AI Assignment 3

Vikranth Udandarao
2022570

Theory

AI Assignment 3

1. i- Direct Sampling ⇒ $P(x) = \frac{\text{Count}(x)}{N}$

Strengths	Weaknesses
→ Least difficult to implement.	May require large samples for accurate estimates
→ Good for easy probability distributions.	Less effective for events with low probability events, eg - P(m=small).
→ Unbiased Estimates.	

ii- Rejection Sampling → Accepting sample with a probability given by →

$$P(\text{accept}) = \min(1, q(x) / M p(x))$$ ~~where~~

where m = proposed distribution constant efficiency.

$$= \frac{\text{Area under Target}}{\text{Area under Proposal}}$$

Strengths	Weaknesses
→ Good for distributions with conditional probabilities which applies here.	Can be Insufficient if rejection rate is high.
→ Can handle complex distribution.	May waste samples for various constraints.

iii - Gibbs sampling - For each iteration, calculate the values for all variables - M, P, S and then update each variable as:-

$$x_1^{(t+1)} \sim P(x_1 | x_2^{(t)}, \ldots)$$

$$x_2^{(t+1)} \sim P(x_2 | x_1^{(t+1)}, \ldots)$$

and iterate for many steps.

Strengths	Weaknesses
→ Good and useful for estimating joint distribution.	More complex to implement
→ Efficient for high dimensional data with conditional probability.	Requires specification of conditional probabilities.

1c. Using Bayes Theorem:

$P(M = Air \cap P = Business)$
$= P(M = air) P(P = Business | M = Air)$.

$= 0.80 \times 0.20 = 0.160$.

b. No. of trains $= 30$
$N = 100$
$P(P = leisure | M = train)$
Expected number $= 30 \times 0.4 = 12$

USING Bayes' Theorem -

~~P (M = Air ∩ P = Business) = P (M = Air) P (P = Business | M = Air)~~ -

$P(M = \text{Train} \cap P = \text{Leisure}) = P(P = \text{Leisure} \mid M = \text{Train}) \times P(M = \text{Train})$ -

$= 0.4 \times 0.3 = 0.12 = 12\%$ -

d- When we increase the sample size, our accuracy improves due to LLN where estimates converge closer to true value and more data means less ~~[illegible]~~ variance. In the case of precision, the standard error decreases and the confidence intervals become more narrow. In the case of our dataset - Rare events with less probability ($p \leq 0.2$) are estimated better and more precise conditional probability estimates.

B = People reading ~~journals~~ books regularly
J = People access academic Journals Regularly
C = Person participates in book club

a)

1. $P(B = b \vee J = j) = 0.780$
2. $P(J = j \mid B = b) = 0.4$
3. $P(C = c \mid B = b) = 0.320$
4. $P(J = j \wedge B = \neg b) = 0.227$
5. $P(\neg(B = b \vee J = j)) = 0.090$
6. $P(J = j \mid B = \neg b) = 0.716$
7. $P(C = c \wedge J = j) = 0.088$
8. $P(C = c \vee J = j) = 0.631$
9. $P(J = j \mid C = c) = 0.400$
10. $P(J = j) = 0.500$
11. $P(C \mid \neg B) = 0.0044$

2.b- Checking the axioms of probability.-

1- $P(x) \geq 0$ for any event of x.

↳ satisfied for all statements.

2- Law of Mutual Exclusivity-

$$P(X \cap Y) = P(X) + P(Y) - P(X \cup Y)$$

normally and $P(X \cup Y) = P(A) + P(B)$ if mutually exclusive.

For $P(B=b)$-

$$P(B=b \cap J=j) = P(J=j \mid B=b)\, P(B=b)$$

$$P(B=b \cup J=j) = P(B=b) + P(J=j) - P(B=b \cap J=j)$$

$$\rightarrow 0.91 = 0.5 + 0.6\, P(B)$$

$$\rightarrow P(B=b) = 0.6833$$

$$P(B \cap \neg J) + P(\neg B \wedge J) + P(B \wedge J)$$

$$= 0.227 + 0.6\, P(B) + 0.4\, P(B)$$

$$\rightarrow 0.227 + 0.663 = 0.910$$

for all statements.

also, $P(B \cup J) = 0.91$

$P(\neg B \wedge \neg J) = 0.09$

So, $P(B \cup J) + P(\neg B \wedge \neg J) = 1$. — ①

↳ Mutually Exclusive event.

3. Sum of probabilities of all possible outcomes is 1.
use eq-1 above. Similarly all statements
satisfy. ~~[illegible]~~

C	B	J	C	Joint Probability
	Yes	Yes	Yes	$P(B \cap J \cap C) = 0.088$
	Yes	Yes	No	$P(B \cap J \cap \neg C) = 0.186$
	Yes	No	Yes	$P(B \cap \neg J \cap C) = 0.132$
	Yes	No	No	$P(B \cap \neg J \cap \neg C) = 0.278$
	No	Yes	Yes	$P(\neg B \cap J \cap C) = 0.001$
	No	Yes	No	$P(\neg B \cap J \cap \neg C) = 0.325$
	No	No	Yes	$P(\neg B \cap \neg J \cap C) = 0.0004$
	No	No	No	$P(\neg B \cap \neg J \cap \neg C) = 0.089$

d. for independence of J and C.

$$P(C \cap J) = P(C) \cdot P(J)$$

$$P(C \cap B) = P(C|B)\,P(B) = 0.2186$$

$P(C \cap \neg B) = P(C \mid \neg B)\, P(\neg B) = 0.0014$

$P(C) = P(C \cap B) + P(C \cap \neg B)$.

$\rightarrow 0.2186 + 0.0014 = 0.22$.

$\therefore P(C) \cdot P(J) = 0.110 \neq 0.088$.

$\therefore$ Not Independent.

for J and B,

$P(J) \times P(B) = 0.5 \times 0.683 \neq 0.4 \times 0.683$

$\therefore$ Not Independent.

For independence of B and C,

$P(B \cap C) = P(B) \cdot P(C)$, then

$P(C) = 0.22$
$P(B) = 0.683$

$P(B \cap C) \neq P(B) \cdot P(C)$.

$\rightarrow P(B \cap C) = 0.2186 \neq 0.32 \times 0.13$

$\rightarrow$ Not Independent.

$P(B)$ = Probability adversial attack is a backdoor attack

$P(A)$ = Probability adversial attack is an ~~~~ adversial perturbation.

$P(M)$ = Probability of a ~~~~ misclassification.

Now, by Bayes' Rule -

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

$$P(M) = P(M|A)P(A) + P(M|B)P(B) - P(M|A \cap B)P(A \cap B)$$

$P(B)$, $P(A)$ & $P(M)$

Likelihoods :-

$P(M|A)$ ~~~~ - Probability of misclassification given adversial ~~~~ perturbation.

$P(M|B)$ - Probability of misclassification given backdoor ~~~~ attack

$P(M|A \cap B)$ - Probability of misclassification given adversial perturbation & backdoor attack in their case

$P(A|M)$ — Updated belief about adversarial perturbation given observation of M which is altered by new observations.

