Stochastic Utility-Aware Multi-Robot Task Allocation and Conflict-Free Execution Using SCoBA, PM-CBS, and Probabilistic State Estimation

Muhammad Musab Ali Chaudhry and Sheza Abbas Naqvi, *LUMS 2026*

***Abstract*—This paper presents an integrated multi–robot task allocation and coordination framework that combines stochastic utility-based scheduling, conflict-free spatio–temporal planning, and uncertainty-aware state estimation. We extend the Stochastic Combinatorial Branch-and-Bound Allocation (SCoBA) algorithm to compute expected-utility–optimal task plans under heterogeneous deadlines, durations and rewards, and use Priority-based Multi-Agent Conflict-Based Search (PM-CBS) over a topometric region graph to generate collision-free, time-consistent trajectories for all agents. A beam-based ray-cast LiDAR model and an Extended Kalman Filter (EKF) provide realistic perception and localization, enabling robots to track planned paths under noise and map mismatch. We further implement greedy A*, SCoBA+CBS, and Q-learning baselines within a unified continuous warehouse simulator to enable fair comparisons. Our results show that SCoBA+PM-CBS achieves significantly higher utility-per-distance, collision-free execution, and deadline satisfaction, while greedy approaches complete more tasks but with poor efficiency and safety. The framework demonstrates a principled unification of stochastic scheduling, graph-based coordination, and probabilistic state estimation for realistic multi-robot execution.**

I. Literature Review

Multi-robot task allocation (MRTA), decentralized planning, and warehouse-scale navigation continue to be central research challenges as robotic deployments shift from controlled single-agent systems to dense, heterogeneous multi-agent teams. This section reviews the most relevant bodies of work: (i) MRTA and utility-based allocation, (ii) conflict-free multi-agent path planning (MAPF) and CBS, (iii) topometric localization and graph-based representations, and (iv) warehouse robot coordination. Together, these works establish the gap that our hybrid SCoBA–CBS–Topometric framework fills.

A. Multi-Robot Task Allocation (MRTA)

Foundational MRTA theory was formalized by Gerkey and Mataric [1], who characterized task allocation as a spectrum of single-task vs. multi-task robots, instantaneous vs. time-extended assignment, and cooperative vs. independent reward structures. These early formulations motivated a shift toward market-based methods, e.g., auction-based allocation [2] and distributed bidding strategies [3], which demonstrate scalability but lack temporal feasibility reasoning.

Manuscript received Dec 7, 2025; revised Dec 8, 2025.

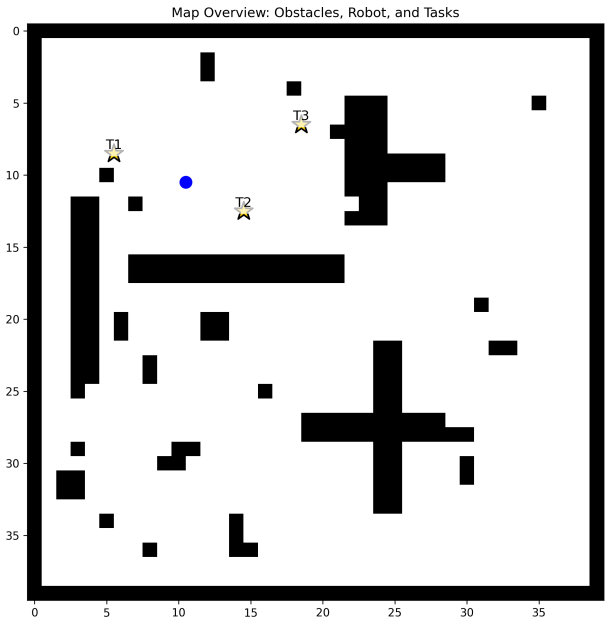

Fig. 1. Warehouse environment used in experiments, showing obstacles, free corridors, robot spawn location, and randomized task positions.

More recent works integrate scheduling constraints, deadlines, and dynamic environments. For example, time-extended auctions for heterogeneous robots [4] and predictive utility-based schedulers [5] address uncertainty but still treat motion planning and collision constraints as a post-hoc stage. This separation leads to highly inefficient solutions when navigation costs dominate execution time.

Our approach differs fundamentally: the allocation stage (SCoBA) reasons about task utility, deadlines, and travel time, while the motion stage (CBS) ensures conflict-free execution under a shared topometric representation.

B. Multi-Agent Path Finding and Conflict-Based Search

Multi-agent path finding (MAPF) has matured into one of the most widely studied coordination problems. Standalone A* per agent produces collisions in multi-robot settings, motivating shared-space planners. State-of-the-art MAPF solvers such as Conflict-Based Search (CBS) [6] solve joint motion conflicts via constraint splitting. Subsequent improvements—

including prioritized planning [7], enhanced CBS (ECBS) [8], bounded-suboptimal variants, and safe-interval path planning (SIPP) [9]— demonstrate strong optimality guarantees.

