Distributed Multi-Robot Exploration Missions: Unified approach using Gaussian Belief Propagation

Master thesis

Sander Boers

Delft University of Technology

TU Delft

TNO

Distributed Multi-Robot Exploration Missions: Unified approach using Gaussian Belief Propagation

by

Sander Boers

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on October 7, 2024, at 13:00.

Faculty:	Faculty of Mechanical Engineering (ME)
Department	Department of Cognitive Robotics
Project duration:	March, 2024 — September, 2024
Thesis committee:	Dr. Ir. Joris Sijs, Dr. Ir. Laura Ferranti, Ir. Wouter Meijer (TNO)

TU Delft TNO

Copyright © S.H. Boers, 2024
All rights reserved.

Preface

This master's thesis, titled "Distributed Multi-Robot Exploration Missions: Unified approach using Gaussian Belief Propagation," is authored by Sander Boers, a student at Delft University of Technology, enrolled in the Robotics Master's program. This research work was conducted from March to September, under the supervision of Joris Sijs, Wouter Meijer and Jelle van Mil, in collaboration with TNO.

The focus of this thesis is on enhancing multi-robot systems using Gaussian Belief Propagation. This research aims to address critical challenges in robot navigation, task allocation, and collision avoidance, as one unified framework.

I would like to express my gratitude to my supervisors for their invaluable time, constructive feedback, and expert guidance, which provided clarity on how to move forward. Our meetings always left me feeling re-energized and motivated, and for that, I thank you. I also extend my appreciation to TNO for providing the necessary resources. Although the Boston Dynamics Spot robot was not used in this research, I enjoyed being in an environment surrounded by such advanced technology.

Finally, I want to thank my family for their unwavering support, not only throughout this project but also throughout my entire educational journey. This thesis represents a significant milestone in my academic journey, and I hope that the findings and methodologies presented will contribute to the field of autonomous systems.

Enjoy reading,

Sander
Delft, September 2024

Table of Contents

Nomenclature

BP Belief Propagation

CBAA Consensus-based auction algorithm

DCOP Distributed Constraint Optimization Problem

DHBA Decentralized Hungarian-based approach

GBP Gaussian Belief Propagation

MAP Maximum A Posteriori

MRBP Multi-Robot Belief Propagation

MRF Markov random field

MRTA Multi-Robot Task allocation

POMDP Partially Observable Markov Decision Process

RL Reinforcement Learning

SLAM Simultaneous Localization and Mapping

SPLAM Simultaneous Planning, Localization, and Mapping

Distributed Multi-Robot Exploration Missions: Unified approach using Gaussian Belief Propagation

S.H. Boers[1], J. Sijs[1], W.J. Meijer[2] and J.D. van Mil[2]

***Abstract*—This work builds upon the Gaussian Belief Propagation (GBP) stack, utilizing it as the core framework for distributed multi-robot exploration missions. The GBP stack's key strength lies in its single-factor graph representation of competencies such as planning, map consensus, and task allocation, making it a powerful tool for intelligent and collaborative robotic behavior. However, previous applications of GBP stack were limited to open environments without obstacles. To address this limitation, we extend the GBP stack by representing the environment as a navigational graph, allowing for task allocation and global path planning in more complex, obstacle-filled environments. This enhancement integrates seamlessly with the single-factor graph approach, enabling both navigational tasks and symbolic tasks, such as opening doors or performing inspections, to be efficiently distributed among a heterogeneous fleet of robots. A feature absent in earlier versions. Simulation results demonstrate improved task coordination, exploration efficiency, and adaptability to varied tasks, showcasing the potential of this extended framework for more challenging multi-robot exploration scenarios.**

***Index terms*—Multi-Robot, Exploration, Factor Graphs, Task allocation, Gaussian Belief Propagation**

I. INTRODUCTION

Exploring hazardous environments, such as collapsed buildings, nuclear sites, or drug labs [1], poses significant risks to human lives. The need for exploration in these areas, whether for search and rescue missions, hazard assessment, or law enforcement, demands innovative solutions to minimize human exposure to danger. Robotics has emerged as an excellent tool in this context, offering the capability to navigate a wide array of terrains with safety and precision.

When considering the deployment of robots in such environments, multi-robot systems offer several advantages over single-robot systems [2], [3]. Firstly, multi-robot systems can cover larger areas more efficiently, leading to faster and more comprehensive data collection, as multiple robots can gather diverse measurements that improve accuracy and completeness. Secondly, the use of multiple robots enhances redundancy and reliability; if one robot fails, others can continue the mission, thereby increasing the overall robustness of the operation.

To fully leverage the potential of multi-robot systems, it is desirable to seek distributed solutions, where each robot operates autonomously with local computation and peer-to-peer communication. This setup not only enhances the system's overall robustness but also its ability to scale [4]. Centralized systems, where a central computer orchestrates all movements, are prone to communication and computation bottlenecks. Additionally, these systems face the risk of complete failure if the central computer encounters a malfunction [5].

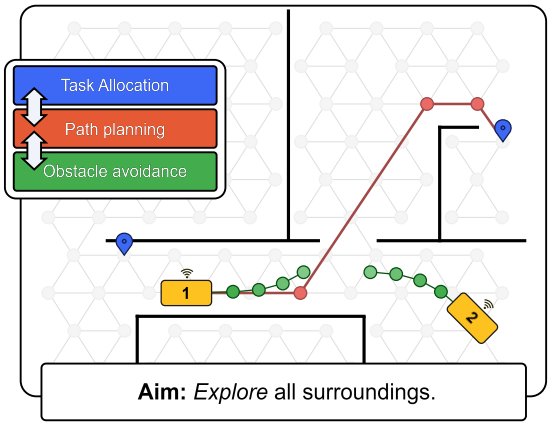


Fig. 1: An illustration of the primary components in a multi-robot exploration mission, highlighting task allocation, path planning, and obstacle avoidance.

Despite their theoretical advantages, distributed multi-robot systems are not yet widely deployed in real-world applications. This limited adoption primarily comes from the inherent complexity of such systems. In exploration missions, robot teams must address several critical competencies simultaneously: Localization, Mapping, Planning, and Coordination (as illustrated in Fig. 1). Each of these competencies presents significant challenges when implemented in a distributed manner. Moreover, the interdependence of these competencies further complicates the system's overall performance. For instance: Accurate localization is crucial for effective planning and efficient planning relies on robust coordination among robots. This intricate web of dependencies has led researchers to explore *coupled approaches* that can address these interrelated challenges cohesively. By coupling the competencies you allow for a joint-optimization over the whole exploration problem.

In this work, we demonstrate how these competencies can be represented as a *factor graph* and linked to create a distributed yet interconnected system. Factor graphs have a rich history in robotics [6], finding applications in diverse areas such as Simultaneous Localization and Mapping (SLAM) [7], tracking [8], structure from motion [9], and motion planning [10]. They are so favourable because of their ability to leverage the locality property in optimization problems, which is advantageous since many robotic optimization problems are local, involving only a subset of variables. Additionally, factor graphs are favourable due to their general applicability, meaning they can model diverse types of problems without being tailored to a specific task. This

This work was supported by TNO.

[1]Cognitive Robotics, Delft University of Technology.
[2]TNO, the Hague, the Netherlands.

inherent generality allows various robotic functions to be modeled together in one system, providing a unified framework for optimization and decision-making. This generality makes them an ideal solution for the problem we are addressing.

Patwardhan et al. recognized these advantages and utilized them to create the *GBP stack* [11], which introduced the concept of writing competencies as factor graphs and stacking them as layers. GBP stands for Gaussian Belief Propagation, a powerful factor graph inference tool for distributed systems [12]. They demonstrated its effectiveness in multi-robot information acquisition missions, yielding promising results. However, the original GBP Stack had limitations that restricted its applicability in complex, real-world scenarios. Most notably, it was designed for environments free of obstacles, lacking the capability to navigate spaces with walls or other obstructions. This constraint highlighted the need for further development to bridge the gap between theoretical models and practical applications.

This work addresses these limitations by investigating necessary additions and modifications to the GBP stack. A key innovation is the transitioning from a continuous model to one incorporating discrete frontier points (specific locations at the boundary between explored and unexplored areas [13]). This shift more accurately reflects the reality of exploration scenarios, where robots must make decisions based on specific, identifiable locations rather than continuous spaces. Consequently, this change necessitated the introduction of a robust task allocation system, enabling the efficient distribution of exploration tasks among multiple robots. This study explores methods for formulating a task allocation algorithm as a factor graph and integrating it within the GBP stack.

II. Technical Background

This section provides a technical overview of factor graphs, Gaussian Belief Propagation and the GBP Planner, which serves as the foundation of the GBP stack. For a more in-depth explanation, the reader is encouraged to view the interactive work of J. Ortiz, T. Evans, and A. J. Davison [12][3]

A. Gaussian Models

When representing uncertainty, gaussian models are often chosen for several key reasons [12]:

1) *Realistic modelling*: Gaussian models accurately reflect the distribution of many physical phenomena and sensor measurements observed in the real world [14].
2) *Mathematical simplicity*: The mathematical structure of Gaussian models is straightforward, making them easy to work with.
3) *Computational Efficiency*: Calculations with Gaussian models can be done quickly using efficient formulas.
4) *Flexibility*: Gaussian models preserve their form under common statistical operations like marginalization, conditioning, and taking products, which makes them versatile for use in robotic systems.

