Auction-Based Multi-Robot Task Allocation and Path Planning for Vehicle Routing

1st Atharv Marathe
Robotics
Oregon State University
Corvallis, Oregon
marathea@oregonstate.edu

2nd Clement Cantil
Robotics
Oregon State University
Corvallis, Oregon
cantilc@oregonstate.edu

3rd Roger Wienaah
Robotics
Oregon State University
Corvallis, Oregon
wienaahr@oregonstate.edu

***Abstract*—In this project, we explore how homogenous multi-agent teams coordinate their movements to navigate to tasks under four different auctioning paradigms. Tasks with random but known durations pop up in random accessible locations at set times, and the robots submit costs-to-go to an auctioneer or team leader that identifies the low bids and awards the subject tasks to the corresponding robots. The four different auctioning methods modify the order in which the auctioneer instructs the team to bid on tasks and awards tasks to winners as follows: (1) robots bid on and are assigned tasks with higher rewards before bidding on and being assigned tasks with lower rewards, (2) robots bid on and are assigned older tasks before bidding on and being assigned newer tasks, (3) robots bid on tasks in a random order as a baseline, and (4) robots bid on all tasks and tasks with lower costs are awarded before tasks with higher costs. We compare the performance of these auctioning methods using RRT, RRT*, and RRT*-Connect to one another in terms of total costs and task list completion time over several trials and present our findings in a box-and=whisker plots..**

I. Introduction

In late 2019, the Centers for Disease Control and Prevention (CDC) reported the first case of 2019 Novel Coronavirus (COVID-19) in the United States. [31] As a consequence of the high transmission rate and virulence of COVID-19, contactless delivery was in vogue because it protected the elderly, immunocompromised, and other high-risk groups from contracting COVID-19, which was spread by exposure to droplets in infected people's breath. [32] From leaving packages on porches instead of delivering them directly to recipients to testing new delivery platforms, companies adopted new business practices and technologies designed to keep their products moving and maintain their respective customer bases.

This push for contactless delivery greatly benefited several companies specialized in the design and manufacture of food delivery robots, including Starship Technologies and Serve Robotics. Alastair Westgarth, CEO of Starship Technologies, stated that as a result of the COVID-19 pandemic, "We saw demand for robot usage just go through the ceiling." [33]

While the COVID-19 pandemic has, thankfully, ended, the demand for contactless delivery services, and thus the demand for food delivery robots, has not waned. On the contrary, analysts anticipate that all industries involved in logistics and delivery services will aggressively adopt contactless delivery services. [34] Moreover, the ripple effect of the COVID-19 pandemic put many restaurants out of business. CNN reported that Technomic, a restaurant research firm, "doesn't foresee the number of restaurants in the US returning to pre-Covid levels even by 2026." [35]



Fig. 1. BellaBots Coordinating in Restaurant Environment [36]

Accordingly, we chose food delivery robots as the motivating example for our project. Unlike Starship robots, however, we confine our food delivery robots to a space enclosed by walls with rooms and connecting corridors the robots must navigate. These constraints make the robots more similar to BellaBots, a line of food delivery robots designed to operate in sit-down restaurants. [36]

In this arrangement, teams of food delivery robots coordinate with human staff members scattered throughout a restaurant to deliver food and drink to customers, return dirty crockery and cutlery to wash stations, and deliver small miscellaneous items to different stations located throughout the restaurant. Teams of autonomous delivery robots coordinate

their routes to ensure that the payloads they carry arrive at their destinations in a timely and cost-effective manner by calculating their paths to the destination and submitting a cost-to-go to an auctioneer or team leader. After receiving all bids, the auctioneer or team leader selects the robot that submitted the lowest bid and awards the subject task to that unit.

In the context of food delivery, minimizing the cost-to-go to a specified area is especially important because travel time is the primary contributor to a delivery robot's cycle time. Similarly, we give due consideration to the energy a robot consumes when traveling to its destination and the wear and tear it slowly accumulates from moving. We make the simplifying assumption that the floor the robots are constrained to traverse is perfectly flat and that the energy the robot would expend to move in any direction is fixed, and we exploit the linear relationship between travel time, energy expended, and wear and tear to use distance traveled as a robot's cost-to-go.

For simplicity, we abstract away humans and other dynamic elements of the restaurant environment. Similarly, we remove elements that would add clutter to the environment like tables and chairs. We generate tasks at random locations to simulate calls to the robot by human staff members and patrons.

Ultimately, we divided our implementation into three parts: (1) implementing discrete and sampling-based path planning algorithms (RRT, RRT*, etc.) and validating functionality in basic simulation, (2) building a basic auctioning system for fixed motion planning tasks based on existing research, and (3) measuring performance using Georgia Tech's Robotarium simulator.

II. Background

This project aims to solve two problems in the multi-robot systems domain, task allocation (*which robot does what*) and path planning (*how to move each robot from its current location to a target location*). This section provides an overview of both problems and their related works.

A. Multi-robot Task Allocation

The problem of deciding which robot should execute a given task in a multi-robot system is called Multi-Robot Task Allocation (MRTA). MRTA aims to coordinate a large number of robots to complete a set of tasks with specific constraints to maximize or minimize a given objective function. Various methods have been employed to solve the MRTA problem, some of these methods include market-based approaches (e.g, [7], [9], [8]), optimization-based methods (e.g, [10], [12], [11]) and behavior-based methods (e.g, [14], [13]). A market-based approach, *auctions* was used in this project for allocating tasks to each robot in the system.