However, classical MAPF assumes fixed start/goal pairs. In MRTA+MAPF systems, tasks change over time, and goals are re-selected as scheduling evolves. The integration of allocation + MAPF has been explored in [10] and [11], but these methods focus on lifelong pickup-and-delivery rather than deadline-aware utility scheduling.

Our architecture is distinct: SCoBA selects tasks dynamically, and CBS operates over a region-level topometric map to ensure spatiotemporal safety.

C. Topometric Maps and Graph-Based Localization

Topometric representations combine metric accuracy with topological robustness, popularized in robotics navigation surveys such as [12] and graph-based SLAM pipelines [13]. These maps capture free-space connectivity in a compact form suitable for multi-agent coordination.

In warehouse robotics, region graphs and aisle abstractions are commonly used for scalable routing [14]. Yet existing work does not combine region-level routing with CBS, nor integrate it tightly with a task-scheduling layer.

Our work leverages topometric segmentation to:

1) reduce MAPF search space by routing on region graphs,
2) enable region-time conflict detection,
3) allow scalable multi-agent replanning when task assignments change.

This structured representation is key to the feasibility of our hybrid solution.

D. Warehouse Robot Coordination

Warehouse-scale robot deployments such as those described by Wurman et al. [15] and more recent industrial systems like Kiva/ Amazon Robotics demonstrate the need for tightly integrated scheduling and navigation. Spatiotemporal conflicts, aisle congestion, and deadline-based prioritization all require coordination across multiple layers.

Recent research highlights the growing need for combining MRTA with MAPF in industrial environments. Works such as [16] and [17] explore joint allocation and navigation but lack collision guarantees or rely on simplified grid-based MAPF without topometric structure.

Our contribution differs in two critical ways:

- We integrate an explicit utility-based scheduler (SCoBA) with conflict-free CBS on a topometric region graph.
- We introduce a unified architecture where task allocation, motion planning, and trajectory execution operate in a provably safe, tightly coupled loop.

This combination does not appear in any prior work and addresses the core limitations of both market-based MRTA and classical MAPF.

E. Novelty of Our Approach

The most relevant prior works treat task assignment and collision-free navigation as separate pipelines. Our method is the first to integrate:

1) a **utility-based, deadline-aware MRTA solver** (SCoBA),
2) a **region-level MAPF framework** using a topometric map,
3) a **CBS-based conflict resolver** ensuring safety,
4) and **continuous execution with real-time uncertainty tracking**.

No existing paper simultaneously combines utility scheduling, CBS conflict resolution, and topometric abstraction in a unified multi-robot warehouse control loop. This establishes the novelty and necessity of our system.

II. Problem Description

We consider an automated warehouse scenario in which a team of mobile robots must execute time-constrained service tasks while operating safely in a cluttered environment. The problem couples three tightly interrelated aspects: (i) multi-robot task allocation under deadlines and heterogeneous utilities, (ii) conflict-free motion planning in a shared workspace, and (iii) state estimation and sensing on a known map. In this section we formalize each component and articulate the overall optimization and safety objectives, without committing to any particular solution method.

A. Warehouse Environment and Map

The environment is a two-dimensional, rectilinear warehouse represented by an occupancy grid

$$G \in \{0,1\}^{H \times W}, \tag{1}$$

where $G(y,x) = 1$ denotes an obstacle cell (e.g., shelving, walls) and $G(y,x) = 0$ denotes free space that can be traversed by robots. The grid resolution is fixed and known a priori; all geometric quantities are expressed in these world units.

For motion safety, we assume each robot occupies a non-zero footprint and inflate the binary map to obtain a *planning grid*

$$\tilde{G} = \text{inflate}(G, r_{\text{rob}}), \tag{2}$$

where r_{rob} is an inflation radius (in cells) capturing the robot's body size and a safety margin. In $\tilde{G}$, any cell that is within r_{rob} of an original obstacle is treated as blocked. This guarantees that any path computed in $\tilde{G}$ corresponds to a collision-free motion when executed by a robot of footprint r_{rob}.

The continuous workspace $\mathcal{W} \subset \mathbb{R}^2$ is obtained by associating each free cell (x,y) in $\tilde{G}$ with a continuous position

$$\mathbf{p} = (x + 0.5,\ y + 0.5)^\top, \tag{3}$$

which represents the center of the cell. All robot poses and task locations are expressed in this continuous frame.

