Multi-Robot Task Allocation in Disaster Response

Siddhartha Cheruvu, Yi Chen, Aaron Arul Maria John
MS Mechanical Engineering - 2020
Arizona State University
Course: MAE 598 -Multi-Robot Systems

I. Abstract

There has been a significant rise in the number of natural disasters over the past few decades. Studies have shown that first 72 hours are very crucial for the emergency response team to save lives and stop further damage to structures, meaning efficient drone scouting and dispatching is highly crucial in maximizing the recovery and rescue of victims. In most of these cases, sending in first human responders is either too dangerous or not practical. To address this issue, UAVs are being used as first responders to inspect and identify key hot spots after disaster has struck a place. The right utilization of available UAVs during this time plays a crucial role for the emergency responders in deciding their actions. To reduce the time and cost of travel of the UAV in this process to cover all tasks, we have explored different Multi-Robot Task Allocation (MRTA) procedures. By formulating the problem as an optimization problem and using other coordination techniques such as Greedy, Hungarian and Simulated Annealing methods, we are able to implement a swarm of robots to be dispatched for missions, while avoiding path or task overlapping to maximize the efficiency in mission execution. The system's robustness is also tested in dynamically changing environments, with scenarios such as newly emerging tasks and in the cases where drones fail. MATLAB and Gazebo environments are primarily used for verifying the performance of each algorithm under different conditions. It is concluded from the simulations that the Hungarian method outperforms the Greedy method in terms of optimizing the path costs in most of the scenarios. Simulated annealing is used as a benchmark in these studies.

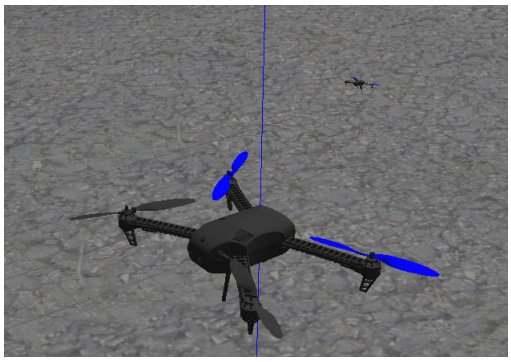

Fig. 1. UAV Iris, Selected Vehicle for Simulation

II. Mathematical Model

In this report, the primary behavior under study is the 'Multi-Robot Task Allocation'.

A. Domain Characteristics:

The disaster response domain is built with the following key characteristics.

1) The number of tasks outnumber the number of agents
2) The locations of tasks are known prior to the execution
3) New tasks can be issued throughout the mission
4) Each agent can execute only one task at a time (Singe Task - ST robots)
5) Each task requires only one agent to be completed (Single Robot - SR tasks)
6) The agent-tasks utilities have intra-schedule dependencies, i.e, the effective utility of an agent for a given task depends upon the other tasks of the same agent

B. Scenario:

Based on the above domain characteristics, a fire-fighting response scenario is created. A small team of fire-fighting UAVs (also referred to as 'agents') are deployed in order to extinguish fires identified across a residential locality of a city. There can be new fires that are continuously discovered during the mission. Another team of scouting UAVs is already considered to be in operation so that they identify any new fires and transmit corresponding locations to the ground station. This information is relayed continuously to all the agents so that the task locations are known prior to the execution.

C. Assumptions and Constraints:

The dimensions of environment are limited to a grid of 100m x 100m in this study. Within these bounds it is reasonable to assume that the agents maintain full communication with the ground station throughout the mission. The communication between the agents and the ground station is used only to transmit the locations of tasks and other agents i.e, the ground station does not act as a central controller. The UAVs are equipped with GPS, IMU and Odometry sensors that help in pinpointing their exact locations.

The UAVs are hovering at a safe height over the debris and are generally unaffected by the obstacles on the ground. Moreover, they are deployed at different heights to alleviate the problem of any probabilities of occurrence of collisions between them. This assumption is considered to simplify the controller design used in path planning of the UAVs.

Further, there is a cost (c_{ij}) associated to each agent-task pair and it is proportional to the distance from the agent's current location to the task location. The task allocation problem is then formulated using an assignment-based double index integer linear programming formulation[6].

