AUCTION-CONSENSUS ALGORITHMS FOR DECENTRALIZED TASK ALLOCATION IN MULTI-ROBOT SYSTEMS

A Thesis

by

JOSE ALBERTO RODRIGUEZ

Submitted in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE IN ENGINEERING

Major Subject: Electrical Engineering

The University of Texas Rio Grande Valley
May 2026

AUCTION-CONSENSUS ALGORITHMS FOR DECENTRALIZED TASK ALLOCATION IN MULTI-ROBOT SYSTEMS

A Thesis
by
JOSE ALBERTO RODRIGUEZ

COMMITTEE MEMBERS

Wenjie Dong
Chair of Committee

Qi Lu
Co-Chair of Committee

Rogelio Soto
Committee Member

Weidong Kuang
Committee Member

Dongchul Kim
Committee Member

May 2026

Copyright 2026 Jose Alberto Rodriguez

All Rights Reserved

ABSTRACT

Rodriguez, Jose A., Auction-Consensus Algorithms for Decentralized Task Allocation in Multi-Robot Systems. Master of Science in Engineering (M.S.E.), May 2026, 54 pp., 18 figures, 27 references.

This thesis investigates methods for improving decentralized multi-robot task allocation (MRTA) through enhancements to auction-consensus coordination. The first contribution of this work introduces the Auction-Consensus Algorithm with Loss Mechanism (ACALM), a deterministic extension that incorporates loss-aware bidding. Through dynamic loss propagation, agents rebid in a manner that better reflects global coordination impact under the Single-Task Robot, Single-Robot Task, Instantaneous Assignment (ST-SR-IA) assumptions. The second contribution replaces handcrafted bidding rules with learned neural bidding policies trained under a centralized training with decentralized execution framework. Using reinforcement learning, agents learn cooperative bidding strategies that improve allocation quality under the Multi-Task Robot, Single-Robot Task, Instantaneous Assignment (MT-SR-IA) assumption. Together, these results demonstrate that both structured loss mechanisms and data-driven bidding adaptation significantly narrow the performance gap between decentralized auction-consensus algorithms and centralized optimal assignments.

DEDICATION

For my loved ones, whose unwavering support and encouragement have made this journey possible.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor, Dr. Dong and co-advisor Dr. Lu, for their guidance and supervision, and the support of the MARS (Multiple Autonomous Robot Systems) lab. Additionally, thanks to Dr. Tarawneh and the financial support provided by the National Science Foundation (NSF) CREST Center for Multidisciplinary Research Excellence in Cyber-Physical Infrastructure Systems (No. 2112650), NSF CISE Minority Serving Institute (MSI) program (No. 2318682), and NSF CISE Expand AI Program (No. 2434916).

TABLE OF CONTENTS

LIST OF FIGURES

Page

CHAPTER I

INTRODUCTION

Multi-Robot Task Allocation (MRTA) is a fundamental problem in cooperative robotics in which a team of autonomous agents must distribute tasks among themselves in order to optimize a global performance objective. Applications of MRTA span aerial drone swarms, warehouse automation, environmental monitoring, and search-and-rescue operations, where efficient coordination directly impacts mission success. As robotic teams scale in size and operate in increasingly complex environments, the need for scalable, robust, and decentralized task allocation strategies becomes critical [1].

Traditionally, MRTA approaches have been broadly classified into centralized and decentralized frameworks [2]. Centralized methods rely on a global coordinator that collects full system information and computes an optimal or near-optimal assignment for all agents. While these approaches can achieve high solution quality, they suffer from several practical limitations, including poor scalability, high communication overhead, and vulnerability to single points of failure. In contrast, decentralized approaches distribute decision-making across agents, allowing robots to operate using local information and limited communication. Such methods naturally scale to large teams and exhibit increased robustness to communication failures and dynamic changes in the environment. However, decentralization often comes at the cost of reduced solution quality due to limited global awareness.

Among decentralized MRTA strategies, auction-based and consensus-driven algorithms have emerged as a powerful class of coordination mechanisms [3]. These methods combine local

bidding processes with distributed conflict resolution to achieve conflict-free task assignments without requiring centralized control. One of the most influential algorithms in this category is the Consensus-Based Auction Algorithm (CBAA) and its extension, the Consensus-Based Bundle Algorithm (CBBA) [4]. These frameworks enable robots to greedily select tasks based on locally computed scores while using consensus to synchronize task ownership across the team. CBBA provides strong convergence guarantees and has been widely adopted due to its simplicity, scalability, and decentralized execution.

Despite their advantages, classical auction-consensus algorithms rely heavily on hand-crafted greedy scoring functions to determine task priorities. While computationally efficient, these heuristics can lead to globally suboptimal allocations, particularly in environments with high task contention or uneven spatial distributions. Small differences in local bid values may cause agents to claim tasks that appear locally optimal but produce poor global outcomes. As a result, improving the bidding mechanisms within decentralized auction frameworks remains an important research challenge.

This thesis investigates two complementary approaches for enhancing decentralized auction-consensus algorithms for MRTA. The first approach introduces a Loss-Aware Auction-Consensus Algorithm (ACALM), which augments traditional greedy bidding with a loss propagation mechanism that captures the cost of missed task opportunities. By accounting for the compounded impact of unsuccessful bids, ACALM dynamically adjusts task priorities to mitigate globally inefficient allocations. The second approach replaces deterministic bidding rules with a learned bidding policy trained using reinforcement learning under a centralized training and decentralized execution framework. Neural network-based bidders enable agents to

learn task valuation strategies directly from data, guiding decentralized coordination toward solutions that more closely approximate global optima.

Together, these contributions aim to bridge classical distributed coordination algorithms with adaptive and data-driven decision-making techniques. By preserving the scalability and robustness of decentralized auction-consensus frameworks while improving solution quality, this work advances the state of the art in multi-robot task allocation.

The remainder of this thesis is organized as follows. Chapter II reviews existing work in decentralized MRTA, auction-consensus methods, and learning-based task allocation. Chapter III formalizes the MRTA problem and defines the system model and performance objectives. Chapter IV presents the proposed Loss-Aware Auction-Consensus Algorithm, including its formulation and theoretical properties. Chapter V introduces the learning-enhanced auction-consensus framework and neural bidding architectures. Chapter VI evaluates both approaches through extensive simulation studies. Chapter VII discusses insights, tradeoffs, and limitations, and Chapter VIII concludes the thesis and outlines future research directions.

CHAPTER II

RELATED WORK

Task allocation in multi-robot systems (MRS) has long been a central research topic in distributed robotics, particularly in scenarios where centralized coordination is infeasible due to communication constraints, scalability limitations, or robustness concerns. Over the past two decades, a wide range of decentralized MRTA strategies have been proposed, with auction-based and consensus-driven frameworks emerging as some of the most influential and practically deployable solutions. More recently, learning-based methods have been introduced to address the limitations of hand-crafted heuristics. This chapter reviews foundational and contemporary approaches relevant to this thesis, highlighting key developments and remaining challenges.

2.1 Auction-Based and Consensus-Driven MRTA Algorithms

Among decentralized task allocation strategies, auction-consensus algorithms have become a dominant paradigm due to their scalability, simplicity, and convergence guarantees. A foundational contribution in this area is the Consensus-Based Auction Algorithm (CBAA) and its extension, the Consensus-Based Bundle Algorithm (CBBA) [4]. These methods integrate local auction-style bidding with maximum consensus protocols to achieve conflict-free task assignments across a robotic team. In CBAA, each agent bids on individual tasks, while CBBA generalizes this approach by allowing agents to construct bundles of multiple tasks executed sequentially along locally optimized routes. Both algorithms guarantee convergence under connected communication topologies and provide a provable worst-case optimality bound for min-sum objectives.

Building upon the CBBA framework, numerous extensions have been proposed to address increasingly realistic operational constraints. To support tasks requiring multiple heterogeneous robots, CBBA was expanded into the Consensus-Based Group Algorithm (CBGA) [5]. Subsequent developments introduced additional constraints, such as payload limitations in CBPA and time-window and resource-load constraints in ICBGA [6][7]. In dynamic environments, algorithms such as C-CBAA and C-CDTP integrated auction-consensus mechanisms with online path planning, enabling agents to update bids as trajectories evolved in the presence of obstacles and collision avoidance requirements [8].

Auction-consensus methods have also been applied to specialized domains. In aerial combat and cooperative targeting scenarios, MTCBAA extended CBAA to handle multi-target and multi-agent engagements while maintaining bounded optimality guarantees [9]. Large-scale benchmarking efforts have evaluated the robustness of auction-based coordination under imperfect communication conditions [10], while formal verification studies have analyzed stability properties of consensus-based bidding protocols [11].

Despite their success, classical auction-consensus algorithms rely fundamentally on greedy, hand-crafted scoring functions to evaluate task desirability. These local heuristics, while computationally efficient, can produce globally suboptimal allocations when task contention is high or spatial distributions are uneven. Early bidding decisions may become locked through consensus, making recovery from poor allocations difficult. Addressing the limitations of greedy bidding remains a central challenge in improving decentralized MRTA performance.