A gaussian distribution can be expressed in the exponential form $p(x) \propto e^{-E(x)}$, with a quadratic energy function $E(x)$ [15]. This energy function can be written in two distinct forms:

$$\underbrace{\mathcal{N}(\mathbf{X}; \boldsymbol{\mu}, \boldsymbol{\Sigma})}_{\text{Moments form}} = \underbrace{\mathcal{N}^{-1}(\mathbf{X}; \boldsymbol{\eta}, \boldsymbol{\Lambda})}_{\text{Canonical form}} \tag{1}$$

With two different energy equation:

$$E_{\text{moments}}(x) = \frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu) \tag{2}$$

$$E_{\text{canonical}}(x) = \frac{1}{2}x^\top \Lambda x - \eta^\top x \tag{3}$$

where $\boldsymbol{\mu}$ is the mean vector, $\boldsymbol{\Sigma}$ the covariance matrix, $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$ the precision matrix and $\boldsymbol{\eta} = \boldsymbol{\Lambda}\boldsymbol{\mu}$ is the information vector.

The Canonical form is significantly more computationally efficient for conditioning and product computation, whereas the Moments form is superior for marginalization. Later, it is observed that conditioning and product computation are performed frequently, which is why the canonical form is preferred.

B. Factor Graphs

Factor graphs serve as a robust representation for systems involving interdependent variables and constraints, particularly useful in probabilistic inference and optimization. A factor graph is an undirected bipartite graph comprising two types of nodes: variable nodes, representing the variables of interest, and factor nodes, representing the functions (or factors) that define the relationships among these variables. The structure of a factor graph allows for the efficient factorization of a joint function $p(\mathbf{X})$ into smaller, manageable components f_s, each depending on a subset $\mathbf{X}_s$ of the variables $\mathbf{X}$:

$$p(\mathbf{X}) = \prod_s f_s(\mathbf{X}_s) \tag{4}$$

In Gaussian factor graphs, these factors take the form of Gaussian distributions:

$$f_s(\mathbf{X}_s) \propto e^{-\frac{1}{2}\left[(\mathbf{z}_s - \mathbf{h}_s(\mathbf{X}_s))^\top \boldsymbol{\Lambda}_s (\mathbf{z}_s - \mathbf{h}_s(\mathbf{X}_s))\right]} \tag{5}$$

where $\mathbf{h}_s(\mathbf{X}_s)$ represents the functional form or measurement function of the factor. $\mathbf{z}_s$ is the observed or expected value, and $\boldsymbol{\Lambda}_s$ is the precision matrix (inverse covariance) of the factor.

$\mathbf{h}_s(\mathbf{X}_s)$ describes the expected measurement based on the involved variables $\mathbf{X}_s$. For example, for a robot measuring the distance to a landmark, the measurement function $\mathbf{h}_s$ could represent the Euclidean distance between the robot's position and the landmark's location. In this context, $\mathbf{z}_s$ is the actual measurement obtained from the sensor.

Note that we can also use factors of this form for priors which are not sensor measurements but assumptions or external knowledge. In this case, $\mathbf{z}_s = 0$, meaning that the factor energy is purely a function of the states.

C. Gaussian Belief Propagation

Gaussian Belief Propagation (GBP) solves an optimization problem by performing inference on Gaussian factor graphs.

[3] https://gaussianbp.github.io/

This is achieved through iterative message passing, where each node in the graph updates its beliefs based on incoming messages from neighboring factors. The optimization process aims to minimize the overall energy of the system, resulting in the most probable configuration of variables. The node-wise computations and message passing makes it suitable for implementation across multiple devices that communicate via networks. GBP, a specific case of loopy belief propagation, has demonstrated excellent performance, achieving exact solutions for the marginal means of all variables with rapid convergence [16]. This is achieved through the iterative execution of four distinct steps, which are:

a) *Variable Belief Update:* A variable $\mathbf{x}_k$ updates its belief by taking the product of all incoming messages from its connected factors:

$$b(\mathbf{x}_k) = \prod_{f \in n(\mathbf{x}_k)} \vec{m}_{f \to k}(\mathbf{x}_k) \tag{6}$$

where $n(\mathbf{x}_k)$ is the set of factors connected to $\mathbf{x}_k$, and $\vec{m}_{f \to k}(\mathbf{x}_k) = \mathcal{N}^{-1}(\mathbf{x}_k; \boldsymbol{\eta}_{f \to k}, \boldsymbol{\Lambda}_{f \to k})$ is the message from a factor to the variable. In the canonical form, this product simplifies to a summation:

$$\boldsymbol{\eta}_k = \sum_{f \in n(\mathbf{x}_k)} \boldsymbol{\eta}_{f \to k}, \quad \boldsymbol{\Lambda}_k = \sum_{f \in n(\mathbf{x}_k)} \boldsymbol{\Lambda}_{f \to k} \tag{7}$$

b) *Variable to Factor Message:* A message from a variable $\mathbf{x}_k$ to a factor f_j is the product of all incoming factor-to-variable messages, excluding the one from f_j:

$$\vec{m}_{\mathbf{x}_k \to j}(f_j) = \prod_{f \in n(\mathbf{x}_k) \setminus f_j} \vec{m}_{f \to k}(\mathbf{x}_k) \tag{8}$$

c) *Factor Likelihood Update:* The likelihood of a factor $f_s(\mathbf{X}_s)$ with a measurement function $\mathbf{h}_s(\mathbf{X}_s)$, observation $\mathbf{z}_s$, and precision $\boldsymbol{\Lambda}_s$ can ben expressed as:

$$\boldsymbol{\eta}_f = \boldsymbol{J}_s^\top \boldsymbol{\Lambda}_s (\boldsymbol{J}_s \mathbf{X}_s^0 + \mathbf{z}_s - \mathbf{h}_s(\mathbf{X}_s^0)), \ \boldsymbol{\Lambda}_f = \boldsymbol{J}_s^\top \boldsymbol{\Lambda}_s \boldsymbol{J}_s \tag{9}$$

where $\boldsymbol{J}_s$ is the Jacobian and $\mathbf{X}_k^0$ is the current state of the variables. This approach linearizes non-linear functions using a first-order Taylor expansion.

d) *Factor to Variable Message:* Finally, the message from a factor to a variable $\mathbf{x}_k$ is given by:

$$\vec{m}_{f \to k}(\mathbf{x}_k) = \sum_{\mathbf{x} \in X_s \setminus \mathbf{x}_k} f_s(\mathbf{X}_s) \prod_{\mathbf{x} \in \mathbf{X}_s \setminus \mathbf{x}_k} \vec{m}_{\mathbf{x} \to f}(\mathbf{x}) \tag{10}$$

This involves taking the product of the factor likelihood and the messages from all other variables, then marginalizing out all variables except $\mathbf{x}_k$.

D. GBP Planner

Patwardhan et al. [17] applied Gaussian Belief Propagation in a multi-robot planning problem to enable robots to plan paths that avoid obstacles and each other. In their approach, each robot's position and velocity over a forward time window are represented as variables in a factor graph (see Fig. 2). The factors in this graph include:

- *dynamics factors*, which ensure smooth trajectories by modelling the robots' motion dynamics.
- *obstacle factors*, which prevent collisions with static obstacles.
- *inter-robot factors*, which ensure robots avoid each other by penalizing trajectories that bring them within a critical distance.

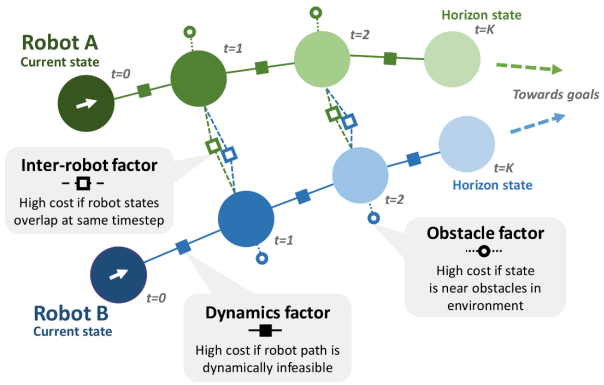


Fig. 2: The factor graph employed in the GBP planner. Circles denote variables containing position and velocity at specific timesteps. Squares represent factors defining the relationships between these variables. [17]

For detailed information on the formulation of the factor equations, please refer to Appendix A.

By using the GBP algorithm, robots communicate and update their plans in a peer-to-peer manner, achieving efficient and collision-free paths without centralized control. This planner forms the foundation on which the stack is built.

III. RELATED WORKS

This section covers related techniques concerning the integration of multiple competences. Additionally, various task allocation algorithms will be examined.

A. Coupled literature works

a) *Partially Observable Markov Decision Processes:* An exemplary case of integrating different competencies is the Simultaneous Localization and Mapping (SLAM) problem, where a robot determines its location within a map while constructing the map itself. SLAM improves inference by incorporating probability distributions of both localization and environmental states. Extending SLAM to include decision-making, known as *Simultaneous Planning, Localization, and Mapping (SPLAM)* [18], enhances autonomy through proactive planning, improved mapping accuracy, and reduced uncertainty. These methods are part of *belief space planning* [19] and often use the framework of *Partially Observable Markov Decision Processes (POMDPs)* [20]. However, POMDPs face challenges such as the difficulty of accurately shaping reward functions and the rapid growth of computational complexity with the size of the state space and number of actions, especially in multi-robot scenarios.

b) *Reinforcement Learning (RL):* RL can be seen as a coupled approach by integrating perception, decision-making, and action into a single framework, thereby eliminating individual modules.