D. Model

Consider there are N agents and M starting tasks. A binary assignment variable x_{ij} is then defined as shown below:

$$x_{ij} = \begin{cases} 1, & \text{if task j is assigned to agent i} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

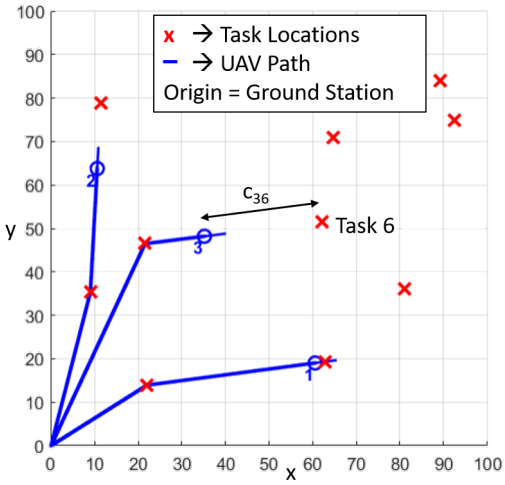

Fig. 2. Visual Representation of MRTA Domain

Then, the mathematical model of the integer linear programming formulation can be given as follows:

Minimize

$$\sum_{i=1}^{N}\sum_{j=1}^{M} c_{ij}x_{ij} \tag{2}$$

Subject to:

$$\sum_{i\in N} x_{ij} = 1, \quad \forall\ i \in N \tag{3}$$

$$\sum_{j\in M} x_{ij} = 1, \quad \forall\ j \in M \tag{4}$$

$$x_{ij} \in \{0, 1\} \tag{5}$$

The model is tweaked to convert this formulation to a multiple-Travelling Salesmen (m-TSP) problem[5] by modifying the constraints as follows. This is because there are in-schedule dependencies for the UAVs which cannot be effectively taken into account by linear assignment problem. Hence, m-TSP provides more accurate way of modeling this problem.

Minimize

$$\sum_{i=1}^{N}\sum_{j=1}^{M} c_{ij}x_{ij} \tag{6}$$

Subject to:

$$\sum_{j=2}^{M} x_{1j} = N, \tag{7}$$

$$\sum_{j=2}^{M} x_{j1} = N, \tag{8}$$

$$\sum_{i \in N} x_{ij} = 1, \quad \forall \ j \in [2, M] \tag{9}$$

$$\sum_{j \in M} x_{ij} = 1, \quad \forall \ i \in [2, N] \tag{10}$$

$$x_{ij} \in \{0, 1\} \tag{11}$$

The constraints (7) and (8) are added to ensure that exactly 'N' agents depart from and return to node 1 i.e, the ground station.

The linear assignment problem can be solved in polynomial time using approaches such as Hungarian Algorithm (Kuhn, 1955)[16] or any standard optimization solvers. These algorithms need the number of tasks and the number of agents to be the same. To overcome this, a set of dummy agents can be created with large positive costs so that the number of rows and columns in the cost matrix are the same. On the other hand, m-TSP is an NP-hard problem.

In addition, in one of the iterations, convex optimization techniques are used to determine the next tasks at each decision making point. The agents decide their next tasks using a decentralized task allocation algorithm to ensure that the overall process is robust and flexible.

III. Theoretical Analysis

Theoretical analysis is performed on the mathematical model and the conclusions that are made about the properties of the model are provided in this section.

A. Convex Optimization

Statement: The m-TSP model described in section II is a Linear Program which is a type of convex optimization problem.

Proof:

The standard form of a Linear Program is:

Minimize

$$\vec{c}^{\,T}\vec{x} + \vec{d} \tag{12}$$

Subject to:

$$\mathbf{A}\vec{x} = \vec{b} \tag{13}$$

$$G\vec{x} \preceq h \tag{14}$$

The objective function (12) and the inequality functions (14) are affine (linear functions with offset).

The variable vector in this optimization problem is

$$\vec{x} = \begin{bmatrix} x_{11} \\ x_{12} \\ . \\ . \\ x_{NM} \end{bmatrix}_{NM \times 1} \tag{15}$$

Objective Function:

The objective function of the model under consideration is

$$\sum_{i=1}^{N} \sum_{j=1}^{M} c_{ij} x_{ij} \tag{16}$$

It can be re-written as:

$$c_{11}x_{11} + c_{12}x_{12} + ... + c_{NM}x_{NM} \tag{17}$$

$$= \vec{c}^{\,T}\vec{x} + \vec{d} \tag{18}$$

where,

$\vec{c} = [c_{11}, c_{12}, ..., c_{NM}]^T$ and

$\vec{d} = \vec{\mathbf{0}}$

Hence, the objective function is affine.

Equality Constraints:

The equality constraints considered in (7), (8), (9) and (10) are all linear functions. Linear functions are both convex and concave. The proof is provided for one of these constraints and the rest of them should follow a similar proof.

$$\sum_{j \in M} x_{ij} = 1, \quad \forall \ i \in [2, N] \tag{19}$$

It can be re-written as:

$$x_{21} + x_{22} + ... + x_{2M} - 1 = 0 \tag{20}$$

which is in the form:

$\mathbf{A}\vec{x} = \vec{b}$

where,

$\mathbf{A}$ is a co-efficient matrix of 0s and 1s, and

$\vec{b}$ = [1]

Hence, the equality constraints are affine.