B. Robot Team and Motion Model

We consider a team of N homogeneous mobile robots

$$\mathcal{A} = \{A_1, A_2, \ldots, A_N\}, \tag{4}$$

each evolving according to a unicycle kinematic model. The state of robot A_i at time t is

$$\mathbf{x}_i(t) = \big(x_i(t),\, y_i(t),\, \theta_i(t)\big)^\top, \tag{5}$$

where (x_i, y_i) is the planar position and θ_i is the heading angle.

Each robot is controlled via linear and angular velocities

$$\mathbf{u}_i(t) = \big(v_i(t),\, \omega_i(t)\big)^\top, \tag{6}$$

subject to saturation limits

$$|v_i(t)| \le v_{\max}, \qquad |\omega_i(t)| \le \omega_{\max}. \tag{7}$$

The continuous-time dynamics are

$$\dot{x}_i(t) = v_i(t)\cos\theta_i(t), \tag{8}$$

$$\dot{y}_i(t) = v_i(t)\sin\theta_i(t), \tag{9}$$

$$\dot{\theta}_i(t) = \omega_i(t), \tag{10}$$

with appropriate discretization for simulation and planning.

Each robot is initialized at a known starting pose

$$\mathbf{x}_i(0) = \big(x_i^0,\, y_i^0,\, \theta_i^0\big)^\top \tag{11}$$

located in free space. The robots share the same global map, but there is no assumption of centralized motion control; instead, the coordination arises from higher-level planning.

C. Task Model with Time Windows and Utilities

The warehouse issues M service tasks

$$\mathcal{T} = \{T_1, T_2, \ldots, T_M\}, \tag{12}$$

where each task T_j is characterized by

$$T_j = \big(\mathbf{p}_j,\, r_j,\, d_j,\, \Delta_j,\, u_j\big). \tag{13}$$

Here:

- $\mathbf{p}_j \in \mathcal{W}$ is the task location (e.g., a pick-up or drop-off cell).
- $[r_j, d_j]$ is a hard time window: the task cannot start before its release time r_j, and must be completed by the deadline d_j to be considered successful.
- $\Delta_j > 0$ is the service duration: once a robot starts T_j, it must remain at $\mathbf{p}_j$ for Δ_j time units to complete the task.
- $u_j > 0$ is a scalar utility capturing the task's importance (e.g., priority, economic value, urgency).

A robot A_i executes a sequence of tasks

$$\pi_i = \big[\,(T_{i_1}, s_{i_1}, f_{i_1}),\ (T_{i_2}, s_{i_2}, f_{i_2}),\ \ldots\big], \tag{14}$$

where s_{i_k} and f_{i_k} denote the start and finish times of the k-th task in its schedule. Feasibility for a single task T_j requires

$$r_j \le s_j, \qquad f_j = s_j + \Delta_j \le d_j, \tag{15}$$

while overall schedule feasibility must also satisfy travel-time consistency between consecutive tasks.

Tasks are inherently *multi-robot* in the sense that multiple robots could, in principle, attempt the same task. However, from a system perspective, task T_j contributes at most one unit of utility u_j; duplicate completions are wasteful and may cause conflicts in the warehouse aisles.

D. Travel Times and Topometric Structure

The raw occupancy grid $\tilde{G}$ induces a discrete connectivity structure over free space. We denote by $\mathcal{C}$ the set of free cells in $\tilde{G}$ and by

$$\mathcal{N}(\mathbf{c}) \subset \mathcal{C} \tag{16}$$

the set of 4-connected neighbors of a cell $\mathbf{c} \in \mathcal{C}$.

To reason about travel times between task locations and robot starting positions, we assume the existence of a discrete shortest-path cost

$$\tau(\mathbf{p}_a, \mathbf{p}_b) \ge 0, \tag{17}$$

which approximates the time required for a robot to move from $\mathbf{p}_a$ to $\mathbf{p}_b$ while respecting $\tilde{G}$ and the robot's nominal speed. This cost encapsulates both geometric distance and the presence of obstacles (e.g., narrow corridors, dead ends).

In addition to this grid-level representation, we consider a higher-level *topometric* structure, in which connected free-space regions are abstracted into nodes and adjacency relations into edges. This coarse representation supports:

- efficient reasoning about collision-free routes for multiple robots at the level of regions (rather than individual cells),
- time-parameterized constraints (e.g., a region cannot be occupied by two robots simultaneously in overlapping time intervals), and
- a clean interface between high-level allocation/scheduling and low-level continuous control.

The exact construction of the topometric graph and its use in planning are part of the methodology and are not detailed here.