This allows robots to learn cooperation strategies directly from their experiences, providing more cohesive multi-robot exploration [21], [22]. The challenges of designing an appropriate reward function and the time-intensive nature of RL training render it impractical for this application.

c) *Distributed Constraint Optimization Problems (DCOP):* A Distributed Constraint Optimization Problem (DCOP) is a framework used to model and solve coordination problems among multiple agents working towards a common goal while adhering to constraints [23]. While DCOPs can be formulated using factor graphs, which shares some structural similarity with Gaussian Belief Propagation (GBP), the methods differ fundamentally. DCOP focuses on optimizing discrete variable assignments and solving constraint satisfaction problems, often using combinatorial methods. In contrast, GBP is a probabilistic inference method designed for continuous variables, excelling in environments with uncertainty and dynamic updates. This makes GBP more suitable for tasks like sensor fusion and multi-robot planning where real-time belief updates and uncertainty handling are critical.

d) *GBP Stack:* Building upon the GBP planner detailed in Section II-D, where collision-free paths are generated, the GBP stack [11] introduces an innovative approach for coupling multiple competencies in multi-robot systems through the use of Gaussian Belief Propagation, formulating the problem as a stack of interconnected factor graphs. The GBP stack enables various aspects of robot operation—such as information acquisition, goal selection, and path planning—to be optimized jointly.

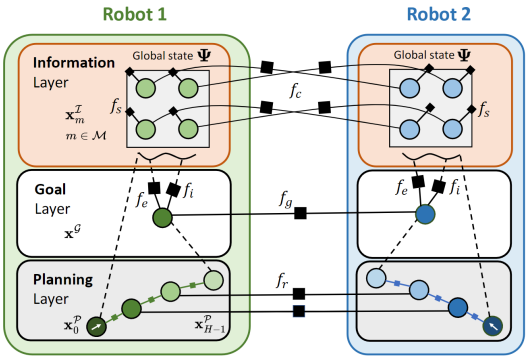


Fig. 3: The stack of factor graphs visualized. The *Information layer* maintains the robot's copy of the global state and reaches a consensus using f_c. The *Goal layer* directs the *Planning layer* for exploration, with the goal position influenced by multiple factors. The planning layer ensures collision-free paths. [11]

In the work of [11], the mission involved multiple robots collectively making measurements of a global state. This global state could represent various matters, such as gas concentration, Wi-Fi signal strength, or ocean depth.

The GBP stack consists of several layers, as seen in Fig. 3. The information layer I allows robots to update their knowledge of the environment based on local measurements and communications from other robots. The goal layer G helps in selecting regions to explore, ensuring that robots do not redundantly target the same areas. The planning layer P operates as described in the GBP planner in Section II-D, ensuring smooth, collision-free navigation through dynamic path generation.

The method discretizes the world by segmenting the environment into square sampling regions m, which forms a 2D grid. For each region m, there is one variable $\mathbf{x}_m^I$ in the information layer representing the region's information:

$$\mathbf{x}_m^I = [\boldsymbol{p}_m^I, \psi_m^I, \zeta_m^I], \tag{11}$$

where $\boldsymbol{p}_m^I$ is the position of the region, ψ_m^I is the global state value, known as the signal value, and $\zeta_m^I \in [0,1]$ indicates the coverage. Here, $\zeta_m^I = 1$ means the region has been covered. An unary signal factor is used to represent a measurement z_s made by the robot of the region m. This factor is represented as:

$$\boldsymbol{f_s} : \boldsymbol{h}_s(\mathbf{x}_m^I) = \mathbf{x}_m^I, \tag{12}$$

$$\boldsymbol{z}_s \sim \mathcal{N}\left([\boldsymbol{p}_m, \psi_m, 1]^\top, \boldsymbol{\Lambda}_s^{-1}\right). \tag{13}$$

The *inter-robot consensus factor*, f_c, stimulates consensus between the beliefs of robots about the same regions and merges information from robots that have explored areas others have not yet reached:

$$\boldsymbol{f_c} : \boldsymbol{h}_c(\mathbf{x}_m^{I_1}, \mathbf{x}_m^{I_2}) = \mathbf{x}_m^{I_1} - \mathbf{x}_m^{I_2}, \tag{14}$$

where $\mathbf{x}_m^{I_1}$ and $\mathbf{x}_m^{I_2}$ are the information layer variables of robots 1 and 2 for region m.

Through the use of an *exploration factor*, f_e, robots are guided to the nearest unexplored region.

$$\boldsymbol{f_e} : \boldsymbol{h}_e(\mathbf{x}^G; \boldsymbol{p}_{m^*}^I) = \mathbf{x}^G - \boldsymbol{p}_{m^*}^I, \tag{15}$$

$$m^* = \underset{m \in \{\arg\min \zeta_{m'}\}}{\arg\min} \parallel \boldsymbol{p}_{m^*}^I - \boldsymbol{p}_0^P \parallel, \tag{16}$$

where $\mathbf{x}^G$ is the goal variable $(x, y$-coordinates), $\boldsymbol{p}_{m^*}^I$ is the position of the nearest unexplored region m^*, and p_0^P is the current position of the robot.

Coordination between robots happens through the *goal diversity factor* f_g, and *collision avoidance factor* f_r (known from Section II-D). The goal diversity factor f_g discourages two connected robots from moving towards the same region, promoting exploratory behavior. It pushes proximate goals somewhat away from each other. This factor is represented as:

$$\boldsymbol{f_g} : \boldsymbol{h}_g(\mathbf{x}^{G_1}, \mathbf{x}^{G_2}) = \begin{cases} 1 - \frac{\parallel \mathbf{x}^{G_1} - \mathbf{x}^{G_2} \parallel}{r_D} & \|\mathbf{x}^{G_1} - \mathbf{x}^{G_2}\| \leq r_D \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

where $\mathbf{x}^{G_1}$ and $\mathbf{x}^{G_2}$ are the goal variables of two robots and r_D is a predefined distance threshold.

As mentioned earlier, there are limitations to this approach. For example, this method fails to handle walls, a common feature in real-world environments. Implementing this is not easy because of how the factors are formulated. Take, for instance, the exploration factor, which looks for the nearest unexplored cell based on Euclidean distance, thus ignoring walls. Another example is the goal diversity factor, which increases the distance between robots' goal variables so that different regions are explored. This factor relies on continuous open space. If walls are

present in the environment, there is a risk of moving the goal variable into a wall or an unreachable section of the map.

B. Task allocation literature works

To approach the problem of task allocation, many different algorithms have been examined. The scope has been set according to the taxonomy for multi-robot task allocation (MRTA) by B. P. Gerkey and M. J. Matarić [24], focusing on single task (ST), single robot (SR), instantaneous allocation (IA), and only those using a decentralized architecture. In this context, ST-SR refers to an individual task that is always performed by one robot, meaning it doesn't require multiple robots working together to complete it. These methods can be broadly categorized into several types [25]:

a) *Auction-Based Methods:* Agents bid on tasks based on local utility calculations, with the consensus-based auction algorithm (CBAA) [26] being a widely adopted method. Each agent computes a utility for each task, broadcasts its bid, and negotiates with others until a consensus is reached on which agent should handle each task. This iterative process ensures that tasks are allocated to the agents that are most capable of completing them. However, the strength of this approach—flexible, decentralized decision-making—comes with challenges. The required rounds of communication between agents can lead to significant overhead, especially in scenarios with limited bandwidth or high agent numbers. This can slow down decision-making and reduce overall system efficiency, particularly in dynamic environments where quick task reallocation is necessary.

b) *Game Theory-Based Methods:* Game Theory-Based Methods: These methods view agents as players in a game, where each agent aims to maximize its own payoff through strategic interactions. This can involve cooperative games, where agents form coalitions, or non-cooperative games, where agents act independently. The goal is often to reach a Nash equilibrium, where no agent can improve its outcome by changing its strategy unilaterally [27]. While these methods provide solid theoretical foundations, they tend to be computationally expensive and less scalable for real-time, multi-robot coordination in dynamic environments. Additionally, the need for complex strategy formulation and convergence to equilibrium can introduce delays, making them unsuitable for systems requiring rapid decision-making and adaptation. Therefore, this approach was not pursued further.

c) *Optimization-Based Methods:* These include deterministic optimization techniques like the Hungarian algorithm and metaheuristic methods such as genetic algorithms and particle swarm optimization. These methods aim to find optimal or near-optimal task allocations by minimizing costs or maximizing utility. Among these, the decentralized Hungarian-based approach stands out. According to S. Ismail and L. Sun [28], the decentralized Hungarian-based algorithm (DHBA) consistently outperforms the consensus-based auction algorithm (CBAA) in terms of converging speed, optimality of assignments, and computational efficiency. This demonstrates that the DHBA is highly effective in achieving fast and scalable task allocation, making it an excellent choice for decentralized systems.

d) *Belief propagation-Based Methods:* J. N. Schwertfeger and O. C. Jenkins [29] proposed *Multi-Robot Belief Propagation* (M-RBP), where robots use Bayesian belief propagation within a Markov random field (MRF) representation to infer task assignments based on local observations and beliefs about other robots' intentions. MRBP-I [30] extends MRBP by incorporating identity constraints, allowing robots to consider individual capabilities and preferences in the task allocation process. Unlike MRBP, which uses an MRF representation, MRBP-I employs a factor graph.

