[A3] - Uncertainty, Bayesian Networks, HMM

Keshav Chhabra(2022247)

a)

Direct Sampling

Easy to operate and fair when applied properly. Performs well with frequent occurrences . However, it does not perform well with infrequent occurrences (e.g., bus travel that is not stressful) and requires sampling from the entire joint distribution.

Rejection Sampling

Handles conditional probabilities well (for example, P(Leisure|Train)=0.4). It is flexible when direct sampling is hard. However, it wastes rejected samples, especially for rare situations, and is not efficient with high rejection rates.

Gibbs Sampling

Good for problems with complicated connections and clear conditions. It is useful for estimating chances like P(Air|Stressed)=0.6. However, it creates samples that depend

on each other and needs a warm-up time. Also, it can get stuck in local peaks.

Direct sampling is best for simple situations like this circumstance . Rejection sampling is good for conditional probabilities but does not work well for rare events. Gibbs sampling is great for complex systems that depend on each other but needs careful adjustment.

b)

Calculating expected leisure train travelers:

Given: P(leisure|train) = 0.400

Sample size = 100

Number preferring train = 30

Expected leisure travelers = 30 × 0.400 = 12 people

Therefore, out of the 30 people who prefer train travel, we would expect 12 people to be traveling for leisure.

c)

Given: P(air) = 0.80

P(business|air) = 0.20

Using multiplication rule:

P(air AND business) = P(air) × P(business|air)

P(air AND business) = 0.80 × 0.20 = 0.160

Therefore, the probability is 0.160 or 0.160 rounded to three decimal places.

d) **To estimate the probability that a person prefers traveling by bus (P(Bus)), we use the information from the dataset:**

The joint probability of preferring bus travel and having low stress is P(Bus and Low Stress) = 0.015.

The conditional probability of having low stress given bus travel is P(Low Stress | Bus) = 0.35.

P(Bus and Low Stress) = P(Bus) * P(Low Stress | Bus)

From this ,

P(Bus) = P(Bus and Low Stress) / P(Low Stress | Bus)

P(Bus) = 0.015 / 0.35 ~ 0.043

Interpretation:

This calculation estimates that about 4.3% of people prefer traveling by bus. The assumption here is that the given conditional probability P(Low Stress | Bus) = 0.35 applies uniformly across bus travelers, and the low-stress level isn't exclusive to certain subsets of bus travelers.

Part 2

- B: A person reads books.
- J: A person accesses academic journals.
- C: A person participates in book clubs.

Given Probabilities:

1. $P(B \cup J) = 0.910$
2. $P(J \cap B) = P(J|B) * P(B) = 0.40 * P(B)$ and $P(B \cap J') = 0.60 * P(B)$
3. $P(C|B) = 0.320$
4. $P(J \cap B') = 0.227$
5. $P(B' \cap J) = 0.090$
6. $P(J|B') = 0.716$
7. $P(C \cap J) = 0.088$
8. $P(C \cup J) = 0.631$
9. $P(J|C) = 0.400$

10. P(J) = 0.500
11. P(C|B') = 0.0044

b) We will check how they obey the Kolmogorov rules of probability. These are:

Non-negativity: P(A)≥0 for any event AA.

Normalization : P(S)=1 where S denotes sample space.

Additivity for disjoint events: If A and B are disjoint, then P(A∪B)=P(A)+P(B)

We will determine P(B) , P(J) and verify that the sum of all probabilities is 1.

From the metrics :

P(B∪J)=0.910

P(~B∩~J)=0.090.

So $P(B \cup J)+P(\sim B \cap \sim J)=1P(B \cup J)+P(\sim B \cap \sim J)=1$, which thus satisfies Normalization.

Break up $P(B \cup J)$:

Using the inclusion-exclusion rule,

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

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

From the statements:

$P(\sim B \cap J)=0.227$implies $P(J)=P(B \cap J)+P(\sim B \cap J)=P(J \mid B)P(B)+0.227$

$P(J)=0.500$ (given).

Finding $P(B)$:

Using $P(B \cap J)=P(J \mid B)P(B)$

$P(B \cap J)=0.40P(B)$.

From $P(B \cup J)=0.910P(B \cup J)=0.910$:

$0.910 = P(B) + P(J) - 0.40P(B)$,

$0.910 = P(B) + 0.500 - 0.40P(B)$,

$0.910 = 0.60P(B) + 0.500$,

$0.60P(B) = 0.410 \Longrightarrow P(B) = 0.4100.60 = 0.683.$

$0.60P(B) = 0.410 \Longrightarrow P(B) = 0.600.410 = 0.683.$

Checking P(J):

From $P(J) = 0.40P(B) + 0.227P(J) = 0.40P(B) + 0.227$:

$P(J) = 0.40(0.683) + 0.227 = 0.2732 + 0.227 = 0.500.$

Check $P(\sim B \cap \sim J)$:

From $P(\sim B \cap \sim J) = 1 - P(B \cup J)$

$P(\sim B \cap \sim J) = 1 - 0.910 = 0.090.$

So, individual probabilities sum to 1 :

