Designing Scalable Belief Space Planning Algorithms for Coordinating Robots with Inconsistent Beliefs

Student Name: Aditya Yadav
Roll Number: 2022041

BTP report submitted in partial fulfillment of the requirements
for the Degree of B.Tech. in Computer Science & Applied Mathematics

BTP Track: Research

BTP Advisor
Dr. Tanmoy Kundu

Indraprastha Institute of Information Technology
New Delhi

Student's Declaration

I hereby declare that the work presented in the report entitled **"Designing Scalable Belief Space Planning Algorithms for Coordinating Robots with Inconsistent Beliefs"** submitted by me for the partial fulfillment of the requirements for the degree of *Bachelor of Technology* in *Computer Science & Applied Mathematics* at Indraprastha Institute of Information Technology, Delhi, is an authentic record of my work carried out under the guidance of **Dr. Tanmoy Kundu**. Due acknowledgments have been given in the report to all material used. This work has not been submitted elsewhere for the award of any other degree.

Aditya Yadav **Place & Date: Delhi, 24.04.2025**

Certificate

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Dr. Tanmoy Kundu **Place & Date: Delhi, 27.11.2024**

Abstract

Multi-robot coordination is crucial for reliable autonomy even in a limited-communication environment in which robots can communicate infrequently and can share limited amount of information between themselves. We develop a technique in which a robot-robot communication is triggered only when a "significant event" is encountered by a robot. A robot seeks information related to a particular event and the helping robot shares its knowledge about the same. Thus, a robot consolidates its belief about an event via gathering information from other robots in the environment. Depending on limited and erroneous sensing, a robot has partial observability of the state of the environment. We model this as a decentralized Partially Observable Markov Decision Process (POMDP) from the perspective of each robot which maintains a "belief" over the state of the environment. Belief is a probability distribution over the states of the environment. New observations, obtained by a robot itself or shared by other robots, evolve the belief of a robot. Considering the erroneous sensing capabilities, a robot needs to consolidate its belief about some significant state of the environment, at times. This enables identify a significant state with more certainty. Previous research proposes sharing all the unshared observations between a given pair of robots. However, limited connectivity restricts the robots to share all the unshared data. Our approach enables a robot consolidate its belief about the significant state by gathering information from its neighboring robots. While the likelihood of occurring the significant state is less than a pre-decided threshold, the robot requests its neighbors for selective information. Experimental results validate the efficacy of our approach for belief consolidation and thus identify significant events with higher likelihood.

Acknowledgments

I would like to express my sincere gratitude to my advisor, **Dr. Tanmoy Kundu**, for his constant guidance, support, and encouragement throughout the course of this thesis. His expertise and thoughtful feedback were invaluable in shaping this work.

I also extend my thanks to my friend **Ayush Singhal**, who has been working on the same topic. His insightful discussions and collaboration have significantly enhanced my understanding of the subject and contributed to the successful completion of this work.

Lastly, I would like to thank my family and friends for their unwavering support and encouragement during the course of this project. Without their constant motivation, this work would not have been possible.

Contents

Introduction

This work tackles the core challenge of achieving joint action consistency in decentralized multi-robot systems operating with inconsistent beliefs and limited communication. In real-world environments such as disaster zones, exploration missions, or surveillance tasks, robots cannot always communicate frequently due to bandwidth limits, noise, or range constraints. This leads to belief divergence—each robot forms a different understanding of the environment, making coordinated action difficult. Traditional approaches, often built on centralized control or assumptions of perfect communication, become impractical in such scenarios due to computational and scalability limitations. To address this, we propose a two-pronged framework that balances coordination, belief synchronization, and minimal communication.

The first method is based on Bayesian games, where robots iteratively exchange partial belief information with one another, using a controlled sharing parameter. The belief update process is implemented in a simulation using a Python script where each robot's belief is stored as a probability vector over possible environment states. At each iteration, robots multiply their belief vector element-wise with that of their peer, scale it by a sharing factor (delta), and normalize the result to ensure the probabilities sum to 1. This iterative process gradually aligns their beliefs, while a divergence metric like KL divergence checks for convergence. Once beliefs are sufficiently aligned, each robot computes expected utility for each action using its current belief and selects the optimal one, ensuring coordinated decision-making with minimal information exchange.