The factor graph structure includes observed variables which indicate task attributes and robot capabilities and unobserved (latent) variables which capture the underlying probability distributions over task assignments. Factor nodes include unary factors, representing the probability of a robot-task assignment based on identity and observations, and pairwise factors, capturing the compatibility between robots' task allocations.

However, the proposed solution does not guarantee optimal assignments of robots to tasks and may even result in multiple robots selecting the same task. Additionally, the formulation of the factor graph in MRBP-I is not solvable with Gaussian belief propagation, as the factors are not written as Gaussian distributions.

IV. PROBLEM STATEMENT

We consider the problem of exploring an area with obstacles using multiple robots operating under a distributed architecture. That is, there is no central computer for control or computation; instead, each robot depends on local information and peer-shared data. We extend the problem to incorporate heterogeneous tasks, making the approach versatile and suitable for real-world applications. This includes tasks beyond the standard *visit-frontier* commands for further exploration, but featuring diverse frontiers such as *open-door* and *perform-inspection*. Not all tasks are executable by every robot. For instance, opening a door necessitates a robot equipped with a robotic arm, while performing an inspection requires a robot with camera capabilities. Therefore, it is essential to consider the specific capabilities of each robot when assigning tasks.

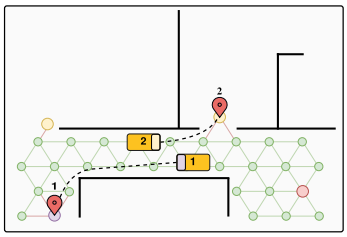
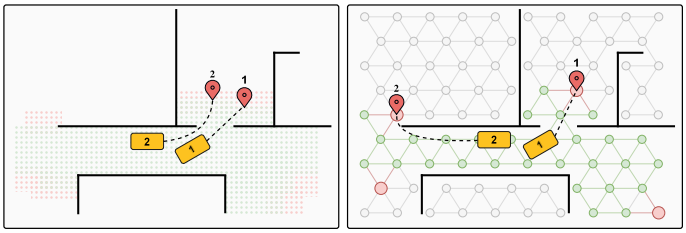


Fig. 4: Illustration of various types of frontier tasks, demonstrating that only specific robots with the necessary capabilities can execute these tasks effectively. For instance, a robot with the yellow capability is designated to handle the corresponding yellow frontier task.

Given the change in the scenario where the environment is no longer obstacle-free, the approach transitions from continuous world positions to a discrete set of identifiable goal locations, corresponding to various types of frontiers. To allocate frontiers effectively, a task allocation algorithm is required. Each task with a specific cost-to-go is distributed to robots within the set that are connected to each other.



(a) Original GBP stack [11] (b) Goal

Fig. 5: Illustration comparing the original method (a), which utilizes continuous position variables, with the goal method (b), featuring a discrete set of locations.

The objective is to minimize the total cost of assigning tasks to robots, formulated as:

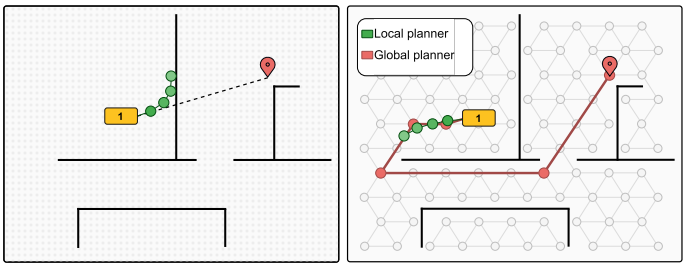
$$\min \sum_{i-1}^{N} \sum_{j=1}^{T} c_{ij} k_{ij} \tag{18}$$

where N is the number of robots, T the number of tasks, c_{ij} the cost of executing task j to robot i, and finally k_{ij} is a binary decision variable where $k_{ij} = 1$ if task j is assigned to robot i, and $k_{ij} = 0$ otherwise. To ensure that each robot is assigned at most one task and each task is assigned to at most one robot:

$$\sum_{j=0}^{T} k_{ij} \leq 1 \forall i \in \{1, 2, ..., T\} \tag{19}$$

$$\sum_{i=0}^{N} k_{ij} \leq 1 \forall j \in \{1, 2, ..., N\} \tag{20}$$

In summary, the new method should include several qualitative improvements: 1) the capability to explore environments with obstacles, and 2) the ability to comprehend and execute heterogeneous tasks. Additionally, specific performance metrics are identified to evaluate improvements over the original method: the *distance traveled* by the robots, as a shorter distance implies quicker exploration, and the *distances between selected goals*, measured at the same time, with the belief that a larger average distance between simultaneously assigned goals indicates more spread-out exploration.



(a) Original GBP stack [11] (b) Goal

Fig. 6: Comparison of the original stack and the goal method for navigation in office-like environments. The original method (a) fails to handle walls.

V. METHODOLOGY

In this section, we elaborate on the improvements made to the GBP stack to extend its applicability to more complex scenarios beyond simple, obstacle-free environments.

A. *Navigational Graph - World layer*

The original stack utilized an information layer where the environment was represented as segmented square regions. This implementation could not model relationships between cells, essential for navigation, nor could it represent obstacles or walls.

To address these limitations, this work employs a navigational graph instead of a 2D grid world. A navigational graph, which is different from a factor graph, consists of nodes and edges, where the nodes represent specific positions within the environment and the edges represent the possible paths or connections between them. This allows for the representation of traversability, enabling the understanding of walls and obstacles. By utilizing a graph, we can accurately calculate actual distances to tasks, rather than relying on the Euclidean distance used in the original stack, which is not accurate in environments with walls.

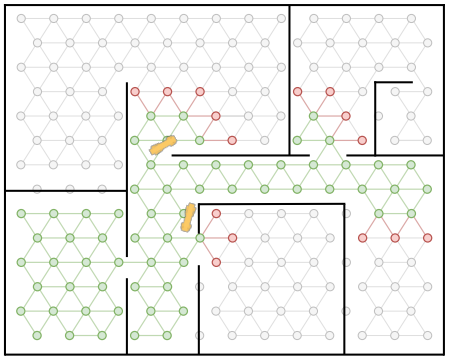


Fig. 7: This figure illustrates the navigational graph built using a triangular grid. Each node represents a specific position, with edges indicating traversability. Green nodes denote visited locations, while red nodes represent frontiers (potential tasks). This graph enables accurate distance calculation for efficient path planning and task allocation.

This graph is built as the robot explores, leaving a trail of nodes with edges as it progresses. In this method, we simplify by assuming that the entire graph is pre-built and that the robot 'discovers' the nodes as it explores. This ensures that all nodes discovered by the robots are in the same positions, eliminating discrepancies in node positioning and ensuring consistency and reliability in task execution. This simplification allows us to focus on the core problem of task allocation rather than the complexities of node sampling algorithms and the need for complex node matching algorithms between robots. A triangular grid was chosen because its six-connected structure offers more flexible movement and better spatial coverage, reducing navigation constraints compared to the four options in a regular grid.

To model this graph within the stack, each node of the navigational graph corresponds to a variable node in the factor graph. This variable encompasses data on the position $\boldsymbol{p}_n^{\mathcal{W}}$, coverage $\zeta_n^{\mathcal{W}}$

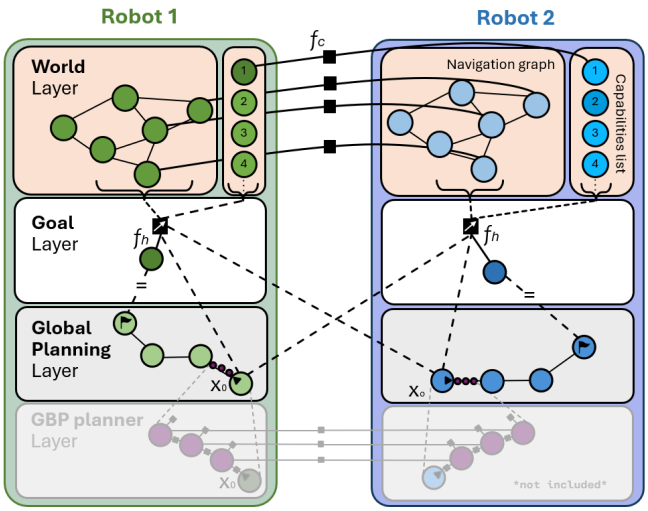


Fig. 8: The improved stack of factor graphs for multi-robot exploration. The updated framework includes several noticeable changes: the integration of a navigational graph for enhanced path planning, the inclusion of a Hungarian factor for efficient task allocation, the consideration of robot capabilities for handling heterogeneous tasks, and the generation of waypoints by the global planner.

and node type (frontier, door, suspicious object) $\varphi_n^{\mathcal{W}}$. The layer has also been renamed to the world layer, reflecting its broader representation of world data, as opposed to the information layer used in the information acquisition mission in the original stack [11].

$$\mathbf{x}_n^{\mathcal{W}} = [\boldsymbol{p}_n^{\mathcal{W}}, \zeta_n^{\mathcal{W}}, \varphi_n^{\mathcal{W}}] \tag{21}$$

The coverage $\zeta_n^{\mathcal{W}}$ can represent three distinct states, each corresponding to a specific integer value: 0 for "unvisited", 1 for "task", and 2 for "visited". Similarly, the node type $\varphi_n^{\mathcal{W}}$ is represented by integers, with each integer corresponding to a particular type of node.