2.2 Enhancements Beyond Greedy Bidding

Several works have sought to improve upon classical auction-consensus frameworks by introducing richer bidding metrics or alternative utility formulations. The Greedy Coalition-

Based Auction Algorithm (GCAA) incorporates dynamic task utilities to better capture evolving mission priorities [12]. Other approaches introduce novel significance or priority measures to influence bidding behavior beyond simple marginal distance or cost reductions [13].

Another emerging direction incorporates uncertainty modeling into the task allocation process. The Bayesian Optimization for Combinatorial Assignment (BOCA) framework models task preferences probabilistically, allowing agents to reason over ambiguous or uncertain reward structures [14]. Although BOCA itself is centralized, it highlights the potential benefits of belief-driven and probabilistic bidding mechanisms for improving allocation robustness.

While these enhancements offer improved expressiveness over fixed greedy scoring, they largely remain heuristic in nature and do not fully resolve the underlying suboptimality introduced by localized decision-making.

2.3 Distributed Optimal and Optimization-Based Approaches

Beyond bounded-suboptimal auction-consensus algorithms, several distributed methods aim to recover globally optimal solutions for specific MRTA formulations. When MRTA reduces to a one-to-one linear assignment problem under the ST-SR-IA assumption, distributed optimization techniques become applicable.

Consensus-based Alternating Direction Method of Multipliers (ADMM) formulations enable agents to maintain local copies of assignment variables while exchanging dual updates across a communication network, converging to the centralized optimal solution [15]. Similarly, distributed implementations of the Hungarian algorithm realize classical assignment solvers through local updates and neighbor communication, achieving polynomial-time optimality [16].

Other distributed optimization approaches, including decentralized genetic algorithms and swarm-inspired metaheuristics, have also been explored [17]. While these methods can

handle complex nonlinear objectives and constraints, they often exhibit slower convergence and higher computational overhead.

Although distributed optimal solvers demonstrate that decentralized global optimality is achievable in constrained settings, their communication requirements, synchronization demands, and computational complexity limit practicality in large-scale or dynamic robotic swarms. Consequently, lightweight auction-consensus frameworks continue to dominate real-world decentralized coordination, motivating efforts to improve their efficiency without sacrificing scalability.

2.4 Learning-Based Multi-Robot Task Allocation

Recent advances in machine learning and reinforcement learning have introduced new opportunities for improving MRTA performance through data-driven policy learning. Rather than relying on handcrafted heuristics, learning-based approaches enable agents to infer task valuation and coordination strategies directly from experience.

Graph neural networks have been widely adopted to represent MRTA problems due to their ability to process variable-sized teams and task sets while respecting permutation invariance [18]. Several works leverage graph-based encoders combined with deep reinforcement learning to learn decentralized allocation policies that generalize across different swarm sizes and environments [19]. Other approaches employ capsule networks, attention mechanisms, and normalization strategies to improve scalability and robustness [20].

Multi-agent reinforcement learning frameworks have also been explored for MRTA, enabling agents to learn cooperative strategies under shared global objectives [21]. Centralized training with decentralized execution (CTDE) has become a common paradigm, allowing agents to leverage global information during training while maintaining decentralized policies at

deployment. These methods have demonstrated promising performance gains over classical heuristics in simulated environments.

Despite these advances, learning-based MRTA approaches often face challenges related to generalization, training stability, and interpretability. Many learned policies remain sensitive to training distributions and may degrade when task densities, workspace geometry, or team sizes change. Moreover, fully end-to-end learned coordination can obscure convergence guarantees and system predictability, which are critical for safety-critical robotic applications.

2.5 Bridging Classical Coordination with Adaptive and Learning-Based Methods

An emerging research direction seeks to combine the robustness and convergence properties of classical decentralized algorithms with the adaptivity of learning-based strategies. Rather than replacing coordination mechanisms entirely, learning components are embedded within established algorithmic frameworks to enhance specific decision processes.

Within auction-consensus systems, this hybrid philosophy motivates improving bidding functions while preserving deterministic consensus layers. By retaining structured coordination protocols, agents benefit from provable convergence and conflict resolution, while learned components provide richer task valuation capabilities than fixed heuristics.

The approaches presented in this thesis follow this hybrid paradigm. The Loss-Aware Auction-Consensus Algorithm augments greedy bidding through explicit loss propagation, improving global efficiency without sacrificing decentralized execution. The learning-enhanced auction-consensus framework embeds neural bidding policies within CBBA, leveraging reinforcement learning to approximate globally optimal bidding strategies while preserving scalable coordination.

Together, these methods contribute to ongoing efforts to unify classical distributed control with modern learning-based intelligence, advancing the effectiveness of decentralized multi-robot task allocation systems.

CHAPTER III

PROBLEM FORMULATION

3.1 Multi-Robot Task Allocation

This thesis considers the Multi-Robot Task Allocation (MRTA) problem in which a team of autonomous robots must cooperatively service a set of spatially distributed tasks. The objective is to determine which robots should execute which tasks while minimizing the total travel distance of the robot team. In decentralized MRTA, robots must determine task assignments through distributed coordination rather than through a centralized planner.

Let:

$$I \triangleq \{1, 2, \cdots, N_r\} \tag{3.1}$$

denote the set of robots, where N_r is the number of robots in the system.

Let:

$$J \triangleq \{1, 2, \cdots, N_t\} \tag{3.2}$$

denote the set of tasks, where N_t is the number of tasks that must be serviced.

Each robot $i \in I$ has an initial position $\boldsymbol{v}_i \in \mathbb{R}^2$, representing its coordinates in a two-dimensional workspace. Each task $j \in J$ is similarly located at position $\boldsymbol{q}_j \in \mathbb{R}^2$. The workspace is therefore modeled as a subset of the Euclidean plane $\mathbb{R}^2$, and travel distances between robots and tasks are measured using the Euclidean norm.

For any vector $\boldsymbol{x} = (x^{(1)}, x^{(2)})^\top \in \mathbb{R}^2$, the Euclidean norm is defined as

$$\|\boldsymbol{x}\|_2 \triangleq \sqrt{(x^{(1)})^2 + (x^{(2)})^2} \tag{3.3}$$

Using this definition, the Euclidean travel cost between robot $i \in I$ and task $j \in J$ is defined as

$$d_{ij} = \left\| \boldsymbol{v}_i - \boldsymbol{q}_j \right\|_2 \tag{3.4}$$

In formulations where robots may execute multiple tasks sequentially, the Euclidean distance between two tasks $k, j \in J$ is defined as

$$d_{N_r+k,j} = \left\| \boldsymbol{q}_k - \boldsymbol{q}_j \right\|_2 \tag{3.5}$$

3.2 Communication Graph

Because the algorithms studied in this thesis operate in a decentralized manner, robots exchange information through local communication links rather than through a centralized server. The communication topology is represented by an undirected graph

$$G_c = (I, E_c) \tag{3.6}$$

where the vertex set corresponds to the set of robots I, and the edge set

$$E_c \subseteq \left\{ \{i, k\} : i, k \in I, i \neq k \right\} \tag{3.7}$$

represents the communication links between pairs of robots.

For any pair of robots $i, k \in I$, define the communication adjacency indicator

$$g_{ik} = \begin{cases} 1 & \text{if } \{i, k\} \in E_c \\ 0 & \text{otherwise} \end{cases} \tag{3.8}$$

Because the communication graph is undirected, the relation

$$g_{ik} = g_{ki} \tag{3.9}$$

holds for all $i, k \in I$.

Throughout this thesis, it is assumed that the communication graph G_c is connected. This assumption guarantees that information exchanged locally between neighboring robots can propagate throughout the entire team via repeated communication rounds.

3.3 Task Allocation Graph

The task allocation itself is represented by a separate graph that describes the travel edges selected in the final solution. This graph is distinct from the communication graph because communication links determine information exchange between robots, whereas allocation edges determine physical movements robots will perform through the workspace.

Define the allocation graph

$$G_a = (V, E_a, w) \tag{3.10}$$

as a weighted directed graph with vertex set

$$V = I \cup J \tag{3.11}$$

The edge set

$$E_a \subseteq (I \times J) \cup (J \times J) \tag{3.12}$$

represents admissible travel edges that may appear in a feasible task allocation.

The weight function

$$w: E_a \to \mathbb{R}_{\geq 0} \tag{3.13}$$

Assigns a nonnegative travel cost to each edge. For robot-task edges, the weight is defined as

$$w(i,j) \triangleq \left\| v_i - q_j \right\|_2, \quad (i,j) \in I \times J \tag{3.14}$$

For task-task edges, the weight is defined as

$$w(k,j) \triangleq \left\| q_k - q_j \right\|_2, \quad (k,j) \in J \times J \tag{3.15}$$

The weight of each edge therefore corresponds to the Euclidean travel distance between its endpoints.

3.4 MRTA Variants

This thesis covers two MRTA variants.