$P(B \cup J) + P(Bc \cap Jc) = 0.910 + 0.090 = 1.$

$P(B \cup J)+P(\sim B \cap \sim J)=0.910+0.090=1.$

Test Additivity for Mutually Exclusive Events

The probabilities for disjoint events (for example $P(\sim B \cap J)P(\sim B \cap J)$ and $P(B \cap \sim J)P(B \cap \sim J)$) are:

$P(\sim B \cap J)=0.227$

$P(B \cap \sim J)=P(B)-P(B \cap J)=0.683-0.40(0.683)=0.683-0.2732=0.4098.$

For $P(B \cup J)=P(B)+P(J)-P(B \cap J)P(B \cup J)=P(B)+P(J)-P(B \cap J)$: $P(B \cup J)=0.683+0.500-0.40(0.683)=0.910.$

This equals the given $P(B \cup J)$, thus showing additivity is indeed true.

Check Non-Negativity All calculated probabilities are not negative:

$P(B)=0.683$

$P(J)=0.500$

$P(\sim B \cap \sim J)=0.090$

d) Event | Probability

$C \cap B \cap J$ |0.088

$C \cap B \cap J'$ | 0.2186

$C' \cap B \cap J$ | 0.0864

$C' \cap B \cap J'$ | 0.0952

$C \cap B' \cap J$ | 0.07264

$C \cap B' \cap J'$ | 0.1541

$C' \cap B' \cap J$ | 0.0437

$C' \cap B' \cap J'$ | 0.0044

Part 3

(a) **Problem Defining Using Bayesian Inference**

We demonstrate using Bayesian inference how new information about backdoor attacks does or does not influence belief that adversarial changes cause a misclassification.

Initial Installation:

Let A show the adversarial changes.

Let B demonstrate the existence of a backdoor trigger.

Let M represent the misclassification alarm.

Initially, ***A and B are independent, so: P(A,B)=P(A)P(B)***

We should compute P(A | M,B), the probability of adversarial changes conditioned on observing a misclassification alarm and the presence of a backdoor attack.

Using Bayes' rule:

$P(A|M,B)=P(M|A,B)P(A|B)/P(M|B)$

Since A and B were original independent.

$P(A|B)=P(A)$

Therefore, the expression gets simplified to:
$P\{A|M,B\}=P(M|A,B)P(A)/P(M|B)$

$P(A|M,B)=P(M|B)P(M|A,B)P(A)$
$P(M|A,B)$: Likelihood of misclassification given both attacks are present.
$P(A)$: Prior probability of adversarial perturbations.
$P(M|B)$: Marginal likelihood of misclassification given only backdoor triggers.

b) **Defining the Probabilities**

1. **Prior Probabilities:**

- $P(A)$: Probability of adversarial perturbations (initially independent).
- $P(B)$: Probability of backdoor triggers (initially independent).

2. **Likelihood Terms:**

- $P(M|A,B)$: Probability of misclassification when both attacks are present.
- $P(M|A,\neg B)$: Probability of misclassification due to adversarial perturbations only.

- $P(M|\neg A,B)$: Probability of misclassification due to backdoor triggers only.
- $P(M|\neg A,\neg B)$: Probability of misclassification with no attack.

3. **Posterior Probability:**

- $P(A|M,B)$: Updated probability of adversarial perturbations given misclassification and backdoor trigger presence.

c) Explaining Away: If B (backdoor trigger) alone sufficiently explains misclassification M, then $P(A|M,B) < P(A|M)$, where A represents adversarial perturbations.

