

Evolutionary-based Coordination of Multi-Robot Systems with Dynamic Constraints

A Technical Report submitted to the Department of Systems and Information Engineering

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Vihar Shah

Spring 2024

Technical Project Team Members

Matthew Heeter

Jose Vallarino

Patrick Sherman

Lauren Bramblett

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Nicola Bezzo, Department of Systems and Information Engineering

Evolutionary-based Coordination of Multi-Robot Systems with Dynamic Constraints

Vihar Shah*, Matthew Heeter*, Jose Vallarino*, Patrick Sherman†, Lauren Bramblett* and Nicola Bezzo*†

*Department of Systems & Information Engineering

†Department of Electrical & Computer Engineering

University of Virginia, Charlottesville, Virginia 22903

Email: {qbq5nv,mnh8eq,jvv3urx,ukw4tc,qbr5kx,nb6be}@virginia.edu

Abstract—Large-scale manufacturing facilities and power plants comprise numerous subsystems that require consistent monitoring and inspection to ensure reliable and secure operations. The use of multiple robotic systems is an expanding, robust, and cost-effective solution for such operations. To enable automated infrastructure monitoring, there are many challenges, such as path planning and task allocation for multiple robots and dynamic constraints. We formulate such challenges as a multiple Traveling Salesman Problem (mTSP) and propose an efficient task assignment solution that uses an evolutionary algorithm considering priority, energy, and time constraints. Our solution incorporates: a digital twin of a monitored environment, A* and artificial potential field methods for path planning and navigation, and human-in-the-loop prioritization. We validate the proposed method with state-of-the-art simulations in Gazebo/ROS for unmanned ground vehicles (UGVs) in cluttered environments.

Index Terms—multiple traveling salesman problem, human-in-the-loop, digital twin, unmanned vehicles

I. INTRODUCTION

For the longevity of an industrial system or facility, it is essential to regularly inspect and monitor various subsystems to ensure their reliability and safety in operation. Traditionally, inspection and monitoring duties have been carried out by humans, who are often required to navigate hazardous conditions, exposing themselves to the possibility of personal harm. Human inspectors may also encounter problems with the efficiency of task allocation and execution without sophisticated allocation algorithms and methodical execution. The advent of digital twin technology presents an opportunity in this sector to recreate complex environments and account for complex constraints, respectively. These developments allow for a significant transition towards automation and robotic aid, reducing the risk of humans being exposed to hazardous scenarios and enhancing the efficiency of operational procedures.

Integration of unmanned ground vehicles (UGVs) in inspection tasks as depicted in Fig.1, managed through automated assignment processes, marks a significant step towards this goal. However, orchestrating such a system involves navigating a complex array of challenges, including efficient

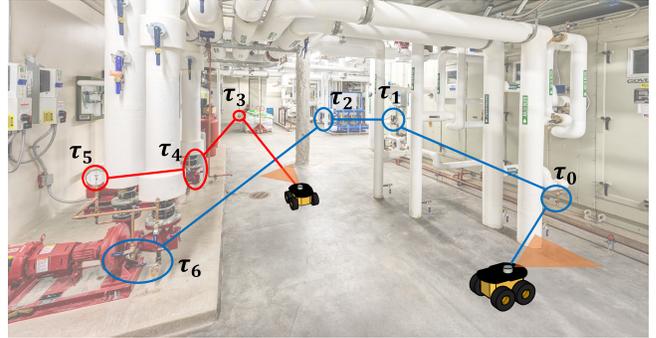


Fig. 1. Pictorial depiction of the problem presented in the paper. A team of ground robots is required to distribute and execute tasks, all while managing constraints related to the environment and the tasks themselves.

path planning between multiple robots and tasks, prioritizing emergency response, and optimizing energy consumption. This paper delves into these challenges, formulating them within the framework of the multiple Traveling Salesman Problem (mTSP). We propose a novel task assignment solution, utilizing an evolutionary algorithm that balances considerations of priority, energy, and time constraints.