The second method models the coordination problem as a factor graph inside a Partially Observable Markov Decision Process (POMDP) framework. Robots are allowed to operate autonomously using local information unless a trigger condition is met (e.g., belief divergence, map change, or victim detection), in which case selective communication is initiated. This approach leverages self-triggered communication and Disjoint Set Union (DSU) data structures to dynamically manage belief clusters. Robots within the same belief cluster merge their information, ensuring internal consistency. The framework is implemented as a layered architecture: starting from Bayesian belief updates on the grid map, communication overlays to manage triggers and gossip protocols, CVT-based frontier allocation to distribute exploration areas, and D* Lite path planning to find optimal movement paths based on each robot's map knowledge.

By combining belief update algorithms, utility-based action selection, graph-based decision-making, and minimal communication design, this system maintains high-level coordination in decentralized robot teams. It scales efficiently with the number of agents, handles real-world communication limitations, and guarantees bounded reward degradation. Together, these techniques form a robust, adaptive, and mathematically grounded solution to decentralized multi-robot planning under uncertainty.

Chapter 1

Initialization

1.1 Problem Cast as a Degenerate Dec-POMDP

We model grid exploration with n robots as the tuple

$$\mathcal{M} = \langle X,\ Z,\ A,\ T,\ O,\ \rho,\ n \rangle,$$

where

X	world state $\{0,1,2\}^{H\times W}$ (static),
Z	joint observation space $\{0,1,2\}^{n\lvert\mathcal{N}_R\rvert}$,
A	joint action space $\{\Uparrow,\downarrow,\leftarrow,\rightarrow,\text{stay}\}^n$,
T	$T(x'\lvert x,a) = \mathbf{1}\{x' = x\}$ (no dynamics),
O	product of i.i.d. cell-wise likelihoods $\mathcal{L}$ (Eq. (**??**)),
ρ	information-gain reward (Def. 1.2).

Because T is deterministic, the belief evolution is *purely Bayesian* and requires no prediction step.

[Local belief tensor] $B_i \in [0,1]^{H\times W\times 3}$ with $\sum_{\sigma=0}^{2} B_i[p,\sigma] = 1\ \forall p$.

1.1.1 Bayesian Measurement Update

For an observation pair $\langle p, z\rangle$ the update is the GPU-batched Eq. (**??**). Lemma **??** guarantees row-stochasticity, hence B_i remains a valid probability tensor.

[Global belief] The joint belief is the product measure $B = \bigotimes_{i=1}^{n} B_i$.

1.2 Information-Theoretic Reward

[Cell-wise sparsification reward] For robot i at step k

$$\rho(B_i) = 1 - \frac{1}{HW}\sum_{p\in\mathcal{G}} \mathrm{KL}\big(B_i[p,:] \,\|\, \delta_{X(p)}\big).$$

The reward is 1 when B_i is fully concentrated ($\mathrm{KL} = 0$) and decreases smoothly otherwise. Lemma 3.1 proves ρ is 1-Lipschitz w.r.t. ℓ_1, a key fact for our communication trigger.

1.3 Layered Deterministic Stack

Why Four Layers?

[label=**L0**,leftmargin=1.2cm]**Belief propagation** (GPU-Bayes) — stochastic. **Communication overlay** (DSU) — deterministic. **Frontier allocation** (CVT / Auction) — deterministic. **Incremental planning** (D*-Lite) — deterministic.

Because layers 2–4 are deterministic *functions* of their inputs, once robots inside each DSU cluster share the same belief slice they *must* output identical targets and paths; see Thm. 3.1. That is the entire trick behind joint-action consistency.

1.4 Self-contained Figure

tikz

[x=6cm,y=1.5cm] //in 0/GPU Bayes/gray!25, 1/Comm DSU/gray!20, 2/CVT / Auction/gray!15, 3/D*-Lite/gray!10 [fill=] (-0.45,) rectangle (0.45,+0.85) node[midway]; [-¿,thick] (0,3.85) – (0,4.3) node[above]Joint action; [-¿,thick] (0,-0.2) – (0,-0.6) node[below]Sensor data;

Figure 1.1: Automatic fallback: four-layer stack drawn with TikZ.

1.5 Main Guarantees

[Joint-action consistency] Assuming loss-free messages and thresholds that force at least one leader per $t_{\max}$ steps, the policy vector $\pi = \langle \pi^1, \ldots, \pi^n \rangle$ is consistent at every step.