Inequality Constraints:

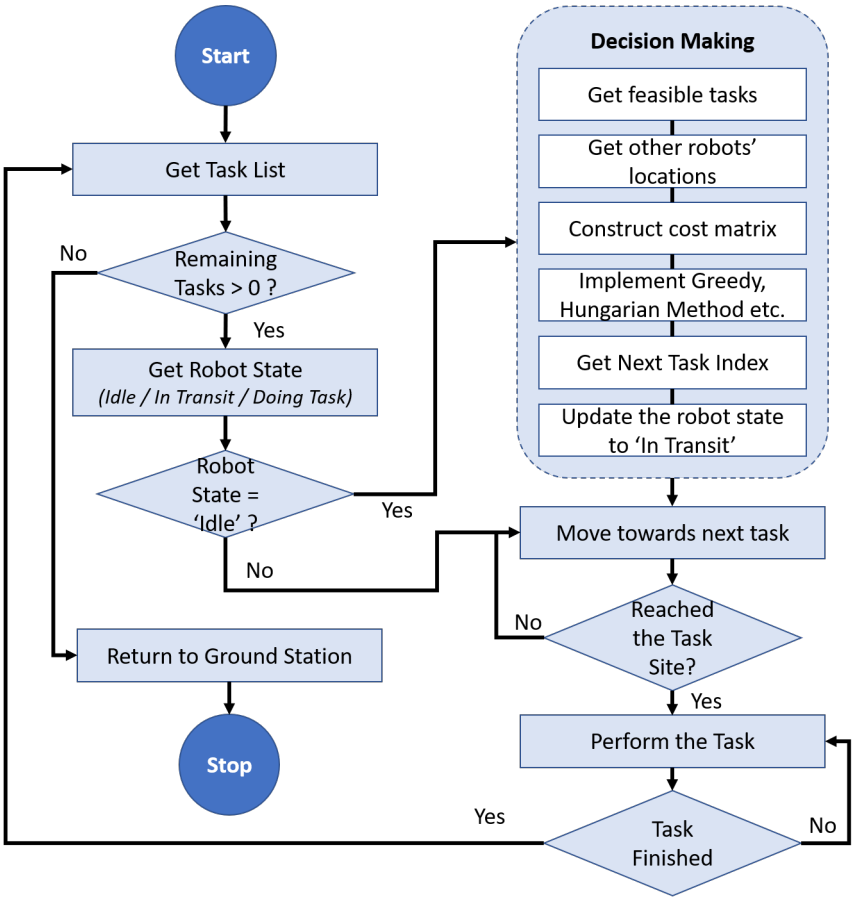

Fig. 3. MRTA Behavior Algorithm under Decentralized Deployment

The values of components in the variable vector are greater than zero. This can be written in the form of inequality constraint as follows.

$$\vec{x} \succeq \vec{0} \tag{21}$$

It can be rewritten as

$$-\vec{x} \preceq \vec{0} \tag{22}$$

which can be further broken down into smaller inequalities.

$$-x_{11} \leq 0, -x_{12} \leq 0..., -x_{NM} \leq 0 \tag{23}$$

$$[-1, 0, 0, ..., 0]\vec{x} \leq 0 \tag{24}$$

$$[0, -1, 0, ..., 0]\vec{x} \leq 0 \text{ and so on...} \tag{25}$$

These are all linear functions of $\vec{x}$. Hence, the inequality constraints are also affine. Therefore, it can be concluded that the problem under consideration is a linear program which is a type of convex optimization problem.

B. Bipartite Graph Matching

The MRTA problem under consideration can be represented with the help of a bipartite graph (or simply bi-graph). This is because the vertices can be divided into two independent and disjoint sets U and V such that every edge connects a vertex in U to one in V. Graph matching can then be performed to allocate the tasks to robots in a conflict-free manner.

Hungarian method can be applied to perform Bipartite graph matching with a complexity of $\mathcal{O}(n^4)$.The edge weights can be determined using the robot-task pair distance at the moment the decision making algorithm is invoked. However, a matrix based formulation of Hungarian method can be easily applied in this case with a complexity of $\mathcal{O}(n^3)$. The optimal solution achieved in both cases is the same.

C. Controller Design

An algorithm is designed in order to drive the multi-robot task allocation behavior towards the desired allocation. The corresponding algorithm under decentralized deployment is shown in Fig.3.