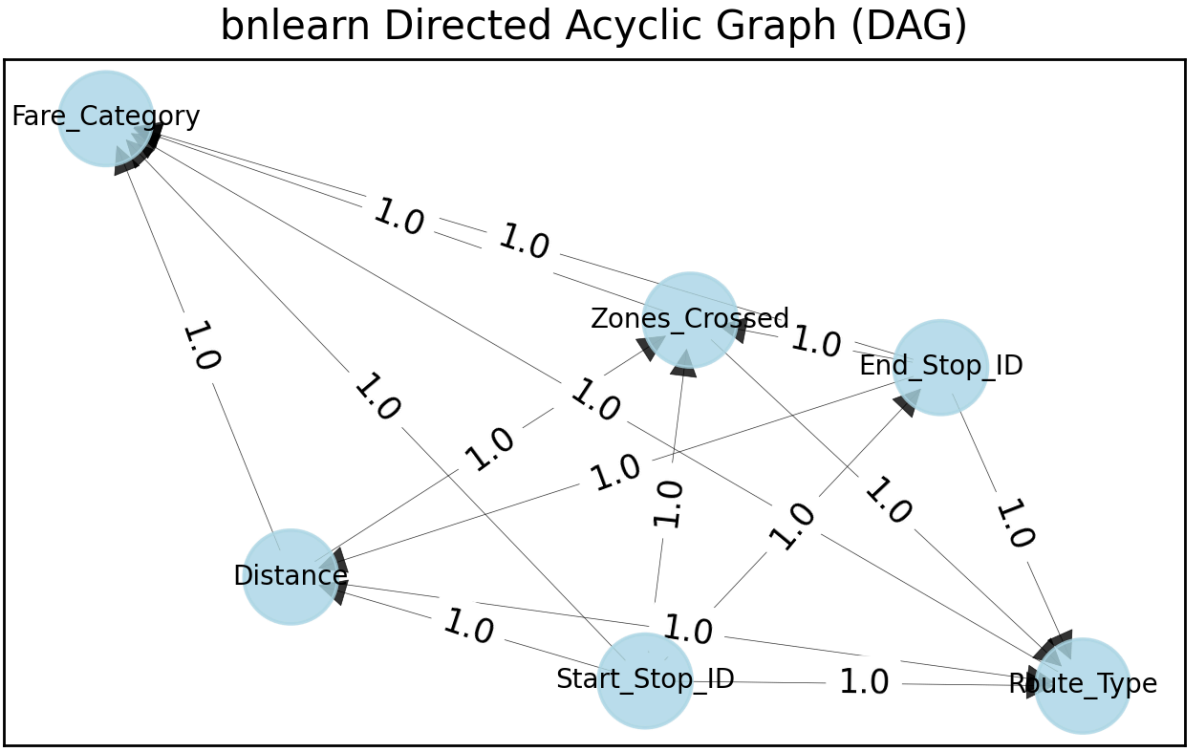
1. Reasoning: Detecting reliable B creates a negative correlation between A and B, even if initially independent.
2. Quantitative Impact: The decrease in $P(A|M,B)$ depends on:
    - The reliability of backdoor triggers $P(M|B)$
    - The prior probabilities of A and B
    - The likelihoods of misclassification under different A and B combinations

This demonstrates selection bias, where conditioning on a common effect (misclassification alarm) induces a negative correlation between independent causes A and B.

Computational Part

Task 1: Construct the initial Bayesian Network (A) for fare classification.

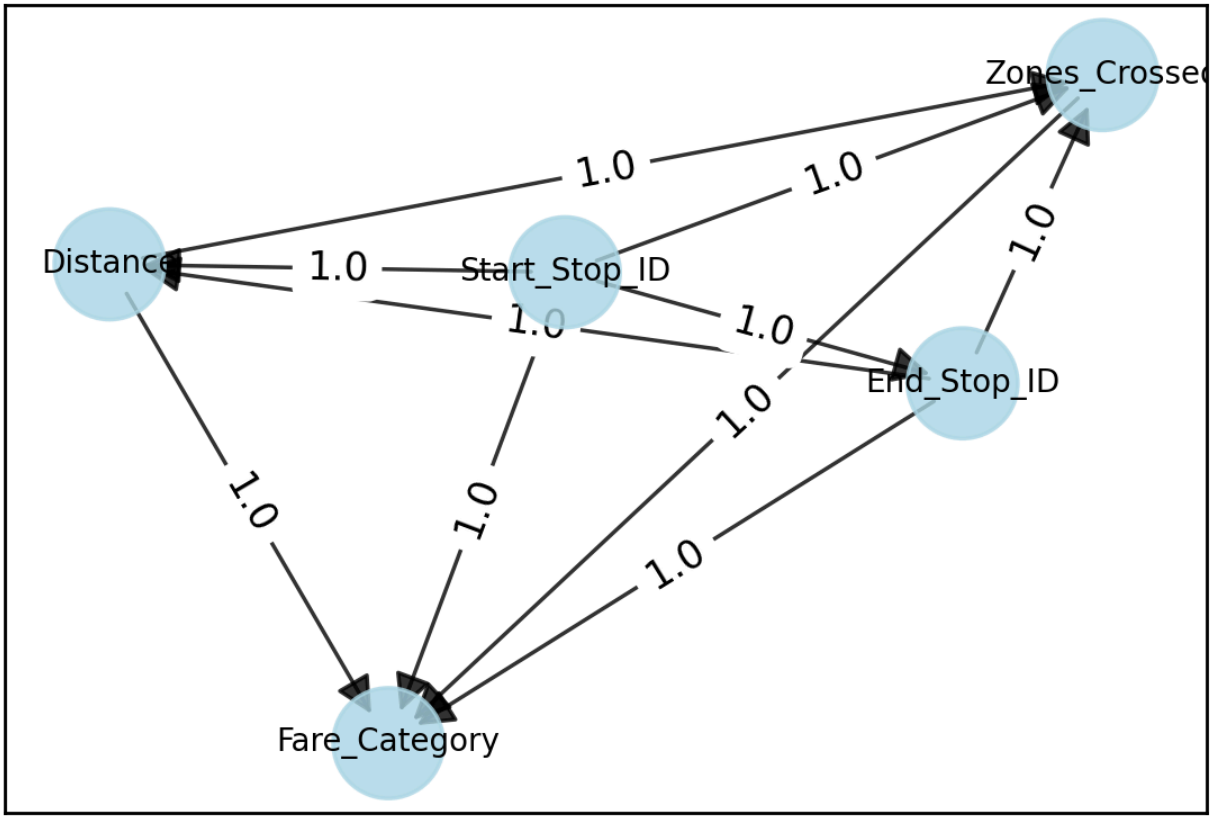
We have taken all the pairs of reasonable dependencies while maintaining the acyclic graph property .




```
Time taken to fit the model in part a) : 2407.20 seconds
```

```
Total Test Cases: 350
Total Correct Predictions: 350 out of 350
Model accuracy on filtered test cases: 100.00%
[+] Done
```

