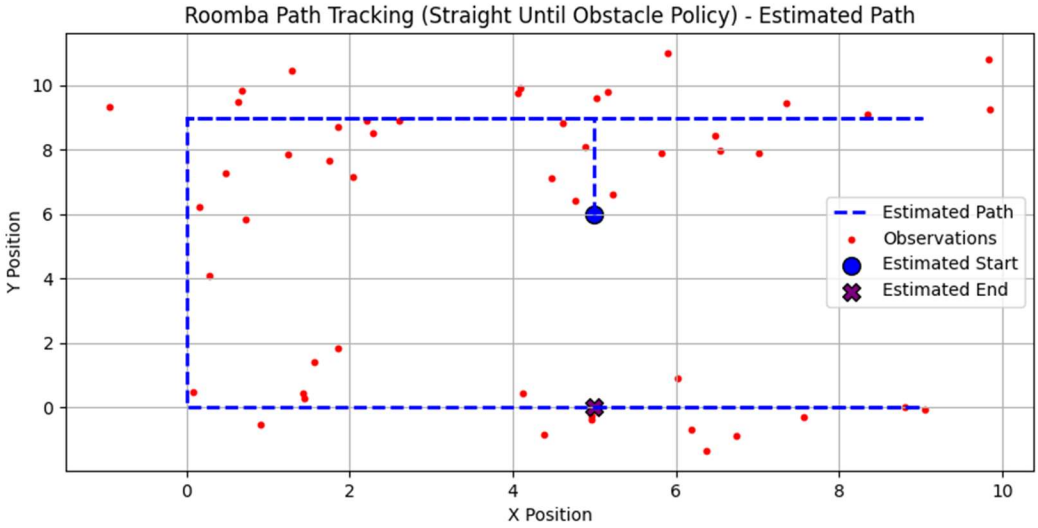
Processing policy: straight_until_obstacle

Tracking accuracy for straight until obstacle policy: 100.00%






Roomba Path Tracking (Straight Until Obstacle Policy) - True Path
Y Position
X Position
True Path
Observations
True Start
True End


Roomba Path Tracking (Straight Until Obstacle Policy) - Estimated Path
Y Position
X Position
Estimated Path
Observations
Estimated Start
Estimated End

(d) Key Observations

1. **Policy Analysis**:
    - The **straight until obstacle** policy consistently achieved higher accuracy due to its predictable transitions, which align closely with the observed data.
    - The **random walk** policy showed moderate accuracy because of the unpredictability of transitions, leading to less alignment with observations.
2. **Seed Variability**: Randomness in the observations and transitions influenced the performance, especially for the random walk policy, which showed a wider range of accuracies (42% to 64%).
3. **Plot Analysis**:
    - Plots illustrate the true path, observed noisy positions, and the Viterbi-estimated path.
    - For the straight policy, the estimated path closely aligns with the true path, even at higher noise levels.
    - The random walk policy struggles to consistently align the estimated path with the true positions.

(e) The **estimated_paths.csv** file contains the following data for each seed value and policy