2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Accounting for Travel Time and Arrival Time Coordination During Task Allocations in Legged-Robot Teams

Shengqiang Chen, Yiyu Chen, Ronak Jain, Xiaopan Zhang, Quan Nguyen, and Satyandra K. Gupta

***Abstract*— Many applications require the deployment of legged-robot teams to effectively and efficiently carry out missions. The use of multiple robots allows tasks to be executed concurrently, expediting mission completion. It also enhances resilience by enabling task transfer in case of a robot failure. This paper presents a formulation based on Mixed Integer Linear Programming (MILP) for allocating tasks to robots by taking into account travel time and ensuring efficient execution of collaborative tasks. We extended the MILP formulation to account for complexities with legged robot teams. Our results demonstrate that this approach leads to improved performance in terms of the makespan of the mission. We demonstrate the usefulness of this approach using a case study involving the disinfection of a building consisting of multiple rooms.**

I. Introduction

Mobile robots are emerging as a useful tool in many applications that require tedious work or pose a threat to human health. Mobile robots, especially wheeled robots are adequate for many applications such as hospitals for disinfection, warehouses for managing shelves, and grocery stores for cleaning [1]–[3]. However, some applications require robots to traverse over challenging terrain and wheeled robots are not adequate for such applications. In the last decade, there have been significant advancements in developing legged robots in both hardware design and control algorithms, realizing robust and agile locomotion in a wide variety of challenging terrains. Therefore, legged robots will be much more suitable for applications such as disinfection of cluttered spaces, pesticide spray, and inspection of construction sites [4].

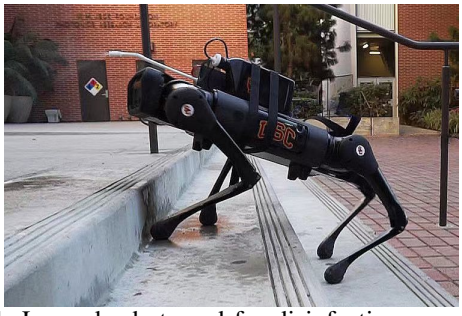


Fig. 1: Legged robot used for disinfection case study

Many applications require the deployment of robot teams to effectively and efficiently carry out missions [5]. The use of multiple robots allows tasks to be executed concurrently, expediting mission completion. It also enhances resilience by enabling task transfer in case of team member failures. Deploying multi-robot teams necessitates solving task allocation problems that consider each team member's capabilities and precedence constraints among tasks. While most missions involve human supervisors, relying solely on them for task allocation can result in unnecessary mission delays. Hence, there is a need for automated task assignment methods to ensure efficiency. These methods must be swift to facilitate rapid replanning in contingency situations. MILPs have proven to be valuable for solving task allocation problems in multi-robot teams (see Section IV for details). Most existing MILP formulations focus on minimizing span time by considering only task execution durations. However, many complex applications introduce additional constraints, including travel time, real time delays, differences in robot capabilities, and support for collaborative tasks. This paper extends the traditional MILP formulation to address mission complexities encountered by legged robot teams.

In missions posing human risks, robots may be dispatched to a location potentially distant from task sites, necessitating travel to and from these locations to perform the tasks. Robots may also need to travel between task locations. In many applications, travel durations represent a significant portion of the task durations. Hence, task allocations must account for travel times. Additionally, certain tasks demand collaboration among multiple robots. Therefore, the task allocation method needs to ensure that each task is allocated an adequate number of robots to carry out tasks. Collaborative tasks can commence only when all required agents are present at the task site. If agents are allowed to partially complete tasks and travel to collaborative tasks, then unnecessary waiting can be reduced by better coordination of arrival time. However, this will require additional travel for the robot to go back to the partially completed task to finish it.

This paper presents a MILP formulation for allocating tasks in legged-robot teams while accounting for travel time. Our focus is on applications with known maps and tasks, including support for collaborative tasks requiring multiple robots. Additionally, our method allows for partial task completions to prevent unnecessary delays in collaborative tasks, facilitating agents' synchronized arrival. We introduce a formulation that operates efficiently for teams of up to ten robots, making it suitable for rapid replanning in cases of agent failure or task execution delays. We evaluate the proposed approach using simulations and illustrate how features of our algorithms lead to superior performance in building disinfection applications (see Figure 1 for the robot

S. Chen, Y. Chen, R. Jain, X. Zhang, Q. Nguyen and S. K. Gupta are with the Department of Aerospace and Mechanical Engineering, University of Southern California, Los Angeles, CA 90089, email: {schen800, yiyuc, ronakjai, xiaopanz, quann, guptask}@usc.edu

used in a prior study involving physical experiments [4]).