Task 2: Prune the initial Bayesian Network (A) to enhance performance.




```
Time taken to fit the model in part b) : 2385.34 seconds
```

```
[bnlearn] >Compute edge strength with [chi_square]
[bnlearn] >5 edges are removed with P-value > 0.05 based on chi_square
```

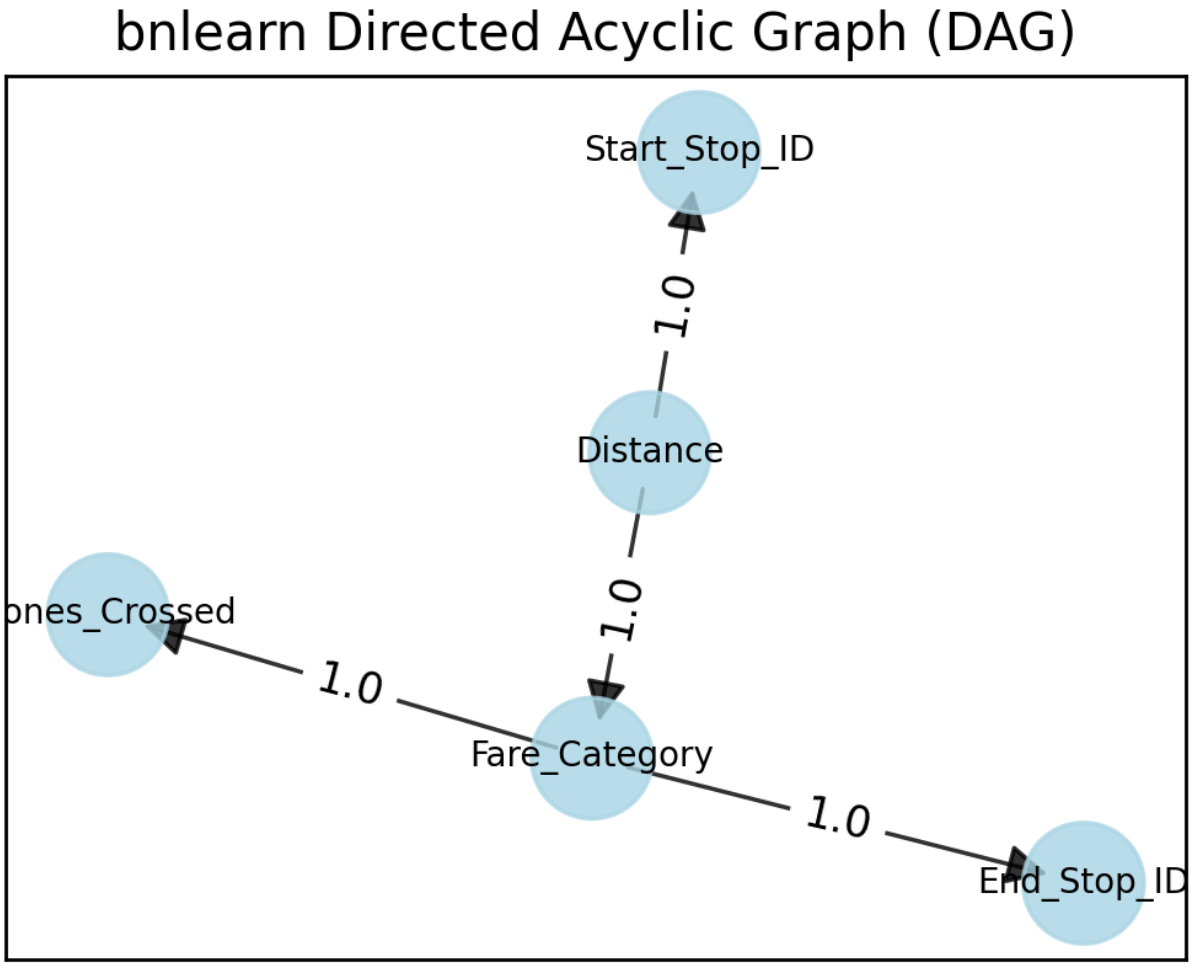
```
Total Test Cases: 350
Total Correct Predictions: 350 out of 350
Model accuracy on filtered test cases: 100.00%
[+] Done
```

Clearly explain the pruning method applied and how it improves the model's efficiency (time taken to fit the data) and/or prediction accuracy.

The pruning method tries to establish independence between variables in the dataset using tests like Chi-square independence test .
With fewer dependencies, the parameter learning step requires estimating fewer CPTs, each involving fewer variables. This drastically reduces the computation time for models with high-dimensional datasets.

So basically **edge pruning was done and 5 edges** were removed as a result of this procedure .