[Bounded cumulative loss] Let $\eta > 0$ and choose the silence window $t_{\max} = -\lambda^{-1} \ln(1 - \lambda\eta/\Delta\rho_0)$. Then $J(k) \geq J_{\text{opt}} - \eta$ for all k.

Detailed proofs live in Chapters 2 and 3.

Chapter 2

Spatial Allocation and Incremental Planning

This chapter expands Layers **3** and **4** of the stack (Fig. 1.1). Section 2.1 formalises the frontier–allocation problem and analyses two deterministic solvers: Centroidal Voronoi Tessellation (CVT) and a Round-Robin Auction. Section 2.2 develops the incremental path planner (D*-Lite) and proves that our re-planning policy preserves optimality under monotone obstacle discovery.

Unless stated otherwise the grid $\mathcal{G}$, notation and assumptions (A_1–A_3) are those of Chapter **??**.

2.1 Frontier Partition as a CVT Problem

2.1.1 Frontiers

[4–frontier set] At step k the *frontier set* is

$$F_k \;=\; \Big\{\, p \in \mathcal{G} \;\Big|\; \underbrace{B_i[p,0]=1}_{\text{free}} \wedge \exists\, q \sim p :\; B_i[q,:] = \big\langle \tfrac{1}{3}, \tfrac{1}{3}, \tfrac{1}{3} \big\rangle \Big\},$$

where $\sim$ denotes 4-neighbourhood.

Frontiers are global: once *one* robot discovers a free cell, everybody will know it after the next DSU gossip sweep (Prop. **??**). Therefore a common F_k is available to all robots at the instant allocation runs.

2.1.2 Energy Formulation

Let $C = \{c_1, \dots, c_n\} \subseteq \mathcal{G}$ denote generator points. The (discrete) *Voronoi region* of robot i is $\mathcal{V}_i(C) = \big\{p \in F_k \mid \|p - c_i\|_1 \le \|p - c_j\|_1,\ \forall j\big\}$. Given a weighting function $w : F_k \to R_{>0}$ (we use $w \equiv 1$) the CVT energy is

$$E(C) \;=\; \sum_{i=1}^{n} \sum_{p \in \mathcal{V}_i(C)} w(p)\, \|p - c_i\|_2^2. \tag{2.1}$$

Algorithm 1 Parallel Lloyd iteration on frontier set F_k

input: $C^{(0)} \leftarrow$ current robot positions $t = 0, 1, \ldots, t_{\max}$ (assign) $\forall p \in F_k$: find $i^\star = \arg\min_i \|p - c_i^{(t)}\|_1$ // GPU arg-min (update) $\forall i$ in parallel:

$$c_i^{(t+1)} = \frac{\sum_{p \in \mathcal{V}_i(C^{(t)})} w(p)\, p}{\sum_{p \in \mathcal{V}_i(C^{(t)})} w(p)} \text{ (if } \mathcal{V}_i = \text{re-seed uniformly)}$$

$E(C^{(t)}) - E(C^{(t+1)}) < \varepsilon$ **break return** clusters $\{\mathcal{V}_i(C^{(t+1)})\}$

2.1.3 Lloyd Iteration

[Monotone descent and termination] Algorithm 1 produces a sequence $\{C^{(t)}\}_t$ such that $E(C^{(t+1)}) \leq E(C^{(t)})$ for all t and terminates in at most $|F_k|$ iterations.

The discrete centroid step minimises (2.1) *for fixed assignments*; conversely the assignment step minimises the energy for fixed generators. Alternating two best-response moves cannot increase the objective, hence monotonicity. Because E is integer-valued under $w \equiv 1$ and bounded below by 0, strict descent happens finitely many times; after at most $|F_k|$ iterations the energy stagnates and the loop exits.

Complexity. GPU assignment is $O(|F_k|)$; centroid updates cost $O(|F_k|)$ as well. Hence the amortised per-iteration work is linear in the frontier size.

2.1.4 Greedy Round-Robin Auction

When F_k is *sparse* Lloyd sometimes collapses generators onto the same cell. We therefore provide a load-balanced fallback.