3.4.1 Single-Task Robot, Single-Robot Task, Instantaneous Assignment (ST-SR-IA)

Under the ST-SR-IA assumption, each robot may be assigned at most one task and each task must be assigned to one robot. In this case, the admissible allocation edges consist only of robot-task edges:

$$E_a^{\mathrm{ST}} \subseteq I \times J \tag{3.16}$$

The allocation graph therefore becomes a directed bipartite graph between robots and tasks.

To represent which edges in the allocation graph are selected in a ST-SR-IA solution, a binary decision variable is introduced.

For robot-task edges, define

$$x_{ij} = \begin{cases} 1 & \text{if task } j \text{ is assigned to robot } i \\ 0 & \text{otherwise} \end{cases}, \quad i \in I, j \in J \tag{3.17}$$

Thus, $x_{ij} = 1$ indicates that the directed edge (i, j) is included in the ST-SR-IA allocation graph.

The assignment constraints are:

$$\sum_{j \in J} x_{ij} \leq 1 \quad \forall i \in I \tag{3.18}$$

$$\sum_{i \in I} x_{ij} = 1 \quad \forall j \in J \tag{3.19}$$

The selected edge set for a ST-SR-IA solution is then defined as

$$E_a^{\mathrm{ST}} = \{(i, j) \in I \times J : x_{ij} = 1\} \tag{3.20}$$

so that E_a^{ST}contains exactly the robot-task edges selected by the assignment.

3.4.2 Multi-Task Robot, Single-Robot Task, Instantaneous Assignment (MT-SR-IA)

MRTA problems with the Multi-Task Robot, Single-Robot Task, Instantaneous Assignment (MT-SR-IA) assumption differs from the ST-SR-IA assumption in that robots may be assigned multiple tasks which will be executed sequentially. Each task still must be assigned to one robot.

Under this formulation, the admissible allocation edge set is

$$E_a^{\mathrm{MT}} \subseteq (I \times J) \cup (J \times J) \tag{3.21}$$

To represent which tasks are assigned to which robots in a MT-SR-IA solution, a binary decision variable is introduced.

Define

$$z_{ij} = \begin{cases} 1 & \text{if task } j \text{ is assigned to robot } i \\ 0 & \text{otherwise} \end{cases}, \quad i \in I, j \in J \tag{3.22}$$

Note that z_{ij} is a task-ownership indicator only: $z_{ij} = 1$ records that task j belongs to robot i's execution path, but does not specify where in the sequence task j appears. The directed edges of the allocation graph under MT-SR-IA are determined entirely by the execution paths $\boldsymbol{p}_i$ defined in (3.24), not by z_{ij} directly.

Let L_t denote the maximum number of tasks that may be assigned to any robot. The assignment constraints become

$$\sum_{j \in J} z_{ij} \leq L_t \quad \forall i \in I \tag{3.23}$$

$$\sum_{i \in I} z_{ij} = 1 \quad \forall j \in J \tag{3.24}$$

The selected edges must form a collection of directed paths originating from robot vertices. For each robot i, define an execution path

$$\boldsymbol{p}_i = [i, j_1, j_2, \cdots, j_{L_i}] = [p_{i,0}, p_{i,1}, p_{i,2}, \cdots, p_{i,L_i}] \tag{3.25}$$

where $j_l \in J$ and $L_i \leq L_t$. The first edge (i, p_{i1}) represents the robot traveling from its initial position to its first assigned task, while each subsequent edge $(p_{i,l-1}, p_{i,l})$ represents travel between consecutive tasks. Each task may appear in at most one robot path.

For the MT-SR-IA formulation, the travel cost associated with robot i's execution path $\boldsymbol{p}_i$ is defined as

$$D(\boldsymbol{p}_i) = w(i, p_{i1}) + \sum_{l=2}^{L_i} w(p_{i,l-1}, p_{i,l}) \tag{3.26}$$

If robot i is assigned no tasks, then $D(\boldsymbol{p}_i) = 0$.

The selected edge set for a MT-SR-IA solution is defined as the union of all directed edges appearing across the robot execution paths:

$$E_a^{\mathrm{MT}} = \bigcup_{i \in I} \{(p_{i,l-1}, p_{i,l}) : l = 1, \cdots, L_i\} \tag{3.27}$$

Each edge $(p_{i,l-1}, p_{i,l}) \in E_a^{\mathrm{MT}}$ is a directed edge in the allocation graph $G_a$with weight $w(p_{i,l-1}, p_{i,l})$ corresponding to the Euclidean travel distance defined in (3.14) and (3.15).

3.5 Global Objective

The objective considered in this thesis is the min-sum objective, which minimizes the total travel cost incurred by the robot team. The algorithms studied in this thesis operate within a decentralized auction-consensus framework and are considered to have converged once no robot changes its internal assignment state between successive iterations. Upon convergence, the final task allocation is represented by the selected edge set

$$E_a^* \subseteq E_a \tag{3.28}$$

which takes the form of E_a^{ST} as defined in (3.20) under the ST-SR-IA assumption, or E_a^{MT} as defined in (3.27) under the MT-SR-IA assumption. The total travel cost of the team is equal to the sum of weights over all edges in the final allocation:

$$C(E_a^*) = \sum_{e \in E_a^*} w(e) \tag{3.29}$$

3.6 Auction-Consensus Structure

The algorithms studied in this thesis solve the MRTA problem in a decentralized manner using an auction-consensus framework. At each iteration, robots first execute a task selection phase, in which they compute bids for tasks and update their local assignment decisions. Robots then execute a consensus phase, in which bid information is exchanged between neighboring robots according to the communication graph G_c. Through repeated iterations of bidding and consensus updates, robots converge to a conflict-free allocation.

Convergence is defined as the condition in which no robot changes its internal assignment state between successive iterations. Although the bidding rules differ between the ACALM algorithm and the learned bidding approach introduced later in this thesis, both methods operate within this same decentralized auction-consensus framework.

CHAPTER IV

AUCTION-CONSENSUS ALGORITHM WITH LOSS MECHANISM

4.1 Motivation

The classical Consensus-Based Auction Algorithm (CBAA) solves ST-SR-IA MRTA problems through greedy reward-based bidding and maximum consensus. While CBAA guarantees convergence and provides a 50% worst-case optimality bound, its greedy selection mechanism may produce globally suboptimal allocations.

The root cause is that CBAA evaluates tasks independently based only on immediate marginal reward. It does not account for the compounded cost of losing multiple preferred tasks in sequence.

ACALM addresses this limitation by introducing a loss-aware bidding mechanism, which adjusts bid values based on the accumulated opportunity cost of previously failed bids.

4.2 Bidding Formulation

4.2.1 Reward Definition

ACALM defines the reward c_{ij} for robot $i \in I$ being assigned task $j \in J$ as:

$$c_{ij} = K - d_{ij} \tag{4.1}$$

where K satisfies:

$$K > \max_{ij} d_{ij} \tag{4.2}$$

This ensures that c_{ij} is always positive, and that reward decreases as distance between robots and tasks increase.

Let robot i's reward vector $\boldsymbol{c}_i$ be denoted as:

$$\boldsymbol{c}_i = \left[c_{i1}, \cdots, c_{iN_t}\right] \tag{4.3}$$

4.2.2 Loss Definition

The main innovation of ACALM is the definition of loss. For robot i, define the initial loss for task j as the difference between the reward of task j and the next best lower reward task:

$$l_{ij} = c_{ij} - \max\left(\boldsymbol{c}_i \cdot \mathbb{I}\left(\boldsymbol{c}_i < c_{ij}\right)\right) \tag{4.4}$$

where $\mathbb{I}(\cdot)$ is the indicator function that is unity if the argument is true and zero otherwise, and $\max(\cdot)$ is a function that returns the maximum scalar value within the input vector. Intuitively, the loss value l_{ij} indicates how much reward is lost if robot i loses task j and must move to its next best alternative. Unlike CBAA which would bid directly using reward c_{ij}, ACALM bids using l_{ij}. This changes the bidding process from who wants the task most to who suffers the most if denied.

4.2.3 Preference Ordering

Each robot constructs a preference vector $\boldsymbol{o}_i$ which contains all tasks $j \in J$ sorted in decreasing order of reward for robot i:

$$\boldsymbol{o}_i = \operatorname{argsort}_{j \in J}\left(-c_{ij}\right) \tag{4.5}$$

such that:

$$c_{i,o_{i1}} \geq c_{i,o_{i2}} \geq \cdots \geq c_{i,o_{iN_t}} \tag{4.6}$$

The preference vector $\boldsymbol{o}_i$ reflects the sequence robot i will follow when selecting tasks. Each robot initially bids on its highest reward task. For robots to keep track of their position within their preference vectors, each robot maintains two variables: s_i and m_i. The variable $m_i \in$

J denotes the task that is currently selected, while the variable s_i represents the position of m_i within $\boldsymbol{o}_i$. Since every robot begins with its highest reward task, the initial conditions for every robot is $s_i = 1$ and $m_i = o_{i,s_i}$.