Our approach leverages the capabilities of a digital twin to accurately represent any environment of interest and employs A* and artificial potential field methods for optimal path planning and subsequent navigation. Inclusion of human-in-the-loop prioritization further enhances the system's flexibility and responsiveness to dynamically changing operational priorities. By validating our method through simulations in Gazebo-ROS, we demonstrate its effectiveness in managing UGVs within realistic and cluttered environments. Fig.1 shows pictorially the intent of this work in which tasks are optimally distributed among two robots that are deployed in mechanical rooms to monitor valves and gauges.

This work makes several key contributions to the field of robotic systems for infrastructure inspection. First, we introduce an advanced task assignment methodology that integrates priority, energy, and temporal considerations, tailored for the coordinated operation of multiple robotic agents. Second, our use of digital twin technology in conjunction with path planning algorithms allows adaptive operational planning in real time. Lastly, we validate a comprehensive

The code repository for this approach and additional explanations are available at https://github.com/UVA-BezzoRobotics-AMRLab/multi_jackal_amcl.

system that not only proposes a theoretical framework, but also demonstrates practical applicability through physics-based simulations.

The rest of the paper is organized as follows: in Section II we provide an overview of the state of the art in multi-robot inspection and digital twin technologies. In Section III we provide an overview of our methods. In Section IV we describe the proposed solution, the inclusion of human operator constraints, and robot control. Section V addresses the realistic environments through a simulation study of our solution. Lastly, Section VI summarizes our research and discusses future work.

II. RELATED WORK

The development of unmanned systems for navigating complex industrial environments leverages diverse technological innovations. [1] introduces a human decision-making behavior model that significantly enhances the operational decision-making within multi-robot systems by integrating human cognitive processes with robotic control. This approach improves human-robot collaboration, optimizing intervention timing based on human insights. While this model seeks to augment human control within robotic operations, our project extends this concept by enhancing the autonomous decision-making capabilities of robots. We aim to reduce the necessity for human intervention by the use of digital twin and advanced algorithms, which allow UGVs to independently adapt and respond to dynamic environments, increasing operational efficiency and reducing human intervention. Meanwhile, [2] discuss the optimization of navigation paths through an energy-aware control framework that manages the energy efficiency of heterogeneous robotic teams. Our project enhances these methods by incorporating a sophisticated task allocation strategy that balances efficiency, safety, and operational demands, tailored to the specific challenges of the situation at hand. Additionally, trajectory optimization in cluttered environments has been addressed by [3] and [4], who apply multi-objective optimization techniques to effectively navigate obstacles. Our approach integrates these methodologies and reintroduces the insights from [5], enhancing navigation capabilities through evolutionary algorithms that optimize paths not only for immediate task execution but also for long-term operational sustainability. Moreover, [6] focuses on multi-robot task scheduling to improve the coordination of robot teams. Unlike previous methods, our work employs real-time data to dynamically adjust task assignments and robot paths based on immediate environmental feedback, significantly enhancing the responsiveness and flexibility of robotic operations in industrial settings.

III. PROBLEM STATEMENT

In this paper, we investigate methods of robotic technologies for precise and efficient inspection of real-world industrial and complex systems. For such systems, a common goal is to find optimal task assignments that minimize the

time robots travel while avoiding obstacles and accounting for human inputs. Formally stated:

Problem: Optimal Path Planning with Obstacles and Human-Inputs: Consider a set of robots R assigned to travel to a set of tasks \mathcal{T} with known positions. Find a policy to assign tasks and plan the motion for each robot so that the longest tour of any robot represented by Q is minimized. This is considered a mTSP [7] with an objective function of:

$$\begin{aligned} \min \quad & Q \\ \text{s.t.} \quad & \sum_{i \in \mathcal{T}^\Sigma} \sum_{j \in \mathcal{T}^\Delta} \omega_{ijk} \cdot \rho_{ijk} \leq Q, \quad \forall k \in R \end{aligned} \quad (1)$$

where ω_{ijk} is the cost of traveling from task i to task j for robot $k \in R$ and ρ_{ijk} is a binary variable that defines if robot k travels from task i to task j . The robots must start and end at their initial depots, represented by the sets \mathcal{T}^Σ and \mathcal{T}^Δ . This is known as a “minMax” optimization problem, where we minimize all robots’ maximum tour cost Q .