Algorithm 2 Round-Robin Auction (F_k arbitrary set, robots fixed)

remaining $\leftarrow F_k$; $\forall i : \mathcal{V}_i \leftarrow t \leftarrow 0$ remaining $\neq i \leftarrow (t \bmod n) + 1$ $p^\star \leftarrow \arg\min_{p \in \text{remaining}} \|p - q_i\|_1$ $\mathcal{V}_i \leftarrow \mathcal{V}_i \cup \{p^\star\}$; remaining $\leftarrow$ remaining $\setminus \{p^\star\}$ $t \leftarrow t + 1$ **return** $\{\mathcal{V}_i\}$

[Bounded imbalance] For any execution of Alg. 2, $\big|\, |\mathcal{V}_i| - |\mathcal{V}_j| \big| \leq 1$ for all i, j.

The assignment is exactly round-robin: cell t (0-based) is always given to robot $i = 1 + t \bmod n$. After $|F_k|$ rounds every robot has $\lfloor |F_k|/n \rfloor$ or $\lceil |F_k|/n \rceil$ cells.

2.2 Incremental Planning with D*-Lite

2.2.1 Notation

Robot i's known occupancy grid at step k induces a weighted digraph $G_k = (V, E, w_k)$ where

$$V = \mathcal{G}, \quad E = \big\{(u, v) \mid u \sim v,\ B_i[u, 0] = B_i[v, 0] = 1\big\}, \quad w_k \equiv 1.$$

The planner maintains

$$g : V \to R_{\geq 0}, \qquad rhs : V \to R_{\geq 0}, \qquad \mathcal{U} \subseteq V \times R^2 \text{ (priority queue)}.$$

2.2.2 Algorithm

Algorithm 3 Incremental **D*-Lite** (simplified)

$rhs(s_{\text{goal}}) \leftarrow 0$; $g \equiv \infty$ $\mathcal{U} \leftarrow \{(key(s_{\text{goal}}), s_{\text{goal}})\}$ $\mathcal{U}$.topKey $< key(s_{\text{start}}) \vee rhs(s_{\text{start}}) \neq g(s_{\text{start}})$ pop u with minimum key from $\mathcal{U}$ $g(u) > rhs(u)$ $g(u) \leftarrow rhs(u)$ $g(u) \leftarrow \infty$; UpdateVertex(u) each predecessor p of u UpdateVertex(p)

Key function and vertex update are exactly those in Koenig & Likhachev (2005); we omit them for space.

2.2.3 Correctness

[Optimality under monotone obstacle growth] Assume cells only switch $\mathsf{unknown} \to \mathsf{obs}$. Then Alg. 3 returns a least-cost path at every re-plan call and runs in $O(|E_k| \log |V|)$ amortised time.

[Idea] Monotone cost increases satisfy the key monotonicity condition (Koenig & Likhachev 2005, Thm. 1). The incremental repair touches only vertices whose shortest-path cost changed, and the binary-heap queue gives the stated complexity.

Path Re-use Policy. Robot i re-plans *only* when (i) it reaches the end of the current path or (ii) any edge on that path is observed as obstacle; otherwise it follows the previously computed prefix. This strategy preserves optimality while greatly reducing queue operations in dense maps.

2.3 Putting Allocation and Planning Together

Algorithm 4 End-to-End Target Selection and Motion for one robot

k mod `ALLOC_INTERVAL` $= 0$ run Alg. 1 (or 2) and receive target $t_i(k)$ $q_i(k) = t_i(k)$ **or** current path invalid run D*-Lite (Alg. 3) on G_k execute *next* edge on path

[Dead-end avoidance] Under Assumption (A**??**) Alg. 4 yields infinite progress: $q_i(k)$ visits a new frontier cell at least every `ALLOC_INTERVAL` steps.

Either the current target remains a frontier and will be reached (eventually triggering a new allocation), or the frontier assignment phase reseeds to a *different* frontier because the old one has been explored. Since $F_k \neq$ by assumption, progress continues indefinitely.

2.4 Summary of Guarantees

Combining Theorems 2.1.3, 2.2.3 and Lemma 2.1.4 with the global results of Chapter **??** yields:

[Layer-3 & 4 soundness]

[label=()]Frontiers are partitioned in $O(|F_k|)$ time with energy $E(C)$ non-increasing. Each robot holds $\lfloor |F_k|/n \rfloor$ or $\lceil |F_k|/n \rceil$ targets (auction fallback). The path to the current target is cost-optimal in G_k and is repaired in $O(|E_k| \log |V|)$ whenever obstacles are discovered.

Consequently the exploration loop in Alg. 4 meets the bounded-loss and joint-action guarantees derived in Chapter **??** while remaining *asymptotically linear* in the frontier size and near-optimal in motion cost.