4.2.4 Loss Propagation

In the case that several auctions are lost consecutively, the initial loss value l_{ij} will not reflect the accumulated loss of several lost bids. For this reason, loss propagation was introduced. When a task is lost for the first time, its loss should be propagated to all previous tasks in $\boldsymbol{o}_i$ to account for the cumulative effect of multiple failed bids. The variable u_i is used to track which tasks have been used for loss propagation already. The variable u_i is initialized as $u_i = 1$ since first task in $\boldsymbol{o}_i$ cannot be propagate loss to any earlier tasks. When $s_i > u_i$, this indicates that the current selected task has not yet been used for loss propagation. If such a task is lost, loss propagation should occur for all previous tasks, u_i is incremented, and s_i is reset so the robot starts again from the beginning of its preference vector with updated loss values.

4.3 Phase 1: Task Selection

The first phase of the ACALM auction-consensus algorithm is shown in Figure 4.1. Phase 1 uses the most up-to-date assignment and winning bid vectors $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$, respectively. Both are vectors of length N_t and are initialized to contain zeros for the first iteration. The winning bid vector $\boldsymbol{y}_i$ will contain the most up-to-date information on the highest placed bid for each task, and the way it is updated will be explained further when discussing Phase 2.

Phase 1 begins by examining the assignment vector $\boldsymbol{x}_i$ to confirm that robot i has not yet been assigned a task (line 2). If a task has already been assigned, the robot i performs no

additional actions during Phase 1. Robots select the next task m_i within their preference vector $\boldsymbol{o}_i$. The bid value they will place will thus be l_{i,m_i}.

Before placing the bid, robot i compares l_{i,m_i} with the corresponding entry in the winning bid vector $\boldsymbol{y}_i$ (line 4). If robot i believes that it can be the new winner for task m_i, the robot updates x_{i,m_i}, and y_{i,m_i} to reflect the new assignment and the new highest bid (lines 5 - 6). Otherwise, robot i evaluates whether loss propagation should occur (lines 8-13). If a task is lost but no loss propagation occurs, the robot simply proceeds to the next task in its preference vector (line 14). This repeats until a task is successfully selected.

Algorithm 1 ACALM Phase 1 for agent i at iteration n

1: **procedure** SELECT TASK
2: **while** $\sum_j x_{ij} = 0$ **do**
3: $m_i = o_{i,s_i}$
4: **if** $l_{i,m_i} > y_{i,m_i}$ **then**
5: $x_{i,m_i} = 1$
6: $y_{i,m_i} = l_{i,m_i}$
7: **end if**
8: **if** $l_{i,m_i} \leq y_{i,m_i}$ **then**
9: **if** $u_i < s_i$ **then**
10: $l_{ij} = l_{ij} + l_{i,m_i},\ \forall j \in \{o_{i,f}\}_{f=1}^{u_i}$
11: $u_i = u_i + 1$
12: $s_i = 0$
13: **end if**
14: $s_i = s_i + 1$
15: **end if**
16: **end while**
17: **end procedure**

Figure 4.1: ACALM Phase 1

4.4 Phase 1.5: Preemptive Loss Propagation

Frequent occurrences of task loss, propagating loss, and rebidding can increase the number of iterations that ACALM requires to achieve convergence to its final solution. To help increase the speed of convergence, preemptive loss propagation is implemented as shown in Figure 4.2. Preemptive loss propagation operates by reviewing the next task r_i that is eligible for loss propagation (line 2). If it is already known that a bid for task r_i will lose against the current highest bid, the corresponding changes to the loss vector l_i are calculated and temporarily stored in vector l'_i (lines 4 - 5). The proposed changes are not applied immediately to l_i because it needs to be verified that the changes will not alter the current task selection. The first condition that must be satisfied is that the proposed updates to l_i do not introduce a new possible winning task earlier in the greedy sequence (line 6). In other words, the currently selected task $m_i = o_{i,s_i}$ must remain the highest reward task capable of being the new winning bid. If this condition is satisfied, the proposed loss propagation can be implemented (lines 7 - 8). Afterward, the task index r_i is incremented to the next candidate task to review whether additional preemptive loss propagation should be performed. Further preemptive loss propagation is allowed only if a second condition is satisfied. This condition ensures that the selected task $m_i = o_{i,s_i}$ remains the only task capable of outbidding the current winners for all tasks earlier than u_i in the sequence $\boldsymbol{o}_i$ (line 10). The first condition guarantees that no higher reward winning option emerges, and the second condition helps the preemptive loss propagation end when there are other winning options later in the preference order which should be attempted first before more loss propagation can occur. Once no more preemptive loss propagation will occur, the bid y_{i,m_i} is updated with the latest l_{i,m_i} so that the most informed bid can be sent during Phase 2 (line 17).

Algorithm 2 ACALM Phase 1.5 for agent i at iteration n

1: **procedure** PREEMPTIVE LOSS PROPAGATION
2: $r_i = o_i[u_i + 1]$
3: **while** $l_{i,r_i} \leq y_{i,r_i}$ **do**
4: $l'_{ij} = l_{ij},\ \forall j \in \mathcal{J}$
5: $l'_{ij} = l'_{ij} + l_{i,r_i},\ \forall j \in \{o_{i,f}\}_{f=1}^{u_i}$
6: **if** $l'_{ij} \leq y_{ij},\ \forall j \in \{o_{i,f}\}_{f=1}^{s_i-1}$ **then**
7: $l_{ij} = l'_{ij},\ \forall j \in \{o_{i,f}\}_{f=1}^{u_i}$
8: $u_i = u_i + 1$
9: $r_i = o_i[u_i + 1]$
10: **if** $l_{ij} \leq y_{ij},\ \forall j \in \{o_{i,f}\}_{f=1}^{u_i}, j \neq o_{i,s_i}$ **then**
11: **continue**
12: **else**
13: **break**
14: **else**
15: **break**
16: **end while**
17: $y_{i,m_i} = l_{i,m_i}$
18: **end procedure**

Figure 4.2: ACALM Phase 1.5

4.5 Phase 2: Consensus

During the consensus process, the winning bid lists are shared between neighboring robots as shown in Figure 4.3. From its own winning bid list and those received from neighbors, robot i finds the list with the highest bid for task m_i which it assigned itself previously in Phase 1. Robot i uses the index $z_{i,m_i} \in I$, which corresponds to the robot index with the highest bid for task m_i, to verify if it is the winner for the task chosen (line 4). If robot i finds the winning robot

index z_{i,m_i} is not equal to its own index, this indicates that it has lost the bid for task m_i (line 5). If a robot loses the bid, it resets its assignment in $\boldsymbol{x}_i$, propagates the loss if necessary, and updates the selection index s_i. Lastly, robot i updates its own winning bid list $\boldsymbol{y}_i$ by keeping the maximum bid values for all tasks from the winning bid lists received from neighbors (line 14).

Algorithm 3 ACALM Phase 2 for agent i at iteration n

1: SEND y_i to k with $g_{ik} = 1$
2: RECEIVE y_k from k with $g_{ik} = 1$
3: **procedure** UPDATE TASK
4: $z_{i,m_i} = \arg\max_k g_{ik} \cdot y_{k,m_i}$
5: **if** $z_{i,m_i} \neq i$ **then**
6: $x_{i,m_i} = 0$
7: **if** $u_i < s_i$ **then**
8: $l_{ij} = l_{ij} + l_{i,m_i}, \ \forall j \in \{o_{i,f}\}_{f=1}^{u_i}$
9: $u_i = u_i + 1$
10: $s_i = 0$
11: **end if**
12: $s_i = s_i + 1$
13: **end if**
14: $y_{ij} = \max_k g_{ik} \cdot y_{kj}, \ \forall j \in \mathcal{J}$
15: **end procedure**

Figure 4.3: ACALM Phase 2

4.6 ACALM Results on ST-SR-IA

4.6.1 Experimental Framework

For the ST-SR-IA formulation associated with ACALM, large-scale randomized datasets were generated in which the number of robots equaled the number of tasks. Swarm sizes of 5, 15,

30, 45, 60, and 120 were evaluated, with 1000 independently generated configurations per size, resulting in 6000 total test instances. Robots and tasks were uniformly sampled within a bounded 15 × 15 two-dimensional workspace. For each configuration, the globally optimal assignment was computed using a Mixed Integer Linear Program (MILP) implemented via the PuLP solver [22], and the optimal total team distance was recorded as a benchmark. Evaluations were conducted with a fully connected communication graph.

Performance was evaluated using two primary metrics. The first is Percent Optimality, defined as the ratio of the decentralized solution team travel cost to the centralized MILP optimal solution. The second is the Number of Auction-Consensus Iterations to Convergence, used as a proxy for communication and coordination overhead.

For ACALM experiments, statistical comparisons were conducted using the Wilcoxon signed-rank test due to the non-normal distribution of performance differences [23]. Significance levels were reported with corresponding confidence intervals.

4.6.2 Sample Test Comparison

Figures 4.4-4.6 show the results for a sample test configuration with a swarm size of 15. Figure 4.4 shows the optimal solution using the MILP solver. Visually, it can be noted that there do not appear to be any excessively long-distance routes in the optimal solution. This highlights how the global optimal solution will often be a balance between each robot attempting to act greedily and being considerate of neighboring robots as well.