II. Related Work

In this paper, we focus on addressing the challenges posed by the Multi-Robot Task Allocation (MRTA) problem. MRTA describes the problem of deciding which robot should execute a given task. MRTA aims to coordinate task assignments in the robot team to complete a set of tasks with specific constraints [6], [7]. These encompass a diverse array of scenarios, including but not limited to the Multiple Traveling Salesman Problem [8], [9], Vehicle Routing Problem [10]–[12], and Job Shop Scheduling Problem [13]–[15]. MRTA problems are broadly categorized into three fundamental types: offline assignment, iteration assignment, and online assignment [16]. In our investigation, we put forth a novel approach that pertains to the offline assignment by accounting for travel time and arrival time coordination.

Approaches related to task assignment, and scheduling in the context of multi-robot task allocation encompass a wide variety of techniques, including Hungarian Algorithms(HA) [17], Mixed Integer Linear Programming (MILP) [18]–[21] and Bio-inspired methods [22], [23]. Recent work has shown that using MILP formulation and solver is a promising approach for solving task allocation problems with complex constraints such as temporal, spatial [24], [25], and precedence constraints [26]–[29]. Previous works shows that MILP is efficient in solving MRTA problems with complex constraints, because a salient advantage of harnessing MILP solvers lies in the availability of robust and efficient open-source solvers [30], [31], which expedite the attainment of desired solutions. We extend previous MILP formulations to handle the requirements of complex missions executed by legged robot teams.

Legged robots offer a distinct set of mobility advantages, rendering them exceptionally well-suited for deployment across a diverse spectrum of demanding environments, as corroborated by prior research [32]–[37]. Their inherent capabilities encompass proficient surveying and inspection, as well as efficient transportation [38]. Legged robots offer agile navigation over rugged terrains, versatile payload integration (e.g., cameras, sensors, end effectors), and enhanced maneuverability in tight spaces compared to wheeled robots. The method presented in this paper enables robots to collaborate on complex tasks.

III. Problem Formulation

A. Terminology

Suppose R be the set of l agents with different capability and mobility, defined as $R = \{R_1, R_2, ..., R_l\}$. Let A be the set of m tasks to be processed, defined as $A = \{A_1, A_2, ..., A_m\}$. Based on the capability of each agent, some types of tasks can only be processed by certain agents. We partition the tasks into n task types, for example, each exploration mapping task is partitioned in the same type, while object removal is considered as another type. The task type list is denoted as $T = \{T_1, T_2, ..., T_n\}$. Each location coordinates are defined by (x_i, y_i) on a working plane.

In a multi-robot collaborative scenario involving agents R and assigned tasks A, as previously described, we assume access to an initial grid sketch and agent status. The task involves multiple agents processing tasks scattered across the grid. Each agent can handle only one task at a time. Additionally, there are multi-agent collaboration tasks requiring at least two agents to collaborate, necessitating simultaneous start, shared paths, and concurrent completion.

Given the restrictions above, we consider the following agent parameters for $R_i \in R$:

1) **Dispatch location:** $O_i \in \mathbb{R}^3$ where the agent is spawned, e.g., centroid of the robot deployed on the map;

2) **Collection location:** $E_i \in \mathbb{R}^3$ where the agent is designed to stop, e.g., centroid of the robot when collected by the truck;

3) **Estimated travel velocity:** $v_i \in \mathbb{R}^+$ representing the velocity of the agent while traveling inside the grid;

4) **Task type capability:** T_i which notes the set of task types that this agent is capable of performing.

We also need to consider the following task parameters for $A_i \in A$:

1) **Estimated task start location:** $S_i \in \mathbb{R}^3$ where the task is estimated to start spatially, e.g., centroid of the obstacle that needs to be removed;

2) **Estimated task finish location:** $F_i \in \mathbb{R}^3$ where the task is estimated to start spatially, e.g., centroid of the obstacle designated destination;

3) **Estimated execution time:** $\Delta_{i,j} \in \mathbb{R}^+$ representing time required to execute task A_i by agent R_j, $\forall R_j \in R$. This parameter can be set to an arbitrarily large number M if the agent is not capable of processing the task, for example, a robot with a camera is not suitable for moving large obstacles.

4) **Number of agents required:** $m_i \in \mathbb{N}^+$ describes the agent needed for execute the task, which is at least 1 for individual tasks and more for collaboration tasks.

5) **Task type:** T_i which notes the exact task type in which this task is categorized in.

For the scenario setup, traveling time is a major consideration of task planning. In order to take care of various types of traveling costs, the cost map is defined to consider the traveling expenses. All agents are expected to have their own travel cost on the grid, plus the start cost and end cost which describes the operation of initializing the agent or shutting the agent down.

1) **Travel cost** $C_{ij} \in \mathbb{R}^+$ includes three parts: travel between tasks, travel from spawn location to task, and travel from last task to end location. C_{ij} represents the time cost when traveling from travel from finish location of task F_j to the start location of task S_i $\forall A_i, A_j \in A$, or the agent spawn location of agent O_j to the first task start location S_i $\forall A_i \in A, R_j \in R$, or the last task finish location of task F_j to the agent end location E_i $\forall R_i \in R, A_j \in A$ respectively. It is derived from the grid location actual distance and preset agent travel velocity.