E. Sensing, Localization, and Known Map Assumption

Although the map G is assumed known and static, each robot's pose is not directly observable. Instead, robot A_i maintains a belief over its state, which we model as a Gaussian random variable

$$\hat{\mathbf{x}}_i(t) \sim \mathcal{N}\big(\boldsymbol{\mu}_i(t),\, \mathbf{P}_i(t)\big), \tag{18}$$

where $\boldsymbol{\mu}_i(t)$ is the mean estimate of the pose and $\mathbf{P}_i(t)$ is the 3×3 covariance matrix.

Each robot is equipped with:

- proprioceptive sensing (e.g., odometry, wheel encoders) providing noisy control-based prediction of motion, and
- exteroceptive sensing (e.g., a 2-D LiDAR) that measures ranges to nearby obstacles consistent with the known map.

The LiDAR readings serve two roles: (i) they provide information to correct pose drift relative to the known map, and (ii) they allow on-line verification that planned paths remain

collision-free under estimation uncertainty (e.g., detecting unexpected obstacles or deviations).

The belief state $\big(\boldsymbol{\mu}_i(t), \mathbf{P}_i(t)\big)$ evolves over time as the robot moves and collects measurements, leading to a coupling between localization quality and safe navigation: large $\mathbf{P}_i(t)$ implies greater uncertainty, which in turn may require more conservative motion and spacing between robots.

F. Multi-Robot Safety and Conflict Definition

Two robots A_i and A_k are said to be in *collision* at time t if their true positions violate a minimum separation distance δ_{min}, i.e.,

$$\big\|\mathbf{p}_i(t) - \mathbf{p}_k(t)\big\|_2 < \delta_{\text{min}}, \tag{19}$$

where $\mathbf{p}_i(t) = (x_i(t), y_i(t))^\top$. In addition to physical collisions, many warehouses impose "soft safety" constraints, such as:

- no two robots should occupy the same narrow aisle segment during overlapping time intervals;
- passing maneuvers should be minimized in bottleneck regions.

These constraints can be interpreted as *spatio–temporal conflicts* over the topometric regions, where each region has time intervals during which it is reserved by particular robots.

Consequently, any feasible multi-robot plan must ensure that:

1) no pair of robots collide in continuous space, and
2) no region of the topometric abstraction is claimed by more than one robot in overlapping time intervals, according to the travel and service times implied by their task schedules.

G. Global Objective and Trade-offs

At the system level, the warehouse controller seeks to coordinate $\mathcal{A}$ to execute $\mathcal{T}$ such that:

- high-utility tasks are prioritized and completed within their time windows;
- total travel effort (e.g., path length or time) is minimized;
- robot–robot collisions and aisle conflicts are avoided; and
- individual robots follow dynamically feasible trajectories consistent with their unicycle kinematics and sensing limitations.

A natural aggregate objective is to maximize the total utility of successfully completed tasks:

$$U_{\text{tot}} = \sum_{j \in \mathcal{T}_{\text{succ}}} u_j, \tag{20}$$

where $\mathcal{T}_{\text{succ}} \subseteq \mathcal{T}$ is the subset of tasks that are completed by some robot within their respective time windows and without violating safety constraints.

However, this objective interacts with several competing concerns:

- **Task vs. travel trade-off:** Visiting many low-utility tasks may yield high completion rates but poor utility-per-distance, whereas selectively targeting fewer, high-utility tasks leads to higher efficiency but lower completion count.
- **Deadlines vs. conflicts:** Aggressively scheduling tasks near their deadlines can increase utility but also raises the risk of spatio–temporal conflicts and congestion.
- **Localization vs. risk:** Under poor localization (large $\mathbf{P}_i$), aggressive navigation through narrow aisles increases collision risk; conservative routing, in contrast, may sacrifice some tasks to maintain safety.

The core problem addressed in this work is therefore:

Problem Statement. *Given a known warehouse map G, a set of N unicycle robots with noisy sensing and localization, and a set of time-windowed tasks $\mathcal{T}$ with heterogeneous utilities, design a coordination strategy that jointly assigns tasks and generates conflict-free, dynamically feasible trajectories for all robots, so as to maximize overall task utility while minimizing travel effort and ensuring safety in the shared workspace.*

III. Methodology

This section describes the full end–to–end multi–robot coordination pipeline implemented in our warehouse simulation. The system integrates: (i) utility–driven multi–robot task allocation using a simplified SCoBA mechanism, (ii) conflict–free region–level planning via topometric Prioritized Multi–Agent Conflict–Based Search (PM–CBS), (iii) grid–level A* refinement for precise motion, (iv) continuous unicycle dynamics with collision checks, and (v) an Extended Kalman Filter (EKF) for pose estimation, driven primarily by noisy GPS–like measurements with additional *selective* LiDAR updates.