B. Consensus algorithm

Each robot maintains a copy of the navigational graph represented in the world layer. When two robots establish a connection, consensus factors are created for each corresponding node in the respective world layer, denoted as f_c in Fig. 8. Following an iteration of Gaussian Belief Propagation, both robots share and update their knowledge about the coverage, and available tasks, leading to a consensus about the world. The factor's measurement function is:

$$\boldsymbol{f_c} : \boldsymbol{h}_c\big(\mathbf{x}_n^{\mathcal{W}_1}, \mathbf{x}_n^{\mathcal{W}_2}\big) = \mathbf{x}_n^{\mathcal{W}_1} - \mathbf{x}_n^{\mathcal{W}_2} \tag{22}$$

We adopt the same communication protocol established in related GBP works, wherein each robot continuously broadcasts its state information. When within communication range, other robots can read this information.

C. Task allocation - Goal layer

The goal layer is responsible for determining the robots' next destination. It comprises of a single variable $\mathbf{x}^G$, containing the x and y coordinates of the goal position. In the original stack, this position converged through three factors (f_e, f_i, f_g), as detailed in Section III-A.d. Which was feasible in open spaces. However, by using the navigational graph, we now have a discrete set of tasks (T). A task allocation algorithm selects the goal position, as it can handle the discrete list of tasks. The decentralized version of the Hungarian algorithm (DHBA) [28] is employed for its instantaneous, local, and optimal characteristics.

This algorithm operates using a cost matrix $\boldsymbol{C}$, where rows represent tasks (T) and columns represent the connected robots (R). The values c_{ij} in the matrix denote the cost of each task-robot combination $\big(T_i, R_j\big)$. These costs are calculated using the navigational graph, the robot's own position $\boldsymbol{p}_{0,A}$ and the position variables received from connected robots, $\boldsymbol{p}_{0,B}, \boldsymbol{p}_{0,C}, \ldots$.

$$C_{\text{example}} = \begin{bmatrix} 21 & 1 & 10 & 43 & 25 \\ 29 & 20 & 47 & 20 & 5 \\ 32 & 16 & 34 & 12 & 40 \\ 31 & 17 & 45 & 39 & 31 \\ 1 & 10 & 36 & 8 & 20 \\ 10 & 5 & 13 & 44 & 24 \end{bmatrix} \tag{23}$$

The Hungarian algorithm is deterministic, ensuring that when two robots share their knowledge, the resulting task assignment will be consistent and identical. This guarantees that no task T_i is assigned to more than one robot R_j, maintaining an efficient and conflict-free task allocation process across robots. The specifics of how the Hungarian algorithm functions can be found in Appendix B.

To integrate the Hungarian algorithm into the stack, it is encapsulated within a factor named *hungarian factor*, $\boldsymbol{f_h}$. This factor computes the optimal node n^* at each timestep t and updates the goal variable $\mathbf{x}^G$ after iterations of GBP.

$$\boldsymbol{f_h} : \boldsymbol{h}_h\big(\mathbf{x}^G; \boldsymbol{p}_{n^*}\big) = \mathbf{x}^G - \boldsymbol{p}_{n^*} \tag{24}$$

$$n^* = \text{hungarian}\big(\mathbf{x}^{\mathcal{W}^{\text{nav}}}, \mathbf{x}^{\mathcal{W}^{\text{cap}}}, \boldsymbol{p}_{0,A}, \boldsymbol{p}_{0,B}, \boldsymbol{p}_{0,C}, \cdots\big) \tag{25}$$

D. Heterogenous tasks

Integrating heterogeneous tasks was straightforward with the use of the Hungarian method and the navigational graph. A capability list $\mathbf{x}^{\mathcal{W}^{\text{cap}}}$ is added to the world layer $\mathcal{W}$ to account for robot capabilities. Each variable in this list represents a robot's capabilities, and each robot maintains a copy of this list. This list is used when tasks other than the visit-frontier task are involved. If a robot lacks the capability to perform a specific task, the corresponding value c_{ij} in the cost matrix $\boldsymbol{C}$ is significantly increased, ideally to a value approaching infinity. This prevents the Hungarian algorithm from allocating that task to the incapable robot.

The capability list is updated when robots connect with each other, in the same manner as other world variables, using consensus factors f_c. This allows robots to learn about each other's capabilities.

E. Global planner

The utilization of the navigational graph allows for the creation of a global planner, a novel feature for the GBP planner [17] and the GBP stack [11]. The global planner takes a start and goal location and outputs waypoints that the local planner uses

to navigate to the goal via the shortest path. We utilize Dijkstra's algorithm due to its guarantees for finding the shortest path and its computational efficiency.

F. Local planner

In this method, the local planner, also known as the GBP planner [17], is omitted. Re-implementing the local planner was not the focus of this work. However, it should be easily integrated within the stack, as it uses the same GBP framework and iterations. In this work, we use discrete timesteps where a robot moves one node further.

G. Algorithm

Robots initialize their GBP stacks with the variables and factors for each layer as previously detailed. At each timestep, every robot executes the procedure outlined in Algorithm 1.

Algorithm 1: For each robot R_i

Let $N(R_i) = \{R_j \mid \|R_i - R_j\| < r_C\}$ be a set of robots within the communication radius of R_i.
while Running **do**
  Manage robot connections and create/delete inter-factors f_c, (f_r).
  Set *unvisited* neighbor nodes as *tasks* ($\zeta_n^{\mathcal{W}} = 1$).
  Read information states of connected robots ($\boldsymbol{p}_{0,j}$).
  Build cost matrix $\mathbf{C}$ and perform task allocation using the Hungarian algorithm. Set $n^* \Rightarrow \boldsymbol{f}_h$.
  if assigned task **do**
    Plan paths with Dijkstra's algorithm.
    Execute GBP iteration.
    Update positions $\boldsymbol{p}_{0,i}$ and nodes by setting the current node to *visited* ($\zeta_n^{\mathcal{W}} = 2$).
  Continuously broadcast information states.
end while

Algorithm 1: The complete algorithm of the the improved stack.

VI. RESULTS

This section presents the results of the implemented methodology in two parts. The first part demonstrates the quantitative improvements. The second part compares the stacks based on the metrics of *average distance between goals* and *distance traveled*, both of which highlight more efficient distributed exploration.

A. Setup

To evaluate the method's competence, a simulator has been developed. The GBP stack[4] was translated from C++ to Python to facilitate rapid prototyping. The navigational graph was implemented using *NetworkX* [31], and the *NiceGUI* [32] library was utilized for visualization. This library allows for rendering a 3D world within the browser using *three.js*.

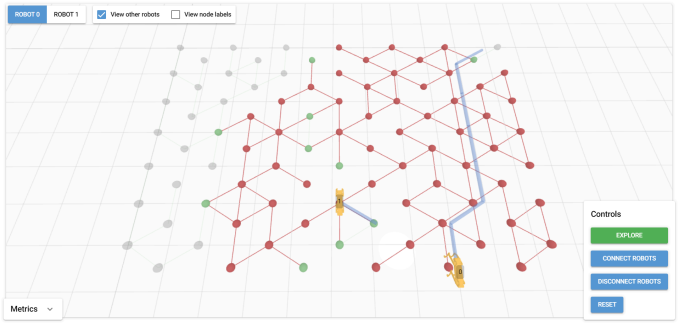
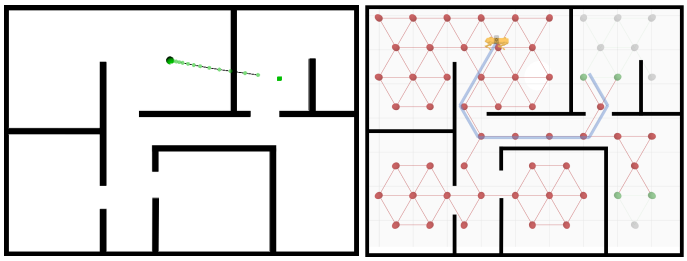


Fig. 9: Screenshot of the simulator highlighting the navigational graph. Red nodes indicate visited areas, green nodes represent discovered frontiers. Two robots are seen exploring the environment, the blue line denotes the planned path.

To directly compare the original stack and the improved stack in Section VI-D,E, identical maps were used for both methods. Each method explores a 100m × 100m open world, with 10m wide grid cells. This method's grid structure was changed from triangular to rectangular for this experiment. A cell is marked as visited when a robot reaches its center, with the goal of visiting every cell. The experiment was conducted with five robots. Two scenarios were tested: in the first, all robots start in the corner; in the second, robots are spawned at random positions. This scenario was repeated five times with different random seeds to ensure reliability.