Task 3: Optimize the Bayesian Network (A) by adjusting parameters or using structure refinement methods.




```
Time taken to fit the model in part c) : 22.37 seconds
```

```
Total Test Cases: 350
Total Correct Predictions: 350 out of 350
Model accuracy on filtered test cases: 100.00%
```

Compare the performance of the optimized Bayesian network with the initial network (A) and explain how the optimization improves the model's accuracy and/or efficiency.

It improves efficiency by maximizing scoring metrics like BIC, the model avoids overfitting, simplifying the CPTs and making computations more efficient. We can see a drastic reduction

from 2407 seconds to 22.37s . Regardless of the system conditions in which it was run , this is a vast improvement .

Part 2)

HMM Question

$S = \{(x, y) \mid 0 \leq x < 10, 0 \leq y < 10, x, y \in \mathbb{Z}\}$

An HMM is a temporal probabilistic model in which the state of the process is described by a single, discrete random variable(Norvig)

Transition Probabilities: $P((x_t, y_t) \mid (x_{t-1}, y_{t-1}))$

The transition probabilities will depend on the movement policy adopted by the robot

---

For RANDOM walk :

The Roomba chooses a random direction (N, E, S, W) and moves one step.

Case 1: (x, y) is not next to an obstacle

- The set of possible next states (S) consists of the four neighbors:

$S = \{(x+1, y), (x-1, y), (x, y+1), (x, y-1)\}$

- All these neighbors are valid because there are no obstacles near (x, y).

Case 2: (x, y) is next to an obstacle

- The set of possible next states (S) includes:
  1. The current position (x, y) (since the Roomba stays in place if an obstacle prevents movement).
  2. All valid neighbors (x', y') such that:

$|x' - x| + |y' - y| = 1 , x' \in [0, 9] , y' \in [0, 9]$

Thus, the set of possible states is:

$S = \{(x, y)\} \cup \{ (x', y') \mid |x' - x| + |y' - y| = 1, x', y' \in [0, 9] \}$

2. Transition Probabilities (P(x', y' | x, y))

The transition probabilities are uniform over the set of possible next states S since the choice is random .

$P(x', y' \mid x, y) = \{ 1 / |S|, \text{if } (x', y') \in S$

{ 0, otherwise.

Where:

- |S| is the size of the set S, which depends on whether (x, y) is next to an obstacle:
  - If (x, y) is not next to an obstacle: |S| = 4.
  - If (x, y) is next to an obstacle: |S| ≤ 4

For STRAIGHT UNTIL OBSTACLE

1. Possible Next States (S)

For a given state (x, y), the possible next states depend on whether the Roomba encounters an obstacle in its current heading or not:

Case 1: (x, y) is not next to an obstacle

- The Roomba moves one step in its current heading direction (North, East, South, or West).
- The possible next state is deterministic:

S = {(x', y')}, (x', y') = (x, y) + MOVEMENTS[heading]

Where MOVEMENTS[HEADING] corresponds to the unit movement in the Roomba's current heading (e.g., North - (0, -1)).

Case 2: (x, y) is next to an obstacle

- The Roomba cannot move in its current direction because it hits an obstacle.
- In this case:
    - The Roomba randomly selects a new valid heading from the other directions.
    - The possible next states (S) are:

$$S = \{(x', y') \mid |x' - x| + |y' - y| = 1, x', y' \in [0, 9]\}$$

This represents all valid neighbors of (x, y) within the grid boundaries.

---

2. **Transition Probabilities (P(x', y' | x, y))**

Case 1: (x, y) is not next to an obstacle

- The Roomba deterministically moves to the next state based on its current heading:

$$P(x', y' \mid x, y) = \{ 1, \text{if } (x', y') = (x, y) + \text{MOVEMENTS[heading]}$$

{ 0, otherwise.

Case 2: (x, y) is next to an obstacle

- The Roomba randomly selects a new heading and moves to one of the valid neighboring states.
- The transition probabilities are uniform across valid neighbors:

$$P(x', y' \mid x, y) = \begin{cases} 1 / |S|, & \text{if } (x', y') \in S \\ 0, & \text{otherwise.} \end{cases}$$

Where:

- $|S|$ is the number of valid neighbors (grid boundaries considered).

EMISSION PROBABILITY

It will model the probability of observing the noisy location captured by the sensors(the evidence variable) given the true location (hidden variable) .

It is given that the observations are modeled as the true position plus Gaussian noise with mean zero and standard deviation σ = 1.0.

2. Emission Probabilities (P(obsx, obsy | x, y))

Assumptions:

- Sensor noise is modeled as a Gaussian distribution with:
  - Mean: The true position of the Roomba (x, y).
  - Standard deviation (σ): Provided in the problem (e.g., σ = 1.0).

The emission probability is given by the probability density function (PDF) of the Gaussian distribution centered at the true position:

$$P(\text{obs}_x, \text{obs}_y \mid x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(x - \text{obs}_x)^2 + (y - \text{obs}_y)^2}{2\sigma^2}\right)$$

Here:

(x, y): The true state of the Roomba.

(obs_x, obs_y): The observed noisy position.