Implementation Notes

- **GPU kernels.** Lloyd's assignment step is a single fused kernel ('argmin_1L_batched'); centroids are computed by a parallel segmented reduction.
- **Sparse maps.** G_k is stored in CSR format; only 4 edges per free cell.
- **Python & Numba.** All outer loops of Alg. 4 are Numba-JIT compiled for clarity without hurting frame rate.

Chapter 3

Adaptive Communication and Consistency

3.1 Trigger and DSU Merge

Metrics $\Delta_B, \Delta_M, \Delta_V$ trigger leadership when exceeding thresholds $(\vartheta_B, \vartheta_M, \vartheta_V)$. A leader contacts at most m neighbours within R_{comm}; disjoint-set FIND/UNION merges clusters.

[Lipschitz loss] For any cell p, $|\rho(B^{\text{old}}) - \rho(B^{\text{new}})| \le \|B^{\text{old}}(p,:) - B^{\text{new}}(p,:)\|_1$.

The KL term in (**??**) is 1-Lipschitz w.r.t. ℓ_1.

[DSU $\Rightarrow$ action consistency] Assume all gossip messages are delivered and the local selector $\pi(\cdot)$ is continuous. One gossip round makes $\pi^i = \pi^j$ for every pair in the same DSU cluster.

Cluster members share an identical belief $\widehat{B}$; continuity forces identical outputs.

3.2 Bounding Reward Degradation

Under exponential KL drift $\mathrm{KL}_k = \Delta\rho_0 e^{-\lambda k}\Delta t$ we integrate until the next compulsory sync:

$$D(t) = \int_0^t \Delta\rho_0 e^{-\lambda\tau}\, d\tau = \frac{\Delta\rho_0}{\lambda}\big(1 - e^{-\lambda t}\big).$$

Demand $D(t) \le \eta$; solving yields horizon

$$t_{\max} = -\lambda^{-1} \ln\Big(1 - \lambda\eta/\Delta\rho_0\Big).$$

[System-level guarantee] With leader polls at period $\le t_{\max}$ the exploration process is *simultaneously* [label=()]

joint-action consistent *and*

η-sub-optimal in cumulative reward.

Algorithm 5 ROBOT-STEP (executed in lock-step)

Sense local cells & update B_i by (**??**) trigger($\Delta_B, \Delta_M, \Delta_V$) become leader; gossip; merge DSU clusters reallocation timer **or** $q_i = t_i$ obtain $\mathcal{V}_i$ via CVT / AUCTION; choose t_i plan & move one step via D*-Lite (Alg. **??**)

3.3 Unified Robot Routine (Pseudo-code)

The four deterministic layers plus synchronous triggering guarantee the properties proved above.

Chapter 4

Belief Updation

4.1 Introduction

In this approach, we aim to reduce the inconsistency between beliefs of multiple non-cooperative robots operating in a common environment. The inconsistency is minimized by sharing partial beliefs iteratively using a modified Bayesian filter approach. Robots select their optimal actions based on the updated beliefs once sufficient convergence is achieved.

4.2 Problem Setting

Consider two agents, A1 and A2, operating in a multi-agent environment with three possible states: $S = \{s1, s2, s3\}$. Each agent can take one of two actions: $A1 = \{a1, a2\}$ for agent 1 and $A2 = \{b1, b2\}$ for agent 2. The utilities of each action in each state are as follows:

4.2.1 Initial Beliefs

- Agent A1: $b_1(s1) = 0.3, b_1(s2) = 0.4, b_1(s3) = 0.3$
- Agent A2: $b_2(s1) = 0.5, b_2(s2) = 0.3, b_2(s3) = 0.2$

4.2.2 Belief Sharing Parameter

A partial sharing parameter δ is introduced to control the extent of belief sharing between robots. Here, $\delta = 0.5$.

State	$u_1(a1, s)$	$u_1(a2, s)$	$u_2(b1, s)$	$u_2(b2, s)$
$s1$	3	2	1	4
$s2$	5	4	2	3
$s3$	2	3	3	1

Table 4.1: Utility Table for Agents A1 and A2

4.3 Bayesian Filter

The Bayesian filter is a recursive algorithm used for estimating the state of a system over time by combining prior knowledge with new observations. The general update formula for belief using the Bayesian filter is:

$$P(X_t|O_{1:t}) = \frac{P(O_t|X_t)P(X_t|O_{1:t-1})}{P(O_t|O_{1:t-1})}$$

Where:

- $P(X_t|O_{1:t})$: Updated belief after incorporating the observation.
- $P(O_t|X_t)$: Likelihood of the observation given the state.
- $P(X_t|O_{1:t-1})$: Prior belief.
- $P(O_t|O_{1:t-1})$: Normalization factor ensuring the total probability sums to 1.