2) **Start cost** $S_i \in \mathbb{R}^+$ represents time cost for anything related to start agent R_i, e.g. initializing the agent and letting

it be ready to move. It is coded as the very first task that all agents should process.

3) **End cost** $E_i \in \mathbb{R}^+$ represents time cost for anything related to shut down agent R_i, e.g. putting the robot back into sleep mode.

Furthermore, tasks may come with precedence constraints, specifying their prerequisites. These constraints are represented by a hierarchical task network (HTN), organized into layers, and stored in a YAML file. Each task belongs to one of three categories: sequential, parallel, or atomic.

B. Problem Formulation

Consider a multi-robot collaborative scenario, where the agents R and tasks A are as defined above. The start time of each task is denoted as $t_i \in \mathbb{N}$. The completion time of the final task is denoted as C_{max}. Binary variable X_{ij} is defined as $X_{ij} = 1$ if A_i is executed by R_j. An auxiliary binary decision variable V_{ijk} is imported to assist problem formulation. With the comprehensive map data and the information at our disposal, our objective is to ascertain optimal solutions that encompass the initiation and conclusion times of tasks, encompassing meticulous allocations of tasks to individual agents. This optimization effort is carried out under strict constraints with the aim of minimizing the overall makespan. It involves considering travel durations, startup delays, and each agent's specific capabilities in handling their tasks.

IV. Approach

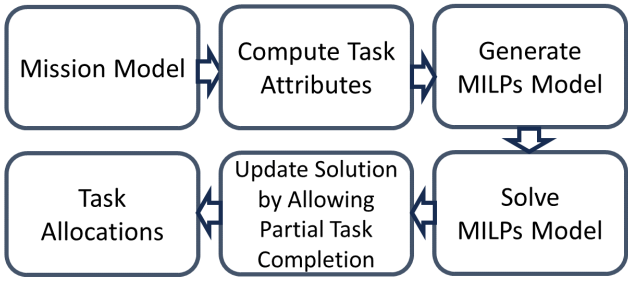


Fig. 2: Overview of the proposed approach

The proposed methodology, illustrated in Figure 2, begins with the mission model as input. It encompasses essential data, including grid parameters, agent specifications, and task details, including execution timelines and capabilities. This data is further enriched with the hierarchical task network model. The geographical information from the map is then used to create a grid distance model, accounting for all possible routes and their distances. This model enables the calculation of travel durations between any two locations, considering agents' dynamic movement speeds and employing the A* path planner. The distance model, along with other data models, is combined to construct a MILP model, as explained in Section IV. Once established, this model is integrated with an open-source MILP solver to produce an optimal solution, addressing task allocation and precise scheduling for the agent ensemble. The resulting schedule, designed for maximum efficiency, guides real-time execution by the agents.

A. MILP Formulation

Our goal is to minimize the completion time C_{max} plus the start time of each task. It is necessary to minimize the sum of all start times because, for an individual agent, it is optimal for the agent to execute tasks and return in its own minimum makespan. By this formulation, the idle time before each task will be minimized. Equation (1) below defines the objective function.

$$\textbf{min} C_{max} + \sum_{i=1}^{m} t_i \tag{1}$$

Equation (2) considers that the completion time must be larger than or equal to the finish time of every task, plus the ending return cost from the last task. In this way, we can find the maximum execution time and validate the makespan.

$$C_{max} \geq t_i + \Delta_{i,j} + \sum_{j=1}^{m} E_{ij} X_{ij}, \forall A_i \in A \tag{2}$$

Equation (3) below defines the precedence constraint. When there is a time precedence relationship between two tasks, Equation (3) states that the latter one may only start when the previous one is finished. The precedence relationship is defined in the hierarchical task network before the scheduler is invoked.

$$t_j \geq t_i + \Delta_{i,j} + C_{ij}, \forall (i,j) \in P \tag{3}$$

Equation (4) defines travel cost constraint, that the start time of each task must be larger than or equal to the travel cost for the robot to go to the task location. The travel cost is either from the dispatch location to the first allocated task, or from a previous task to the current task. All travel costs are defined in the section above.

$$t_i \geq \sum_{j=1}^{m} C_{ij} X_{ij}, \forall A_i \in A \tag{4}$$

Equation (5) below defines the exact-agent constraint. Each task should be only executed by one robot for a non-collaborative task, and each task should be executed by exactly the amount of agents required for collaboration tasks.

$$\sum_{j=1}^{m} X_{ij} = m_i, \forall A_i \in A \tag{5}$$

Equation (6) (7) states that no two tasks can be simultaneously executed by the same agent. We introduce V as an auxiliary binary decision variable and M is an arbitrarily large number. If any two tasks for the same agent are executed with the overlapped period, then the two equations will result in a contradiction.