The drawbacks of the CBAA method are shown in Figure 4.5, with many routes being very short, meaning those robots likely won the bids for their greedy task, and several of the remaining routes being large in distance, meaning those robots were likely forced to take

inconvenient tasks from what remained available. An example is the route from robot $r13$ to task $t12$ along the left side of Figure 4.5, which is a route with a significantly large distance.

The results shown in Figure 4.6 depict how ACALM can better replicate the optimal solution, with only a few minor task allocation differences from the optimal solution for the given sample configuration.

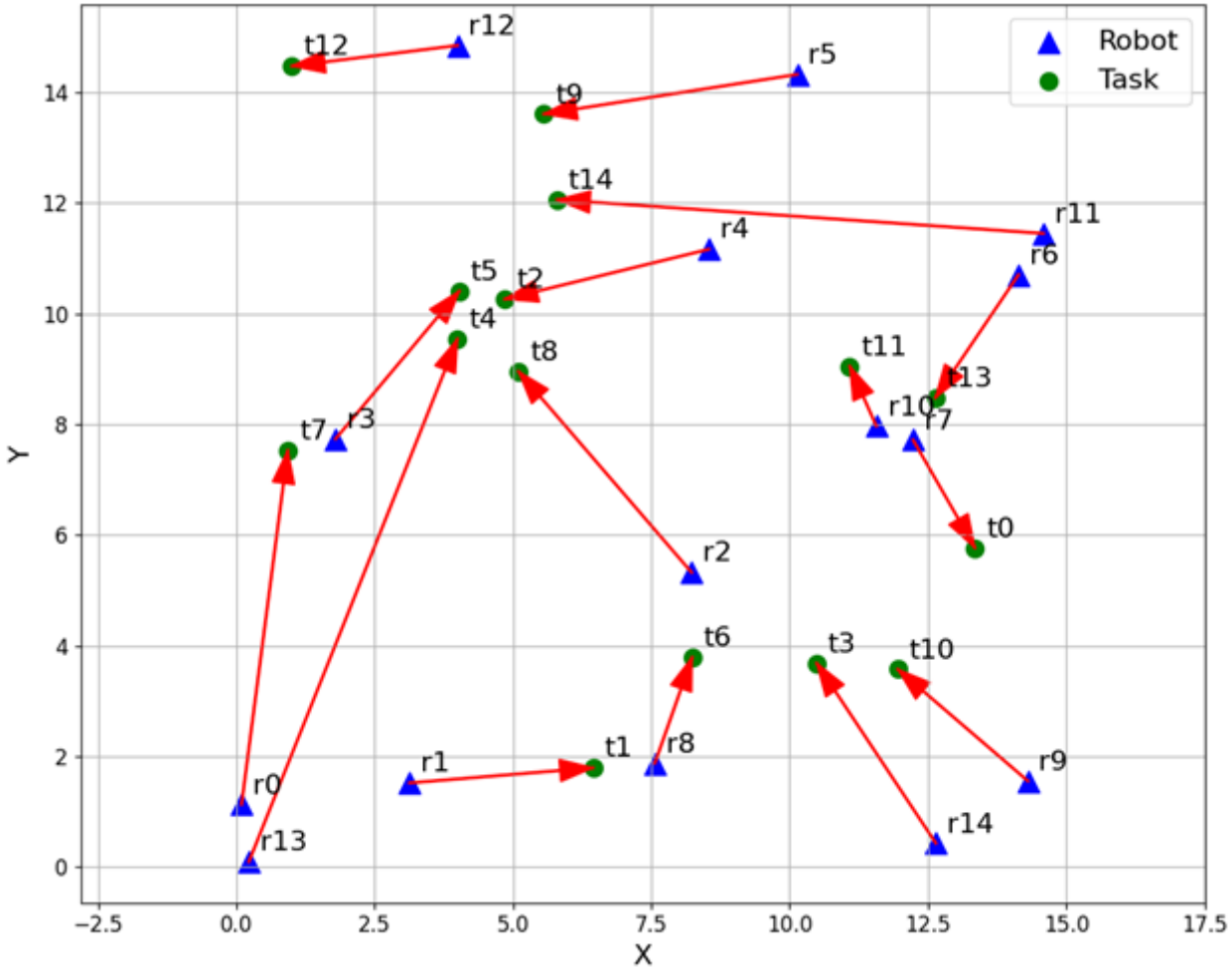

Figure 4.4: Example Optimal Routes for ST-SR-IA Configuration

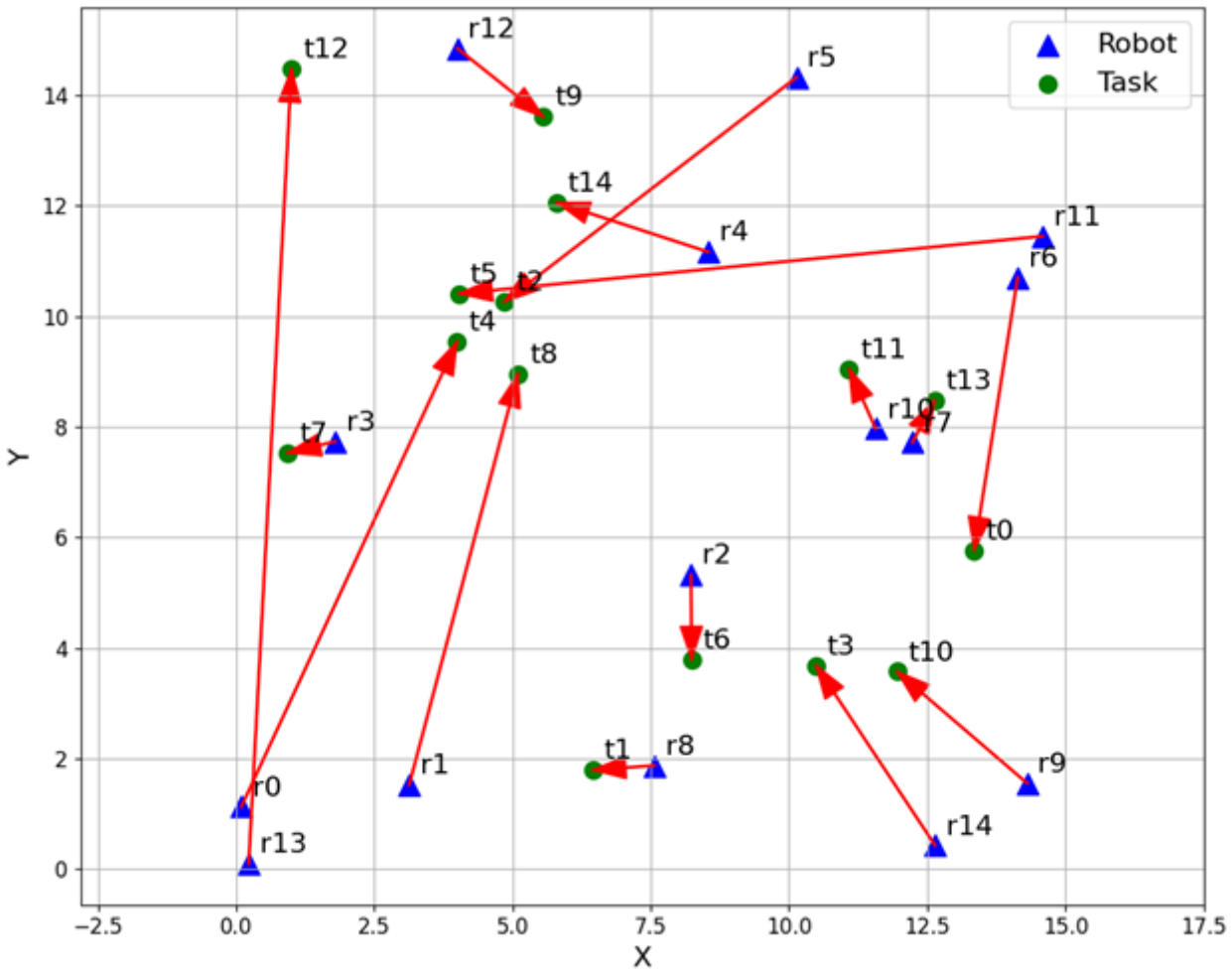

Figure 4.5: CBBA Solution for Example ST-SR-IA Configuration

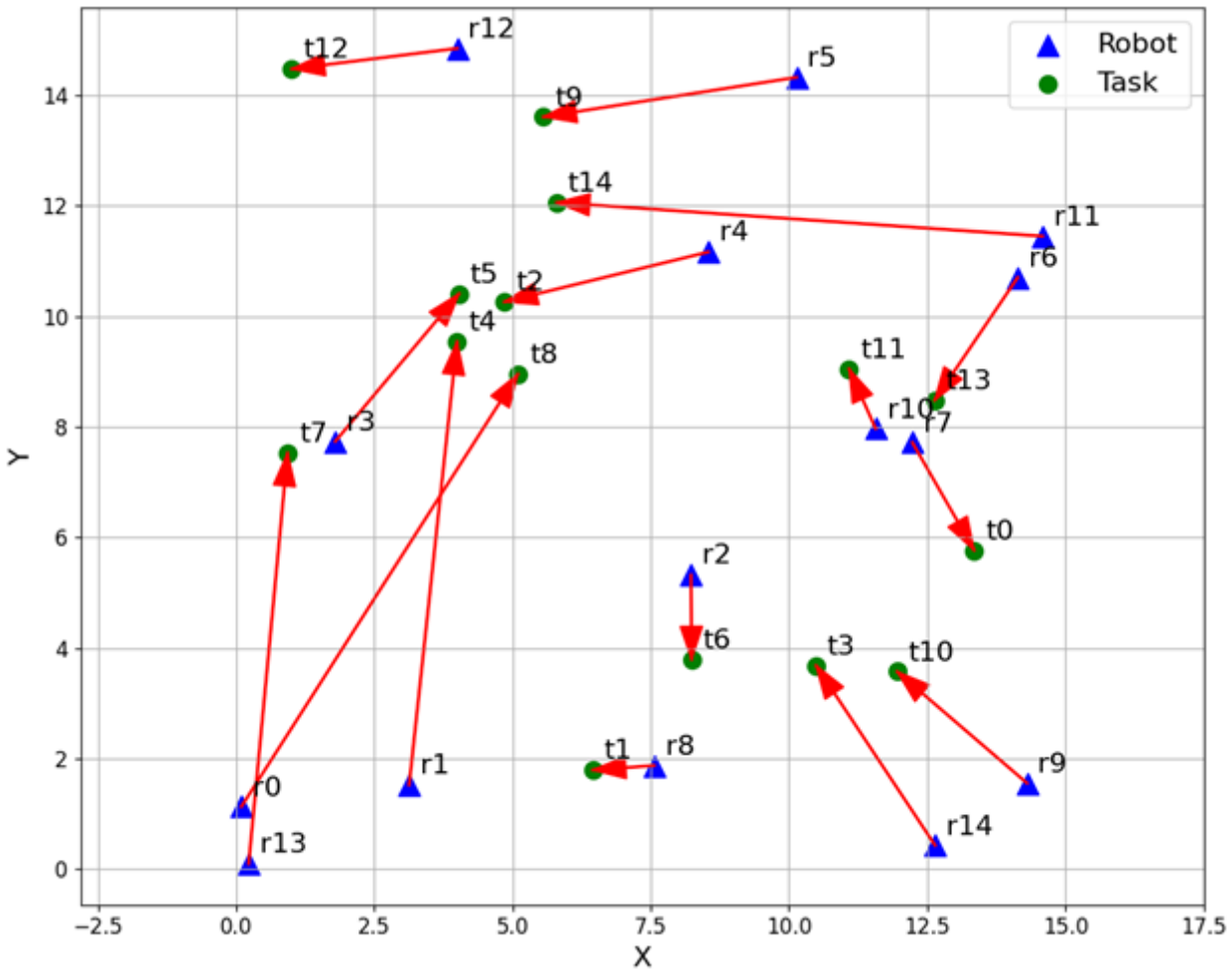

Figure 4.6: ACALM Solution for Example ST-SR-IA Configuration

4.6.3 Convergence Speed Comparison

The statistical results for the 6000 total test configurations are shown in Figures 4.7-4.8. The statistical significance is explicitly indicated by asterisks ($p < 0.001$). Additionally, the notch on each plot indicates the 95% confidence interval of the medians. If the notches of two boxes do not overlap, this indicates a statistically significant difference between the medians.

Figure 4.7 shows that the number of iterations required for ACALM to converge increases significantly with swarm size. This is due to the dynamic loss propagation mechanism that introduces additional bidding cycles as agents attempt to recover from task rejections.