4.4 Belief Update Process (Modified Bayesian Filter)

At each iteration, agents exchange partial beliefs. The updated belief for agent A1 considering information from agent A2 is calculated as:

$$b_1'(s) = \frac{\delta \cdot b_1(s) \cdot b_2(s)}{\sum_{s \in S} \delta \cdot b_1(s) \cdot b_2(s)}$$

Similarly, for agent A2:

$$b_2'(s) = \frac{\delta \cdot b_2(s) \cdot b_1(s)}{\sum_{s \in S} \delta \cdot b_2(s) \cdot b_1(s)}$$

This formulation ensures that the beliefs are updated in a probabilistically consistent manner and always sum to 1 after normalization.

4.5 Measuring Belief Convergence

The inconsistency between the beliefs of two agents is measured using KL Divergence:

$$D_{KL}(b_1' \parallel b_2') = \sum_{s \in S} b_1'(s) \log \left(\frac{b_1'(s)}{b_2'(s)} \right)$$

The belief update process is repeated until the KL divergence between the beliefs of the agents falls below a specified threshold, indicating convergence.

4.6 Utility Calculation and Action Selection

Once the beliefs converge (i.e., KL divergence is sufficiently low), the agents compute their utilities for each action based on the updated beliefs and select the optimal action.

For Agent 1:

$$U_1(a) = \sum_{s \in S} b'(s) u_1(a, s)$$

For Agent 2:

$$U_2(b) = \sum_{s \in S} b'(s) u_2(b, s)$$

The actions yielding the highest utility are chosen by the agents.

4.7 Summary of Steps

1. Initialize beliefs for all agents.
2. At each iteration, share partial beliefs using δ.
3. Update beliefs using the modified Bayesian filter formula.
4. Calculate KL Divergence to measure belief convergence.
5. Repeat steps 2-4 until KL divergence is below the threshold.
6. Compute utilities based on the converged beliefs.
7. Select the optimal actions for each agent.

This approach ensures that the non-cooperative agents can reduce belief inconsistency through probabilistic belief merging and then make optimal decisions based on the merged beliefs.

4.8 Code for Belief Update and Simulation

The following Python code was used for the belief update and simulation process, which includes the calculation of the KL divergence and belief convergence:

Algorithm 6 Belief Update and Simulation

Data: Initial beliefs b_1, b_2, sharing factor δ, iterations N
Result: Final beliefs b_1, b_2, KL divergence history
function NORMALIZE(b)
    return $b / \sum b$
end function
function KL-DIVERGENCE(b_1, b_2)
    $D \leftarrow 0$
    for each state i **do**
        $D \leftarrow D + b_1[i] \cdot \log(b_1[i]/b_2[i])$
    end for
    return D
end function
function UPDATE-BELIEF(b_1, b_2, δ)
    $b_{\text{new}} \leftarrow \delta \cdot b_1 \cdot b_2$
    return NORMALIZE(b_{new})
end function
$history\_A1 \leftarrow [b_1]$
$history\_A2 \leftarrow [b_2]$
$KL\_history \leftarrow [\,]$
for $i = 1$ to N **do**
    $b_1' \leftarrow$ UPDATE-BELIEF(b_1, b_2, δ)
    $b_2' \leftarrow$ UPDATE-BELIEF(b_2, b_1, δ)
    $KL \leftarrow$ KL-DIVERGENCE(b_1', b_2')
    Append KL to $KL\_history$
    $b_1 \leftarrow b_1'$
    $b_2 \leftarrow b_2'$
    Append b_1 to $history\_A1$
    Append b_2 to $history\_A2$
end for