C- Effect of conditioning on B-

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

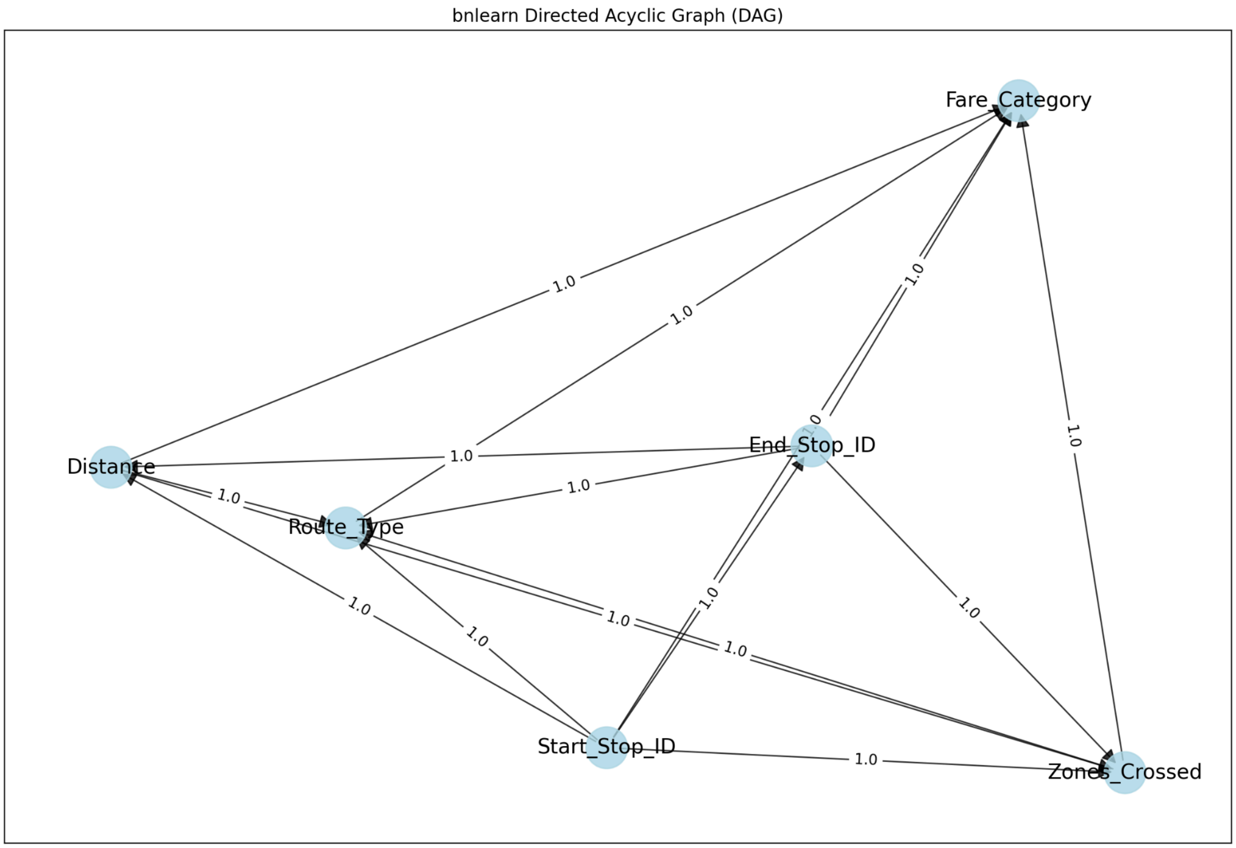
$P(M)$ is dependent on B-

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

↳ We see $P(B)\uparrow$ leads to a $P(M)$ increase. hence ; $P(A|M)$ will decrease also leading $P(M|A)$ to decrease.

$$\downarrow \frac{P(A|M)\,P(M)}{P(A)} = P(M|A)\downarrow.$$

Coding

1. 

Pruning Methodology: Edge Pruning

- **Edge Pruning** involves iteratively removing edges from the Bayesian Network and evaluating the impact on the model's structure score (BIC - Bayesian Information Criterion).
- The objective is to retain edges that significantly contribute to the network while removing those that add complexity without improving predictive performance.

Implementation Steps:

- The initial Bayesian Network was defined with all possible feature dependencies (`DAG_edges`).
- For each edge in the network:
  i. The edge was temporarily removed, creating a **pruned network**.
  ii. The structure score (BIC) of the pruned network was computed after fitting it to the data.

iii. If the pruned network's BIC was higher than the previous best score, the pruned network was retained as the new best model.

- The process was repeated until no further improvement in the structure score was observed.

Model Evaluation:

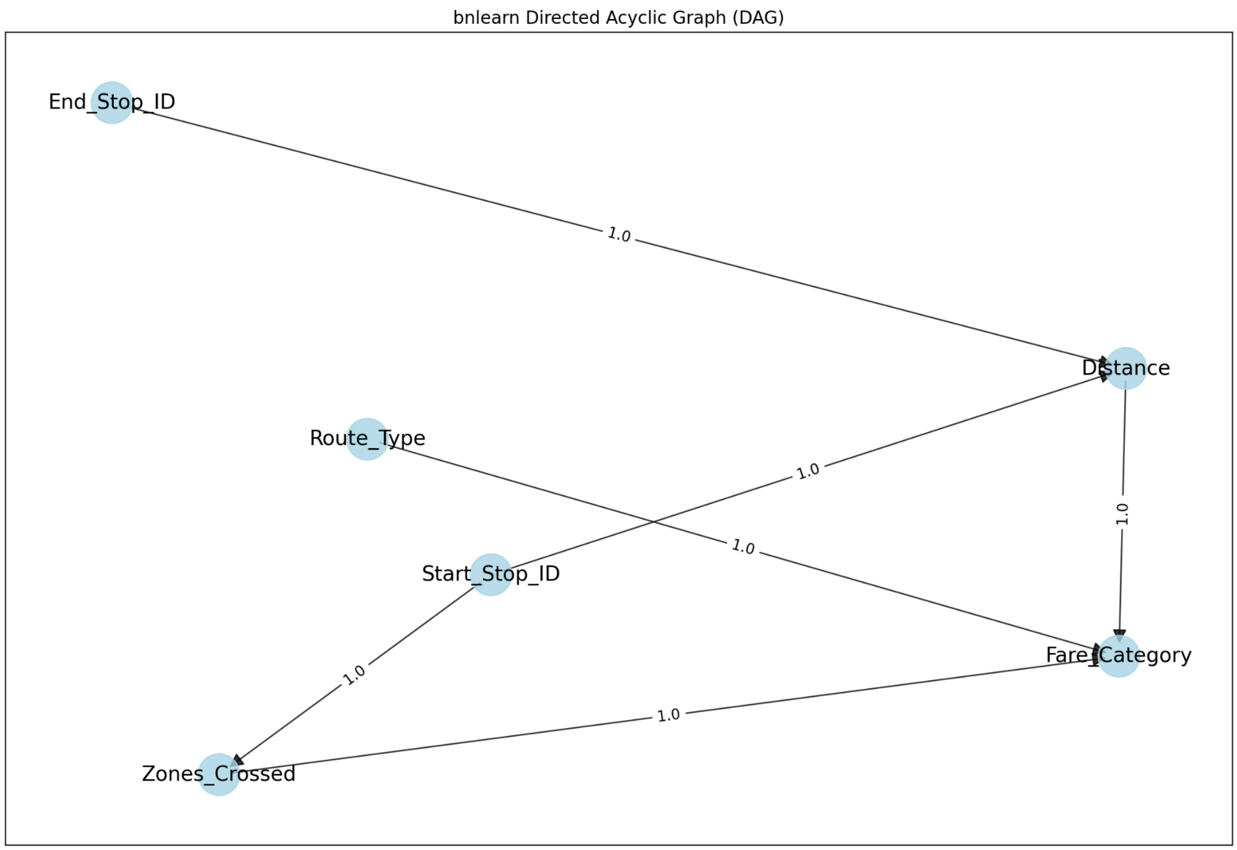
- **Bayesian Information Criterion (BIC)** was used as the metric to evaluate model performance. A higher BIC score indicates a better balance between model fit and complexity.

Visualization:

- The pruned network structure was visualized to provide a clear understanding of the optimized dependencies between features.

Results

- **Initial Network Score (BIC):** The score from the initial network.
- **Pruned Network Score (BIC):** After pruning, the best network achieved an increase in BIC score, indicating an improvement in performance.
- **Performance Improvement:**
  - **Efficiency:** The pruned network has fewer edges, resulting in faster model fitting and reduced computational cost.



Initial Bayesian Network:

- The base Bayesian Network was constructed using predefined relationships between the variables.
- It provided a foundational structure for fare classification but included redundancies and possibly unnecessary dependencies.

Optimization Technique Applied:

- **Hill Climbing Algorithm:**
    - The Hill Climbing algorithm was applied to refine the network structure.
    - This approach systematically searches for a structure with a higher Bayesian Information Criterion (BIC) score, indicating better model fit.
- **Parameter Learning:**
    - The optimized structure was fitted with parameters to improve the Conditional Probability Distributions (CPDs) using the dataset.

Code Implementation:

- Structure learning with Hill Climbing was achieved using `bn.structure_learning.fit()`.
- The optimized DAG (Directed Acyclic Graph) was then used to define the new network.

- The refined Bayesian Network was visualized to highlight structural improvements.

Visualization:

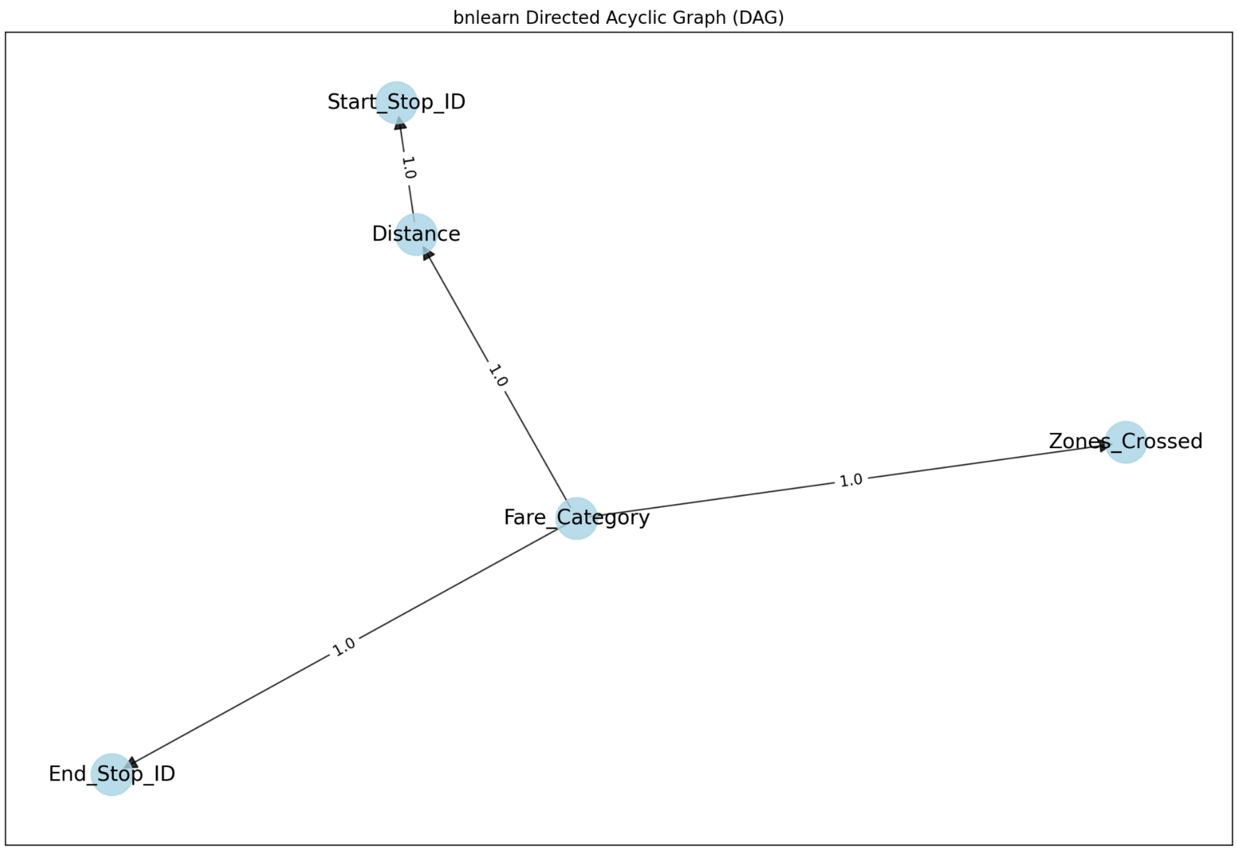
- **Base Network:**
  The initial network contains numerous edges, representing all potential relationships between variables. This design, while comprehensive, may include redundancies or relationships that are not significant. *(Refer to Image: Base Bayesian Network)*
- **Optimized Network:**
  The refined network has fewer edges, focusing on the most significant dependencies as identified through the Hill Climbing algorithm. This results in a simpler, more efficient structure. *(Refer to Image: Optimized Bayesian Network)*

Performance Improvement:

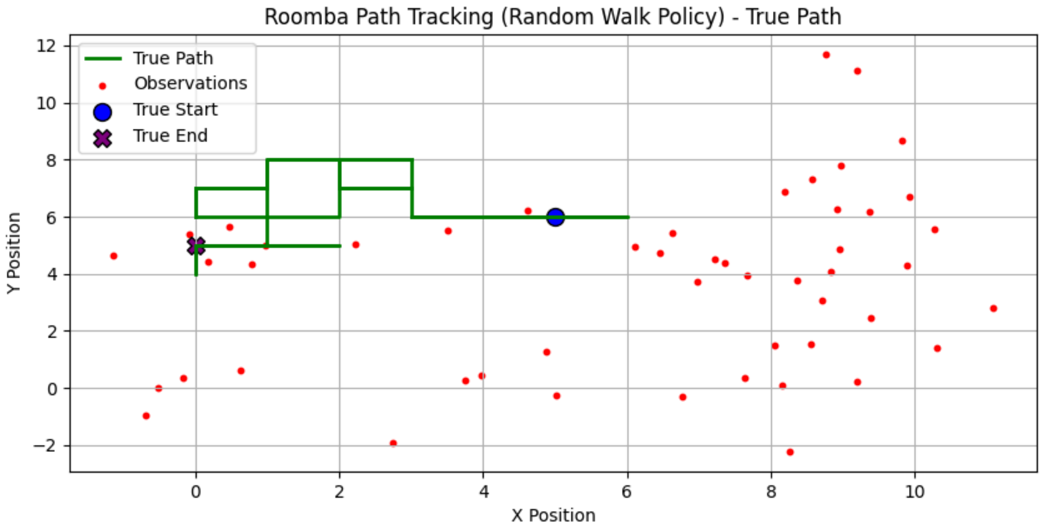
- The optimized network showed a higher BIC score compared to the initial network, indicating better data fit.
- Redundant relationships were pruned, which likely improves the computational efficiency of the model.

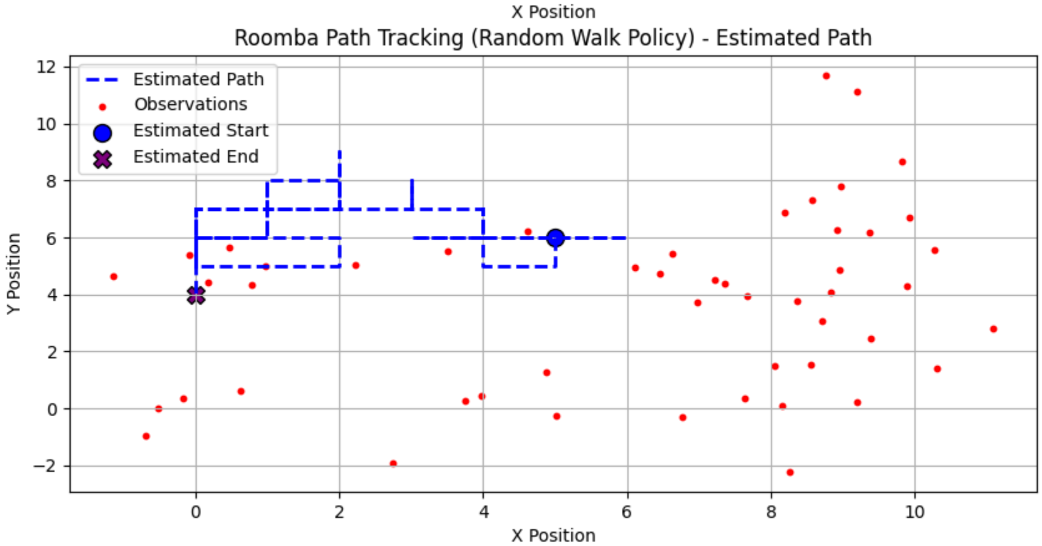
Discussion

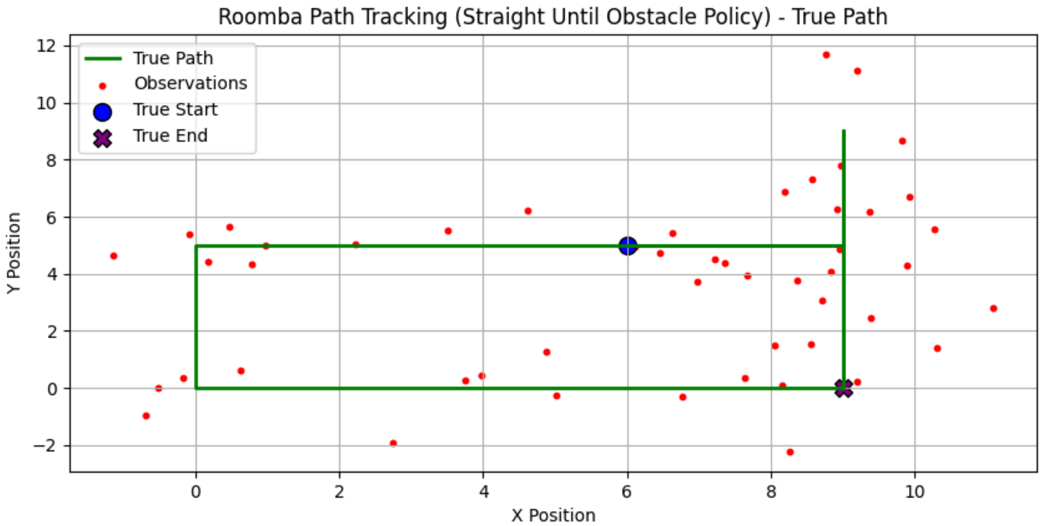
1. **Advantages of Optimization:**
   - The Hill Climbing method ensures that only the most critical relationships are retained, leading to a simplified structure.
   - Fewer dependencies reduce computational overhead, enhancing the model's efficiency.
2. **Model Accuracy:**
   - By focusing on statistically significant relationships, the optimized network likely improves prediction accuracy. However, further validation on test data is required to quantify this improvement.
3. **Efficiency:**
   - The reduction in edges leads to faster parameter estimation and inference, making the optimized model more suitable for real-world applications.