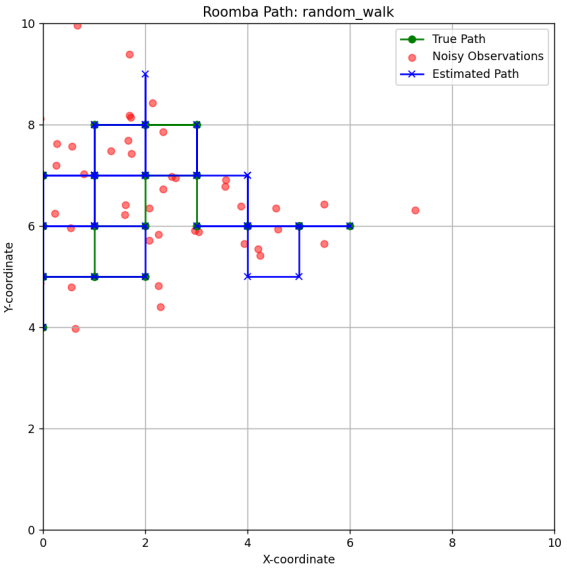
σ^2: The variance of the noise.

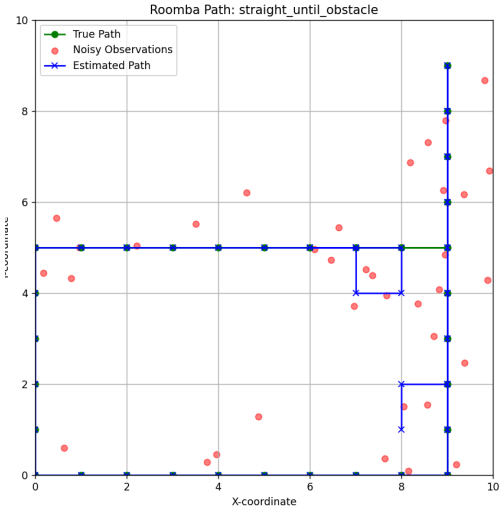
If we are allowed to make the assumption that the noise in x and y are independent then

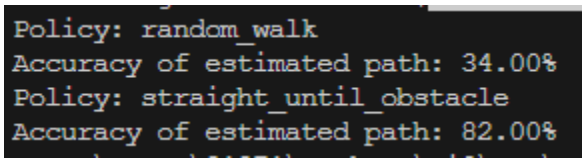
$$P(\text{obs}_x, \text{obs}_y \mid x, y) = P(\text{obs}_x \mid x) \cdot P(\text{obs}_y \mid y)$$

$$P(\text{obs}_x \mid x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x - \text{obs}_x)^2}{2\sigma^2}\right)$$

$$P(\text{obs}_y \mid y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y - \text{obs}_y)^2}{2\sigma^2}\right)$$





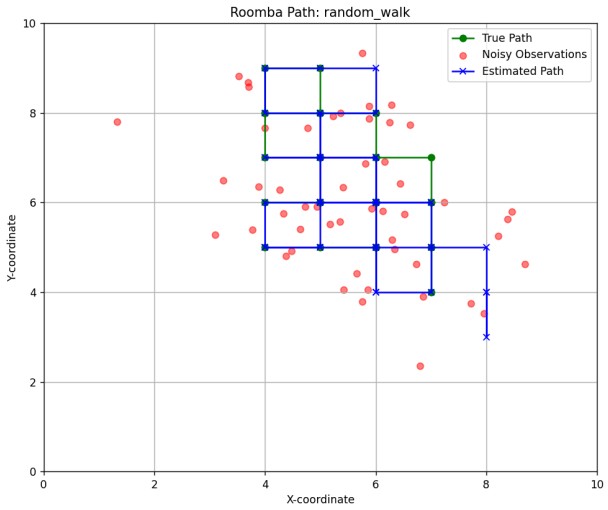


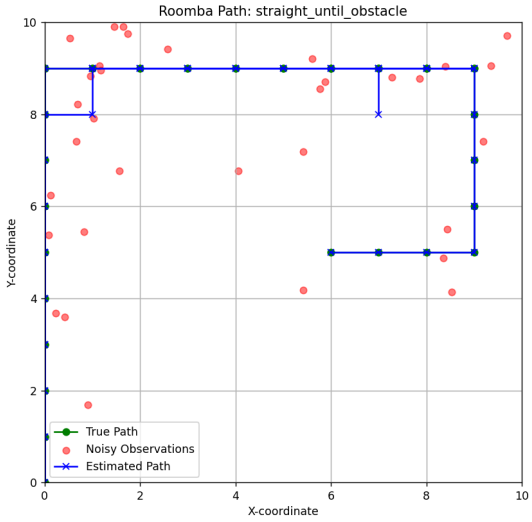
seed : 111

The image shows that the "straight_until_obstacle" policy has an accuracy of estimated path of 82.00%, which is higher than the 34.00% accuracy of the "random_walk" policy.

The graph on the right also visually demonstrates that the "straight_until_obstacle" path follows the target path more closely compared to the "random_walk" path . This may happen because the true path is more accurately traversed approximately by the "straight_until_obstacle" .