B. Auctions:

In market-based approaches, robots on the team act as self-interested agents operating in a virtual economy in which tasks and team resources are exchanged over the market in pursuit of individual profit. Auctions are the most common mechanisms used in market-based approaches. In an auction, a set of items is put on the market by an auctioneer in an *announcement* phase, and the participants can make an offer for these items by submitting bids to the auctioneer [19]. Once all bids are received or a pre-specified deadline has passed, the auction is then cleared in the *winner determination* phase by the auctioneer who decides which items to award and to whom. In multi-robot applications, the items for sale are typically tasks, roles, or resources. The bid prices reflect each robot's cost or utilities associated with completing a task, satisfying a role, or utilizing a resource. There are many different types of auctions, which are typically differentiated based on the items that are auctioned and the structure of the auction. Tasks may be allocated in single-item auctions(e.g, [20], [21]), multi-item auctions (e.g., [8], [22]) or combinatorial auctions (e.g., [23], [24]). Sequential single-item auctions are relatively simple and only permit the allocation of one task at a time while multiple tasks may be allocated in parallel in a given round in multi-item auctions. Conversely, combinatorial auctions involve combinations of tasks being auctioned as a group. The auction method used in this project is a multi-item auction were multiple tasks are allocated to multiple robots in parallel at each auction round.

C. Path planning:

Path planning algorithms attempt to establish paths and movements that allow mobile robots to reach a goal while navigating autonomously and safely in an environment. These algorithms generate an optimal solution for a robot to transition from an initial state to a final state while avoiding static and dynamic obstacles present in the environment. In multi-robot systems, path planning extends beyond individual robot motion to account for coordination, collision avoidance, and task completion efficiency. Existing multi-agent path planning algorithms include the Conflict Based Search algorithm [15], A* extensions [16] and [17], and the Increasing Cost Tree Search algorithm [18].

D. Combined task allocation and path planning approaches

The combined problem of multi-robot task allocation and path planning has been studied and addressed by many researchers. For instance, Camisa et al. [27] proposed a collaborative task allocation and path planning method for unmanned aerial vehicles (UAVs) and unmanned ground vehicles (UGVs). However, their work did not consider the environmental obstacles. Turpin et al. [25] introduced a concurrent task allocation and trajectory planning method that enables robots to combine time-optimized trajectories and speed adjustments. Nawaz et al. [26] explored the application of Markov Decision Processes (MDP) for addressing the unified problem of task allocation and path planning in underwater environments. Also, Aryan et al [28] tackled task allocation and path planning issues by combining the advanced SMT solver in [29] and the improved Conflict-Based Search (CBS) algorithm in [30].

For this project we employed an auction-based MRTA algorithm adapted from [6] and used RRT, RRT*, RRT-Connect, and RRT*-Connect for the multi-robot path planning.

III. Algorithm Description

This section describes the algorithms implemented in this paper and specifies how they connect to each other and why they were selected.

A. Task Allocation

We required a simple auction-based framework that allows either a dedicated auctioneer or a team leader to instruct robots to bid on a task, receive bids, and award the subject task to the winning robot. We arranged for task generation to occur in a process external to the auctioning algorithm to mimic calls to the robot to travel to given destinations, likely initiated by human staff members and patrons per our motivating example.

We required four variations of the same general auctioning method. Each method differs from the others in how it prioritizes tasks for bidding and assignment. We defined these four auctioning methods as follows: (1) robots bid on and are assigned tasks with higher rewards before bidding on and being assigned tasks with lower rewards, (2) robots bid on and are assigned older tasks before bidding on and being assigned newer tasks, (3) robots bid on tasks in a random order as a baseline, and (4) robots bid on all tasks and tasks with lower costs are awarded before tasks with higher costs.

This necessitated a four-stage auctioning process, which we opted to base on the algorithms presented in Presentation Paper 13, *Extending Behavior Trees with Market-Based Task Allocation in Dynamic Environments* [6] due to their similarity to steps in common ascending bid auctions (also known as English auctions). Common ascending bid auctions have the advantage of being straightforward in both concept and implementation. Furthermore, we found their computational complexity relatively low for our purposes, with the possible exception of Method 4.

All robots bid on discrete tasks located at different coordinates. The auctioneer (1) announces an auction is starting and sends a list of tasks to the robots to bid on, (2) collects the bids the robots submit, (3) ranks the bids for each task, and (4) awards tasks to groups of robots based on their capabilities and the cost of performing said tasks. Unlike ascending bid auctions, the auctioneer seeks to minimize costs incurred by robot teams. For the purposes of this project, we consider only each robot's cost-to-go to the target location as stated in Section I.

Assigned robots submit bids with infinite cost-to-go to the auctioneer to ensure they do not receive a new task if they have already been assigned to or are executing a task. In Method 4, robots bids are marked as bids retroactively if they are assigned to a task in the current Selection iteration.

Costs-to-go are computed by each robot using RRT, RRT*, or RRT*-Connect. All robots use the same algorithm in any given experiment. Varying path planners allows us to determine which planner performs best in the specific environment selected.

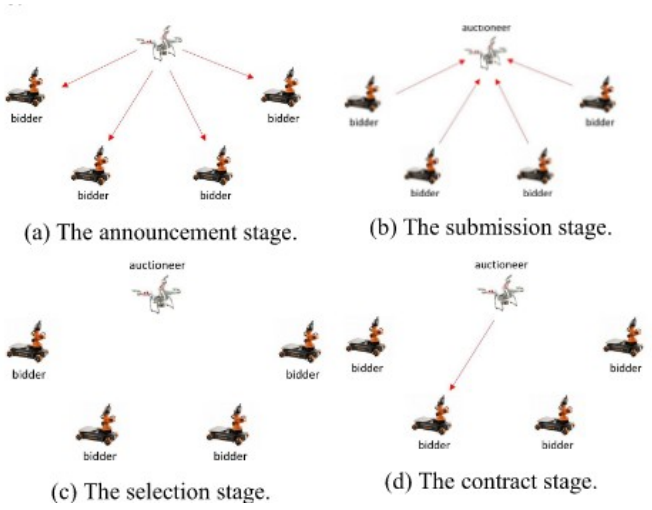


Fig. 2. Auction Stages [6]

In Figure 2, Stage 1 is the Announcement Stage, in which the auctioneer creates and broadcasts a list of unassigned tasks (L_1 refers to the task list and l_i refers to the individual tasks) to all robots (R_1 refers to the list of all robots and r_i refers to the individual robots). At program initialization, the auctioneer receives an initial list of tasks (T_1 refers to the input task list) and their associated rewards (P_1 refers to the input rewards list, p_i refers to the input individual rewards, and V_1 refers to the existing rewards list). At this point, all tasks are unallocated and all robots are available. Later, new task lists are passed to the auctioneer as they are created. The auctioneer runs auctions soon after it receives a task list and reruns auctions after a ten second delay if the auction is unsuccessful (i.e., if tasks remain on the task list, indicating no robot was awarded those tasks). For the purposes of this project, tasks are tuples corresponding to accessible (x, y) destinations in the environment. The Announcement Stage algorithm is shown in Algorithm 1.