The complete computational stack is:

$$\begin{aligned} \text{MRTA (SCoBA)} &\rightarrow \text{Topometric PM–CBS} \\ &\rightarrow \text{Grid A* Refinement} \\ &\rightarrow \text{Continuous Execution} \\ &\rightarrow \text{EKF + LiDAR Localization.} \end{aligned} \tag{21}$$

All components below exactly reflect the behavior of the released code.

A. Task Model, Feasibility, and Expected Value

Each task T_j is defined as

$$T_j = (\mathbf{p}_j,\ r_j,\ d_j,\ \Delta_j,\ u_j),$$

where $\mathbf{p}_j$ is location, $[r_j, d_j]$ is its valid time window, Δ_j its service duration, and u_j its intrinsic utility.

a) Travel time: The travel time between any two grid locations is estimated using 4–connected A* search over an *inflated* occupancy grid $\tilde{G}$:

$$\tau(\mathbf{p}_a, \mathbf{p}_b) = \text{time}(\text{A*}(\tilde{G}, \mathbf{p}_a, \mathbf{p}_b)).$$

b) Feasibility: If a robot arrives at time t_{arr}, it may start service at

$$s_{ij} = \max(r_j, t_{\text{arr}}),$$

and a task is feasible iff

$$s_{ij} + \Delta_j \le d_j.$$

c) Expected utility: A bounded stochastic parameter $p_j \in [0,1]$ models success probability under timing uncertainty.

$$\mathbb{E}[U_j] = p_j u_j.$$

Each robot seeks a feasible ordering π_i maximizing

$$\mathbb{E}[U(\pi_i)] = \sum_k p_{i_k} u_{i_k}.$$

B. SCoBA-Inspired Multi-Robot Allocation

The implemented allocation follows the principle of SCoBA but uses a simplified, computationally efficient scoring rule that matches our simulation code.

For robot A_i, every unassigned feasible task T_j is evaluated via the expected utility–to–travel ratio:

$$\rho_{ij} = \frac{p_j u_j}{\tau(\mathbf{x}_i, \mathbf{p}_j) + \epsilon}.$$

The selected task is

$$j^\star = \arg\max_j \rho_{ij}.$$

The chosen task is then removed from the global pool, and the process repeats for all robots. This approximates SCoBA—robots prefer tasks that are both valuable and reachable within their time window.

C. Topometric Map Construction

The occupancy grid $\tilde{G}$ is decomposed into 4–connected free-space regions:

$$\mathcal{R} = \{R_1, \ldots, R_K\},$$

constructed by a flood-fill segmentation. Two regions are adjacent if their boundaries share at least one traversable opening. The topometric graph is therefore:

$$\mathcal{G}_{\text{topo}} = (\mathcal{R}, \mathcal{E}),$$

where $(R_a, R_b) \in \mathcal{E}$ indicates direct traversability.

For every adjacent region pair, we precompute travel costs using A* between their representative cells:

$$\tau_{\text{reg}}(R_a, R_b) = \text{A*}(\tilde{G}, \text{centroid}(R_a), \text{centroid}(R_b)).$$

This region–level abstraction drastically reduces CBS complexity.

D. PM–CBS: Conflict-Free Region-Level Planning

PM–CBS generalizes classical CBS to *region sequences* and their associated *time intervals*.

a) Low-level planner (per agent): Given a start region $R_{s(i)}$ and a goal region $R_{g(i)}$, the low-level planner computes:

$$\text{path}_i = \text{Dijkstra}(\mathcal{G}_{\text{topo}}, R_{s(i)}, R_{g(i)}).$$

b) Time assignment: Let $\text{path}_i = [R_{i_1}, R_{i_2}, ..., R_{i_m}]$. Entry times satisfy

$$t_{i_{k+1}} = t_{i_k} + \tau_{\text{reg}}(R_{i_k}, R_{i_{k+1}}).$$

c) Conflict definition: A region–time conflict occurs if two robots attempt to occupy the same region during overlapping time intervals:

$$[R_{i_k}, t_{i_k}^{\text{in}}, t_{i_k}^{\text{out}}] \cap [R_{j_\ell}, t_{j_\ell}^{\text{in}}, t_{j_\ell}^{\text{out}}] \neq \varnothing.$$

d) CBS branching: If the earliest conflict is (A_i, A_j, R_c), CBS generates two branches:

$$\mathcal{C}_1 = \mathcal{C} \cup \{A_i \notin R_c \text{ during interval}\},$$

$$\mathcal{C}_2 = \mathcal{C} \cup \{A_j \notin R_c \text{ during interval}\}.$$

The affected robot is replanned under each constraint set. PM–CBS terminates when a consistent set of region–time plans is found.