$$t_k - \Delta_{i,j} - C_{ik} - t_i \geq -M(2 - X_{ij} - X_{kj}) - M(1 - V_{ikj}) \quad \forall A_i, A_k \in A, R_j \in R \tag{6}$$

$$t_i - \Delta_{k,j} - C_{ik} - t_k \geq -M(2 - X_{ij} - X_{kj}) - M(V_{ikj}) \quad \forall A_i, A_k \in A, R_j \in R \tag{7}$$

Equation (8) is the miscellaneous constraint. It defines time as a natural integer, and decision variables are binary.

$$t_i \in \mathbb{N}, C_{max} \in \mathbb{N}, X_{ij} \in \{0,1\}, V_{ijk} \in \{0,1\} \quad (8)$$

B. Updating Allocation to Allow Partial Task Completion

In multi-robot collaboration scenarios, we often face situations where one robot must wait for another to join for collaboration. To reduce wait times, we employ a strategy where the early-arriving robot may perform part of a task, collaborate, and then return to complete the task if the combined travel and partial task duration is shorter than the anticipated wait time. This dynamic approach minimizes idle periods and is continuously refined during schedule generation using the MILP Solver, ensuring optimal task allocation and operational efficiency.

For two robots being assigned in a collaboration task A_k, assume the robot with vacant waiting time is R_i and the other late-arrival robot is R_j, do the following Algorithm 1:

Algorithm 1 Algorithm for updating allocation to allow partial task completion

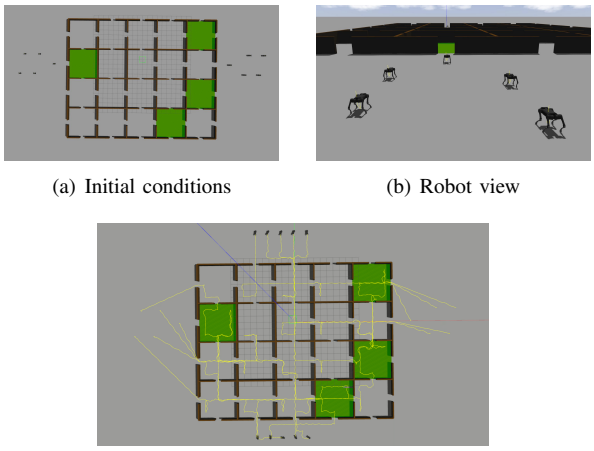
Input: Original Schedule
Output: Updated Schedule
After MILP Solver schedule is generated :
LOOP Process
for R_i, R_j serving collaborative task A_k **do**
  start time of A_k is t_k, arrival time of R_i is t_i
  if $(t_k > t_i)$ **then**
    $\alpha \leftarrow t_k - t_i$
    for any breakable task A scheduled after A_k **do**
      suppose the previous task served by R_i is a_{pre}
      if $(t_{pre} + \Delta_{i,pre} + C_{A_{pre},A} + C_{A,A_k} < \alpha)$ **then**
        schedule partial of A between A_{pre} and A_k
        $\alpha \leftarrow t_{pre} + \Delta_{i,pre} + C_{A_{pre},A} + C_{A,A_k}$
      end if
    end for
  end if
  rerun the schedule to minimize idle time
end for
return *Schedule*

First, we check if R_i has a wait time right before A. If there is a wait time, then this wait time may be optimized by breaking down other tasks. For every other global task scheduled after the start time of A, check if it is breakable into parts. For every breakable task, check if it can fit in the waiting time before A including travel from the task before T to the breakable task location, and the travel from the breakable task location to A. If any task can be plugged into the vacant space, fill the space with a partial of A so that there is no waiting time for a_i. For every other breakable task for R_i coming in later, loop the process to find the lowest available makespan schedule. After checking all available breakable tasks, we find the minimal makespan for all iterations regarding the wait time for R_i. Next, for collaboration task A involving agents R_i and R_j, we adjust the allocation to permit partial task completion, obtaining the optimal local schedule. Subsequently, we apply the algorithm iteratively for every other pair of collaboration task robots, minimizing idle time to determine the lowest available makespan schedule. The final task allocation schedule is then returned to the planner, and we conduct a MILP solver verification to assess global optimization.

Our formulation is implemented using the Google OR-TOOLS CP-SAT solver [30]. It enables us to effectively address the intricate challenges of considering travel time, multi-robot collaboration, and partial task completion. We estimate travel times based on the known grid, and our proposed scheduler dynamically adjusts partial tasks to adhere to collaboration constraints. Moreover, we account for agent initialization and shutdown times, ensuring comprehensive modeling of real-world scenarios. Notably, our formulation yields optimal solutions within mere seconds, showcasing its potential for a seamless transition into an online scheduling framework.