Stage 2 is the Submission Stage, in which robots create and send bids based on the task list they received in Stage 1. Each robot computes the cost of performing each task in the provided list based on the objective function. Here, the reported cost is simply the cost-to-go returned by one of the path planning algorithms described in a later subsection of this document. Each robot passes the auctioneer a single message containing its cost for each individual task on the provided list. Here, L_2 is the cost associated with each task the robot is bidding on. The Submission Stage algorithm is shown in Algorithm 2.

Stage 3 is the Selection Stage, in which the auctioneer assigns teams of robots to tasks based on the contents of their bids in accordance with one of the four methods described previously. As written, the auctioneer iterates through the task list sequentially from the first position to the last position. If some tasks are not assigned, they remain on the list considered

Algorithm 1 Auctioning: Announcement [6]

Input: The set of new tasks T_1 and the set of associated rewards P_1
Output: Broadcast the task list L_1
if Method 1 **then**
  if $L_1$is not None **then**
    $L_1 = L_1 \cup T_1$
    $V_1 = P_1 \cup T_1$
  else
    $L_1 = T_1$
  Get Sorted Indices of V_1, Ascending
  $L_1 = L_1$(Sorted Indices)
else if Method 2 **then**
  if $L_1$is not None **then**
    $L_1 = L_1 \cup T_1$
  else
    $L_1 = T_1$
else if Method 3 **then**
  if $L_1$is not None **then**
    $L_1 = L_1 \cup T_1$
    $L_1 =$ Shuffle(L_1)
  else
    $L_1 =$ Shuffle(T_1)
else if Method 4 **then**
  if $L_1$is not None **then**
    $L_1 = L_1 \cup T_1$
  else
    $L_1 = T_1$
Broadcast $L_1$to all robots

Algorithm 2 Auctioning: Submission [6]

Input: The task list L_1
Output: The cost of each task L_2
Initialize a task list with bids L_2
for each $l_i$in L_1 **do**
  if Robot Unavailable **then**
    $bid = \infty$
  else
    $bid =$ Cost based on path planner
  $L_2 \leftarrow L_2 \cup \{l_i, bid\}$
Respond to auctioneer with message L_2

Algorithm 3 Auctioning: Selection [6]

Input: The tasks list with cost L_2
Output: The robot's awarded task l_i
if $min(bids == \infty)$ **then** pass
else
  if Method 1 *or* Method 2 *or* Method 3 **then**
    for each $l_i$in L_2 **do**
      $r_3 = r_1$(Index of $min(bid)$)
      L_2(Index of $min(bid)) = \infty$
      Inform Winning Robot It Has Won l_i
  else
    for each i in $RangeofLength(L_1)$ **do**
      for each l_i in L_1 **do**
        if O_3 is not None **then**
          $O_3 \leftarrow O_3 \cup (min(bid))$
          $W_3 \leftarrow W_3 \cup$ (Index of $min(bid)$)
        else
          $O_3 = (min(bid))$
          $W_3 =$ (Index of $min(bid)$)
      $r_3 = R_1(W_3)$(Index of $min(O_3)$)
      $l_i = L_2$(Index of $min(O_3)$
      Inform Winning Robot It Has Won l_i
      Change All of the Winning Robot's Bids to ∞
      Change All of the Assigned Task's Bids to ∞
  Read or initialize the list of selected robots L_3
  for each t_j in L_1 **do**
    if $count(L_3(t_j)) < V(t_j)$ **then**
      $L_3 \leftarrow L_3 \cup \{t_j, r_i, bid_j\}$
    else
      if $L_2(bid_j) < L_3(bid_j)$ **then**
        update L_3 with $(t_j, r_i, bid_j) in L_2$
  if receive all bids **then**
    Inform all robots in L_3 to accept the assigned tasks
    clear L_3

Algorithm 4 Auctioning: Contracting [6]

Input: The robot's awarded task l_i
Output: Task execution
lookup stored path corresponding to l_i and execute

in the following auction. That done, the auctioneer passes the winning robots their assigned tasks.

Here, r_3 is the selected robot, W_3 is the list of minimum bid indices per task, w_3 is an individual minimum bid index per task, and O_3 is the list of minimum bids per task, and o_3 is an individual minimum bid per task. The Selection Stage algorithm is shown in Algorithm 3.

Stage 4 is the Contract Stage, in which the winning robots commence performance of their assigned tasks. One robot can only have one task assigned to it at any given time. The Contract Stage algorithm is shown in Algorithm 4.

B. Rapidly-exploring Random Tree (RRT)

The Rapidly-exploring Random Tree (RRT) is a sampling-based algorithm used for path planning [2]. RRT enables agents to quickly plan paths from one point to another in an environment where goal locations may be unknown. This allows robots to navigate environments that are not known a priori.

RRT and the selected RRT variants are appropriate for this application because they allow for reconfiguration of the environments the robots are operating in between runs. If we added tables, chairs, and other similar obstacles to the map, using these sampling-based planners would allow us to

simulate the robot moving through a changing space, albeit one that does not change while the robot is online.