Chapter 5

Results

Results

Simulation Report

```
 ===== Simulation Report =====
Elapsed :  3.01s Steps:  200
Coverage :  56.60% (5660/10000)
Victims hit :  28/50
Confirmed :  [(2, 42), (3, 14), (4, 30), (6, 8), (7, 41), (12, 68), (15, 40), (20,
39), (21, 50),
     (24, 83), (25, 76), (26, 90), (28, 82), (29, 80), (31, 38), (40, 40), (55, 14),
(56, 33),
     (61, 79), (63, 10), (67, 60), (68, 30), (69, 37), (73, 43), (73, 68), (83, 22),
(85, 34), (88, 57)]
Comms :  min=40, max=202, avg=138.00
Positions :  [(80, 3), (99, 34), (7, 11), (9, 1), (18, 45), (34, 52), (89, 10), (85,
42), (27, 32), (33, 32)]
============================
```

Visualization Results

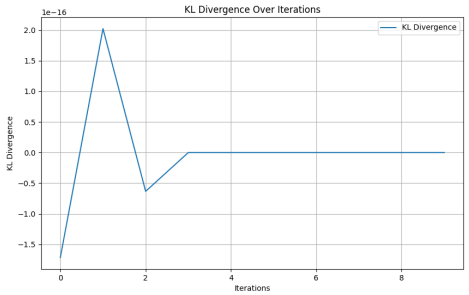

Figure 5.1: KL Divergence Over Iterations

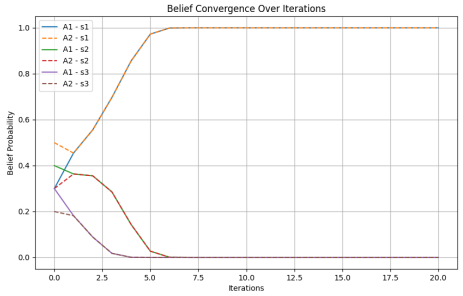

Figure 5.2: Belief Convergence for Agent A1

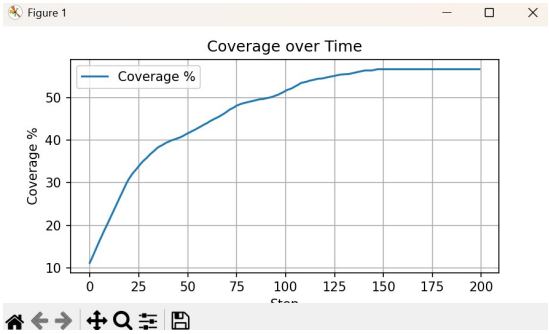

Figure 5.3: Belief Convergence for Agent A2

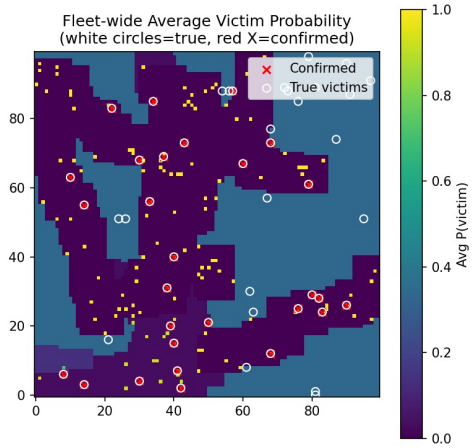

Figure 5.4: Final Belief Comparison Between A1 and A2

Bibliography

[1] Khen Elimelech and Vadim Indelman, "Simplified Decision Making in the Belief Space Using Belief Sparsification," *The International Journal of Robotics Research*, vol. 0, no. 0, pp. 1–27, 2022, doi: 10.1177/02783649221076381.

[2] Y Narahari, *Game Theory and Mechanism Design*, vol. 4, IISc Lecture Notes Series, World Scientific Publishing Co. Pte. Ltd., Singapore, 2014, ISBN 8902_9789814525046. Available online: `https://www.worldscientific.com/worldscibooks/10.1142/8902`.

[3] Moran Barenboim and Vadim Indelman, "Adaptive Information Belief Space Planning," in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI-22)*, [Location of the Conference], 2022, pp. [Page Numbers]. Technion Autonomous Systems Program, Technion - Israel Institute of Technology, Haifa 32000, Israel. Email: moranbar@campus.technion.ac.il, vadim.indelman@technion.ac.il.

[4] Tanmoy Kundu, Moshe Rafaeli, and Vadim Indelman, "Multi-Robot Communication-Aware Cooperative Belief Space Planning with Inconsistent Beliefs: An Action-Consistent Approach," *arXiv*, Mar. 9, 2024, arXiv:2403.05962v1.