V. Results

A. Evaluation of System Scalability



(a) Initial conditions (b) Robot view

(c) Generated path according to schedule

Fig. 3: Test-case with 10 quadruped robots, 20 rooms and 4 collaboration tasks with 5 blocked doors

We want to test how our system performs with respect to a wide variety of complexity drivers in the problem. We created a scenario generator that can create a wide variety of synthetic test-cases by randomly selecting scene attributes. Instead of a real-world example, we purposely generated test-cases by placing rooms in grid patterns by randomly selecting the number of rooms and room sizes. A representative test-case is depicted in Figure 3(a). Doors are placed between rooms to make sure that robots can travel from room to room. The number of rooms is directly related to tasks that need to be performed by the team. The system can assign certain rooms specialized tasks that require multiple collaborating robots. This allows us to control the number of collaborative tasks. The system is able to randomly select the robot dispatch location and robot collection location. This allows us to control the initial and final travel time. Scenes generated

by the scenario generator are able to randomly generate scenes with a wide variety of complexity and statistically characterize our system's performance.

Our ensemble of agents comprises two types of quadruped robots, specifically the Unitree AlienGO robot and the Unitree A1 robot, as illustrated in Figure 3(b). In this test-case, all agents are engaged in observation tasks, involving traversing rooms and collecting data over randomly assigned time intervals, which are determined prior to scheduling. It's noteworthy that each agent exhibits unique efficiency characteristics, causing their actual execution times to deviate by approximately $\pm 20\%$ from the predefined values.

The simulation environment is constructed using Gazebo, leveraging the ROS (Robotic Operating System) middleware alongside Python and the Google OR-TOOLS CP-SAT Solver for optimal solution determination. Visualization of the simulation occurs within Gazebo itself. The simulation is conducted with 13th Gen Intel(R) Core(TM) i9-13900 CPU with 24 cores, and NVIDIA GeForce RTX 4090 GPU.

In Figure 3(a), we present the initial conditions for our experiments. The agents are strategically spawned on opposite sides of the grid, following the designated path illustrated in Figure 3(c). A representative task assignment, along with corresponding execution times, is outlined in Figure 4. The makespan is quantified as the time interval commencing when all agents enter the grid and concluding when all agents successfully return to their respective ending locations.

The experimental parameters are subject to variation as follows: the side length of each square room, denoted as s, varies within the mathematical range $s \in [4, 8]$. For the grid dimensions, the number of rows, denoted as n_r, falls within the range $n_r \in [3, 5]$, and the number of columns, denoted as n_c, lies within $n_c \in [2, 4]$. Random door blockages on the grid are quantified by the number of blocked doors, expressed as n_b, with $n_b \in [2, 7]$. The number of tasks, represented as n_a, is contingent upon the number of rooms and agents, and n_r is limited to the range $n_r \in [2, \lfloor n_a/2 \rfloor]$. Spawn and return locations are generated once the grid is determined. With the grid's center consistently located at $(0, 0)$, the spawn area for A1 robots is randomly selected as $([-18 + \min_x, -2 + \min_x], [-10, 10])$, while for AlienGO robots, it is chosen as $([2 + \max_x, 18 + \max_x], [-10, 10])$. The return area for A1 robots is randomly designated as $([-5, 5], [-16 + \min_y, -2 + \min_y])$, and for AlienGO robots, it is set as $([-5, 5], [2 + \max_y, 16 + \max_y])$. The number of collaboration tasks, denoted as n_c, is determined within the range $n_c \in [2, \lfloor n_a/3 \rfloor]$.

The efficiency gain of a new task allocation is calculated as the percentage reduction in total makespan compared to the previous allocation. To assess the efficiency gains achieved through collaborative task execution, we introduce a compensation rate denoted as α. This parameter quantifies the reduction in task completion time when collaborating compared to working in isolation. The parameter is user-selected based on how collaboration contributes to task advancement, reflecting user preferences. For instance, if an individual robot typically requires 100 seconds to complete a task, with $\alpha = 0.2$, collaborative work reduces the execution time to 20 seconds when two agents are involved. It's important to note that collaborative tasks do not consider the $\pm 20\%$ variance in individual efficiency, so the execution duration under any two collaborating robots remains the same. We evaluate this test-case using three different schedulers: the original classical MILP scheduler, a MILP scheduler that accounts for travel time, and a MILP scheduler that incorporates collaboration with partial task completion.

We will now present results to characterize system performance.