The RRT algorithm works as follows. The algorithm initializes a tree T with an initial state x_{init}, which is the robot's current position. A random sample of the state space is taken $x_{rand} \in X$, and the nearest node $x_{nearest}$ in the tree is found. If x_{rand} is more than some distance δ from $x_{nearest}$ the sample is steered towards $x_{nearest}$ until it is within δ resulting in x_{new}. Next, the node, x_{new}, and the edge, $(x_{new}, x_{nearest})$, are checked for collisions. If both the node and edge are collision free, the node and edge are added to the graph. Lastly, if the node x_{new} is within δ of the goal node x_{goal}, the edge (x_{new}, x_{goal}) is checked for collision. If the edge is collision free, it is added to the tree, and a path is returned from the initial state x_{init} to the goal state x_{goal}. The algorithm can also be biased towards the goal by adding a probability to sample the goal node instead of taking a random sample. The RRT algorithm is shown in Algorithm 5.

Algorithm 5 Rapidly-exploring Random Trees (RRT)

Input: Initial state x_{init}, goal state x_{goal}, step size δ
Output: Path from x_{init} to x_{goal}
$T \leftarrow \{x_{init}\}$
while dist$(x_{new}, x_{goal}) > \delta$ **do**
  $x_{rand} \leftarrow$ randomSample()
  $x_{nearest} \leftarrow$ nearestNeighbor(T, x_{rand})
  $x_{new} \leftarrow$ steer$(x_{nearest}, x_{rand}, \delta)$
  if collisionFree$(x_{nearest}, x_{new})$ **then**
    Add x_{new} to T with edge $(x_{nearest}, x_{new})$
    if dist$(x_{\text{new}}, x_{\text{goal}}) < \delta$
    and collisionFree$(x_{\text{new}}, x_{\text{goal}})$ **then**
      Add edge (x_{new}, x_{goal}) to T
      return getPath(x_{init}, x_{goal})
return *Failure*

C. RRT*

The RRT* algorithm is an optimized variant of RRT which utilizes rewiring to improve path quality [5]. Compared to RRT, RRT* provides asymptotic optimality, meaning the solution will approach the optimal path as a greater number of iterations are performed.

The RRT* algorithm works as follows. The algorithm works very similarly to RRT, but instead of connecting x_{new} to the closest node, it checks surrounding nodes to find the parent node x_{parent} which provides the lowest-cost path to x_{new}. Once x_{new} is added, the tree checks nearby nodes to see if passing through x_{new} will reduce their path cost. This rewiring step reduces path costs and prevents branches from moving backwards. The algorithm continues this loop until it can connect to x_{goal}, and then returns a path from x_{init} to x_{goal}. The RRT* algorithm is shown in Algorithm 6.

D. RRT-Connect

RRT-Connect is a bidirectional version of RRT [3]. Rather than growing a single tree from the start node, RRT-Connect grows two trees. The advantage of RRT-Connect over RRT is that the search space is much smaller. If the distance from the goal to the start location is R, then RRT will have to search a space of approximately πR^2, whereas RRT-Connect will search a space of approxmately $\pi(\frac{R}{2})^2$.

The algorithm for RRT-Connect works as follows. The algorithm initializes two trees, T_{start} and T_{goal}, with the root nodes being the start node x_{init} and goal node x_{goal}, respectively. The extend algorithm shown in 8 the generates and checks a sample point x_{new} for validity. If the point is valid, it adds it to the tree (either T_{start} or T_{goal}) and returns a status if the tree advanced, was trapped, or if the sample was close enough to directly reach the tree. The algorithm then checks if x_{new} can connect to the opposite tree, shown in the connect algorithm 7. If it can, the point is connected and the path is returned, otherwise the connect algorithm returns **Trapped** and the algorithm continues. The algorithm repeats this process swapping which tree is expanded with each iteration. The trees in RRT-Connect grow in the same way as RRT and so are not guaranteed to return an optimal path. The path returned will depend on the point at which the two trees connect to one another. The RRT-Connect algorithm is shown in Algorithm 9.

Algorithm 6 RRT*

Input: Initial state x_{init}, goal state x_{goal}, step size δ
Output: Asymptotically optimal path from x_{init} to x_{goal}
$T \leftarrow \{x_{init}\}$
while dist$(x_{new}, x_{goal}) > \delta$ **do**
  $x_{rand} \leftarrow$ randomSample()
  $x_{nearest} \leftarrow$ nearestNeighbor(T, x_{rand})
  $x_{new} \leftarrow$ steer$(x_{nearest}, x_{rand})$
  if collisionFree(x_{new}) **then**
    $x_{parent} \leftarrow$ chooseParent(T, x_{new})
    if collisionFree(x_{new}, x_{parent}) **then**
      Add x_{new} to T with edge (x_{parent}, x_{new})
      for $x_{near} \in$ neighborhood(T, x_{new}) **do**
        if cost(x_{new}) + cost(x_{new}, x_{near}) < cost(x_{near}) **and** collisionFree(x_{new}, x_{near}) **then**
          rewire(x_{new}, x_{near})
return getPath(x_{init}, x_{goal})

Algorithm 7 connect

Input: Tree T, target state x_{target}
Output: Reached if connection is made, else **Trapped**
while True **do**
  $S \leftarrow$ extend(T, x_{target})
  if $S \neq$ **Advanced then**
    return S

E. RRT*-Connect

RRT*-Connect is a bidirectional, asymptotically optimal sampling-based motion planning algorithm that combines the

Algorithm 8 extend

Input: Tree T, target sample x_{target}
Output: Status of Tree resulting from adding new point
$x_{nearest} \leftarrow$ nearestNeighbor(T, x_{target})
$x_{new} \leftarrow$ steer$(x_{nearest}, x_{target})$
if collisionFree$(x_{nearest}, x_{new})$ **then**
    Add x_{new} to T with edge $(x_{nearest}, x_{new})$
    if $x_{new} = x_{target}$ **then**
        return Reached
    else
        return Advanced
else
    return Trapped