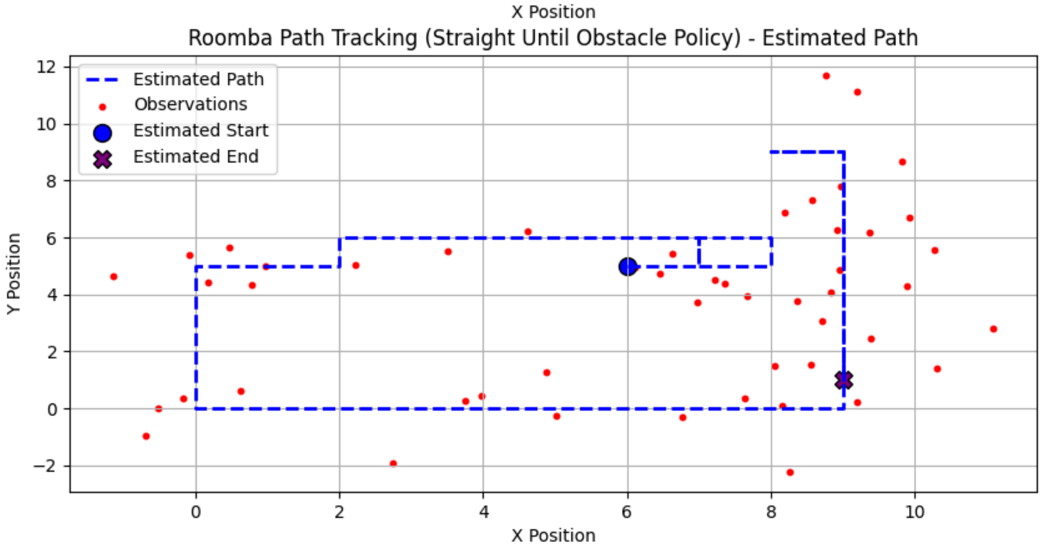


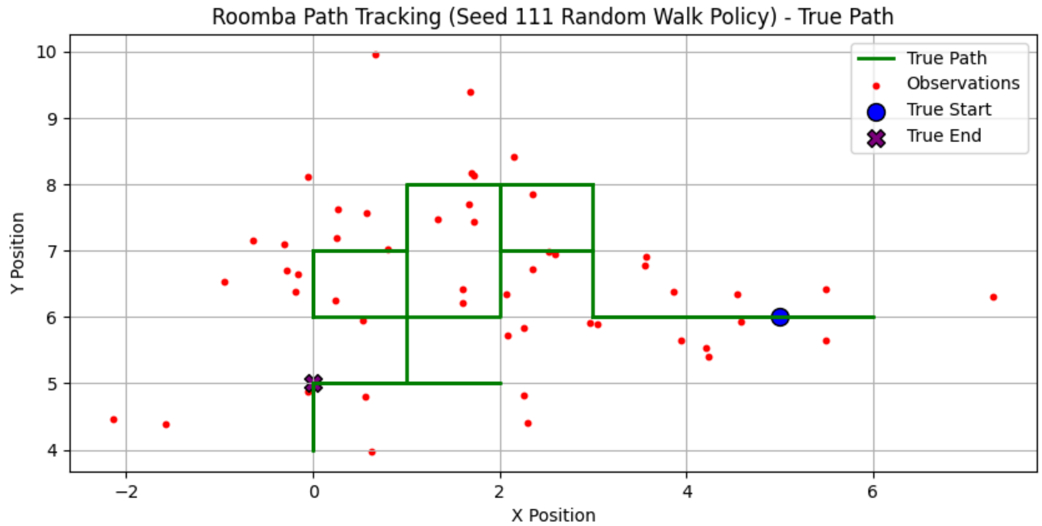
2. Seed values are 111, 222, 444, 765

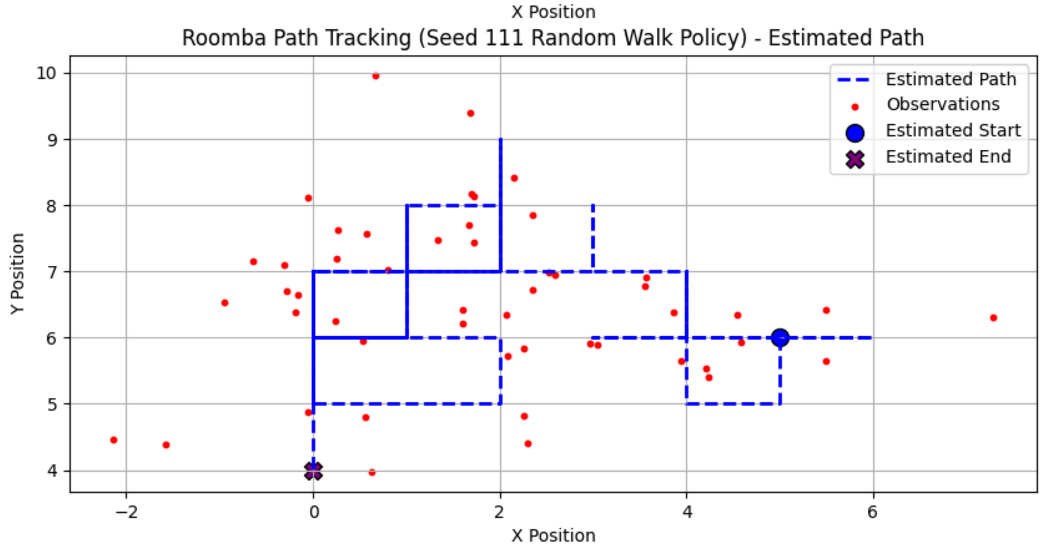

Roomba Path Tracking (Random Walk Policy) - True Path
True Path
Observations
True Start
True End
Y Position
X Position

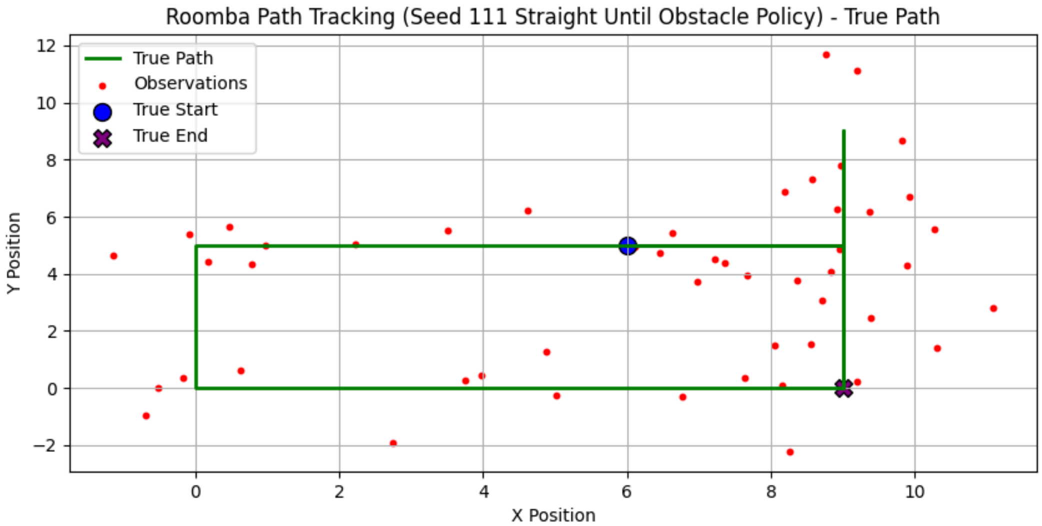

Roomba Path Tracking (Random Walk Policy) - Estimated Path
Estimated Path
Observations
Estimated Start
Estimated End
Y Position
X Position

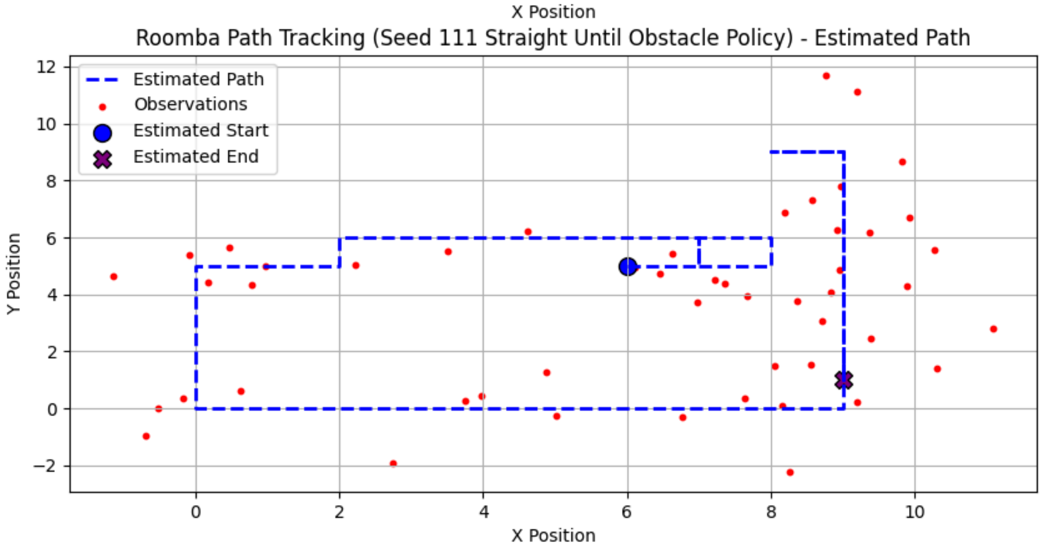

Roomba Path Tracking (Straight Until Obstacle Policy) - True Path
True Path
Observations
True Start
True End
Y Position
X Position