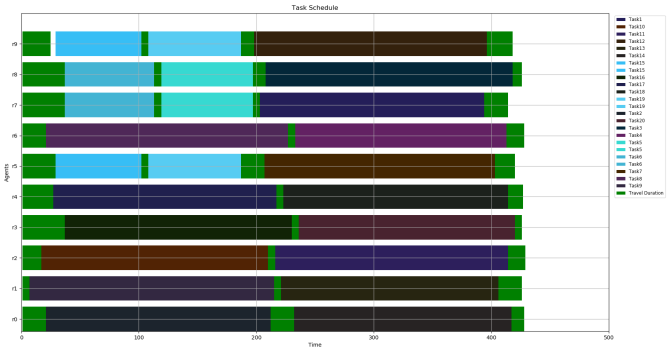


Fig. 4: Representative task allocation for a scene

1) Plan Quality: The sample result depicted in Figure 4 offers an optimal solution to a complex problem involving 10 agents and 20 tasks. Green color is for traveling tasks, blue is collaboration tasks, and all other dark colors are unique tasks by a single agent. The randomized grid consists of 20 rooms and 4 collaboration tasks. As per the generated schedule, all 10 agents start their tasks at $t = 0$s and successfully return to the base by $t = 426$s, traversing the path illustrated in Figure 3(C). Each agent receives a directed schedule and follows the path generated by the A* algorithm. Upon reaching the assigned room, agents work for a duration specified by the model. This test-case does not involve partial task completion, yet it exhibits a significant efficiency boost. The original scheduler, based only on task duration modeling, yields a makespan of 559s, while the travel time-aware scheduler reduces it to 508s, resulting in a 23% efficiency increase in this case.

2) Comparison with Baseline: Based on the setup above, our objective is to conduct a comprehensive comparison of makespan differences across various environmental and task configurations. The initial run of the scheduler focuses solely on the task execution duration model (scheduler 1), which is a straightforward task for most open-source MILP solvers. The resultant overall makespan serves as our baseline benchmark. Subsequently, we employ the scheduler with the incorporation of travel time considerations (scheduler 2), and we record the resulting makespan. Furthermore, the experiment is performed again with a choice of penalty factor α on collaboration tasks (scheduler 3). In this multi-faceted evaluation process involving three distinct schedulers under identical conditions, we discern variations in makespan. We quantify efficiency as a percentage reduction of the overall

makespan between any two schedulers.

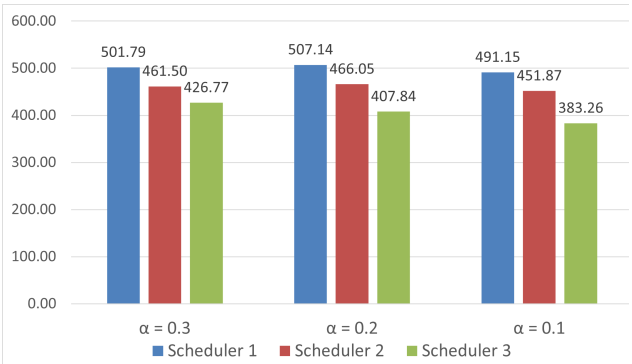


Fig. 5: Makespan Comparison between Three Schedulers by Changing Penalty Factor

To assess the significance of the penalty factor α, we conducted 500 test-cases for three distinct α values, measuring the average overall makespan, as presented in Figure 5. For $\alpha = 0.3$, task scheduling with scheduler 1 resulted in an average makespan of 501 seconds, which was reduced to 461 seconds in scheduler 2. Introducing a penalty rate of $\alpha = 0.3$ for scheduler 3 further decreased the average makespan to 427 seconds. The figure illustrates that the average efficiency increase from scheduler 1 to the scheduler 2 is 7.6%, and the increase from the scheduler 2 to scheduler 3 is 7.4%. The overall efficiency improvement achieved by our scheduler over the original scheduler is 14.5%. For $\alpha = 0.2$, the overall efficiency increase for our proposed scheduler is 19.2%, and increase 21.8% for $\alpha = 0.1$.

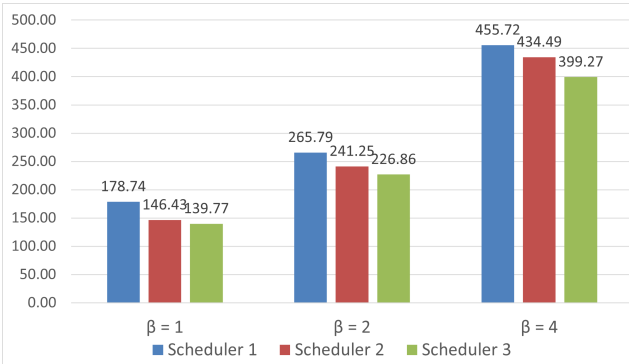
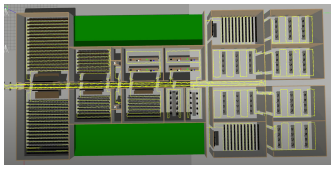
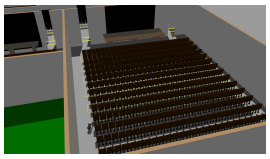


Fig. 6: Makespan Comparison between Three Schedulers by Changing Duration Scale Ratio

To assess the task duration model's relationship with the map scale, we introduce the duration scale ratio β, which represents the ratio of the average execution time in the duration model to the average travel time on the grid for all agents. We generate task duration models based on three different β values, each with a $\pm 20\%$ variance for individual agents, and conduct 500 test-cases on random grids while keeping α fixed at 0.3. The average overall makespan is presented in Figure 6. For a duration scale ratio of $\beta = 1$, the average efficiency increases by 17.6% from scheduler 1 to scheduler 2, by 4.4% from scheduler 2 to the scheduler 3, and the overall efficiency improvement over the original scheduler is 21.3%. The overall efficiency increase for our proposed scheduler is 14.0% for $\beta = 2$ and 11.9% for $\beta = 4$.