Algorithm 9 RRT-Connect

Input: Initial state x_{init}, goal state x_{goal}, max iterations N
Output: Path from x_{init} to x_{goal}
$T_{start} \leftarrow \{x_{init}\}$
$T_{goal} \leftarrow \{x_{goal}\}$
for $i = 1$ to N **do**
    $x_{rand} \leftarrow$ randomSample()
    $x_{new} \leftarrow$ extend(T_{start}, x_{rand}) ▷ *Grows the tree in the same way as RRT, with the newly sampled point*
    if not **Trapped then**
        if connect (T_{goal}, x_{new}) = **Reached then** ▷ *If x_{new} is close enough to the other tree to connect to one of the nodes*
            return getPath$(T_{start}, T_{goal}, x_{new})$
    swap(T_{start}, T_{goal}) ▷ *Swaps which tree you are expanding for the next iteration of the loop*
return *Failure*

exploration efficiency of RRT-Connect with the asymptotic path optimality guarantees of RRT* [4]. Similar to RRT-Connect, the algorithm grows two trees, one from the start node x_{init} and one from the goal node x_{goal}, but like RRT*, the algorithm rewires the Trees to improve the returned path.

The algorithm for RRT*-Connect works as follows. The algorithm initializes two trees, T_{start} and T_{goal}, with the root nodes being the start node x_{init} and the goal node x_{goal}, respectively. The extendStar algorithm shown in 11 the generates and checks a sample point x_{new} for validity. If the point is valid, the algorithm looks for the best parent node to connect the new point to. Once a parent has been found, the algorithm adds the point to the tree (either T_{start} or T_{goal}) and returns a status if the tree advanced, was trapped, or if the sample was close enough to directly reach the tree. The algorithm then checks if x_{new} can connect to the opposite tree, shown in the connectStar algorithm 10. If it can, the point is connected and the path is returned, otherwise the connect algorithm returns **Trapped** and the algorithm continues. The algorithm repeats this process swapping which tree is expanded with each iteration. As iterations continue, the rewiring improves the path quality, making RRT*-Connect asymptotically optimal. Unlike RRT-Connect, the RRT*-Connect algorithm does not terminate until the maximum number of iterations has been reached. This altered termination condition allows RRT*-Connect to continue rewiring the tree to improve path quality even once an initial path is discovered from the start to the goal. The RRT*-Connect algorithm is shown in Algorithm 12.

Algorithm 10 connectStar

Input: Tree T, target state x_{target}
Output: Reached if connection is made, else **Trapped**
while True **do**
    $(S, x_{new}) \leftarrow$ extendStar(T, x_{target})
    if $S \neq$ **Advanced then**
        return S

Algorithm 11 extendStar

Input: Tree T, target sample x_{target}
Output: Status
$x_{nearest} \leftarrow$ nearestNeighbor(T, x_{target})
$x_{new} \leftarrow$ steer$(x_{nearest}, x_{target})$
if collisionFree$(x_{nearest}, x_{new})$ **then**
    $X_{near} \leftarrow$ neighborhood(T, x_{new})
    $x_{min} \leftarrow x_{nearest}$
    $c_{min} \leftarrow$ cost$(x_{nearest})$ + dist$(x_{nearest}, x_{new})$
    for $x_{near} \in X_{near}$ **do**
        if collisionFree(x_{near}, x_{new}) **then**
            if cost(x_{near}) + dist$(x_{near}, x_{new}) < c_{min}$ **then**
                $x_{min} \leftarrow x_{near}$
                $c_{min} \leftarrow$ cost(x_{near}) + dist(x_{near}, x_{new})
    Add x_{new} to T with edge (x_{min}, x_{new})
    for $x_{near} \in X_{near}$ **do**
        if cost(x_{new}) + dist$(x_{new}, x_{near}) <$ cost(x_{near})
        and collisionFree(x_{new}, x_{near}) **then**
            rewire(x_{new}, x_{near})
    if $x_{new} = x_{target}$ **then**
        return Reached
    else
        return Advanced
else
    return Trapped

Algorithm 12 RRT*-Connect

Input: Initial state x_{init}, goal state x_{goal}, max iterations N
Output: Asymptotically optimal path from x_{init} to x_{goal}
$T_{start} \leftarrow \{x_{init}\}$
$T_{goal} \leftarrow \{x_{goal}\}$
for $i = 1$ to N **do**
    $x_{rand} \leftarrow$ randomSample()
    $x_{new} \leftarrow$ extendStar(T_{start}, x_{rand})
    if not **Trapped then**
        connectStar(T_{goal}, x_{new})
    swap(T_{start}, T_{goal})
return getPath(T_{start}, T_{goal})