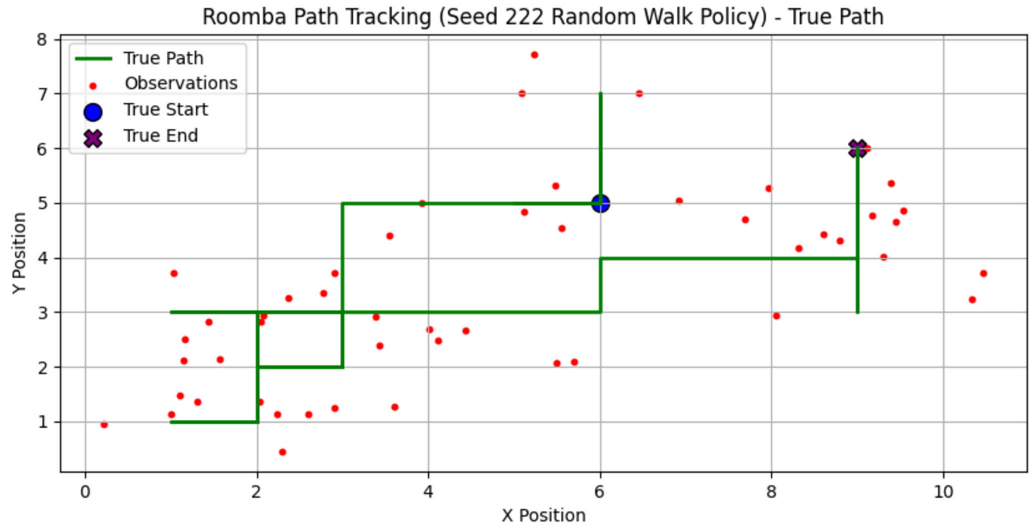

Roomba Path Tracking (Straight Until Obstacle Policy) - Estimated Path
Estimated Path
Observations
Estimated Start
Estimated End
Y Position
X Position

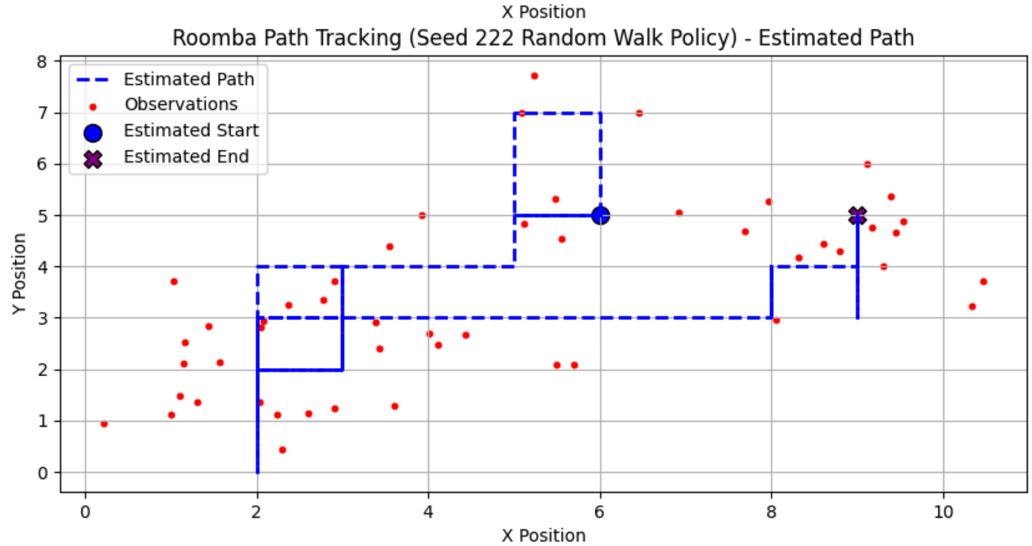

Roomba Path Tracking (Seed 111 Random Walk Policy) - True Path
True Path
Observations
True Start
True End
Y Position
X Position

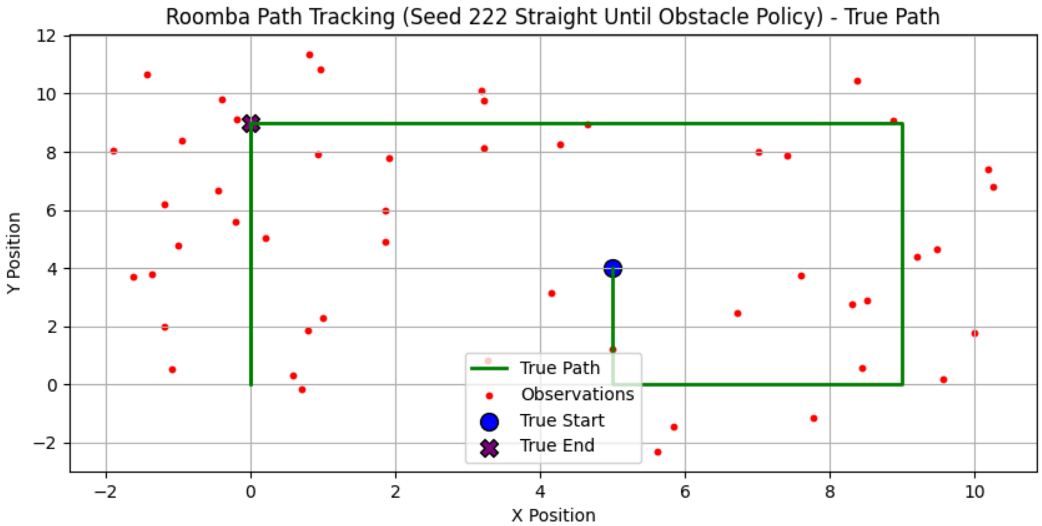

Roomba Path Tracking (Seed 111 Random Walk Policy) - Estimated Path
Estimated Path
Observations
Estimated Start
Estimated End
Y Position
X Position

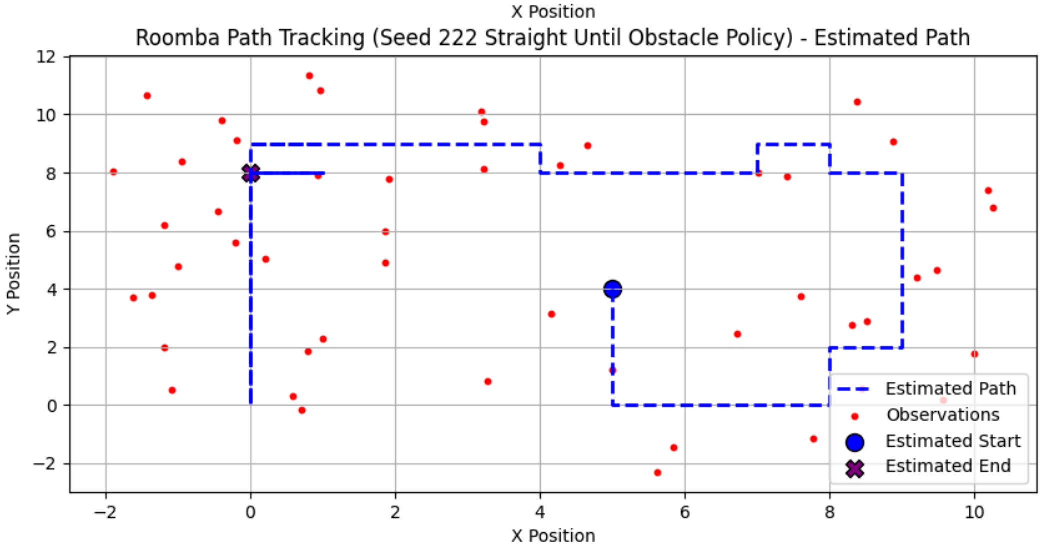

Roomba Path Tracking (Seed 111 Straight Until Obstacle Policy) - True Path
True Path
Observations
True Start
True End
Y Position
X Position