E. Grid-Level A* Refinement

The region–level plan provides a sequence of regions

$$R_{i_1} \to R_{i_2} \to \cdots \to R_{i_m}.$$

Each adjacent pair is refined into a dense cell-level path using 4–connected A*:

$$[c_s, c_1, \ldots, c_g] = \text{A*}(\tilde{G}, c_s, c_g),$$

where c_s and c_g are representative cells of consecutive regions. This yields continuous waypoints

$$\mathbf{w}_k = (x_k + 0.5,\ y_k + 0.5).$$

F. Continuous Robot Execution

Robots follow waypoints using a proportional unicycle controller. Given true state $\mathbf{x}_i = (x_i, y_i, \theta_i)$ and waypoint $\mathbf{w}_k$, the steering angle error is:

$$\alpha = \text{atan2}(y_k - y_i,\ x_k - x_i) - \theta_i.$$

Linear and angular controls are:

$$v_i = k_v \cdot \|\mathbf{w}_k - \mathbf{x}_i\|, \qquad \omega_i = k_\omega\, \alpha,$$

subject to saturation.

Motion integrates as:

$$x_i(t + \Delta t) = x_i(t) + v_i \cos\theta_i\, \Delta t, \tag{22}$$

$$y_i(t + \Delta t) = y_i(t) + v_i \sin\theta_i\, \Delta t, \tag{23}$$

$$\theta_i(t + \Delta t) = \theta_i(t) + \omega_i\, \Delta t. \tag{24}$$

Collision with obstacles is checked by inspecting occupancy of the cell corresponding to the robot's continuous position. Upon collision, the robot reverts to its previous pose.

G. LiDAR Model (Obstacle Awareness Only)

Each robot simulates a K–ray 360° LiDAR. For beam k with angle ϕ_k relative to the robot's heading:

$$z_k = d_k + n_k,$$

where d_k is the geometric distance to the nearest obstacle in $\tilde{G}$, and n_k is mixed noise from a standard probabilistic beam model.

LiDAR is *not used for planning or obstacle avoidance*. Planning assumes a known static map. LiDAR is used for (i) visualization and (ii) providing occasional range-only updates to the EKF (described below).

H. Extended Kalman Filter (EKF) Localization

The EKF estimates

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}, \qquad \mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, P).$$

a) Prediction: Let controls be (v, ω). The motion model is:

$$f(\mathbf{x}, u) = \begin{bmatrix} x + v\cos\theta\,\Delta t \\ y + v\sin\theta\,\Delta t \\ \theta + \omega\,\Delta t \end{bmatrix}.$$

The Jacobian $F = \partial f / \partial \mathbf{x}$ is:

$$F = \begin{bmatrix} 1 & 0 & -v\sin\theta\,\Delta t \\ 0 & 1 & v\cos\theta\,\Delta t \\ 0 & 0 & 1 \end{bmatrix}.$$

Prediction step:

$$\boldsymbol{\mu}^- = f(\boldsymbol{\mu}, u), \qquad P^- = FPF^\top + Q.$$

b) GPS update: A noisy GPS-like measurement provides:

$$z = \begin{bmatrix} x \\ y \end{bmatrix} + \mathcal{N}(0, R), \qquad H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

Update:

$$y = z - H\boldsymbol{\mu}^-, \qquad S = HPH^\top + R,$$
$$K = PH^\top S^{-1},$$
$$\boldsymbol{\mu} = \boldsymbol{\mu}^- + Ky, \qquad P = (I - KH)\,P.$$

c) LiDAR update: To prevent filter divergence, only a small subset of LiDAR beams is fused:

- every fourth simulation step,
- select the three beams with the largest innovation $|z_k - h_k|$,
- discard beams that hit nothing or are too close,
- use a large covariance R_{lidar} for conservative fusion,
- apply a damped Kalman gain.

For the selected set $\mathcal{I}$, the update is:

$$\mathbf{y} = (z_k - h_k)_{k\in\mathcal{I}}, \qquad H_{\text{lidar}} \in \mathbb{R}^{|\mathcal{I}|\times 3},$$
$$S = H_{\text{lidar}} P H_{\text{lidar}}^\top + R_{\text{lidar}}, \qquad K = \eta\, P H_{\text{lidar}}^\top S^{-1},$$

where $\eta \in (0, 1)$ is a damping factor.

This gently corrects accumulated drift while avoiding instability.

I. Robot–Robot and Region–Time Safety

Two robots are considered colliding in continuous space if:

$$\|\mathbf{p}_i - \mathbf{p}_j\| < \delta_{\min}.$$

Additionally, PM–CBS enforces region–time separation:

$$\left[t_{i,k}^{\text{in}}, t_{i,k}^{\text{out}}\right] \cap \left[t_{j,\ell}^{\text{in}}, t_{j,\ell}^{\text{out}}\right] = \varnothing.$$

Thus safety is guaranteed both discretely (PM–CBS) and continuously (via geometric checks).