Moreover, the policy should take into account the following constraints: 1) *priority constraints*, which places hierarchical importance on tasks; 2) *time constraints*, which factor in task completion time; and 3) *precedence constraints*, which specify the order certain tasks must be executed. The policy is executed within a given map of the environment, but local obstacle avoidance is enabled to account for small environmental changes. Our final algorithm considers all of these factors and produces optimal paths for each robot.

IV. APPROACH

Our proposed architecture to solve this problem consists of formulating an algorithm to assign a set of tasks to each robot so that the total operation time is shared and minimized. Fig. 2 illustrates a high-level overview of our approach.

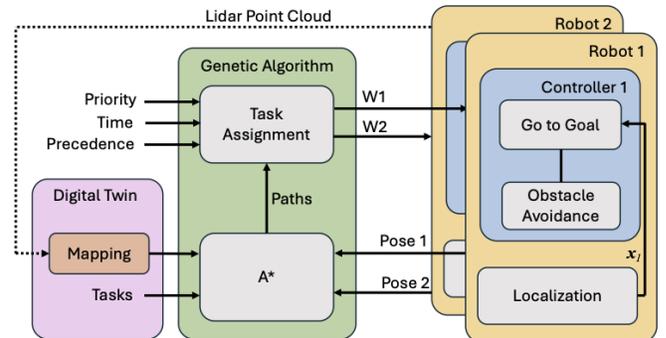


Fig. 2. Architecture Diagram

The first step of our approach is to create a digital twin representation by mapping the environment, using a robot with a scanner such as lidar, and recording the necessary tasks based on the coordinates of the map. The Genetic Algorithm (GA) block optimizes the distance traveled using A* path planning and task assignment algorithms. The paths are then sent to the robots’ controller, which uses a go-to-goal algorithm with artificial potential fields (APF) to move

the robots while avoiding obstacles along the path using robots' scanners. A robot's position x_k is estimated, using a particle filter-based technique, given that the robots do not have access to ground truth position data during mission operations. The estimated positions are communicated to the path-planning algorithm to ensure that the robots continue following the generated paths. Our algorithms are explained in more detail in the following sections.

A. Solving mTSP using Genetic Algorithm

Addressing the challenges inherent in the mTSP for an industrial environment requires algorithmic strategies that prioritize efficiency in computation and task allocation. The mTSP, an extension of the traditional traveling salesman problem, is recognized for its NP-hard classification, indicating a problem where the solution space expands exponentially with the addition of each task, which further increases when introducing multiple agents. Efficiently solving the mTSP consists of two main steps: (1) Using the A* algorithm to pre-process the fastest path calculations between any starting position and city or any city to a different city (2) Using a GA to allocate tasks to agents using the path calculations.

1) *A* Path-Planning*: The A* algorithm [8] is used to find the shortest path between each task and the robots' start and final position. Given that a twin digital map of the environment is available, we use a cost function to determine optimal paths. The cost function is the sum of the actual cost associated with the distance traveled and a heuristic cost, which in our case is the Manhattan distance from one point to the next. Fig. 3(a) shows an example of all optimal paths between tasks and robots.

2) *Task Allocation*: Our research introduces a novel approach to addressing the mTSP with additional priority, task completion time, and precedence constraints through the adaptation of a GA, a computational technique that emulates the evolutionary process of natural selection. Within this framework, tasks are analogized to "cities", and autonomous robots to "agents", thereby framing the problem as one of distributing a set of tasks (cities) among multiple robots (agents) in an optimal sequence. The core mechanism of our GA involves the generation of an initial population comprising random sequences of task allocations across the robots. These sequences are evaluated based on a fitness criterion (ω_k), which, in our context, is defined by the cost of task distribution. Specifically, the total task cost for a robot required to complete the most time-intensive task sequence (2). The general solution for the total cost is calculated by taking the sum of the A* distances between the tasks in the path divided by robot velocities. Sequences that facilitate a balanced but efficient distribution of tasks are deemed to possess better (lower) fitness levels and survive.

$$C = \max(\omega_k) \quad \forall k \in R \quad (2)$$

The selection process within our GA adopts a tournament-style evaluation, where random subsets of sequences (C_1, C_2)

face one another to identify the most efficient allocations (minimum fitness), which are then preserved as selected parents (s) for the current generation (3).

$$s = \min(C_1, C_2) \quad (3)$$

Offspring sequences are generated through an "order crossover" mechanism, which integrates random segments of the parent sequences into the offspring, ensuring efficient task distribution patterns are inherited. Additionally, to introduce variability and potential for discovering more efficient solutions, a mutation process is randomly applied to the offspring sequences, swapping two tasks within a path. This iterative process of selection, crossover, and mutation continues across a predefined number of generations, leading to a set of paths with the best fitness or task distribution efficiency shown in Fig. 3(b). The result is an adaptable solution to the mTSP that is well-suited to the dynamic and time-sensitive demands of industrial environments.