(a) complete simulation with proposed scheduler



(b) Robot working in theater classroom

Fig. 7: Disinfection simulation in first floor of SGM Hall

B. Case Study

In order to validate the proposed scheduler in real applications, we performed a case study in a disinfection application. In the case study, as shown in 7, we modeled the floor plan similar to the first floor of a building at the University of Southern California. 6 Unitree AlienGo Robot are deployed at the front door of the building, and they will apply disinfection sprays to rooms in the building, including 2 theater classrooms, 4 normal classrooms, 2 labs, 6 tech labs, 2 conference rooms, and 2 restrooms. Due to the specialty of tech labs, all tech labs require 2 robots to work together for disinfection. Our goal is to find a schedule with minimum execution time so that all the rooms can be disinfected.

Using our multi-agent collaboration scheduler, an optimal schedule is generated in 10 seconds and all the allocations will be sent to the robot. The robots follow scheduled paths generated by the A* algorithm to complete tasks, walking through rooms, disinfecting, and moving to the next area. The reported overall execution duration is 1478 seconds.

At $t = 300$s the agent in the restroom reported a failure and the scheduler is invoked for reallocation. The schedule is then updated in 10 seconds and the reported estimated finish time is updated to be 1760 seconds. Then, at $t = 600$s human intervention occurred and the agent was reported to be fixed and deployed immediately from the restroom. The schedule is updated again with a total makespan of 1540 seconds. Figure 7(a) shows the planned path for all agents as they return to base in the simulation. The simulation shows that all robots are working as desired and collaboration tasks are also finished as required while encountering emergencies.

VI. Conclusions

We demonstrate that travel time considerations and constraints related to the execution of collaborative tasks can be integrated into MILP formulation. We are able to solve problems involving 10 agents and 20 tasks in ten seconds on a moderate PC. This current approach is based on a deterministic framework. We would like to extend this approach to handle uncertainties in task execution duration and failures and validate it in real-world scenarios.

Acknowledgement: This work is supported in part by National Science Foundation Grant IIS-2133091. The opinions expressed are those of the authors and do not necessarily reflect the opinions of the sponsors.

REFERENCES

[1] M. Guettari, I. Gharbi, and S. Hamza, "Uvc disinfection robot," *Environmental Science and Pollution Research*, vol. 28, pp. 40 394–40 399, 2021.

[2] Í. R. da Costa Barros and T. P. Nascimento, "Robotic mobile fulfillment systems: A survey on recent developments and research opportunities," *Robotics and Autonomous Systems*, vol. 137, p. 103729, 2021.

[3] A. Joon and W. Kowalczyk, "Design of autonomous mobile robot for cleaning in the environment with obstacles," *Applied Sciences*, vol. 11, no. 17, p. 8076, 2021.

[4] Y. Chen, A. Pandey, Z. Deng, A. Nguyen, R. Wang, P. Thonapalin, Q. Nguyen, and S. K. Gupta, "A semi-autonomous quadruped robot for performing disinfection in cluttered environments," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers, 2021.

[5] E. F. Flushing, L. M. Gambardella, and G. A. Di Caro, "Simultaneous task allocation, data routing, and transmission scheduling in mobile multi-robot teams," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1861–1868.

[6] H. Chakraa, F. Guérin, E. Leclercq, and D. Lefebvre, "Optimization techniques for multi-robot task allocation problems: Review on the state-of-the-art," *Robotics and Autonomous Systems*, p. 104492, 2023.

[7] H. Aziz, A. Pal, A. Pourmiri, F. Ramezani, and B. Sims, "Task allocation using a team of robots," *Current Robotics Reports*, vol. 3, no. 4, pp. 227–238, 2022.

[8] S. C. Sarin, H. D. Sherali, and A. Bhootra, "New tighter polynomial length formulations for the asymmetric traveling salesman problem with and without precedence constraints," *Operations research letters*, vol. 33, no. 1, pp. 62–70, 2005.

[9] O. Cheikhrouhou and I. Khoufi, "A comprehensive survey on the multiple traveling salesman problem: Applications, approaches and taxonomy," *Computer Science Review*, vol. 40, p. 100369, 2021.

[10] I.-M. Chao, B. L. Golden, and E. A. Wasil, "The team orienteering problem," *European journal of operational research*, vol. 88, no. 3, pp. 464–474, 1996.

[11] A. M. Khamis, A. M. Elmogy, and F. O. Karray, "Complex task allocation in mobile surveillance systems," *Journal of Intelligent & Robotic Systems*, vol. 64, pp. 33–55, 2011.

[12] M. Gam, D. Lefebvre, L. Nabli, and A. J. Telmoudi, "A petri nets based approach for the optimisation of surveillance patrols," *International Journal of Sensor Networks*, vol. 36, no. 4, pp. 181–193, 2021.