Three different kinds of decision making algorithms are tested.

1) Greedy Algorithm
2) Hungarian Algorithm

Then, Simulated Annealing algorithm is used as a benchmark for comparing the performance of these two algorithms. The results of this study are provided in the section IV.

IV. Validation in Simulations

Simulations are performed in MATLAB and Gazebo in order to validate the model properties described in section III.

A. MATLAB Simulation

The optimization problem is validated to be of convex optimization type by solving the task allocation using convex optimization solver and comparing the results with those of the Hungarian method. It is found that the results are exactly the same for both the techniques. Since it is known that the Hungarian method provides optimal solution for a linear assignment problem, it can be concluded that the problem in hand is indeed a Linear Program. The task allocation results of one of the MATLAB simulations with 3 robots and 10 tasks are provided in Fig.4.

1) Greedy Algorithm:

- If any robot, 'i' remains unassigned, find the task 'j' with the lowest cost.
- Assign task 'j' to robot 'i' and remove it from the consideration.
- Go to the first step.

The greedy algorithm forces each idle robot to take the nearest available task. Hence, it doesn't take the positions of other robots into account. One of the advantages of this method is that it is easy to implement in an online computation scenario. Since, the robots do not need other robots' positions it is decentralized decision making. The downside, however, is that choosing the nearest task is not always optimal and can result in redundancies. As it can be seen in Fig.5., the robot 1's last task has high edge distance and it could have been taken up by robot 3.

2) Hungarian Algorithm:

- If any robot, 'i' remains unassigned, find the task 'j' using the utilities of all the other robots.
- Construct a cost matrix where rows correspond to a robot ID and the columns correspond to a task.
- If the number of robots R is not equal to the number of tasks remaining T, add dummy rows / columns with very large cost to make the cost matrix square.
- Apply row and columns transformations as per Hungarian Method and find the next task for the robot.
- Go to the first step.

The Hungarian Algorithm is observed to provide a better solution in terms of total cost compared to the greedy algorithm. Since, the utilities of all the other robots are considered in the decision-making of each robot, this method provides an improved total cost. However, the solution is sub-optimal in multi-robot scenario. The route taken by the robots under this decision making can be seen in Fig.6.

3) Simulated Annealing: In order to include a benchmark to gauge the performance of the used methods, a Simulated-Annealing algorithm is used to find the optimal paths in each scenario.

Simulated annealing is very effective in finding global optima in the presence of large numbers of local optima."Annealing" refers to an analogy with thermodynamics, specifically with the way that metals cool and anneal. Simulated annealing uses the objective function of an optimization problem instead of the energy of a material.

- Initialize the problem with a random allocation.
- Initialize an annealing temperature T and a hyper parameter α. T starts high and is gradually decreased according to an "annealing schedule", T_{new} = T*.α.
- Interchange a set of nodes in the path and find the cost of this neighboring solution.
- If it is an improvement over the original cost, accept the new allocation.
- If not, accept the new allocation anyway based on a probability.