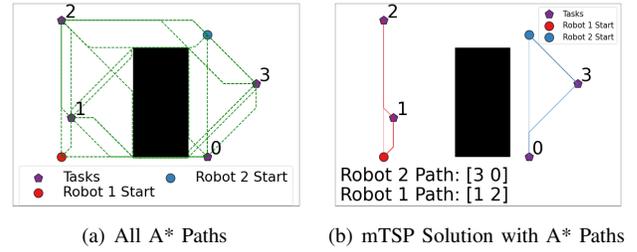


Fig. 3. GA Task Allocation

B. Human-Input Constraints

In addition to the GA which provides a solution to the mTSP, we introduce operator-based constraints to influence the task allocation by modifying the GA's functions and our evaluation of fitness. In the following subsections, we will use a hospital example to motivate priority, task time, and precedence constraints alongside figures of the new paths: τ_0 is delivering paperwork to a nurse, τ_1 is picking up paperwork, τ_2 is a vital medicine delivery, and τ_3 is taking a picture of a gauge.

1) *Priorities*: The first of such constraints allows the operator to prioritize a task at predefined levels. For ease, we demonstrate with 4 priority levels (0=no priority, 1=important task, 2=critical task, 3=emergency). The goal is to have a higher priority task visited by *any* robot earlier than otherwise. In the hospital, consider τ_0 and τ_2 to have priorities of 3. With no priority, ω_{ijk} is calculated by the A* distance divided by velocity from one task to another. To include this priority, we revise the evaluation of cost, ω_{ijk} , by modifying its calculation (5), based on priority, γ , and how far the robot travels, p , before completing that task:

$$p = \omega_{ijk} + \sum_{\forall \tau_i, \tau_j \in \mathcal{T}_k}^i \omega_{\tau_i \tau_j k} \quad (4)$$

We inflate such cost, ω_{ijk} , for any priority, $\gamma_j > 0$ by:

$$\omega_{ijk} = f_\gamma(p) = \gamma_j p \quad (5)$$

Where $f_\gamma(p)$ is a function of priority and $\tau_i, \tau_j \in \mathcal{T}_k$ represent the robot's tasks that take place before ω_{ij} . For ease, if $\gamma_j = 0$, ω_{ijk} is not revised, otherwise our function, $f_\gamma(p)$, is linear. The function of priority, $f_\gamma(p)$, can be set by the operator. Since "priority" and "emergency" are subjective terms, the function allows custom configurations for the growth of change in cost based on priority. As a brief example, a more aggressive prioritization could have $f_\gamma(p) = p^{\gamma_j}$ or a logistic function. This formulation increases the perceived travel duration the robot undertakes to reach a prioritized task as its position in the path is further away - resulting in poorer fitness. As a result of the priorities to τ_0 and τ_2 , the agents' path orders have swapped (Fig. 4(a)), with respect to the general solution (Fig. 3(b)). Furthermore, this methodology of handling priority constraints allows the robot to consider performing low-priority tasks before high-priority ones if the detour is negligible. In Fig. 4(a), Robot 1 completes τ_2 before τ_1 and Robot 2 completes task τ_0 before τ_3 . The fitness of both paths can be justified by the strategy of prioritizing tasks (5). Consider τ_1 , which is on the way to τ_2 but is overlooked. Given the absence of task completion times, the GA prefers to tackle τ_2 first, however, it should ideally finish τ_1 during the journey if $\gamma_1 > 0$. Fig. 4(b) explores this scenario with $\gamma_1 = 1$. This result meets our expectations as τ_1 is visited before τ_2 . Although there is a small increase in distance traveled on the way to τ_2 , the fitness of this tour is higher since the robot takes a small detour to the high-priority task and minimizes the weighted objective of total path distance and priority preferences.