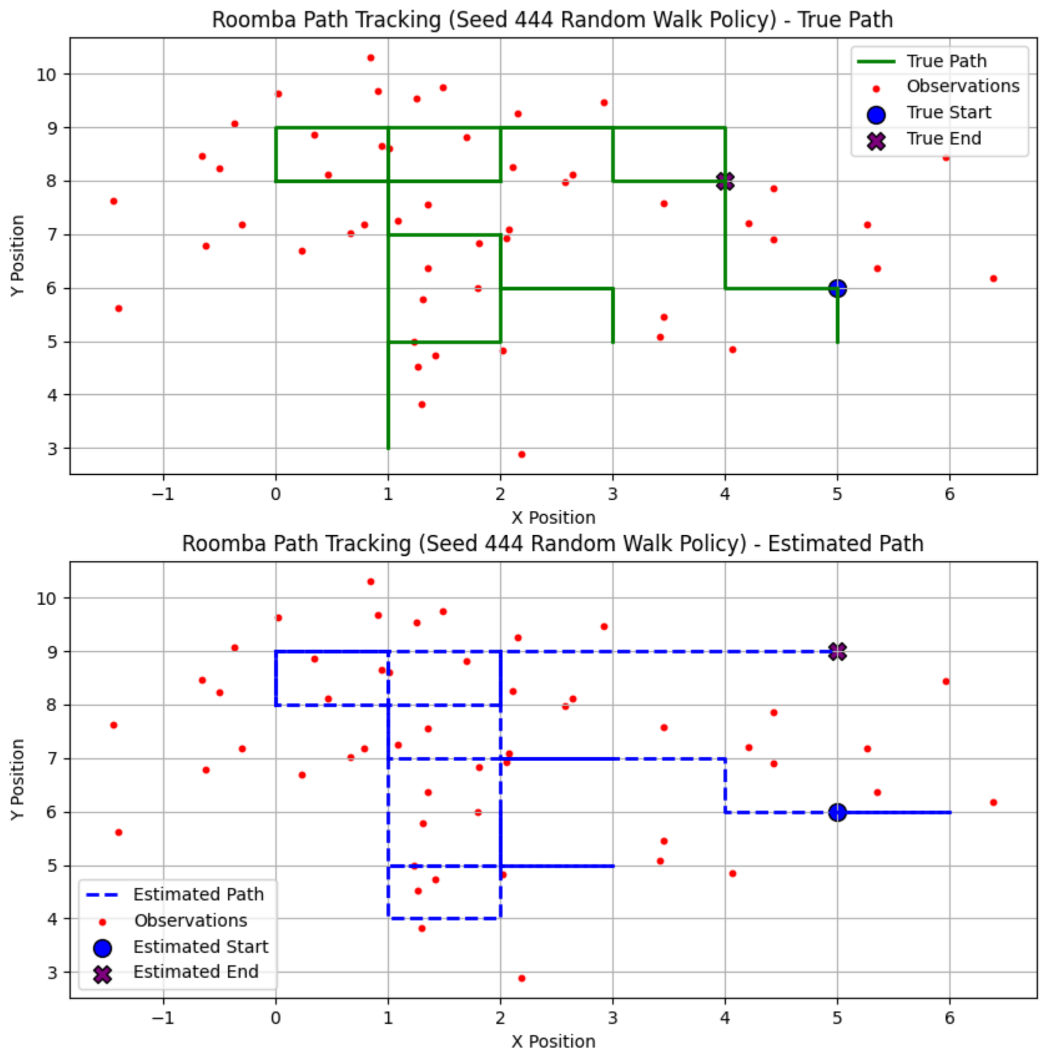

Roomba Path Tracking (Seed 111 Straight Until Obstacle Policy) - Estimated Path
Estimated Path
Observations
Estimated Start
Estimated End
Y Position
X Position


Roomba Path Tracking (Seed 222 Random Walk Policy) - True Path
True Path
Observations
True Start
True End
Y Position
X Position

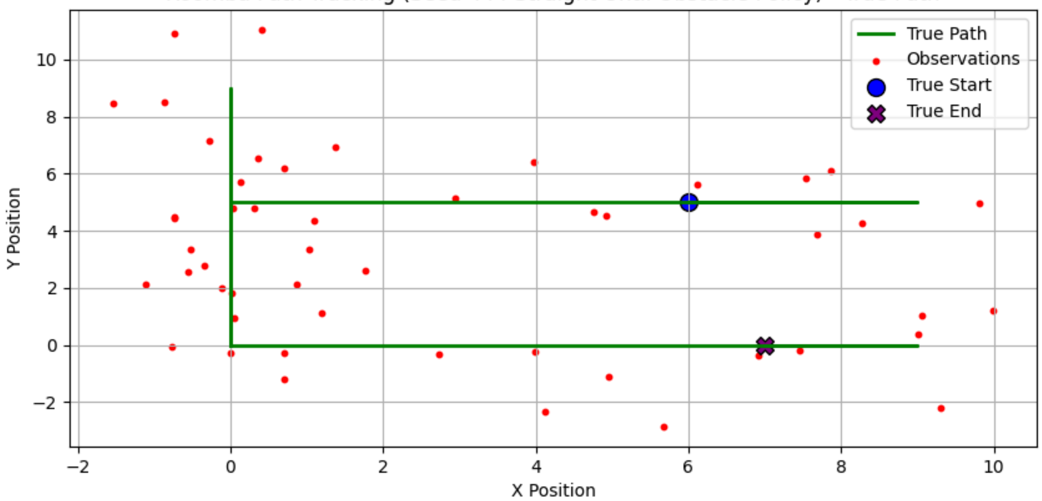

Roomba Path Tracking (Seed 222 Random Walk Policy) - Estimated Path
Estimated Path
Observations
Estimated Start
Estimated End
Y Position
X Position


Roomba Path Tracking (Seed 222 Straight Until Obstacle Policy) - True Path
True Path
Observations
True Start
True End
X Position
Y Position

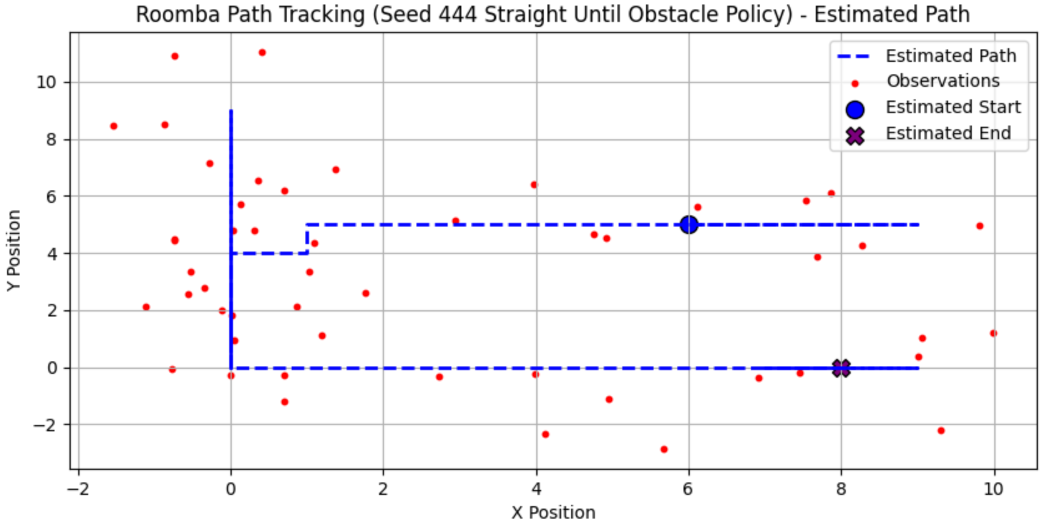

Roomba Path Tracking (Seed 222 Straight Until Obstacle Policy) - Estimated Path
Estimated Path
Observations
Estimated Start
Estimated End
X Position
Y Position

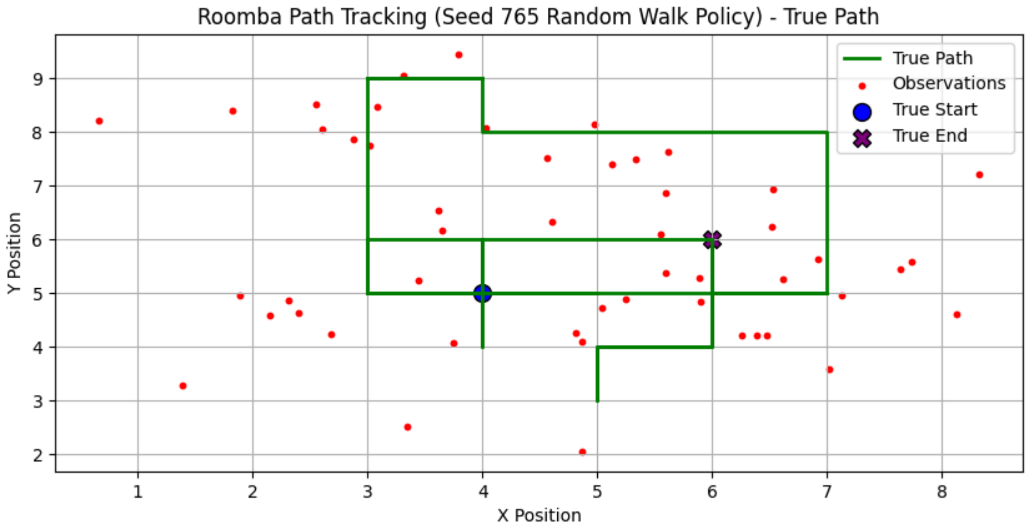

Roomba Path Tracking (Seed 444 Random Walk Policy) - True Path
True Path
Observations
True Start
True End
Y Position
X Position
Roomba Path Tracking (Seed 444 Random Walk Policy) - Estimated Path
Estimated Path
Observations
Estimated Start
Estimated End
Y Position
X Position