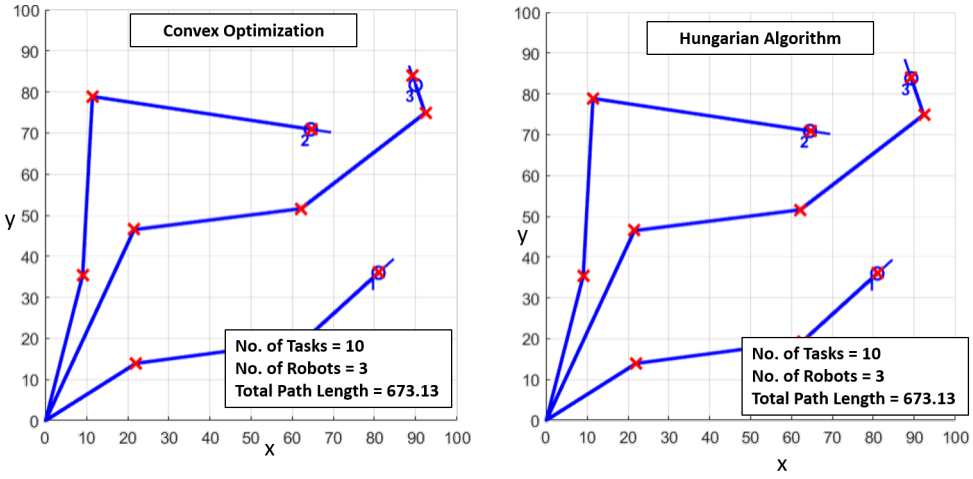

Fig. 4. Comparison of Convex Optimization and Hungarian Method

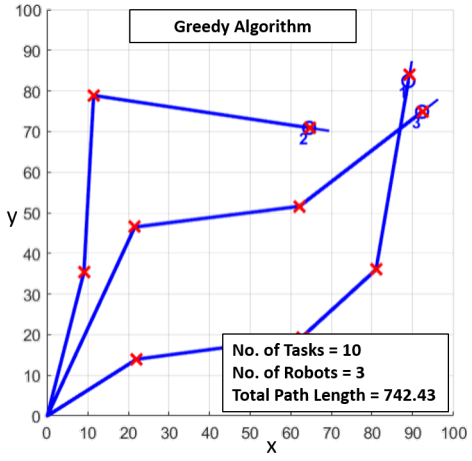

Fig. 5. Greedy Algorithm - Task Allocation

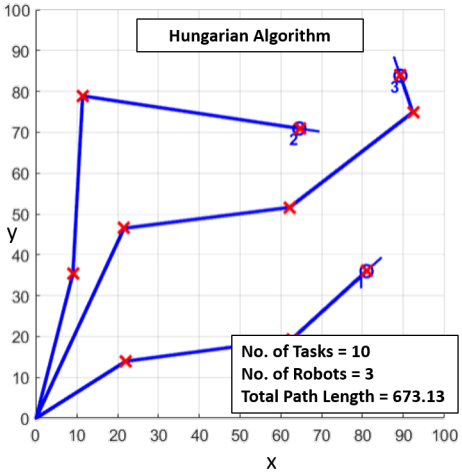

Fig. 6. Hungarian Algorithm - Task Allocation

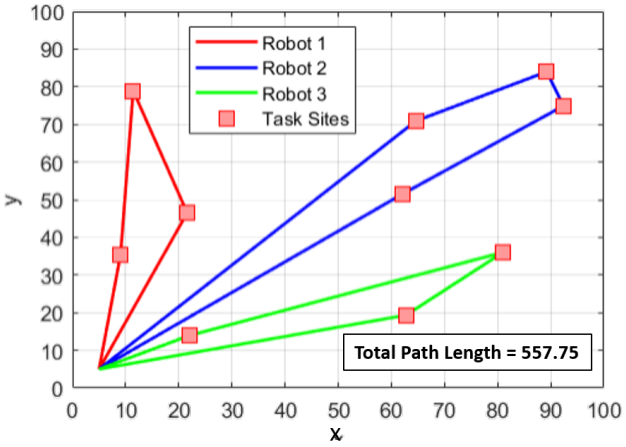

Fig. 7. Simulated Annealing - Task Allocation

- The probability decreases exponentially with the "badness" of the move, which is the amount ΔE by which the solution is worsened (i.e., energy is increased.)
- Temperature T is also used to determine the probability. At higher values of T, worse solutions are more likely to be accepted and as T decreases exponentially after each iteration, the probability decreases and the solution converges to the global optima.

The optimal path for the scenario under consideration identified through simulated annealing is shown in Fig.7.

4) Performance Evaluation: The performance of Greedy and Hungarian algorithms are evaluated

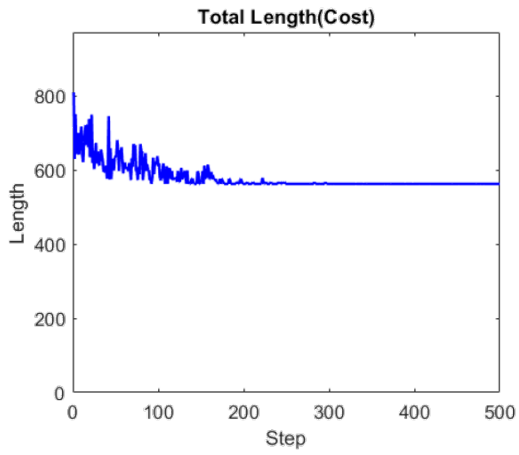

Fig. 8. Simulated Annealing - Total Cost Improvement

Iteration	Total Cost (Combined Path Length)			Percentage Difference	
	Greedy	Hungarian	Simulated Annealing (SA)	Greedy vs SA	Hungarian vs SA
1	742.4	673.1	557.8	33.11%	20.69%
2	808.2	753.9	601.9	34.27%	25.26%
3	808.4	690.4	585.3	38.11%	17.95%
4	749.4	805.6	600.7	24.76%	34.12%
5	744.0	654.1	575.0	29.39%	13.75%
6	651.0	650.4	561.7	15.90%	15.80%
7	686.7	598.8	549.0	25.09%	9.07%
8	681.6	645.9	578.9	17.75%	11.57%
9	602.9	598.4	555.3	8.56%	7.75%
10	758.9	705.0	632.6	19.98%	11.45%
11	850.1	754.2	693.6	22.56%	8.73%
12	780.9	644.2	562.5	38.83%	14.52%
Average	738.7	681.1	587.8	**25.69%**	**15.89%**