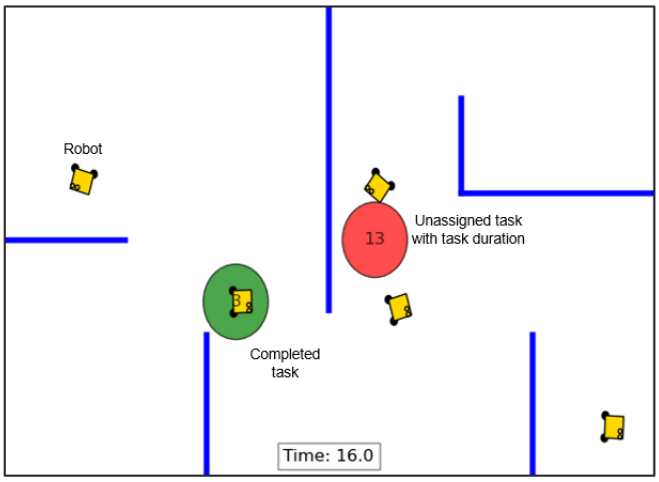


Fig. 3. Simulator with Robots and Tasks

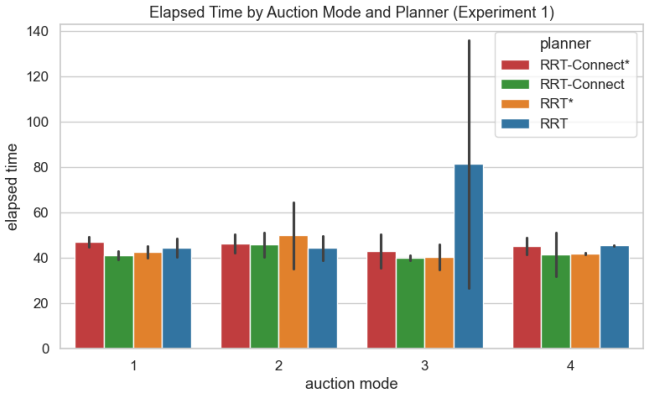


Fig. 4. Tasks Completion Time by Each Auction Mode and Path Planner

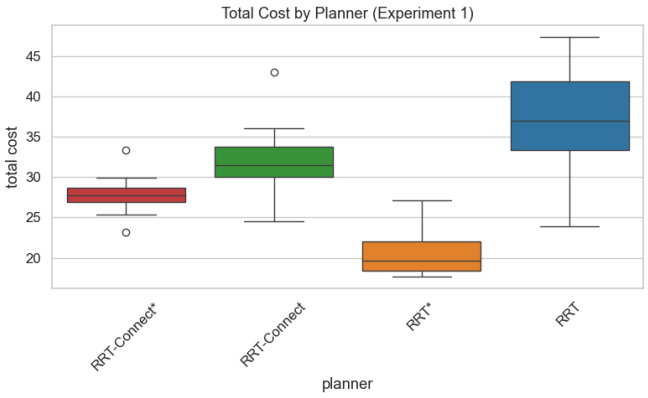


Fig. 5. Total Cost by Each Path Planner

IV. Simulation and Results

A. Simulator

To evaluate the performance of the four path-planning algorithms in the market-based auctioning system (RRT, RRT*, RRT-Connect, and RRT*-Connect) we use the existing simulator for the Robotarium at Georgia Tech [1]. The Robotarium simulator allows us to run multi-agent experiments with up to 20 agents and handles collision avoidance and low-level control for us. Collision avoidance is achieved with the built-in barrier certificate functions of the Robotarium [1]. The barrier certificates prevent the agents from colliding while still allowing them to maintain a trajectory as close as possible to the original input. The in-built hybrid controller allows us to drive agents to nodes on our path planner without having to compute wheel velocities ourselves. Using the Robotarium simulator also allows us to submit experiments to test our algorithms on real robotic hardware allowing us to avoid the pitfalls of the sim-to-real gap. Figure 3 shows the layout of the simulator with the robots and associated tasks used for our experiments.

B. Experimental Trials and Results

In our experimental runs, we varied the starting locations of the robots, the number and frequency of new tasks, the auction modes, and the path planning algorithms. The four auction modes employed in this project are listed below.

- Mode 1: Task cost minimized in highest to lowest reward priority
- Mode 2: Task cost minimized in oldest task to newest task priority
- Mode 3: Random task assignment
- Mode 4: Task cost minimized without priority

For the first simulation, we conducted two trials for each auction mode with five robots and nine tasks. The starting locations of the robots were spread out along one side of the map, with three tasks appearing every ten seconds. All four path planning algorithms were used for each trial and each task required one robot to complete it. The time taken to complete all tasks with each auction mode and planner for this simulation is shown in Figure 4. The total path cost for each planner is also shown in Figure 5. It can be seen that RRT-Connect had the fastest completion time for all tasks in the various auction modes. This is due to the bidirectional and non-rewiring nature of RRT-connect, enabling robots to move quickly and execute assigned tasks. The RRT planner in auction mode three produced significant variances because RRT finds paths to task locations without considering the cost of those paths. As a result, robots that are farther away from a task may win bids over robots that are closer due to the paths generated by the RRT planner. In terms of path cost, RRT* and RRT*—Connect found paths with the least cost compared to the other path planning algorithms. This is because of the asymptotic optimality of the two algorithms.

For the second simulation, we conducted ten trials per path planning algorithm employing only auction mode four. We used five robots for nine tasks in this experiment. The starting positions of the robots were randomized and three tasks appeared every ten seconds. The total time taken by each planner to complete the assigned tasks is shown in Figure 6 and the total path cost for each path planning algorithm is also