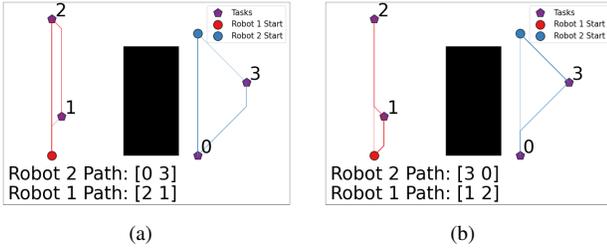


Fig. 4. Example of priority constraints. In (a), the priorities for τ_0 and τ_2 increase so that $\gamma_0 = \gamma_2 = 3$. In (b), in addition to the increase in priorities, we increase the priority of τ_1 so that $\gamma_1 = 1$.

2) *Time*: Similarly, a time constraint allows the operator to input the time it takes to complete each task. This impacts the GA by varying the amount of time, t_j , a robot spends on each destination task, j , and could otherwise be traveling. By representing this trade-off as an increased cost to the next task, ω_{ijk} (5) is updated and formulated as:

$$\omega_{ijk} = \omega_{ijk} + t_j \quad (6)$$

This results in an increase to the cost before being evaluated for priority and eventually for fitness. Focusing on the observations in Fig. 4(b), it is unrealistic and potentially unwanted to expect a robot to perform a low-priority task on

the way to an emergency task. The example held task times to 0 seconds, purely evaluating priority. Consider the example in Fig. 4(b) with the addition of a 3-minute task completion time to τ_1 , $t_1 = 180$. The result shown in Fig. 5(a) shows the algorithm returns to favor completing the emergency at τ_2 before the more time-consuming, lower priority τ_1 . As a separate example, consider no task priorities where $t_0 = 300$. This modifies the standard solution in Fig. 3(b) to share task work more efficiently as seen in Fig. 5(b) where Robot 1 handles all tasks except the time-consuming τ_0 .

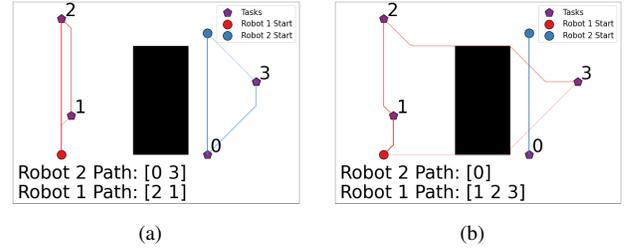


Fig. 5. Example of time constraints. As shown, the mTSP solution from Fig. 4(b) changes when the time to complete τ_1 is increased to 180s shown in (a) and in (b) where the time to complete τ_0 is increased to 300s.

3) *Precedence*: Lastly, the operator can specify if a robot needs to visit any task before a specified task in the form of precedence constraints. This is accomplished by modifying three fundamental functions in the GA. First, *generateUniquePaths* for the initial population is tailored to consist of paths that follow precedence but allow randomness outside the constraint. Next, *orderCrossover* is modified to pass on successful parent path sequences only if the crossover section selected does not overlap with a precedence task. Lastly, the *repairChild* function, responsible for handling duplicate and missing tasks, is modified to include a check to ensure that precedence constraints are satisfied. If they are not satisfied, the tasks are swapped within a robot's path or between robots. This is generated in the GA to evaluate paths that meet these precedence constraints and swap any two cities that are in the incorrect order as needed by the constraint. Note that mutations are not allowed. Since the GA relies on randomization to be effective, this methodology of handling precedence constraints allows randomization until it breaks our constraint by generating valid random paths for the initial population and promoting order crossovers among non-precedence cities. Fig. 6(a) is an extension of the general solution in Fig. 3(b) but requires a single robot to pick up the paperwork before delivering it to the nurse (τ_1 before τ_0) and must deliver it to the nurse in order to deliver the medicine (τ_0 before τ_2). Fig. 6(b) shows another example in which τ_3 must be completed before task τ_2 .