Fig. 9. Comparative Study of the Employed Algorithms

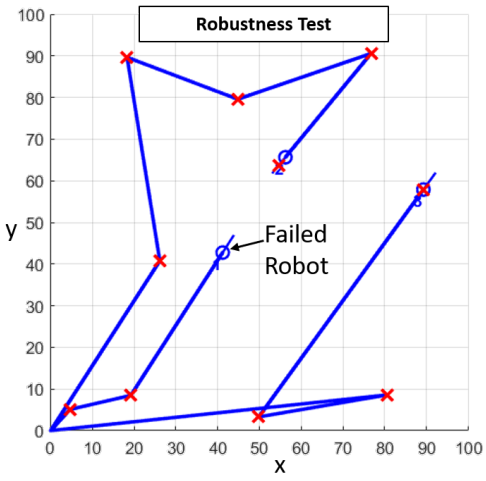

Fig. 10. Robustness Test of the Algorithms

Fig. 11. Dynamic Task Allocation - Randomly Emerging Tasks

by using Simulated Annealing as the benchmark. Several simulations with different sets of task locations are used in this comparison. It is observed that while Simulated Annealing provides the optimal path, Hungarian method provides paths which are on average, 15.9 percent longer. Greedy algorithm generally provides worse solutions with an average path 25.7 percent longer compared to that of the annealing method.

5) Additional Simulations: The robustness of the algorithms is tested by intentionally disabling one of the robots during the mission. It is observed that the remaining robots update their task schedules and finish all the tasks in the absence of the failed robot. The results of this simulation are provided in Fig.10.

Additional simulations are run where new tasks are dynamically created based on a probability factor. The robots are observed to take up new tasks dynamically and finish all the tasks. The results of this simulation are provided in Fig.11.

B. Gazebo simulation

1) Simulation setup: In addition to the validation performed in Matlab, we developed a more complex simulation with higher fidelity in Gazebo with PX4 unmanned aerial vehicle platform, accompanied with Python and ROS languages. The virtual UAV model used is the quadcopter IRIS, which came equipped with position sensor (IMU) and way point following controller. The ROS nodes were designed to be easily scalable to different number of tasks, robots and the size of the work space. Our miniature Gazebo simulation en-

vironment is as shown in Fig.12, where the robots are initially spawned at 3 distinct locations, and the task locations, which in our case are the disaster sites, are placed inside the 10 by 10 3D space with a square pattern, to demonstrate the difference in task allocation algorithm. The dimension of the space were shrunk for visualization purposes, such that the drone movements are more visible. In our particular showcase, we set the number of task M to be 12, and number of robots N to be 3.

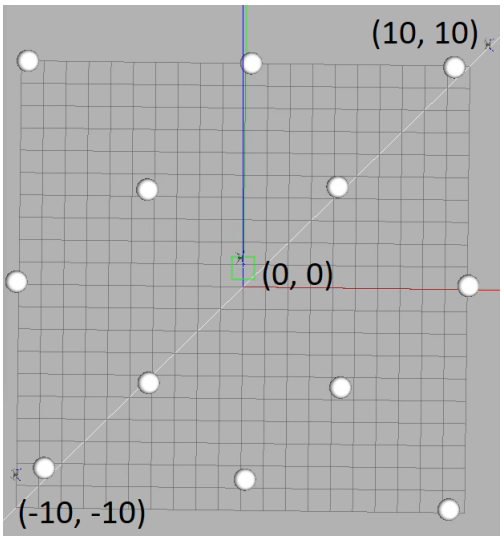

Fig. 12. Simulation environment in Gazebo: white spheres represent task location, while drones spawn along the white diagonal line with labeled positions.

2) ROS node composition: The ROS communication network architecture is composed of a master in charge of assigning tasks and nodes that are in charge of controlling each robot, and it is constructed in the way that can be scaled to add more robots to the network. The ROS communication network construction is shown in the Fig.13.

While the general structure of the algorithm is similar to Fig.3, in ROS there are some slight modification, which may help to provide a more realistic model in the real world, where disturbances and delay exist.