Across all configurations, ACALM exhibits a mean iteration-to-robot ratio close to one (approximately 0.98 for 5 robots and 1.03 for 60 robots), indicating that the algorithm typically requires about one full iteration cycle per robot in the swarm. Interestingly, at larger scales (e.g., 120 robots), this ratio drops below one (approximately 0.78), suggesting improved scalability as collective convergence becomes more efficient relative to swarm size. In contrast, Figure 4.7 also demonstrates that the original CBAA and similar GCAA method maintains a much lower iteration count across different swarm sizes. The original CBAA maintains a much lower iteration-to-robot ratio—around 0.54 for 5 robots and decreasing to 0.09 for 120 robots—demonstrating faster consensus but less iterative adjustment during task contention. While this leads to reduced communication overhead, it also limits reallocation adaptability compared to ACALM. The GCAA method follows a similar trend to CBAA.

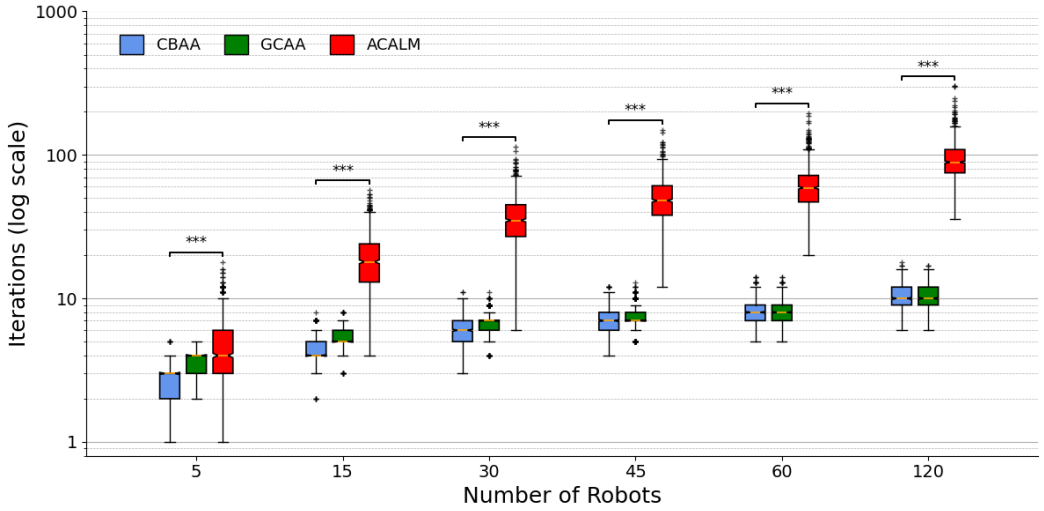

Figure 4.7: Iterations vs Number of Robots for ST-SR-IA

4.6.4 Percent Optimality Comparison

Figure 4.8 compares the percent optimality of both algorithms across the test configurations. The ACALM method displays a median optimality greater than 95%, even at a

swarm size of 120 robots. Meanwhile, the CBAA and GCAA methods see a gradual decline in optimality as the number of agents increases, indicating their reduced effectiveness in larger-scale problems.

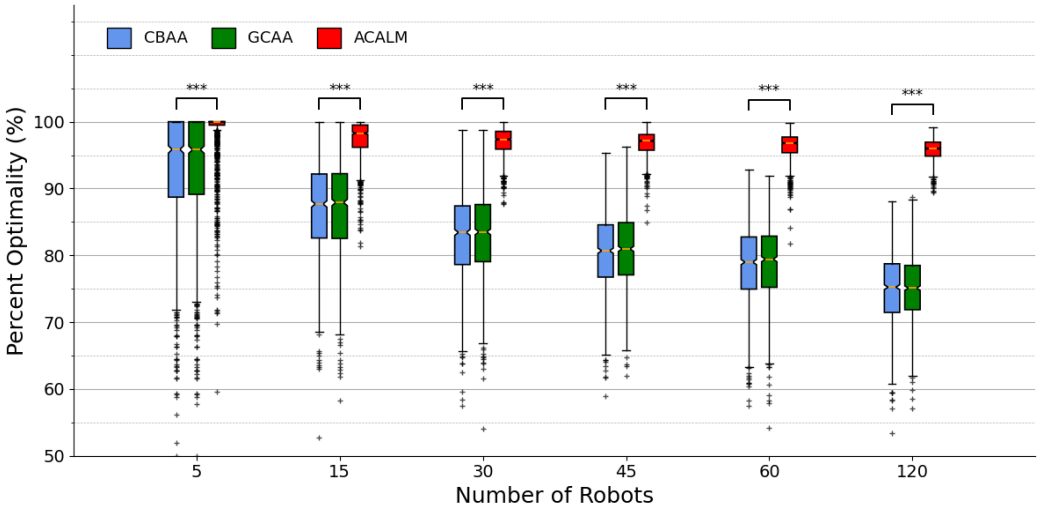

Figure 4.8: Percent Optimality vs Number of Robots for ST-SR-IA

CHAPTER V

AUCTION-CONSENSUS ALGORITHM WITH LEARNED BIDDING

5.1 Motivation

Chapter IV introduced ACALM, which improves classical CBAA by incorporating a deterministic loss-propagation mechanism. While loss propagation partially corrects greedy misallocations, it remains heuristic and manually designed as well as unsuited for MT-SR-IA.