The resulting paths satisfy the precedence constraints by generating valid paths, avoiding invalid crossovers, repairing invalid paths, and preventing mutation. The logic in the first function's modifications is seen in Alg. 1. The *Generate Unique Paths* algorithm is modified from its original form by initializing a population of precedence valid paths. The set, *precedence* contains sets of precedence cities, α , or lack

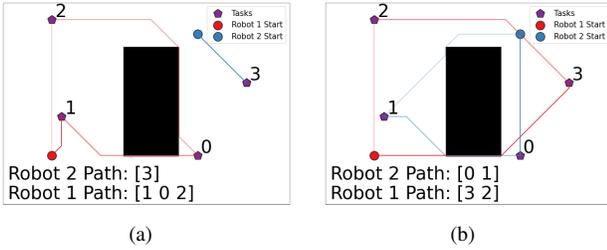


Fig. 6. Example of precedence constraints. The mTSP solution changes when τ_1 must occur before τ_0 in (a) and when τ_3 must occur before τ_2 in (b).

thereof, for each city. Note line 6 in the algorithm references $precedence_i$, or the set of precedence cities for city i . The $addAvailableSlot$ procedure finds an open slot in the path for the same agent as the other precedence tasks and adds the task at that location, the $rand$ function chooses a random member from the set, and the $remove$ procedure takes a task away from $remainingCities$ to satisfy the $while$ condition in line 3 of the algorithm:

Algorithm 1 Generate Unique Paths

Require: $numCities, numAgents$

Require: $precedence$

- 1: $paths \equiv \emptyset$
 - 2: $remainingCities \leftarrow \{0, \dots, numCities - 1\}$
 - 3: **while** $remainingCities$ is not empty **do**
 - 4: $i = rand(remainingCities)$
 - 5: **if** $precedence_i \neq \emptyset$ **then**
 - 6: **for each** $\alpha \in precedence_i$ **do**
 - 7: $addAvailableSlot(paths, \alpha)$
 - 8: $remove(remainingCities, \alpha)$
 - 9: $addAvailableSlot(paths, i)$
 - 10: $remove(remainingCities, i)$
-

C. Robot Control

The next step is to have the robots follow their assigned paths. This approach has two main components: localization using Adaptive Monte Carlo Localization (AMCL) and control using Artificial Potential Fields (APF).

1) *Localization*: Localization is a critical component for the robots to follow their assigned paths, as it allows the robots to understand where they are in relation to the environment. This is implemented using a probabilistic localization technique, AMCL [9]. It utilizes sensor data, such as lidar scans, to estimate the robots' positions within the given map. The process involves generating a set of particles representing possible robot poses, weighting them based on sensor data likelihood, resampling to emphasize high-weighted particles, and updating the robots' pose estimate iteratively as new data arrive. AMCL refines the robots' position estimates by repeatedly comparing sensor data with the map, adjusting the particle cloud, and converging towards the most likely robot pose within the environment. Determining the robots' positions is essential for safe and efficient navigation and interaction with obstacles.

2) *APF Controller*: As mentioned in Section IV, the mTSP utilizes estimations of initial positions but AMCL helps provide dynamic updates to robot position. Now that the robots' positions x_k are known, the next component is to control the robots to go to their assigned tasks. We coordinate the movement of each robot using an artificial potential field (APF) method. The idea is that any object placed at any point in this potential field desires to move to a position of lower potential. This can be calculated in a vector field based on the gradient of the potential function for every point q in the field at a particular time t :

$$\nabla U(\mathbf{q}) = \left[\frac{\partial U}{\partial q_1}(\mathbf{q}), \dots, \frac{\partial U}{\partial q_m}(\mathbf{q}) \right]' \quad (7)$$

A robot's velocities at a point q is calculated as the negative gradient of the potential field or the sum of all forces acting on the field and leverages three main objectives: 1) go-to-goal represented as an attractive field, 2) collision avoidance between robots, and 3) obstacle avoidance represented as repulsive fields. In this method, the total force acting on the k^{th} robot is generally formulated as:

$$\mathbf{F}(\mathbf{x}_k) = -\nabla U(\mathbf{x}_k) = \beta_1 F_k^1 + \beta_2 F_k^2 + \beta_3 F_k^3 \quad (8)$$

considering β_n is a weighting coefficient for force F_k^n where each F_k^n corresponds to the k^{th} objective listed previously. For our approach, this translates to a potential field generated by combining repulsive obstacle and robot avoidance fields with an attractive go-to-goal field.