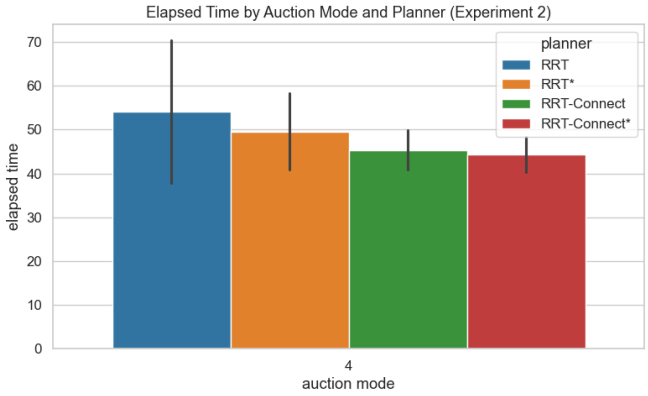


Fig. 6. Tasks Completion Time by Auction Mode 4 and Path Planner

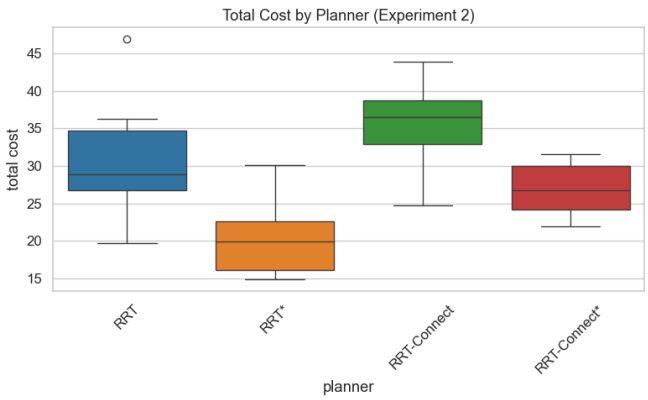


Fig. 7. Total Cost by Each Path Planner using Auction Mode 4

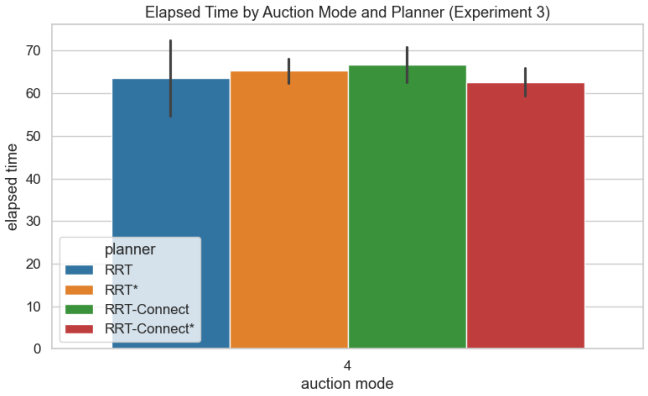


Fig. 8. Tasks Completion Time by Auction Mode 4 and Path Planner

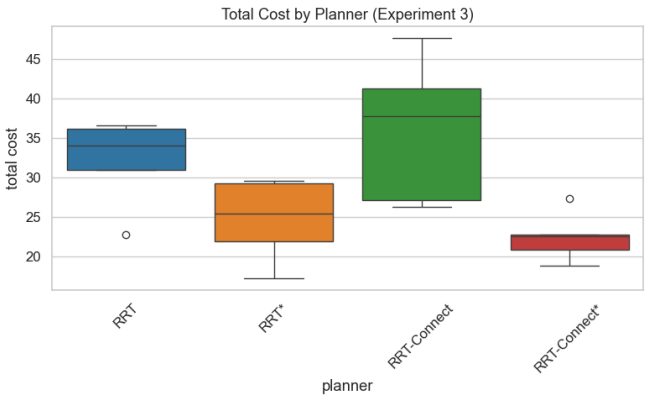


Fig. 9. Total Cost by Each Path Planner using Auction Mode 4

shown in Figure 7. Auction mode four showed low variances when multiple trials were conducted. The differences in task completion time and total path costs were due to the nature of the path-planning algorithms used. Bidirectional planners completed the tasks more quickly than unidirectional planners, while the asymptotically optimal planners (RRT* and RRT*-Connect) achieved better results regarding the total path cost for completing all tasks.

The last simulation involved five trials per path planning algorithm with nine tasks and two robots. The auction mode implemented in this simulation was auction mode four with randomized starting positions for each robot. Figure 8 shows the elapsed time by the auction mode and planner. The total cost by planner is shown in Figure 9.

V. Conclusion

This project explored the effectiveness of auction-based multi-robot task allocation and path planning for vehicle routing using RRT, RRT*, RRT-Connect, and RRT*-Connect. We implemented four auctioning methods that varied based on task prioritization and cost minimization. Through simulation-based testing using the Robotarium platform, we evaluated the performance of the combined multi-robot task allocation and path planning approaches in terms of task completion time and total path cost. RRT-Connect achieved the fastest task completion times due to its bidirectional search, while RRT* and RRT*-Connect produced lower path costs due to their asymptotic optimality. Auction mode 4 (cost minimization without priority) showed the most consistent performance while auction mode 3 had significant variances in task completion time. The results demonstrate that combining auction-based task allocation with effective path planning improves the performance of the multi-robot system in complex environments. For future work, we intend to investigate the performance of our solution in dynamic environments.

References

[1] S. Wilson et al., "The Robotarium: Globally Impactful Opportunities, Challenges, and Lessons Learned in Remote-Access, Distributed Control of Multirobot Systems," in IEEE Control Systems Magazine, vol. 40, no. 1, pp. 26-44, Feb. 2020, doi: 10.1109/MCS.2019.2949973.

[2] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Tech. Rep. TR 98-11, Department of Computer Science, Iowa State University, 1998.

[3] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," Proceedings ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), San Francisco, CA, USA, 2000, pp. 995-1001 vol.2, doi: 10.1109/ROBOT.2000.844730. keywords:

Path planning;Computer science;Space exploration;Algorithm design and analysis;Humans;Animation;Robotic assembly;Buildings;Tree graphs;Kinematics,

[4] S. Klemm et al., "RRT*-Connect: Faster, asymptotically optimal motion planning," 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO), Zhuhai, China, 2015, pp. 1670-1677, doi: 10.1109/ROBIO.2015.7419012. keywords: Planning;Algorithm design and analysis;Probabilistic logic;Legged locomotion;Search problems;Joining processes,

[5] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli and S. Teller, "Anytime Motion Planning using the RRT*," 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 2011, pp. 1478-1483, doi: 10.1109/ICRA.2011.5980479.

[6] T. Wang, D. Shi, and W. Yi, "Extending behavior trees with market based task allocation in dynamic environments," in Proceedings of the 2020 4th International Symposium on Computer Science and Intelligent Control, 2020, pp. 1–8.

[7] B. Gerkey, M. Mataric, Sold!: auction methods for multirobot coordination, IEEE Transactions on Robotics and Automation 18 (5) (2002) 758–768. doi:10.1109/TRA.2002.803462.

[8] H.-L. Choi, L. Brunet, J. P. How, Consensus-based decentralized auctions for robust task allocation, IEEE Transactions on Robotics 25 (4) (2009) 912–926. doi:10.1109/TRO.2009.2022423.

[9] S. Botelho, R. Alami, M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement, in: Proceedings 1999 IEEE International Conference on Robotics and Automation, 1999, pp. 1234–1239. doi:10.1109/ROBOT.1999.772530.

[10] J. Wang, Y. Gu, X. Li, Multi-robot Task Allocation Based on Ant Colony Algorithm, Journal of Computers 7 (9) (2012) 2160–2167. doi: 10.4304/jcp.7.9.2160-2167.