This chapter introduces a learning-enhanced auction-consensus framework in which the deterministic bidding function is replaced by a neural bidding policy trained using reinforcement learning. Unlike ACALM, which modifies CBAA's reward structure analytically, this approach learns bidding behavior directly from interactions with the MT-SR-IA MRTA environment defined in Chapter III.

5.2 Auction-Consensus Structure with Learned Bidding

The auction-consensus structure remains unchanged, with the algorithm still consisting of two phases. With robots now being able to have multiple tasks assigned, robots, or agents as commonly referred to in reinforcement learning, need to track which tasks they have assigned to themselves and in what order. This is done with the bundle state variable $\boldsymbol{b}_i \in (J \cup \{0\})^{L_i}$ where the tasks are ordered based on when they were assigned. Each agent maintains four state variables: The winning bid vector $\boldsymbol{y}_i$, the bundle $\boldsymbol{b}_i$, and path $\boldsymbol{p}_i$, and the winning agent vector $\boldsymbol{z}_i$. The winning agent vector $\boldsymbol{z}_i \in I^{N_t}$ tracks who agent i believes is the winning robot for each task.

5.2.1 Neural Bidding Policy

In contrast to classical CBBA, where bids are computed using deterministic marginal gain functions, we construct a partial observation $\boldsymbol{o}_i(t)$ from the agent's local state and task features. The function $\phi(\cdot)$ constructs the partial observation from the agent's local state and task positions $\boldsymbol{q}_j \; \forall \; j \in J$. The resulting observation is represented as $\boldsymbol{o}_i(t) \in \mathbb{R}^{N_t \times F}$, where N_t is the number of tasks and F is the feature dimension per task.

The neural bidding policy $\pi_\theta(\cdot)$ maps the partial observation to a bid vector $\boldsymbol{c}_i \in \mathbb{R}^{N_t}$, producing one scalar bid value per task. A higher c_{ij} value is interpreted as a stronger preference for assigning the task j to agent$i$. For tasks already contained in the agent's bundle, the corresponding bid outputs are masked prior to task selection by setting their effective bid values to $-\infty$. The CBBA structure is retained to coordinate task ownership across agents, while learning replaces the handcrafted scoring function.

5.2.2 Phase 1: Learned Bundle Construction

Phase 1 consists of constructing partial observations from current agent states, computing task bids using the learned neural policy, and greedily selecting tasks for self-assignment as shown in Figure 5.1.

Algorithm 1 Learned Bundle Construction for Agent i at Iteration t

procedure BUILD BUNDLE($\mathbf{z}_i(t-1), \mathbf{y}_i(t-1), \mathbf{b}_i(t-1), \mathbf{p}_i(t-1)$)
  $\mathbf{y}_i(t) = \mathbf{y}_i(t-1)$
  $\mathbf{z}_i(t) = \mathbf{z}_i(t-1)$
  $\mathbf{b}_i(t) = \mathbf{b}_i(t-1)$
  $\mathbf{p}_i(t) = \mathbf{p}_i(t-1)$

  Construct partial observation features:
  $\mathbf{o}_i(t) = \Phi(\mathbf{y}_i(t), \mathbf{z}_i(t), \mathbf{b}_i(t), \mathbf{p}_i(t), Q)$

  Compute bids using learned neural policy:
  $\mathbf{c}_i(t) = \pi_\theta(\mathbf{o}_i(t))$

  for all $j \in \mathcal{J}$ **do**
    $h_{ij}(t) = \mathbb{I}(c_{ij}(t) > y_{ij}(t))$
  end for

  while $|\mathbf{b}_i| < L_t$ **do**
    if $\sum_{j \in \mathcal{J}} h_{ij}(t) = 0$ **then**
      break
    end if
    $J_i = \arg\max_{j \in \mathcal{J}} \; c_{ij}(t) \cdot h_{ij}(t)$
    $n_{i,J_i} = \arg\min_n \; \Delta D(\mathbf{p}_i \oplus_n \{J_i\})$
    $\mathbf{b}_i = \mathbf{b}_i \oplus_{\text{end}} \{J_i\}$
    $\mathbf{p}_i = \mathbf{p}_i \oplus_{n_{i,J_i}} \{J_i\}$
    $y_{i,J_i}(t) = c_{i,J_i}(t)$
    $z_{i,J_i}(t) = i$
  end while
end procedure

Figure 5.1: Learned Bundle Construction

Each agent evaluates all available tasks by computing task-specific features, such as distances relative to its current position and current route, before generating bids. The agent

compares these bids against its current winning bid list and selects tasks for which it can place higher bids until its bundle capacity L_t is reached. Newly selected tasks are inserted into the route at the position that minimizes the increase in path length, denoted $\Delta D(\cdot)$.

$$\Delta D(\boldsymbol{p}_i \oplus_n \{j\}) = D(\boldsymbol{p}_i \oplus_n \{j\}) - D(\boldsymbol{p}_i) \tag{5.1}$$

5.2.3 Phase 2: Consensus

Phase 2 consists of agents executing a consensus phase to resolve conflicts over task ownership using the standard CBBA communication update rules. During this phase, agents exchange their current winning bid vectors $\boldsymbol{y}_i$ and winning agent vectors $\boldsymbol{z}_i$. Based on the information received, each agent updates its local variables $\boldsymbol{y}_i$, $\boldsymbol{z}_i$, $\boldsymbol{b}_i$, and $\boldsymbol{p}_i$ according to the CBBA update, reset, or leave rules.

Unlike classical CBBA, no diminishing marginal gain (DMG) condition is enforced on the learned bidding policy. Therefore, the standard convergence and optimality guarantees of CBBA do not necessarily apply. The consensus mechanism is retained to promote consistency of task assignments across agents under decentralized communication.

5.3 Centralized Training with Decentralized Execution (CTDE)

The neural bidding model is trained using a Centralized Training with Decentralized Execution (CTDE) framework, which is well-suited for multi-agent cooperation. During training, all information required to evaluate global team performance, including the optimal assignment and optimal team distance computed via a MILP solver, is available to the learning algorithm. This allows the system to shape rewards based on how closely the team's collective decisions approximate the optimal task allocation solution.

CTDE is implemented by separating the information available to the actor and critic networks. The actor (neural bidder) receives only the robot's partial observation constructed from local state and task-related features, mirroring the information available during decentralized deployment. The critic, however, is trained with a global observation summarizing the overall system state and global performance. Global observation is used only to stabilize learning and to promote cooperative behavior during training. As a result, robots can learn coordinated bidding behaviors using global feedback while still executing a fully decentralized auction-consensus algorithm during deployment.

5.4 Reinforcement Learning Formulation

The learned bidding scheme can be modeled as a multi-agent reinforcement learning problem in which each robot repeatedly produces task bids that influence the distributed auction-consensus allocation. Because each robot observes only local information and partial task context, the overall coordination process is naturally viewed as a decentralized partially observable Markov decision process (Dec-POMDP). At each auction-consensus iteration t, robot i receives a partial observation $\boldsymbol{o}_i(t)$ derived from its local state and task features, and outputs a bid vector over tasks using a parameterized neural policy. The environment then executes the auction phase and consensus update rules, producing new internal state variables and advancing the system to the next iteration. The actor-environment interactions keep going until an iterations limit is reached, which is considered a timeout, or until convergence is achieved, which is defined as when no changes to state variables occur between two iterations.

In this formulation, the robot's action corresponds to the bid values produced by the neural bidder, while the environment transition is induced by the auction-consensus mechanism

itself. Learning is driven by a global reward that evaluates the quality of the team allocation as the auction converges.

5.4.1 Reward Design

The reward function is designed to measure incremental improvements in collective performance as the auction-consensus process evolves. Rather than rewarding only the final allocation, the learning signal is provided at each iteration t using a weighted sum of normalized components that reflect improvement in team travel distance, task assignment coverage, and agreement with a centralized optimal reference solution. Specifically, the reward at iteration t is defined as

$$r(t) = w_D r_D(t) + w_C r_C(t) + w_A r_A(t) \tag{5.2}$$

where $r_D(t)$ measures normalized improvement in team travel distance, $r_C(t)$ captures progress toward assigning tasks (coverage), and $r_A(t)$ measures agreement with the MILP optimal assignment. Each component is scaled to lie in $[-1,1]$, and the weights w_D, w_C, w_A control the relative importance of each component. This reward shaping provides a dense training signal that encourages policies to improve allocations throughout the auction, guiding learning toward higher-quality decentralized outcomes without directly enforcing centralized decisions.

5.4.2 Policy Optimization

Policy optimization is performed using Proximal Policy Optimization (PPO), a stable policy-gradient method that is well suited for continuous-valued outputs such as task bids. Training proceeds over multiple epochs, each consisting of a batch of randomly generated MRTA worlds. For each world, the current bidding policy is executed within the auction-

consensus environment until either convergence is reached or a maximum iteration limit triggers a timeout. The resulting trajectories contain sequences of partial observations, bid actions, global observations, and shaped rewards, which are used to update both the actor and critic networks via PPO.