[13] D. Applegate and W. Cook, "A computational study of the job-shop scheduling problem," *ORSA Journal on computing*, vol. 3, no. 2, pp. 149–156, 1991.

[14] L. Shen, S. Dauzère-Pérès, and J. S. Neufeld, "Solving the flexible job shop scheduling problem with sequence-dependent setup times," *European journal of operational research*, vol. 265, no. 2, pp. 503–516, 2018.

[15] M. L. Pinedo, *Scheduling*. Springer, 2012, vol. 29.

[16] 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.

[17] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[18] N. Atay and B. Bayazit, "Mixed-integer linear programming solution to multi-robot task allocation problem," 2006.

[19] M. Lippi and A. Marino, "A mixed-integer linear programming formulation for human multi-robot task allocation," in *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*, 2021, pp. 1017–1023.

[20] N. Dhanaraj, S. V. Narayan, S. Nikolaidis, and S. K. Gupta, "Contingency-aware task assignment and scheduling for human-robot teams," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 5765–5771.

[21] M. Lippi, P. Di Lillo, and A. Marino, "A task allocation framework for human multi-robot collaborative settings," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 7614–7620.

[22] Z. Li and X. Li, "Genetic algorithm for task allocation and path planning of multi-robot system," *J. Math. Sci*, vol. 4, no. 1, pp. 34–38, 2016.

[23] S. Mirjalili and S. Mirjalili, "Genetic algorithm," *Evolutionary Algorithms and Neural Networks: Theory and Applications*, pp. 43–55, 2019.

[24] C. Zhang and J. A. Shah, "Co-optimizating multi-agent placement with task assignment and scheduling." in *IJCAI*, 2016, pp. 3308–3314.

[25] S. Li, D. Park, Y. Sung, J. A. Shah, and N. Roy, "Reactive task and motion planning under temporal logic specifications," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 12 618–12 624.

[26] B. Hayes and B. Scassellati, "Autonomously constructing hierarchical task networks for planning and human-robot collaboration," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 5469–5476.

[27] L. Johannsmeier and S. Haddadin, "A hierarchical human-robot interaction-planning framework for task allocation in collaborative industrial assembly processes," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 41–48, 2016.

[28] Y. Cheng, L. Sun, C. Liu, and M. Tomizuka, "Towards efficient human-robot collaboration with robust plan recognition and trajectory prediction," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2602–2609, 2020.

[29] Y. Cheng, L. Sun, and M. Tomizuka, "Human-aware robot task planning based on a hierarchical task model," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1136–1143, 2021.

[30] F. Didier, L. Perron, S. Mohajeri, S. A. Gay, T. Cuvelier, and V. Furnon, "Or-tools' vehicle routing solver: a generic constraint-programming solver with heuristic search for routing problems," 2023.

[31] A. Hosny and S. Reda, "Automatic milp solver configuration by learning problem similarities," *Annals of Operations Research*, pp. 1–28, 2023.

[32] C. D. Bellicoso, M. Bjelonic, L. Wellhausen, K. Holtmann, F. Günther, M. Tranzatto, P. Fankhauser, and M. Hutter, "Advances in real-world applications for legged robots," *Journal of Field Robotics*, vol. 35, no. 8, pp. 1311–1326, 2018.

[33] R. Zimroz, M. Hutter, M. Mistry, P. Stefaniak, K. Walas, and J. Wodecki, "Why should inspection robots be used in deep underground mines?" in *Proceedings of the 27th International Symposium on Mine Planning and Equipment Selection-MPES 2018*. Springer, 2019, pp. 497–507.

[34] C. Gehring, P. Fankhauser, L. Isler, R. Diethelm, S. Bachmann, M. Potz, L. Gerstenberg, and M. Hutter, "Anymal in the field: Solving industrial inspection of an offshore hvdc platform with a quadrupedal robot," in *Field and Service Robotics: Results of the 12th International Conference*. Springer, 2021, pp. 247–260.

[35] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, p. eabk2822, 2022.

[36] A. Bouman, M. F. Ginting, N. Alatur, M. Palieri, D. D. Fan, T. Touma, T. Pailevanian, S.-K. Kim, K. Otsu, J. Burdick *et al.*, "Autonomous spot: Long-range autonomous exploration of extreme environments with legged locomotion," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 2518–2525.

[37] A. Agha, K. Otsu, B. Morrell, D. D. Fan, R. Thakker, A. Santamaria-Navarro, S.-K. Kim, A. Bouman, X. Lei, J. Edlund *et al.*, "Nebula: Quest for robotic autonomy in challenging environments; team costar at the darpa subterranean challenge," *arXiv preprint arXiv:2103.11470*, 2021.

[38] B. Debogórski, M. Fiedeń, J. Szrek, and J. Wodecki, "A small legged robot for inspection purposes," in *IOP Conference Series: Earth and Environmental Science*, vol. 1189, no. 1. IOP Publishing, 2023, p. 012004.