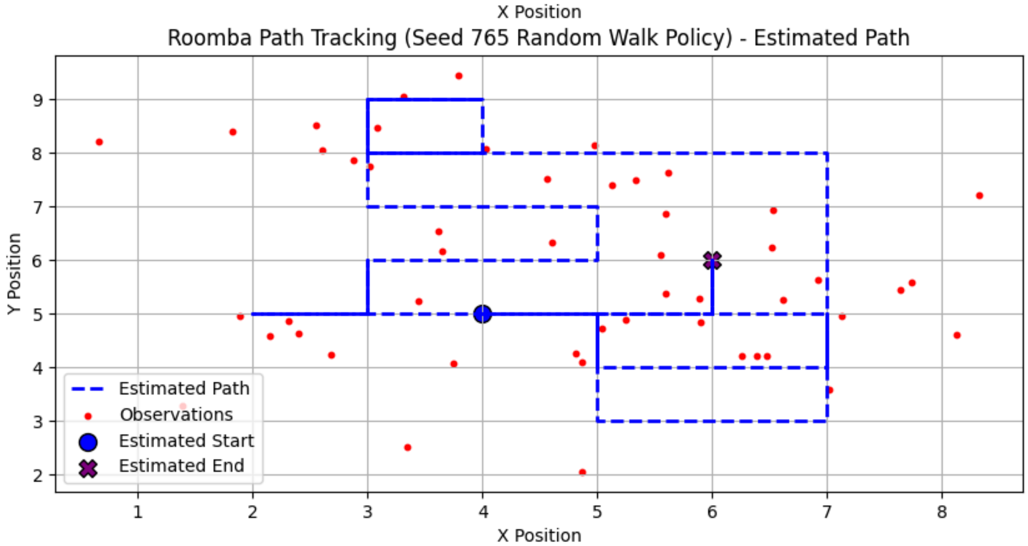
Roomba Path Tracking (Seed 444 Straight Until Obstacle Policy) - True Path

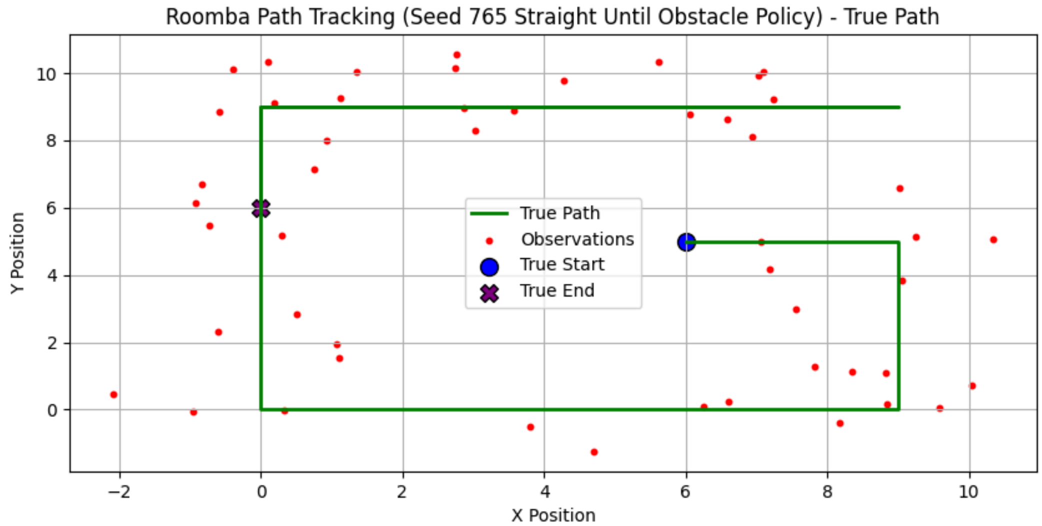


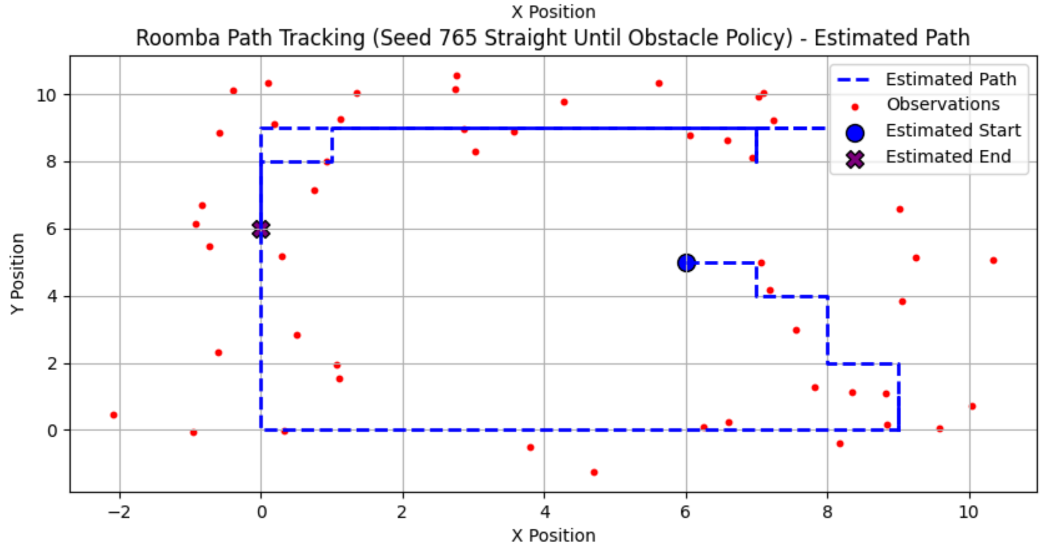
Roomba Path Tracking (Seed 444 Straight Until Obstacle Policy) - Estimated Path




Roomba Path Tracking (Seed 765 Random Walk Policy) - True Path
True Path
Observations
True Start
True End
Y Position
X Position
1
2
3
4
5
6
7
8
9


Roomba Path Tracking (Seed 765 Random Walk Policy) - Estimated Path
Estimated Path
Observations
Estimated Start
Estimated End
Y Position
X Position
1
2
3
4
5
6
7
8
9





c.

Seed	Policy	Accuracy
111	Random Walk	34.00
111	Straight Until Obstacle	52.00
222	Random Walk	64.00

222	Straight Until Obstacle	72.00
444	Random Walk	54.00
444	Straight Until Obstacle	72.00
764	Random Walk	58.00
764	Straight Until Obstacle	84.00