V. CASE STUDIES

Our proposed architecture was validated in simulation using Gazebo and RViz with a digital twin map of a complex environment depicted in Fig. 7. Gazebo experiments allow us to implement realistic and high-fidelity experiments. The simulation uses two robots that are tasked to travel to five tasks located throughout the map. We set up three simulation trials to verify each constraint: 1) Priority, 2) Time, 3) Precedence. The goals and the paths that the robots are assigned to are visualized in RViz, seen in Fig. 7(b). This example has no constraints.

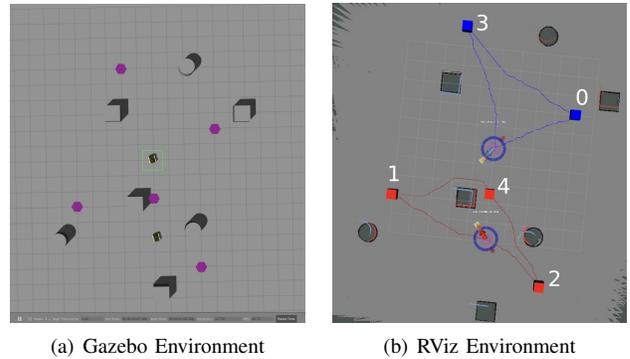


Fig. 7. Simulation Set-Up

The robots will begin to move once the A* paths are allocated. The simulation finishes once all tasks have been

met and the robots are at their starting positions. These simulations provide us with crucial insight in how the robots travel to the tasks and if they are following the correct paths while avoiding obstacles.

Priority Constraint Results: The first trial consists of changing the priority constraints while omitting all other constraints. The priority assignment is as follows: $\gamma_2, \gamma_4 = 3$, indicating an emergency, while $\gamma_0, \gamma_1, \gamma_3 = 0$. The paths the robots follow can be seen in Fig. 8(a). Cities with higher priorities are visited first while maintaining an efficient solution: robot 1 travels to τ_4 first, followed by τ_0 and τ_3 , and robot 2 travels to τ_2 then τ_1 .

Time Constraint Results: The next trial explores changing the task completion time while setting other constraints to have no impact. The time assignment is as follows: $t_0, t_1, t_2, t_3 = 20$ and $t_4 = 60$. The results of this trial can be seen in Fig. 8(b), where robot 1 is assigned to travel to τ_3 , τ_0 , then τ_1 and robot 2 travels to τ_2 then τ_4 . As seen in the difference between Fig. 7(b) and 8(b), Robot 2 is allocated a path that takes the long completion time of τ_4 into account to maximize the overall task efficiency of all robots.

Precedence Constraint Results: The precedence trial explores requiring order to task completion for any robot considering completing that set of tasks. The precedence constraints are as follows: τ_1 must be done before τ_2 , and τ_0 must be done before τ_4 . The results of this trial can be seen in Fig. 8(c), robot 1 travels to τ_3 , τ_0 , then τ_4 , and robot 2 travels to τ_1 then τ_2 . This leads to a completely different solution than the general solution in Fig. 7 by moving τ_4 to Robot 1's path.

Mixed Constraint Results: The mixed constraint trial explores a combination of the constraints to simulate realistic conditions. The constraints are as follows: τ_1 before τ_3 ; $\gamma_0 = 3, \gamma_2 = 1$; and, $t_4 = 30$ seconds. The results of this trial can be seen in Fig. 8(d), robot 1 travels to τ_0 , τ_1 then τ_3 , and robot 2 travels to τ_2 then τ_4 .

The solution utilizes each robot to handle each priority task first and divides the precedence tasks and time-consuming task amongst the nearest robots from the priority tasks.

VI. CONCLUSION AND FUTURE WORK

In this work, we have presented a framework for efficient task planning and allocation for multiple robots and complex constraints. To solve this mTSP we have used a genetic algorithm strategy with user-input constraints, including priority, time, and precedence. Once a solution is obtained, the robots follow the assigned paths using an artificial potential field method. A digital twin representation of the environment is used to create paths in a realistic environment for simulation.