B. Result 1 - Complex environments

The first quantitative improvement is the method's capacity to handle environments with walls , enabling navigation in complex environments with obstacles, as demonstrated in Fig. 10.



(a) Original GBP stack [11] (b) Improved GBP stack

Fig. 10: Screenshots of the GBP stacks in action. The original GBP stack (a) attempts to move through walls, while the improved GBP stack (b) uses a global planner to navigate around obstacles efficiently.

C. Result 2 - Heterogenous tasks

Robots are given specific capabilities, which the task allocation algorithm considers when assigning tasks. For instance, in the scenario depicted in Fig. 11, robot 2 is significantly closer to the door than robot 1. However, because robot 2 lacks the necessary arm to open the door, this task is assigned to robot 1. Once a door is opened, it remains open, but if it closes, the node is marked unvisited again, and the task is reassigned.

[4]github.com/aalpatya/gbpstack

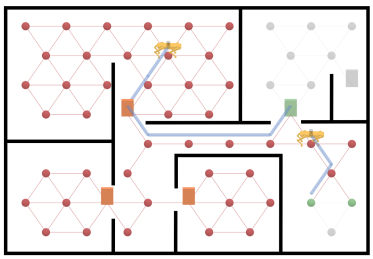


Fig. 11: Screenshot illustrating various types of frontiers, specifically doors. The robot on the far right is closer to the door but lacks the necessary capability, hence it is not assigned to that task.

D. Result 3 - Improved coordination

The original stack selected the next location to explore by choosing the nearest unexplored cell. This approach could lead to multiple robots choosing the same location, resulting in suboptimal exploration. In contrast, the improved stack divides the world into discrete task positions and uses task allocation to ensure no two robots select the same task. This ensures that robots explore different parts of the world, enhancing coordination.

To evaluate, the average distance between selected goals of all robots over time was measured. The results, however, did not show significant differences, as noted in Table I. This is because, while the original method occasionally chose the same goal, it also often selected goals on the opposite side of the map, leading to a relatively balanced average distance.

TABLE I: COMPARISON

	Corner		Random	
	Original [11]	This work	Original [11]	This work
Distance Between Goals [m]	**28.9 ± 16.8**	25.8 ± 12.7	34.3 ± 19.9	**38.9 ± 18.9**
Distance Traveled [m]	358.2 ± 6.9	**222.0 ± 14.7**	399.8 ± 10.2	**220.0 ± 8.5**

E. Result 4 - Faster exploration

The metric of *distance traveled* offers a more conclusive assessment of the effectiveness of the implemented method. We use the metric *distance traveled* over *exploration time* to ensure that velocity and acceleration do not influence the results. As shown in Table I, there is a significant improvement in both scenarios. On average, the improved stack results in a 42% reduction in distance traveled compared to the original method, demonstrating its effectiveness.

The trajectories of the robots can be seen in Fig. 12 and Fig. 13, where one of the random trajectories has been selected. It is noticeable that the original method more frequently takes the same paths. The less direct paths in the original method are due to the local planner.

VII. DISCUSSION

Although the current methodology effectively addresses the objectives outlined in the problem statement, some goals are not entirely fulfilled with this approach. The ambition to integrate all competences into a unified optimization problem, and as it currently is implemented is debatable. Specifically, the discrete task list solution via the Hungarian factor is more an encapsulation of the Hungarian algorithm than an inherent part of an optimization framework. Similarly, the global planner's path planning based on Dijkstra's algorithm is not formulated as a factor graph optimization. These types of problems may not be ideally suited for Gaussian Belief Propagation (GBP) due to its limitations in handling the discrete nature of such tasks and the inability to enforce hard constraints.

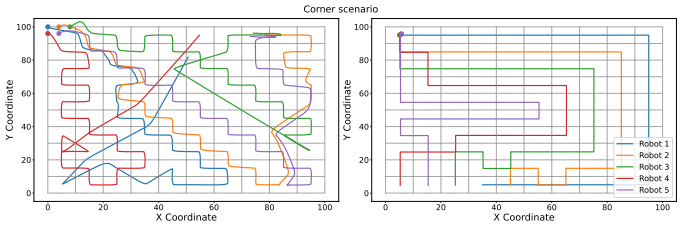


Fig. 12: Robot trajectories in the corner scenario, comparing the original stack (left) with the improved stack (right).

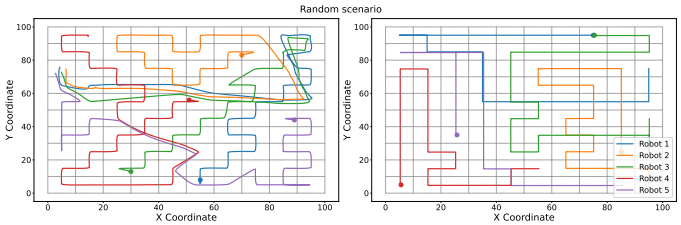


Fig. 13: Robot trajectories in the random scenario, comparing the original stack (left) with the improved stack (right).

Recent literature has explored solutions for these issues in the form of discrete-continuous factor graph solvers [33]. However, these methods remain centralized and do not adopt the Gaussian form. Despite these challenges, there remains potential for the future application of Gaussian Belief Propagation.

There are numerous problems that fit well within the Gaussian factor graph framework, making them ideal candidates for future work. For example, this work assumes perfect localization; however, integrating Simultaneous Localization and Mapping (SLAM) into the framework would be advantageous. This integration would eliminate the need for a pre-built map. It does bring new challenges such as map merging and intelligent landmark sampling.

Another area for future work involves deploying the framework on real robots. Although the framework has been designed in a distributed fashion, it currently runs on a single computer. Implementing message passing in a distributed robot network, as has been demonstrated in previous research [34], would be a critical next step.

VIII. CONCLUSION

This work presents significant advancement in the field of multi-robot exploration by addressing key limitations of the original GBP stack. The improved stack incorporates discrete fron-

tier points in a navigational graph and a robust task allocation system based on the decentralized hungarian algorithm.

The implementation of Gaussian Belief propagation offers several key advantages, including parallel asynchronous processing of information across multiple robots, which enables robust and distributed decision-making. Additionally, the framework achieves significant scalability thanks to its distributed architecture.

The results demonstrate that the improved stack significantly outperforms the original in several critical aspects. Notably, the ability to navigate environments with walls and obstacles. The introduction of capability-based task allocation enabling applicability in real-world scenarios. Results also show the reduction of distance traveled by an average of 42%, showcasing its efficiency and improved coordination.

REFERENCES

[1] W. Meijer, A. Kemmeren, J. van Bruggen, T. Haije, J. Fransman, and J. van Mil, "Situational Graphs for Robotic First Responders: an application to dismantling drug labs," *arXiv preprint arXiv:2404.17395*, 2024, doi: 10.48550/arXiv.2404.17395.

[2] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on robotics*, vol. 21, no. 3, pp. 376–386, 2005, doi: 10.1109/TRO.2004.839232.

[3] D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schulz, and B. Stewart, "Distributed multirobot exploration and mapping," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1325–1339, 2006, doi: 10.1109/CRV.2005.36.

[4] N. Atanasov, J. Le Ny, K. Daniilidis, and G. J. Pappas, "Decentralized active information acquisition: Theory and application to multi-robot SLAM," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 4775–4782. doi: 10.1109/ICRA.2015.7139863.

[5] R. Zlot, A. Stentz, M. Dias, and S. Thayer, "Multi-robot exploration controlled by a market economy," vol. 3, pp. 3016–3023, 2002, doi: 10.1109/ROBOT.2002.1013690.

[6] F. Dellaert, "Factor graphs: Exploiting structure in robotics," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, no. 1, pp. 141–166, 2021, doi: 10.1146/annurev-control-061520-010504.

[7] F. Dellaert, M. Kaess, and others, "Factor graphs for robot perception," *Foundations and Trends® in Robotics*, vol. 6, no. 1–2, pp. 1–139, 2017, doi: 10.1561/2300000043.

[8] J. Pöschmann, T. Pfeifer, and P. Protzel, "Factor graph based 3d multi-object tracking in point clouds," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 10343–10350. doi: 10.48550/arXiv.2008.05309.

[9] A. Baid *et al.*, " GTSFM: Georgia Tech Structure from Motion." [Online]. Available: https://github.com/borglab/gtsfm

[10] M. Mukadam, J. Dong, X. Yan, F. Dellaert, and B. Boots, "Continuous-time Gaussian process motion planning via probabilistic inference," *The International Journal of Robotics Research*, vol. 37, no. 11, pp. 1319–1340, 2018, doi: 10.48550/arXiv.1707.07383.

[11] A. Patwardhan and A. J. Davison, "A distributed multi-robot framework for exploration, information acquisition and consensus," *arXiv preprint arXiv:2310.01930*, 2023, doi: 10.48550/arXiv.2310.01930.

[12] J. Ortiz, T. Evans, and A. J. Davison, "A visual introduction to Gaussian belief propagation," *arXiv preprint arXiv:2107.02308*, 2021, doi: 10.48550/arXiv.2107.02308.

[13] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*, 1997, pp. 146–151. doi: 10.1109/CIRA.1997.613851.