The design idea of the master node is as followed: It keeps track of each robot's location, as well as their status of handling the task. The task assignment function evaluates the cost of a robot executing a task, given the task position and robot's position. Then the task with the minimum cost is assigned to the robot. If the robot tells the master that it has accomplished the previous task, then the master starts sending the new task location to the robot, until all tasks are completed. Once that happens, the master announce that the mission is finished, thus all robots start moving to their home position. Note that the sequence of task assignment is different depending on each of the task allocation algorithm. The lower level control, which are the robot controllers in this case, are working at a higher frequency than the master, as a design principle in order to prevent loss of message and have higher reaction rate to the environment.

3) Task Allocation Models: With the ROS architecture, we implemented two kinds of task allocation algorithm: online and offline allocation, which are the variant of the greedy and Hungarian algorithm introduced in section III. In the offline version (Hungarian method) of the algorithm, the tasks are assigned among all robots before the mission starts, where the cost of executing a task is evaluated with the robot's starting position. This implementation is different in the way where all tasks are assigned at the start of a mission instead of online computation, due to implementation limitation. On the other hand, the online task allocation assigns one task at a time, and the costs of executing each task are evaluated with the robot's current location. The simulation results are presented in the video format attached in the submission.

4) Results and Discussion: It can be easily noted that the offline model is largely less efficient than the online version, due to the fact that costs to perform each task is evaluated with robot's initial location. What happens is that if one agent spawns closer to more tasks, it will be assigned most of the tasks even though it means the mission would have been faster if the tasks were shared.

From the simulation, we also noticed that a slight offset in the drone current position can alter the task assignment dramatically, especially in the offline case. For example, if a drone hovers slightly away from the center of the field, it is less likely to be assigned tasks than the ones that are closer to the center, and thus closer to more positions. Consequently, the mission time cost and path length increases for the offline allocation algorithm.

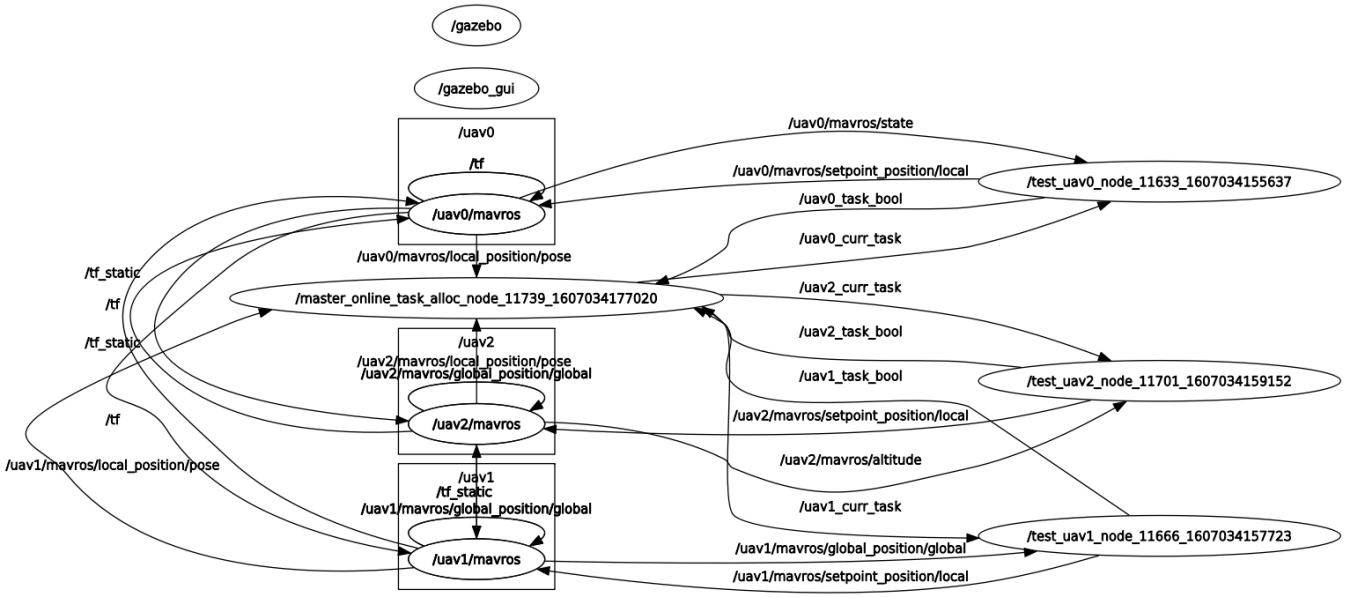

Fig. 13. Node Graph of the Gazebo Simulation: The master node (center circle) is in charge of the task allocation, and each UAV node (right three circles) accepts command from master and control the UAV.