Possible future extensions of this work include: adding additional constraints such as time of day for completion, heterogeneous robots with different capabilities, and the inclusion of humans performing tasks alongside the robots. Experiments with real robots in a real-world environment are also recommended to ensure that our framework is robust for real-world applications.

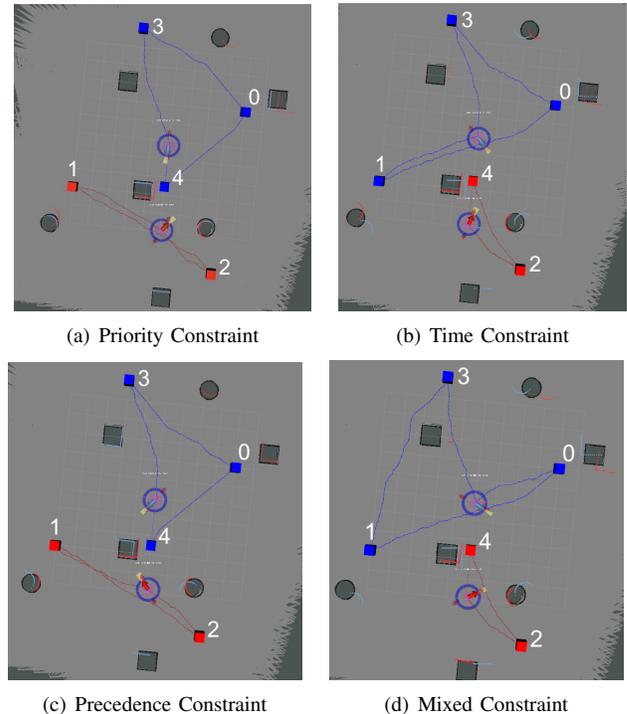


Fig. 8. Gazebo Simulation Results from RViz

ACKNOWLEDGMENT

This work is sponsored by EnterAR and we thank them and Siemens for their technical assistance and useful discussions throughout this project.

REFERENCES

- [1] J. Huang, W. Wu, Z. Zhang, and Y. Chen, "A human decision-making behavior model for human-robot interaction in multi-robot systems," *IEEE Access*, vol. 8, pp. 197 853–197 862, 2020.
- [2] T. X. Lin, E. Yel, and N. Bezzo, "Energy-aware persistent control of heterogeneous robotic systems," in *2018 Annual American Control Conference (ACC)*. IEEE, 2018, pp. 2782–2787.
- [3] M. E. A. Boudjellel and T. Chettibi, "Optimal trajectory planning for a mobile robot in presence of obstacles using multi-objective optimization techniques," in *2016 8th International Conference on Modelling, Identification and Control (ICMIC)*. IEEE, 2016, pp. 509–514.
- [4] I. Thammachantuek and M. Ketcham, "Path planning for autonomous mobile robots using multi-objective evolutionary particle swarm optimization," *Plos one*, vol. 17, no. 8, p. e0271924, 2022.
- [5] W. Ghadir, J. Habibi, A. G. Aghdam, and Y. Zhang, "Time-efficient trajectory optimization in patrolling problems with non-prespecified depots and robots," in *2016 24th Mediterranean Conference on Control and Automation (MED)*. IEEE, 2016, pp. 1047–1052.
- [6] Y. Zhang and L. E. Parker, "Multi-robot task scheduling," in *2013 IEEE international conference on robotics and automation*. IEEE, 2013, pp. 2992–2998.
- [7] L. Bramblett, B. Miloradovic, P. Sherman, A. V. Papadopoulos, and N. Bezzo, "Robust online epistemic replanning of multi-robot missions," *arXiv preprint arXiv:2403.00641*, 2024.
- [8] M. Juliá, A. Gil, L. Payá, and O. Reinoso, "Local minima detection in potential field based cooperative multi-robot exploration," *International Journal of Factory Automation, Robotics and Soft Computing*, vol. 3, 2008.
- [9] L. P. Matias, T. C. Santos, D. F. Wolf, and J. R. Souza, "Path planning and autonomous navigation using amcl and ad," in *2015 12th Latin American Robotics Symposium and 2015 3rd Brazilian Symposium on Robotics (LARS-SBR)*. IEEE, 2015, pp. 320–324.