[14] E. T. Jaynes, *Probability theory: The logic of science*. Cambridge university press, 2003.

[15] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, F. Huang, and others, "A tutorial on energy-based learning," *Predicting structured data*, vol. 1, no. 0, 2006.

[16] J. Du, S. Ma, Y.-C. Wu, S. Kar, and J. M. Moura, "Convergence analysis of belief propagation on Gaussian graphical models," *arXiv preprint arXiv:1801.06430*, 2018.

[17] A. Patwardhan, R. Murai, and A. J. Davison, "Distributing collaborative multi-robot planning with Gaussian belief propagation," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 552–559, 2022, doi: 10.48550/arXiv.2203.11618.

[18] A. Agha *et al.*, "Nebula: Quest for robotic autonomy in challenging environments; team costar at the darpa subterranean challenge," *arXiv preprint arXiv:2103.11470*, 2021, doi: 10.48550/arXiv.2103.11470.

[19] T. Regev and V. Indelman, "Multi-robot decentralized belief space planning in unknown environments via efficient re-evaluation of impacted paths," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 5591–5598. doi: 10.1109/IROS.2016.7759822.

[20] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial intelligence*, vol. 101, no. 1–2, pp. 99–134, 1998, doi: 10.1016/S0004-3702(98)00023-X.

[21] D. He, D. Feng, H. Jia, and H. Liu, "Decentralized exploration of a structured environment based on multi-agent deep reinforcement learning," in *2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS)*, 2020, pp. 172–179. doi: 10.1109/ICPADS51040.2020.00032.

[22] A. H. Tan, F. P. Bejarano, Y. Zhu, R. Ren, and G. Nejat, "Deep reinforcement learning for decentralized multi-robot exploration with macro actions," *IEEE Robotics and Automation Letters*, vol. 8, no. 1, pp. 272–279, 2022, doi: 10.1109/LRA.2022.3224667.

[23] F. Fioretto, E. Pontelli, and W. Yeoh, "Distributed constraint optimization problems and applications: A survey," *Journal of Artificial Intelligence Research*, vol. 61, pp. 623–698, 2018.

[24] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The International journal of robotics research*, vol. 23, no. 9, pp. 939–954, 2004, doi: 10.1177/0278364904045564.

[25] G. M. Skaltsis, H.-S. Shin, and A. Tsourdos, "A review of task allocation methods for UAVs," *Journal of Intelligent & Robotic Systems*, vol. 109, no. 4, p. 76–77, 2023, doi: 10.1007/s10846-023-02011-0.

[26] H.-L. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE transactions on robotics*, vol. 25, no. 4, pp. 912–926, 2009, doi: 10.1109/TRO.2009.2022423.

[27] V. Singhal and D. Dahiya, "Distributed task allocation in dynamic multi-agent system," in *International Conference on Computing, Communication & Automation*, 2015, pp. 643–648. doi: 10.1109/CCAA.2015.7148452.

[28] S. Ismail and L. Sun, "Decentralized hungarian-based approach for fast and scalable task allocation," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2017, pp. 23–28. doi: 10.1109/ICUAS.2017.7991447.

[29] J. N. Schwertfeger and O. C. Jenkins, "Multi-robot belief propagation for distributed robot allocation," in *2007 IEEE 6th international confer-*

ence on development and learning, 2007, pp. 193–198. doi: 10.1109/DEVLRN.2007.4354060.

[30] J. A. Berry, E. A. Olson, A. Gilbert, and O. C. Jenkins, "A Case of Identity: Enacting Robot Identity with Belief Propagation for Decentralized Multi-Agent Task Allocation," in *2023 32nd IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 2023, pp. 2373–2379. doi: 10.1109/RO-MAN57019.2023.10309580.

[31] A. Hagberg, P. Swart, and D. S Chult, "Exploring network structure, dynamics, and function using NetworkX," 2008.

[32] F. Schindler and R. Trappe, "NiceGUI: Web-based user interfaces with Python. The nice way.." [Online]. Available: https://github.com/zauberzeug/nicegui

[33] K. J. Doherty, Z. Lu, K. Singh, and J. J. Leonard, "Discrete-continuous smoothing and mapping," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 12395–12402, 2022.

[34] R. Murai, J. Ortiz, S. Saeedi, P. H. Kelly, and A. J. Davison, "A robot web for distributed many-device localisation," *IEEE Transactions on Robotics*, 2023.

[35] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1–2, pp. 83–97, 1955, doi: 10.1002/nav.3800020109.

APPENDIX A - GBP PLANNER FACTORS

Pose Factor

The Pose Factor connects the robot's current state and its horizon state, anchoring the trajectory at these specific points. This factor ensures that the optimization process respects the robot's initial and final states during a planning horizon, as these cannot be changed.

$$\boldsymbol{h}_p(\mathbf{x}_k) = \mathbf{x}_k, \quad \boldsymbol{\Lambda}_p = \sigma_p^{-2}\mathbf{I} \tag{26}$$

Where:

- $\mathbf{x}_k$ represents the state of the robot at time k, which is defined as $\mathbf{x}_k = [x_k, y_k, \dot{x}_k, \dot{y}_k]$, where x_k and y_k are the positional coordinates, and $\dot{x}_k$ and $\dot{y}_k$ are the corresponding velocities.
- $\sigma_p^{-2}\mathbf{I}$ is the precision matrix, with σ_p being the standard deviation associated with the pose uncertainty. In this factor, the value of σ_p is set to a very small number, such as 1×10^{-15}, which gives the anchoring behavior.

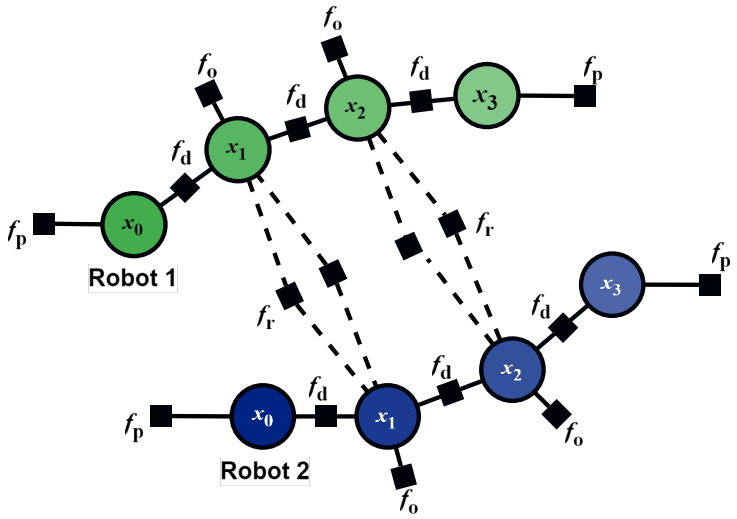


Fig. 14: Illustration of the variables and factors used in GBP Planner.

Dynamics Factor

The Dynamics Factor enforces smoothness by connecting consecutive states in time, thereby ensuring the trajectory is dynamically feasible. This factor is derived from a noise-on-acceleration model of dynamics.

$$\boldsymbol{h}_d(\mathbf{x}_k, \mathbf{x}_{k+1}) = \boldsymbol{\Phi}(t_{k+1}, t_k)\mathbf{x}_k - \mathbf{x}_{k+1} \tag{27}$$

$$\boldsymbol{\Lambda}_d = \begin{pmatrix} \frac{1}{3}\Delta t_k^3 \mathbf{Q}_d & \frac{1}{2}\Delta t_k^2 \mathbf{Q}_d \\ \frac{1}{2}\Delta t_k^2 \mathbf{Q}_d & \Delta t_k \mathbf{Q}_d \end{pmatrix}^{-1} \tag{28}$$

with $\Delta t_k = t_{k+1} - t_k$, $\mathbf{Q}_d = \sigma_d^2 \mathbf{I}$, and

$$\boldsymbol{\Phi}(t_b, t_a) = \begin{pmatrix} \mathbf{I} & (t_b - t_a)\mathbf{I} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \tag{29}$$

Where:

- Δt_k is the time difference between consecutive states.
- $\mathbf{Q}_d = \sigma_d^2 \mathbf{I}$ represents the process noise covariance matrix.
- $\boldsymbol{\Phi}(t_b, t_a)$ is the state transition matrix from time t_a to time t_b.

Obstacle Factor

The Obstacle Factor represents the robot's distance from static obstacles, penalizing states that bring the robot too close to these obstacles.

$$\boldsymbol{h}_o(\mathbf{x}_k) = \begin{cases} 1 - \frac{d_o(\mathbf{x}_k)}{r_R} \text{ if } d_o(\mathbf{x}_k) \leq r_R \\ 0 \qquad\qquad \text{otherwise} \end{cases} \tag{30}$$

$$\boldsymbol{\Lambda}_o = (t_k \sigma_r)^{-2} \mathbf{I} \tag{31}$$

Where:

- $d_o(\mathbf{x}_k)$ is the signed distance from the robot's state $\mathbf{x}_k$ to the nearest obstacle. A pre-computed 2D signed distance field image with the positional coordinates ($\mathbf{x}_k[0{:}2]$) are used to obtain this value.
- r_R is the robot's radius.