In the "straight_until_obstacle" policy, the deterministic movement along straight lines reduces the search space of possible paths for the Viterbi algorithm, leading to higher accuracy. In contrast, the frequent direction changes in the "random_walk" policy significantly expand the search space, making it more challenging for the algorithm to converge on the true path






```
Accuracy of estimated path: 56.00%
Policy: straight_until_obstacle
Accuracy of estimated path: 80.00%
```

Seed : 150

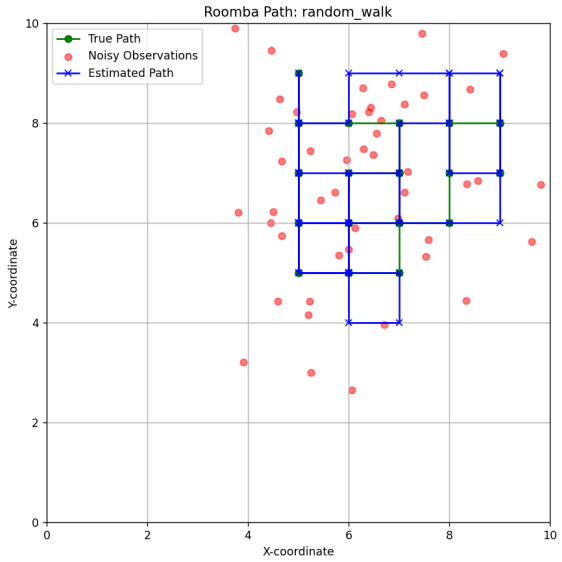
The image shows that the "straight_until_obstacle" policy has an accuracy of estimated path of 80.00%, which is higher than the 56.00% accuracy of the "random_walk" policy.

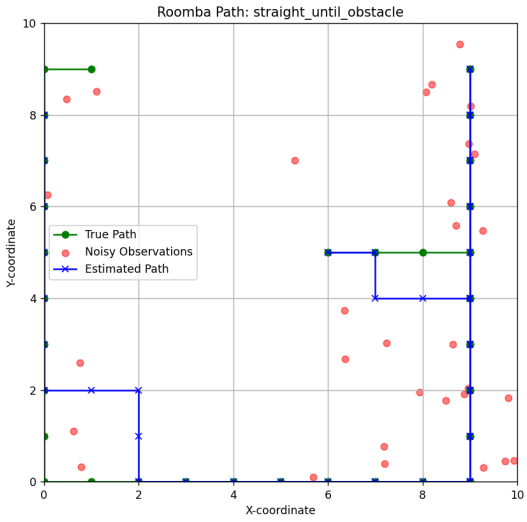
The graph on the right also visually demonstrates that the "straight_until_obstacle" path follows the target path more closely compared to the "random_walk" path . This may happen because the true path is more accurately traversed approximately by the "straight_until_obstacle" .

In the “straight_until_obstacle” policy, the deterministic movement along straight lines reduces the search space of possible paths for the Viterbi algorithm, leading to higher

accuracy. In contrast, the frequent direction changes in the "random_walk" policy significantly expand the search space, making it more challenging for the algorithm to converge on the true path

---

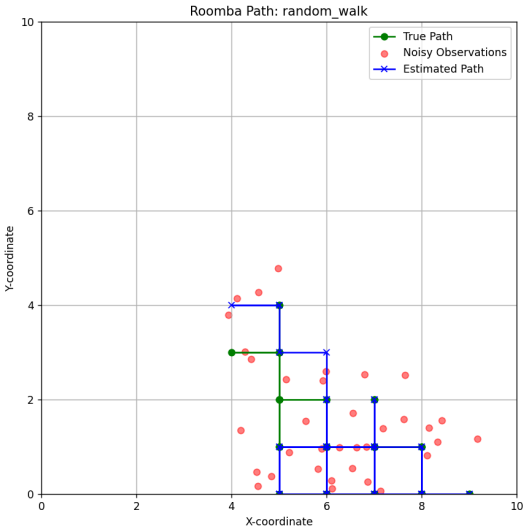


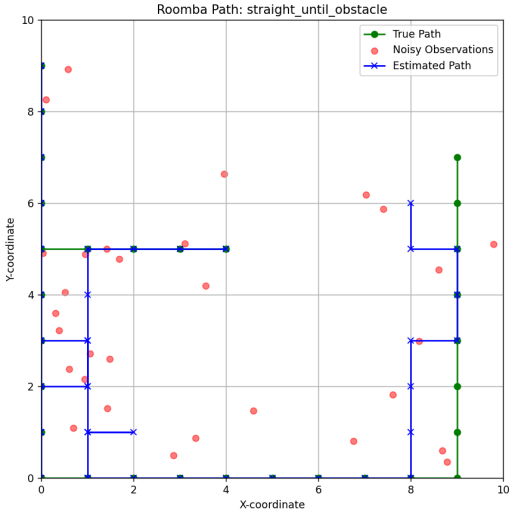



```
Policy: random_walk
Accuracy of estimated path: 50.00%
Policy: straight_until_obstacle
Accuracy of estimated path: 56.00%
```