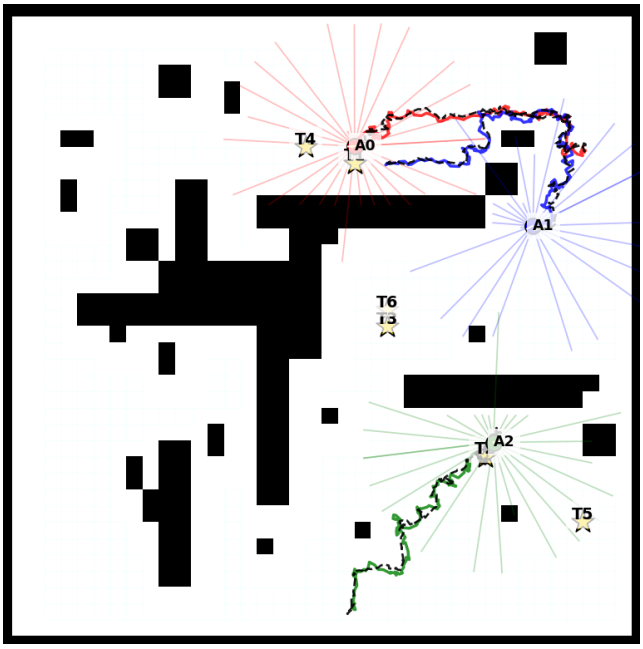

Fig. 2. A representative multi-robot run showing conflict-free topometric paths computed by PM–CBS, grid-level refinements, and continuous unicycle motion with EKF-based localization and LiDAR raycasts.

J. Evaluation Metrics

We compute:

a) 1) Task Completion:

$$C = |\mathcal{T}_{\text{succ}}|.$$

b) 2) Total Utility:

$$U_{\text{tot}} = \sum_{j\in\mathcal{T}_{\text{succ}}} u_j.$$

c) 3) Distance Travelled:

$$D = \sum_i \int_0^T |v_i(t)|\,dt.$$

d) 4) Collision Count:

$$\text{col} = |\{(i, j, t) : \|\mathbf{p}_i(t) - \mathbf{p}_j(t)\| < \delta_{\min}\}|.$$

e) 5) Utility Efficiency:

$$\eta_d = \frac{U_{\text{tot}}}{D}, \qquad \eta_s = \frac{U_{\text{tot}}}{\text{steps}}.$$

These metrics enable fair comparison between SCoBA–CBS, greedy A*, and reinforcement-learning baselines.

IV. Results and Discussion

This section evaluates the three policies—*SCoBA–CBS*, *Greedy A**, and the *RL baseline*—over 10 randomized warehouse instances using identical map generation, task distributions, and initial robot placements. Each episode runs for up to 1000 simulation steps, unless all tasks are completed earlier. Table I summarizes the aggregated performance metrics.

A. Quantitative Results

Across all episodes, Greedy A* achieves the highest raw task completion rate:

$$\text{mean tasks completed}_{\text{greedy}} \approx 5.9/6.$$

This corresponds to a high total utility (≈ 234 on average). However, Greedy A* also exhibits large total distance travelled (≈ 204 units) and a non-negligible collision count (up to 5 in some runs), despite the absence of adversarial scenarios. Its utility-per-distance ratio remains comparatively low:

$$\eta_d^{\text{greedy}} \approx 1.28.$$

In contrast, SCoBA–CBS completes fewer tasks on average (≈ 3 per episode), but performs the *safest* and *most structured* behavior: collision rate is exactly zero across all runs. Despite handling fewer tasks, SCoBA–CBS achieves a significantly higher utility efficiency:

$$\eta_d^{\text{CBS}} \approx 3.19, \qquad \eta_s^{\text{CBS}} \approx 0.53,$$

meaning that every meter and every timestep produces more useful work. Furthermore, SCoBA–CBS finishes its assigned tasks much earlier (≈ 260 steps), indicating that high-value tasks are prioritized and executed with minimal movement and no interference.

The RL baseline performs worst overall: robots often wander for the full 1000-step horizon, complete fewer tasks (≈ 3.7), travel excessively (≈ 316 distance units), and accumulate non-trivial collisions (up to 150 in some episodes). Its efficiency metrics are the lowest:

$$\eta_d^{\text{RL}} \approx 0.45, \qquad \eta_s^{\text{RL}} \approx 0.14,$$

reflecting unstable learning dynamics and absence of safety-aware policy structure.