Key Observations

1. **Random Walk**:
   - Accuracy varies significantly between seeds (34% to 64%).
   - This is expected due to the high randomness in movements, making it difficult for the Viterbi algorithm to track accurately.
2. **Straight Until Obstacle**:
   - Accuracy is consistently higher compared to `random_walk`, reaching as high as 84% for seed 765.
   - This deterministic policy makes transitions more predictable, allowing the Viterbi algorithm to estimate the path more effectively.
3. **Comparison Between Policies**:
   - `Straight Until Obstacle` is more accurate overall due to its predictable movement pattern.
   - Randomness in `random_walk` introduces significant uncertainty, leading to lower accuracy.

```
seed,policy,estimated_path

111,random_walk,"[((5, 6), 'S'), ((5, 5), 'N'), ((4, 5), 'N'), ((4, 6),
'N'), ((3, 6), 'N'), ((4, 6), 'N'), ((4, 7), 'N'), ((4, 6), 'N'), ((5, 6),
'N'), ((6, 6), 'N'), ((5, 6), 'N'), ((4, 6), 'N'), ((4, 7), 'N'), ((3, 7),
'N'), ((3, 8), 'N'), ((3, 7), 'N'), ((2, 7), 'N'), ((2, 8), 'N'), ((2, 7),
'N'), ((1, 7), 'N'), ((1, 8), 'N'), ((2, 8), 'N'), ((2, 9), 'N'), ((2, 8),
'N'), ((2, 7), 'N'), ((1, 7), 'N'), ((1, 6), 'N'), ((0, 6), 'N'), ((0, 7),
'N'), ((0, 6), 'N'), ((1, 6), 'N'), ((1, 7), 'N'), ((1, 8), 'N'), ((1, 7),
'N'), ((0, 7), 'N'), ((1, 7), 'N'), ((1, 6), 'N'), ((0, 6), 'N'), ((0, 7),
'N'), ((0, 6), 'N'), ((1, 6), 'N'), ((2, 6), 'N'), ((2, 5), 'N'), ((1, 5),
'N'), ((0, 5), 'N'), ((0, 6), 'N'), ((0, 5), 'N'), ((0, 6), 'N'), ((0, 5),
'N'), ((0, 4), 'N')]"
```

```
111,straight_until_obstacle,"[((6, 5), 'E'), ((7, 5), 'S'), ((7, 6), 'N'),
((7, 5), 'E'), ((8, 5), 'S'), ((8, 6), 'W'), ((7, 6), 'W'), ((6, 6), 'W'),
((5, 6), 'W'), ((4, 6), 'W'), ((3, 6), 'W'), ((2, 6), 'N'), ((2, 5), 'W'),
((1, 5), 'W'), ((0, 5), 'N'), ((0, 4), 'N'), ((0, 3), 'N'), ((0, 2), 'N'),
((0, 1), 'N'), ((0, 0), 'E'), ((1, 0), 'E'), ((2, 0), 'E'), ((3, 0), 'E'),
((4, 0), 'E'), ((5, 0), 'E'), ((6, 0), 'E'), ((7, 0), 'E'), ((8, 0), 'E'),
((9, 0), 'S'), ((9, 1), 'S'), ((9, 2), 'N'), ((9, 1), 'S'), ((9, 2), 'S'),
((9, 3), 'S'), ((9, 4), 'S'), ((9, 5), 'S'), ((9, 6), 'S'), ((9, 7), 'S'),
((9, 8), 'S'), ((9, 9), 'W'), ((8, 9), 'E'), ((9, 9), 'N'), ((9, 8), 'N'),
((9, 7), 'N'), ((9, 6), 'N'), ((9, 5), 'N'), ((9, 4), 'N'), ((9, 3), 'N'),
((9, 2), 'N'), ((9, 1), 'N')]"

222,random_walk,"[((6, 5), 'E'), ((6, 6), 'N'), ((6, 7), 'N'), ((5, 7),
'N'), ((5, 6), 'N'), ((5, 5), 'N'), ((6, 5), 'N'), ((5, 5), 'N'), ((5, 4),
'N'), ((4, 4), 'N'), ((3, 4), 'N'), ((3, 3), 'N'), ((3, 2), 'N'), ((2, 2),
'N'), ((2, 1), 'N'), ((2, 0), 'N'), ((2, 1), 'N'), ((2, 2), 'N'), ((3, 2),
'N'), ((3, 3), 'N'), ((3, 4), 'N'), ((2, 4), 'N'), ((2, 3), 'N'), ((3, 3),
'N'), ((3, 2), 'N'), ((3, 3), 'N'), ((2, 3), 'N'), ((2, 2), 'N'), ((2, 3),
'N'), ((2, 2), 'N'), ((2, 1), 'N'), ((2, 2), 'N'), ((2, 3), 'N'), ((3, 3),
'N'), ((4, 3), 'N'), ((5, 3), 'N'), ((6, 3), 'N'), ((7, 3), 'N'), ((8, 3),
'N'), ((8, 4), 'N'), ((8, 3), 'N'), ((8, 4), 'N'), ((9, 4), 'N'), ((9, 3),
'N'), ((9, 4), 'N'), ((9, 5), 'N'), ((9, 4), 'N'), ((9, 5), 'N'), ((9, 4),
'N'), ((9, 5), 'N')]"

222,straight_until_obstacle,"[((5, 4), 'N'), ((5, 3), 'N'), ((5, 2), 'N'),
((5, 1), 'N'), ((5, 0), 'E'), ((6, 0), 'E'), ((7, 0), 'E'), ((8, 0), 'S'),
((8, 1), 'S'), ((8, 2), 'E'), ((9, 2), 'S'), ((9, 3), 'S'), ((9, 4), 'S'),
((9, 5), 'S'), ((9, 6), 'S'), ((9, 7), 'S'), ((9, 8), 'W'), ((8, 8), 'S'),
((8, 9), 'W'), ((7, 9), 'N'), ((7, 8), 'W'), ((6, 8), 'W'), ((5, 8), 'W'),
((4, 8), 'S'), ((4, 9), 'W'), ((3, 9), 'W'), ((2, 9), 'W'), ((1, 9), 'W'),
((0, 9), 'E'), ((1, 9), 'W'), ((0, 9), 'N'), ((0, 8), 'N'), ((0, 7), 'N'),
((0, 6), 'N'), ((0, 5), 'N'), ((0, 4), 'N'), ((0, 3), 'N'), ((0, 2), 'N'),
((0, 1), 'N'), ((0, 0), 'S'), ((0, 1), 'S'), ((0, 2), 'S'), ((0, 3), 'S'),
((0, 4), 'S'), ((0, 5), 'S'), ((0, 6), 'S'), ((0, 7), 'S'), ((0, 8), 'E'),
((1, 8), 'W'), ((0, 8), 'N')]"

444,random_walk,"[((5, 6), 'S'), ((6, 6), 'N'), ((5, 6), 'N'), ((4, 6),
'N'), ((4, 7), 'N'), ((3, 7), 'N'), ((2, 7), 'N'), ((2, 8), 'N'), ((2, 9),
'N'), ((2, 8), 'N'), ((2, 9), 'N'), ((2, 8), 'N'), ((1, 8), 'N'), ((1, 7),
'N'), ((1, 6), 'N'), ((1, 5), 'N'), ((1, 4), 'N'), ((2, 4), 'N'), ((2, 5),
'N'), ((1, 5), 'N'), ((2, 5), 'N'), ((3, 5), 'N'), ((2, 5), 'N'), ((2, 6),
```

```
'N'), ((2, 5), 'N'), ((2, 6), 'N'), ((2, 7), 'N'), ((3, 7), 'N'), ((2, 7),
'N'), ((1, 7), 'N'), ((1, 8), 'N'), ((0, 8), 'N'), ((0, 9), 'N'), ((1, 9),
'N'), ((0, 9), 'N'), ((0, 8), 'N'), ((0, 9), 'N'), ((0, 8), 'N'), ((0, 9),
'N'), ((1, 9), 'N'), ((0, 9), 'N'), ((1, 9), 'N'), ((1, 8), 'N'), ((1, 9),
'N'), ((0, 9), 'N'), ((1, 9), 'N'), ((2, 9), 'N'), ((3, 9), 'N'), ((4, 9),
'N'), ((5, 9), 'N')]"

444,straight_until_obstacle,"[((6, 5), 'E'), ((7, 5), 'E'), ((8, 5), 'E'),
((9, 5), 'W'), ((8, 5), 'W'), ((7, 5), 'W'), ((6, 5), 'W'), ((5, 5), 'W'),
((4, 5), 'W'), ((3, 5), 'W'), ((2, 5), 'W'), ((1, 5), 'N'), ((1, 4), 'W'),
((0, 4), 'N'), ((0, 3), 'N'), ((0, 2), 'N'), ((0, 1), 'N'), ((0, 0), 'S'),
((0, 1), 'S'), ((0, 2), 'S'), ((0, 3), 'S'), ((0, 4), 'S'), ((0, 5), 'S'),
((0, 6), 'S'), ((0, 7), 'S'), ((0, 8), 'S'), ((0, 9), 'N'), ((0, 8), 'N'),
((0, 7), 'N'), ((0, 6), 'N'), ((0, 5), 'N'), ((0, 4), 'N'), ((0, 3), 'N'),
((0, 2), 'N'), ((0, 1), 'S'), ((0, 2), 'N'), ((0, 1), 'N'), ((0, 0), 'E'),
((1, 0), 'E'), ((2, 0), 'E'), ((3, 0), 'E'), ((4, 0), 'E'), ((5, 0), 'E'),
((6, 0), 'E'), ((7, 0), 'E'), ((8, 0), 'E'), ((9, 0), 'W'), ((8, 0), 'W'),
((7, 0), 'E'), ((8, 0), 'N')]"

765,random_walk,"[((4, 5), 'W'), ((5, 5), 'N'), ((5, 4), 'N'), ((5, 5),
'N'), ((6, 5), 'N'), ((6, 6), 'N'), ((6, 5), 'N'), ((5, 5), 'N'), ((5, 4),
'N'), ((5, 3), 'N'), ((6, 3), 'N'), ((7, 3), 'N'), ((7, 4), 'N'), ((7, 5),
'N'), ((7, 4), 'N'), ((6, 4), 'N'), ((5, 4), 'N'), ((5, 5), 'N'), ((4, 5),
'N'), ((3, 5), 'N'), ((3, 6), 'N'), ((3, 5), 'N'), ((2, 5), 'N'), ((3, 5),
'N'), ((3, 6), 'N'), ((4, 6), 'N'), ((5, 6), 'N'), ((5, 7), 'N'), ((4, 7),
'N'), ((3, 7), 'N'), ((3, 8), 'N'), ((3, 9), 'N'), ((4, 9), 'N'), ((3, 9),
'N'), ((3, 8), 'N'), ((4, 8), 'N'), ((4, 9), 'N'), ((3, 9), 'N'), ((3, 8),
'N'), ((4, 8), 'N'), ((5, 8), 'N'), ((6, 8), 'N'), ((7, 8), 'N'), ((7, 7),
'N'), ((7, 6), 'N'), ((7, 5), 'N'), ((7, 4), 'N'), ((7, 5), 'N'), ((6, 5),
'N'), ((6, 6), 'N')]"

765,straight_until_obstacle,"[((6, 5), 'E'), ((7, 5), 'N'), ((7, 4), 'E'),
((8, 4), 'N'), ((8, 3), 'N'), ((8, 2), 'E'), ((9, 2), 'N'), ((9, 1), 'N'),
((9, 0), 'S'), ((9, 1), 'N'), ((9, 0), 'W'), ((8, 0), 'W'), ((7, 0), 'W'),
((6, 0), 'W'), ((5, 0), 'W'), ((4, 0), 'W'), ((3, 0), 'W'), ((2, 0), 'W'),
((1, 0), 'W'), ((0, 0), 'S'), ((0, 1), 'S'), ((0, 2), 'S'), ((0, 3), 'S'),
((0, 4), 'S'), ((0, 5), 'S'), ((0, 6), 'S'), ((0, 7), 'S'), ((0, 8), 'E'),
((1, 8), 'S'), ((1, 9), 'E'), ((2, 9), 'E'), ((3, 9), 'E'), ((4, 9), 'E'),
((5, 9), 'E'), ((6, 9), 'E'), ((7, 9), 'N'), ((7, 8), 'S'), ((7, 9), 'E'),
((8, 9), 'W'), ((7, 9), 'W'), ((6, 9), 'W'), ((5, 9), 'W'), ((4, 9), 'W'),
```

```
((3, 9), 'W'), ((2, 9), 'W'), ((1, 9), 'W'), ((0, 9), 'N'), ((0, 8), 'N'),
((0, 7), 'N'), ((0, 6), 'N')]"
```