Inter-robot Factor

The Inter-robot Factor penalizes states where robots come within a critical distance of each other, ensuring collision avoidance between robots.

$$\boldsymbol{h}_r\big(\mathbf{x}_k^A, \mathbf{x}_k^B\big) = \begin{cases} 1 - \frac{d_r(\mathbf{x}_k^A, \mathbf{x}_k^B)}{r^*} \text{ if } d_r\big(\mathbf{x}_k^A, \mathbf{x}_k^B\big) \leq r^* \\ 0 \qquad\qquad\qquad\quad \text{otherwise} \end{cases} \tag{32}$$

$$d_r\big(\mathbf{x}_k^A, \mathbf{x}_k^B\big) = \parallel \mathbf{x}_k^A - \mathbf{x}_k^B \parallel \tag{33}$$

Where:

- $d_r\big(\mathbf{x}_k^A, \mathbf{x}_k^B\big)$ is the Euclidean distance between the states of robots A and B at time k.
- $r^* = 2r_R + \epsilon$, with ϵ being a small safety distance.

These factors collectively ensure the GBP Planner's efficiency and effectiveness in generating collision-free, smooth trajectories for multiple robots in dynamic environments. The iterative process of Gaussian Belief Propagation within and between robots allows for decentralized and scalable multi-robot planning.

APPENDIX B - HUNGARIAN ALGORITHM

The Hungarian algorithm is a combinatorial optimization algorithm used to solve the assignment problem. The assignment problem involves finding the most cost-effective way to assign n tasks to n agents such that the total cost is minimized (or the total profit is maximized). The algorithm efficiently finds an optimal solution for the assignment problem with a time complexity of $O(n^3)$.

Algorithm 2: Hungarian algorithm Pseudocode

Initialization step

Initialize the cost matrix `C` of size `n x n`

For each row `i` in `C`
  Subtract the minimum value of the row from each element in the row
For each column `j` in `C`
  Subtract the minimum value of the column from each element in the column
while the number of covering lines is less than `n`
  Cover all zeros in the matrix using a minimum number of horizontal and vertical lines
  if the number of covering lines is equal to `n` **then**
    Optimal assignment is possible
    Go to the assignment step
  else
    Find the smallest element not covered by any line
    Subtract this smallest element from all uncovered elements
    Add this smallest element to all elements covered by both a horizontal and a vertical line
  end if
end while

Assignment step

Construct the optimal assignment from the zero elements in the matrix

For each row `i`
  For each column `j`
    if `C[i][j] == 0` **then**
      Assign task `j` to agent `i`
    end if
  end
end

Algorithm 2: The Hungarian algorithm [35]

Appendix C - Simulation Environment

This appendix provides a detailed overview of the simulator developed to test and visualize the coordination of robots using Gaussian Belief Propagation. The simulator is built upon a translated version of the GBP library by A. Patwardhan and A. J. Davison [11], originally written in C++ and adapted into Python for enhanced flexibility during development.

The decision to translate the GBP library into Python, while reducing the algorithm's efficiency, significantly increased the freedom to prototype new features and experiment with the system's behavior. To enable visualization of the robots and their task coordination, the simulator uses a 3D environment powered by the NiceGUI Python library. Within this virtual world, several critical elements can be observed:

- The **navigational graph**, represents the environment as nodes and edges, where green nodes indicate frontiers, red nodes mark visited locations, and grey nodes are unknown (not yet discovered). The edges represent traversable paths, accounting for obstacles, enabling the robots to plan accurate routes based on actual distances.
- The **planned paths** for each robot as they move towards their assigned tasks.
- **3D models** of Boston Dynamics Spot robots were chosen for visualization, allowing users to track robot movements and evaluate task coordination in real-time (see Fig. 16).

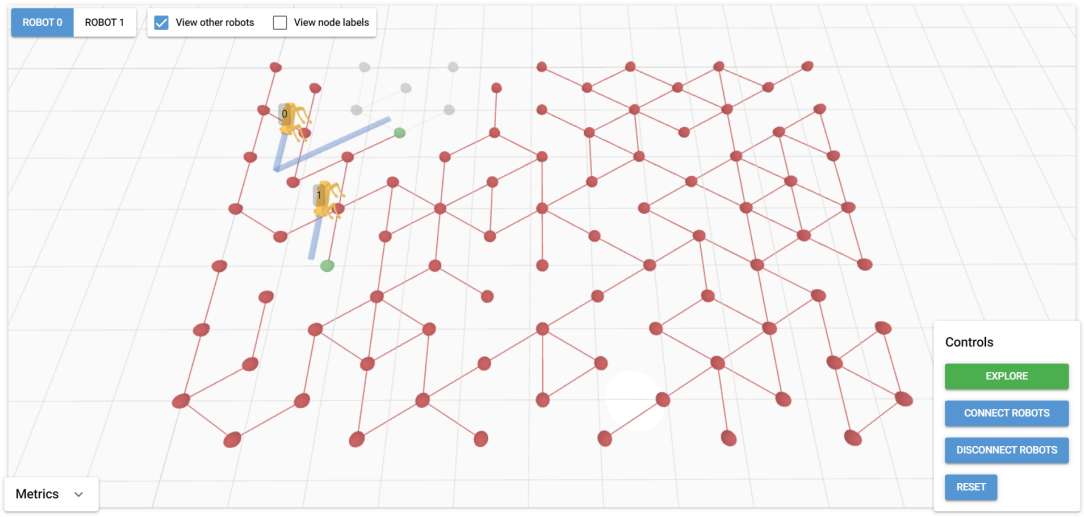


Fig. 15: The developed simulator interface.

As shown in Fig. 15, users can select the view of any robot in the system. This feature allows them to observe the world from the robot's perspective, including the areas it has visited and its current task assignments. Because the system is fully distributed, the robot only knows about the environment based on its exploration and the information shared with it by other robots. Additionally, a checkbox enables the display of other robots' positions, offering a broader perspective of the entire system. For debugging purposes, another checkbox can be activated to show the node labels on the navigational graph, which are critical for the task allocation process.

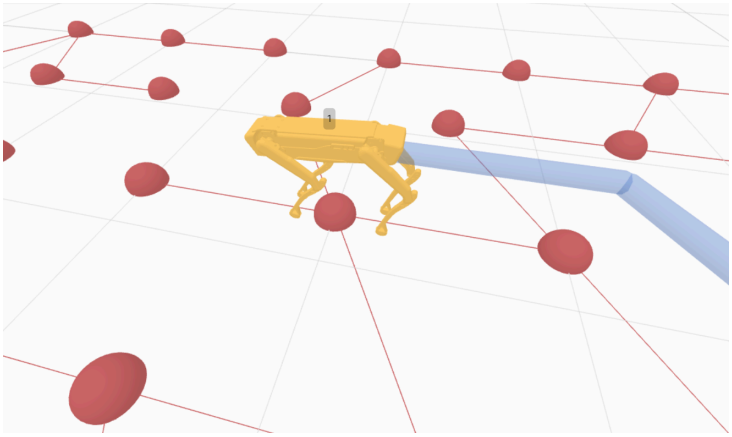
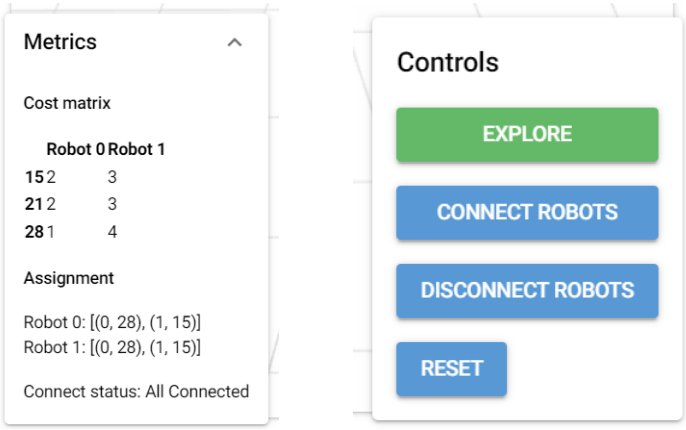


Fig. 16: 3D model of the Boston Dynamics Spot robot integrated into the simulator.

The interface also provides several control buttons that allow users to interact with the simulation:

- The **Explore button** advances the simulation by one timestep. During each step, robots communicate (if connected to others), share map data, and perform task allocation. The robots then execute a GBP iteration, moving one node closer to their assigned tasks.
- The **Connect button** enables communication between robots, allowing them to exchange map information and update their task allocations accordingly. This ensures that the robots can collaborate effectively and avoid redundant tasks.
- The **Reset button** clears the memory of all robots, forcing them to rediscover the environment from scratch. This is useful for testing and debugging new task allocation or pathfinding strategies.



(a) Metrics panel (b) Control panel

Fig. 17: (a) Metrics Panel: This panel displays the cost matrix, showing the costs for all robots to all tasks, along with the actual task assignments calculated by each robot. (b) Control Panel: This panel contains buttons to control the simulator)

The bottom left corner of the simulator displays various metrics about the robots and their decision-making processes. Users can see the **cost matrix** for the selected robot, which shows the costs of different task assignments based on the robot's position and capabilities. This area also shows the final **task allocation** and the **connection status** of the selected robot. These metrics provide detailed insights into how decisions are being made and how coordination is achieved across the system.