Seed : 142

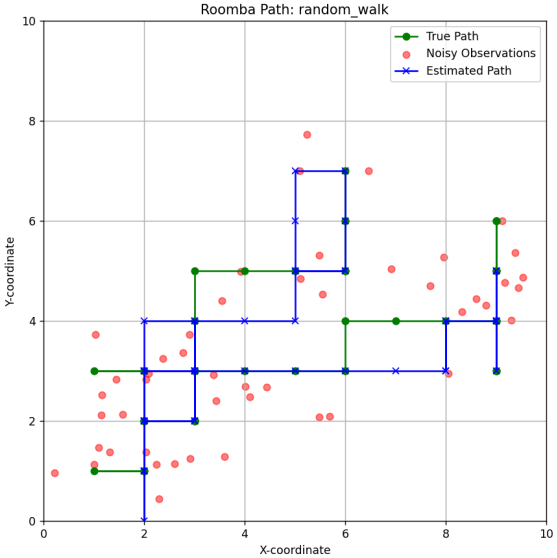
The same reasoning for the above part applies to this .

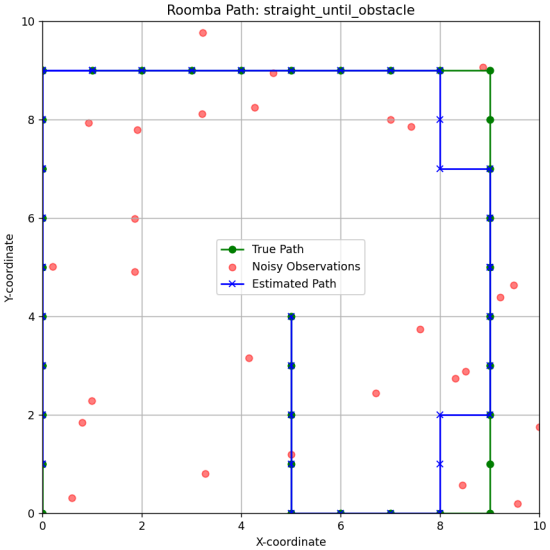





```
Accuracy of estimated path: 44.00%
Policy: straight_until_obstacle
Accuracy of estimated path: 60.00%
```

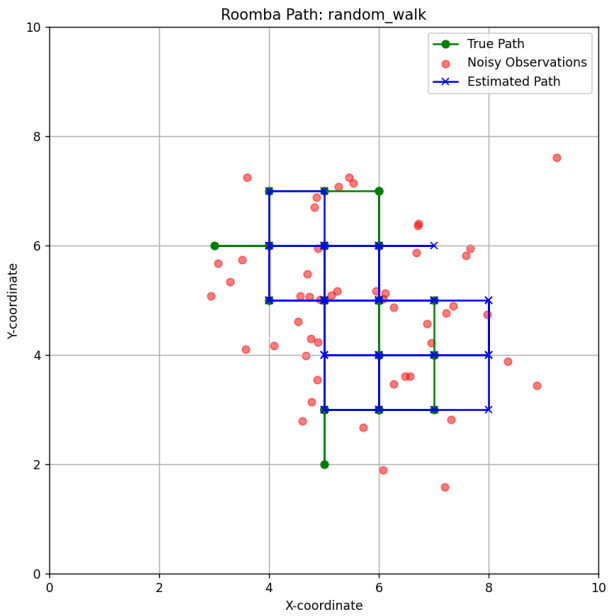
Seed : 180

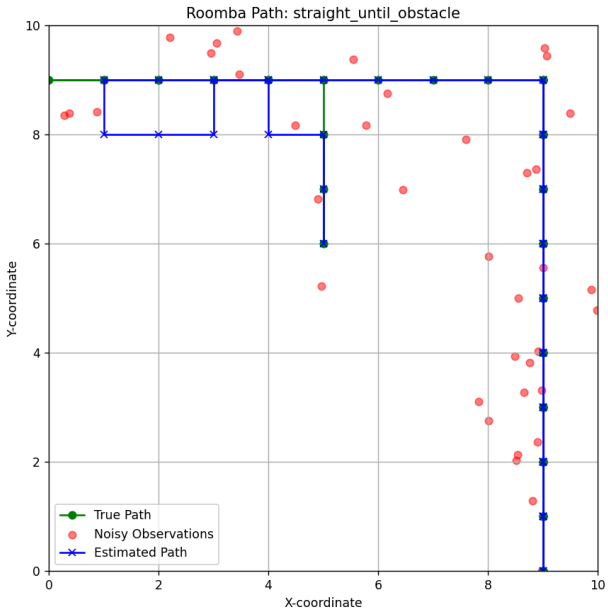




```
Policy: random_walk
Accuracy of estimated path: 64.00%
Policy: straight_until_obstacle
Accuracy of estimated path: 60.00%
```

Seed : 222






```
Policy: random_walk
Accuracy of estimated path: 42.00%
Policy: straight_until_obstacle
Accuracy of estimated path: 78.00%
```

Seed : 300