[11] M. Kasahara, K. Cai, Online Supervisory Control with Optimal Task Assignment for Efficient and Adaptive Warehouse Automation, Vol. 63, 2020, pp. 90–93. doi:10.11511/jacc.63.0_90.

[12] X. Kong, Y. Gao, T. Wang, J. Liu, W. Xu, Multi-robot Task Allocation Strategy based on Particle Swarm Optimization and Greedy Algorithm, in: 2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC), 2019, pp. 1643–1646. doi: 10.1109/ITAIC.2019.8785472.

[13] F. Aljalaud, H. A. Kurdi, Autonomous task allocation for multi-uav systems based on area-restricted search behavior in animals, Procedia Computer Science 191 (2021) 246–253. doi:https://doi.org/10.1016/j.procs.2021.07.031.

[14] B. B. Werger, M. J. Mataric, Broadcast of local eligibility: behaviorbased control for strongly cooperative robot teams, in: Proceedings of the fourth international conference on Autonomous agents, 2000, pp. 21–22. doi:10.1145/336595.336621.

[15] Sharon, G., Stern, R., Felner, A., Sturtevant, N.R.: Conflict-based search for optimal multi-agent pathfinding. Artif. Intell. 219, 40–66 (2015)

[16] Standley, T.S.: Finding optimal solutions to cooperative pathfinding problems. In: AAAI Conference on Artificial Intelligence, pp. 173–178 (2010)

[17] Erokhin, A., Erokhin, V., Sotnikov, S., & Gogolevsky, A. (2018). Optimal multi-robot path finding algorithm based on a. In Proceedings of the Computational Methods in Systems and Software (pp. 172–182). Springer.

[18] Sharon, G., Stern, R., Goldenberg, M., Felner, A.: The increasing cost tree search for optimal multi-agent pathfinding. Artif. Intell. 195, 470–495 (2013)

[19] M. B. Dias, R. Zlot, N. Kalra and A. Stentz, "Market-based multirobot coordination: A survey and analysis", Proceedings of the IEEE, vol. 94, no. 7, pp. 1257-1270, 2006.

[20] Lagoudakis MG, Berhault M, Koenig S, Keskinocak P, Kleywegt AJ. Simple auctions with performance guarantees for multirobot task allocation. In: 2004 IEEE/RSJ international conference on intelligent robots and systems (IROS)(IEEE Cat. No. 04CH37566). IEEE; 2004;1:698–705.

[21] Koenig S, Tovey C, Lagoudakis M, Markakis V, Kempe D, Keskinocak P, Kleywegt A, Meyerson A, Jain S. The power of sequential single-item auctions for agent coordination. In: AAAI. 2006 p. 1625–1629

[22] Viguria A, Maza I, Ollero A. Set: An algorithm for distributed multi-robot task allocation with dynamic negotiation based on task subsets. In: Proceedings 2007 IEEE international conference on robotics and automation. IEEE; 2007. p. 3339–3344

[23] Berhault M, Huang H, Keskinocak P, Koenig S, Elmaghraby W, Griffin P, Kleywegt A. Robot exploration with combinatorial auctions. In: Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453). IEEE; 2003;2:1957–1962.

[24] Wen X, Zhao Z-G. Multi-robot task allocation based on combinatorial auction. In: 9th International conference on control, mechatronics and automation (ICCMA). IEEE; 2021. p. 27–32.

[25] M. Turpin, N. Michael, and V. Kumar, "Capt: Concurrent assignment and planning of trajectories for multiple robots," The International Journal of Robotics Research, vol. 33, no. 1, pp. 98–112, 2014.

[26] F. Nawaz and M. Ornik, "Multiagent, multitarget path planning in markov decision processes," IEEE Transactions on Automatic Control, vol. 68, no. 12, pp. 7560–7574, Dec. 2023.

[27] A. Camisa, A. Testa, and G. Notarstefano, "Multi-robot pickup and delivery via distributed resource allocation," IEEE Transactions on Robotics, vol. 39, no. 2, pp. 1106–1118, Apr. 2023.

[28] A. Aryan, M. Modi, I. Saha, R. Majumdar, and S. Mohalik, "Optimal integrated task and path planning and its application to multi-robot pickup and delivery," arXiv preprint arXiv:2403.01277, 2024.

[29] L. De Moura and N. Bjørner, "Z3: An efficient smt solver," in International conference on Tools and Algorithms for the Construction and Analysis of Systems. Springer, 2008, pp. 337–340.

[30] H. Zhang, J. Chen, J. Li, B. Williams, and S. Koenig, "Multi-agent path finding for precedence-constrained goal sequences," in International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2022.

[31] Centers for Disease Control and Prevention. "COVID-19 Timeline." CDC Museum, https://www.cdc.gov/museum/timeline/covid19.html. Accessed 19 Mar. 2025.

[32] Centers for Disease Control and Prevention. "About COVID-19." Centers for Disease Control and Prevention, 13 Nov. 2024, https://www.cdc.gov/covid/about/index.html. Accessed 19 Mar. 2025.

[33] Xtalks. "How COVID-19 Gave Rise to Food Delivery Robots." Xtalks, https://xtalks.com/how-covid-19-gave-rise-to-food-delivery-robots-2903/. Accessed 19 Mar. 2025.

[34] Yi Jiang, Po-Lin Lai, Ching-Chiao Yang, Xinchen Wang, Exploring the factors that drive consumers to use contactless delivery services in the context of the continued COVID-19 pandemic, Journal of Retailing and Consumer Services, Volume 72, 2023, 103276, ISSN 0969-6989, https://doi.org/10.1016/j.jretconser.2023.103276. (https://www.sciencedirect.com/science/article/pii/S0969698923000231)

[35] Valinsky, Jordan. "Major Restaurant Chains That Have Closed Down Locations Across the U.S." CNN Business, CNN, 25 Feb. 2023, https://www.cnn.com/2023/02/25/business/restaurant-closures/index.html.

[36] Pudu Robotics. "BellaBot." Pudu Robotics, Pudu Robotics, https://www.pudurobotics.com/product/detail/bellabot.