However, in the online version, we observed that the initial location of the drones does not affect the mission time as much as the offline version, since the tasks are assigned on the fly depending on the location of the drone at a given time. On the other hand, the simulation provided different insight than the analysis model, where there exists disturbances and noises in sensor, environment and control loop, which can change the solution by a significant amount depending on the scenario of the problem.

V. Conclusions

In this report, we invoked some useful models and properties from prior work, and using the knowledge as a stepping stone to design several task allocation algorithms, in search of an optimal way for our disaster response problem. We then developed simulations in MATLAB and Gazebo to test our models, showing the effectiveness of each algorithm.

However, there are some limitations that we encountered during the process. While optimal assignment can be found offline, it is difficult to deploy the same strategy within a dynamical environment efficiently, due to the time cost to obtain a good solution. In the case of the Gazebo simulation, we assumed that each robot operates at a different height, thus alleviating the potential problem of inter robot collisions. In order for our current model to be widely applicable for types of robots, future work is needed to incorporate dynamic collision avoidance system, to avoid each other and possible undiscovered obstruction.

VI. Team Members' Contribution

- Siddhartha:
  - Section II (A, B, C) - (Problem Formulation, Domain Creation, Assumptions and Constraints)
  - Section III (A) - (Convex Optimization)
  - Section IV (A-2, A-3, A-4) - (Coded Hungarian Algorithm, Simulated Annealing, included dynamic task allocation for newly emergent tasks)
- Yi:
  - Section II (D) - (Mathematical Model - Integer Linear Program)
  - Section III (C) - (Controller Design)
  - Section IV (B) - (Gazebo ROS Simulation Code)

- Aaron:
  - Section II (D) - (Mathematical Model - mTSP)
  - Section III (B) - (Bipartite Graph Matching)
  - Section IV (A-1) - (Coded Greedy Algorithm, performed robustness test and Data Collection)

REFERENCES

[1] H.-C. Jang, Y.-N. Lien, and T.-C. Tsai, "Rescue information system for earthquake disasters based on manet emergency communication platform," in Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing: Connecting the World Wirelessly. ACM, 2009, pp. 623–627.

[2] E. Gil Jones, M. Bernardine Dias, Anthony Stentz, "Learning-enhanced Market-based Task Allocation for Oversubscribed Domains", 2007

[3] E. Gil Jones, M. Bernardine Dias, Anthony Stentz, "Time-extended multi-robot coordination for domains with intra-path constraints", 2010.

[4] Payam Ghassemi, Souma Chowdhury, "Multi-Robot Task Allocation in Disaster Response: Addressing Dynamic Tasks with Deadlines and Robots with Capacity Constraints", 2020

[5] T. Bektas, The multiple traveling salesman problem: an overview of formulations and solution procedures, Omega 34 (3) (2006) 209–219.

[6] G. Ayorkor Korsah, Anthony Stentz and M. Bernardine Dias, "A comprehensive taxonomy for multi-robot task allocation", 2013.

[7] Antidio Viguria, Ivan Maza and Anibal Ollero, Distributed Service-Based Cooperation in Aerial/Ground Robot Teams Applied to Fire Detection and Extinguishing Missions, Advanced Robotics 24 (2010) 1–23

[8] M. Bernardine Dias, Robert Zlot, Nidhi Kalra, and Anthony Stentz, Market-based multirobot coordination: a survey and analysis, Proceedings of the IEEE (Volume: 94, Issue: 7, July 2006)

[9] Maja J. Mataric´ , Gaurav S. Sukhatme And Esben H. Østergaard. "Multi-Robot Task Allocation in Uncertain Environments" (2003)

[10] D.F.Votaw, Jr. and A. Orden, "The Personnel Assignment Problem"

[11] Bradley Woosley and Prithviraj Dasgupta, "Integrated real-time task and motion planning for multiple robots under path and communication uncertainties", 2017.

[12] Maren Bennewitz, Wolfram Burgard, Sebastian Thrun, "Optimizing Schedules for Prioritized Path Planning of Multi-Robot Systems", 2001.

[13] Hector I. A. Perez-imaz, Paulo A. F. Rezeck, Douglas G. Macharet, Mario F. M. Campos, "Multi-robot 3D Coverage Path Planning for First Responders Teams", 2016.

[14] Ayman El shenawy, Khalil Mohamed, Hany M. Harb, "Exploration Strategies of Coordinated Multi-Robot System: A Comparative Study", 2018.

[15] Bradley Woosley, Prithviraj Dasgupta, John G. Rogers III, Jeffrey Twigg, "Multi-robot information driven path planning under communication constraints", 2019

[16] Kuhn HW (1955) The Hungarian method for the assignment problem. Naval Research Logistics Quarterly 2: 83–97.