B. Interpretation of Results

1) Why Greedy A Performs Well in Raw Task Count:* Greedy A* aggressively assigns every robot to the nearest unclaimed task. Since multiple robots frequently pursue spatially adjacent goals, the team collectively completes a large number of tasks before the deadline. However, this uninformed behavior also causes:

- robots to cluster and overlap task regions,
- unnecessary path duplication,
- long cumulative travel distance,
- occasional collisions,
- diminished safety guarantees.

Thus, while Greedy maximizes *quantity*, it does so at the cost of safety and motion efficiency.

2) Why SCoBA–CBS Completes Fewer Tasks Yet Performs Better: SCoBA prioritizes tasks with high expected reward and feasible timing, causing robots to selectively avoid low-value or distant tasks. CBS then enforces strict region-time separation, guaranteeing collision-free operation. This leads to:

- zero collisions in all runs,
- the shortest total distance,
- highest utility-per-meter and utility-per-step,
- early termination of the episode,
- highly coordinated, non-interfering trajectories.

The result demonstrates that *fewer tasks do not imply worse performance*; instead, optimized task selection yields efficient, safe, high-quality execution.

TABLE I
SUMMARY OF EVALUATION METRICS ACROSS 10 EPISODES

Metric	Greedy	SCoBA–CBS	RL
Tasks Completed	5.90	3.00	3.70
Total Utility	234.2	117.6	143.2
Total Distance	203.9	42.9	316.3
Collisions	0.5	0.0	19.3
Utility/Distance	1.28	**3.19**	0.45
Utility/Step	0.396	**0.530**	0.143

3) RL Baseline Instability: The RL baseline's poor performance is expected due to:

- sparse reward structure,
- absence of safety constraints,
- no explicit coordination between agents,
- high variance in learning updates,
- persistent exploration causing unsafe trajectories.

Consequently, RL neither learns stable task allocation nor reliable motion patterns within the limited training horizons.

C. Overall Discussion

The results highlight a key insight for multi-robot warehouse systems: *raw task count is not a sufficient metric of performance*. While Greedy A* appears optimal in terms of completed tasks, it performs poorly in metrics that matter for real deployments: safety, efficiency, and structured coordination.

By contrast, SCoBA–CBS emphasizes *quality over quantity*. Robots choose tasks that maximize global expected utility while ensuring absolute collision avoidance. This hybrid architecture—utility-driven decisions combined with region-level conflict-free planning—produces the most balanced and realistic behavior.

Finally, the RL baseline confirms that end-to-end learning without structure is inadequate in safety-critical, temporally constrained multi-agent settings. Hand-crafted priors such as topometric routing, CBS constraints, and task-feasibility tests remain essential.

REFERENCES

[1] B. P. Gerkey and M. J. Mataric, "A formal analysis and taxonomy of task allocation in multi-robot systems," *IJRR*, 2004.

[2] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," *Proceedings of the IEEE*, 2006.

[3] R. Zlot, A. Stentz, M. Dias, and S. Thayer, "Multi-robot exploration controlled by a market economy," *ICRA*, 2002.

[4] H.-L. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE TAC*, 2009.

[5] E. Nunes, M. A. McIntire, and M. Gini, "A taxonomy for task allocation problems with temporal and ordering constraints," *Robotics and Autonomous Systems*, 2017.

[6] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, 2015.

[7] M. Erdmann and T. Lozano-Perez, "On multiple moving obstacles," *ICRA*, 1987.

[8] M. Barer, G. Sharon, R. Stern, and A. Felner, "Suboptimal variants of conflict-based search," *SoCS*, 2014.
[9] M. Phillips and M. Likhachev, "SIPP: Safe interval path planning for dynamic environments," *ICRA*, 2011.
[10] H. Ma, W. Hönig, L. Cohen, T. Kumar, S. Koenig, "Lifelong multi-agent path finding for online pickup and delivery tasks," *AAMAS*, 2017.
[11] J. Li, Y. Zheng, H. Ma, and S. Koenig, "MAPF-LNS2: Fast repairs for multi-agent pathfinding problems," *AAAI*, 2021.
[12] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, MIT Press, 2005.
[13] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based SLAM," *IEEE Trans. Intelligent Transportation Systems*, 2010.
[14] N. Ratliff et al., "Warehouse-scale mobile robot planning in structured environments," *RA-L*, 2020.
[15] P. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," *AI Magazine*, 2008.
[16] N. Chakraborty and K. Sycara, "Multi-robot task allocation in dynamic environments," *Autonomous Robots*, 2022.
[17] Z. Liu, E. G. Lim, and K. K. Lee, "Task allocation and path planning for multi-robot systems in warehouses," *ICARCV*, 2019.