Conceptually, each PPO rollout simulates the decentralized allocation process: robots generate bids from local observations, the auction phase updates bundles and tentative winners, and the consensus phase propagates bid information through the communication graph. The critic uses global state information during training to estimate value functions and reduce gradient variance, improving stability and learning efficiency. After training, only the actor network (the learned bidding policy) is retained and evaluated under decentralized execution, demonstrating whether the learned bidder generalizes across unseen environments and larger swarm sizes.

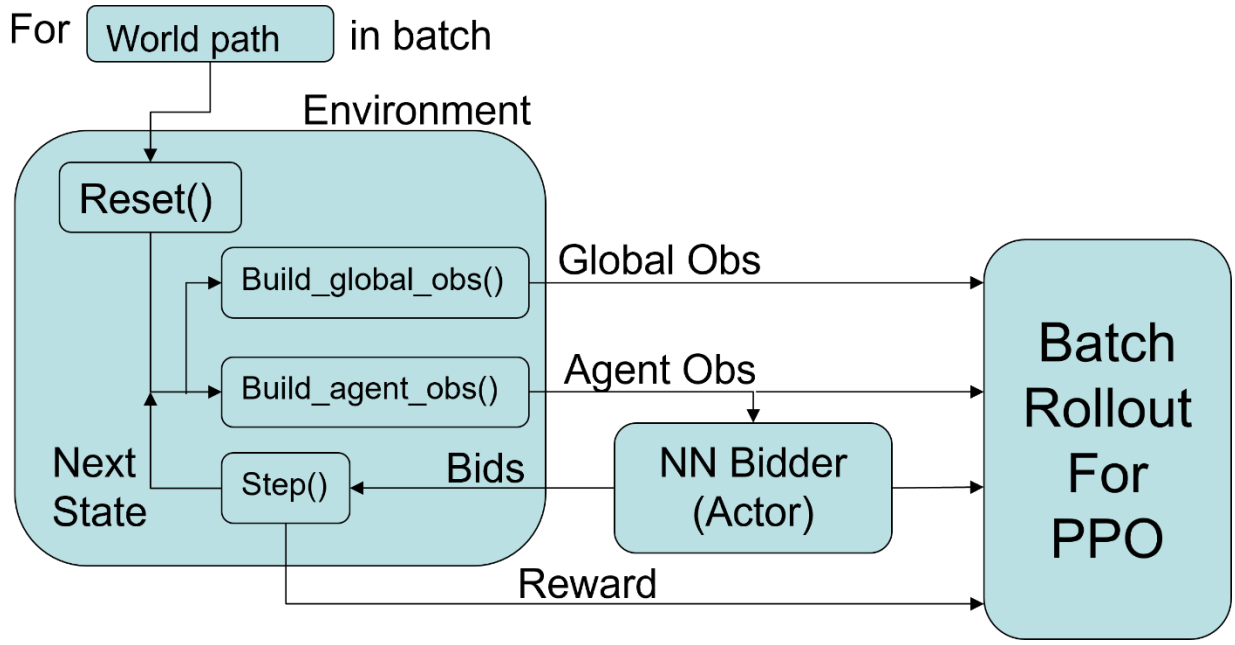

Figure 5.2: Training Diagram

5.5 Neural Architectures Evaluated

To understand how model structure impacts decentralized bidding performance, multiple neural architectures were evaluated as candidate bidding policies. All learned bidding policies were trained on a dataset of 1000 randomly generated 2D MRTA worlds, each containing 5 agents and a variable number of tasks (10–20). Robot and task locations were uniformly sampled within square workspaces with side lengths varying from 25 to 55 units, introducing significant geometric diversity and preventing overfitting to a single task density or workspace scale. The architectures were a Neural Additive Model (NAM) for interpretability and simplicity, a Long Short-Term Memory (LSTM) based model for memory and belief-state learning, and a Set Transformer model for permutation-invariant global context via attention. This comparison highlights both the potential and limitations of learned bidding within auction-consensus coordination.

5.5.1 Neural Additive Model (NAM)

The Neural Additive Model (NAM) is included as a lightweight and interpretable baseline for learned bidding [24]. Its additive structure decomposes the predicted bid into contributions from individual input features, as shown in Figure 5.3, making it possible to inspect how specific task attributes influence bid magnitude. This interpretability is useful in MRTA settings, where understanding the policy's decision logic can improve trust and facilitate debugging. In practice, the NAM achieved strong average performance but exhibited occasional convergence failures in which the auction-consensus process did not reach a consistent assignment before the maximum iteration limit. These timeouts suggest that while the NAM can learn effective local bidding heuristics, its limited representational capacity may struggle to

consistently produce bids that interact stably with the consensus dynamics across all environments.

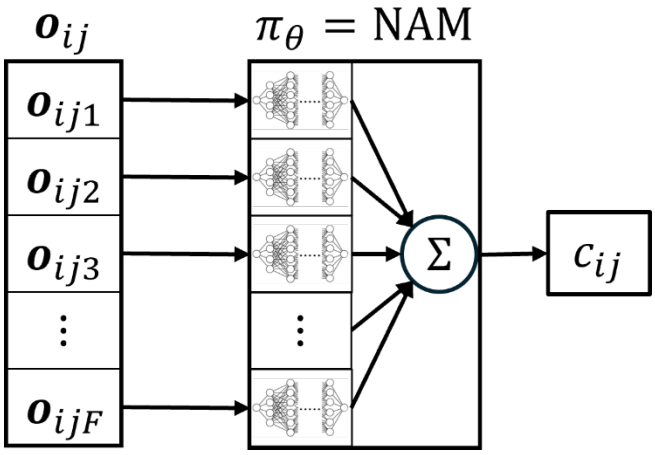

Figure 5.3: Neural Additive Model as Neural Bidder

5.5.2 LSTM-Based Model

An LSTM-based bidder is evaluated to address the partial observability inherent in decentralized MRTA. Since each robot's observation provides only local and incomplete information, the allocation process can be viewed as a Dec-POMDP, where effective decision-making often requires memory [25]. The LSTM maintains a hidden state and cell state that evolve over time, enabling the policy to build an implicit belief state from a history of partial observations. This belief-state representation is then passed through an MLP head to produce task bids at each iteration.

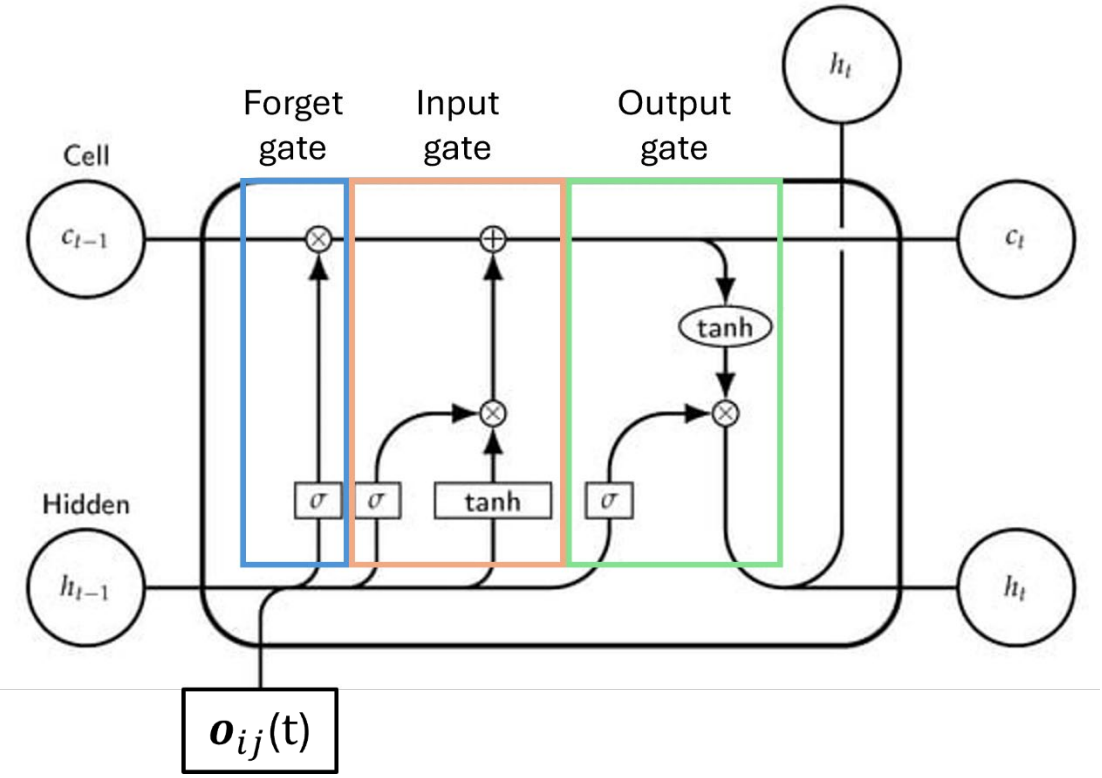

Figure 5.4: LSTM Cell Diagram [26]

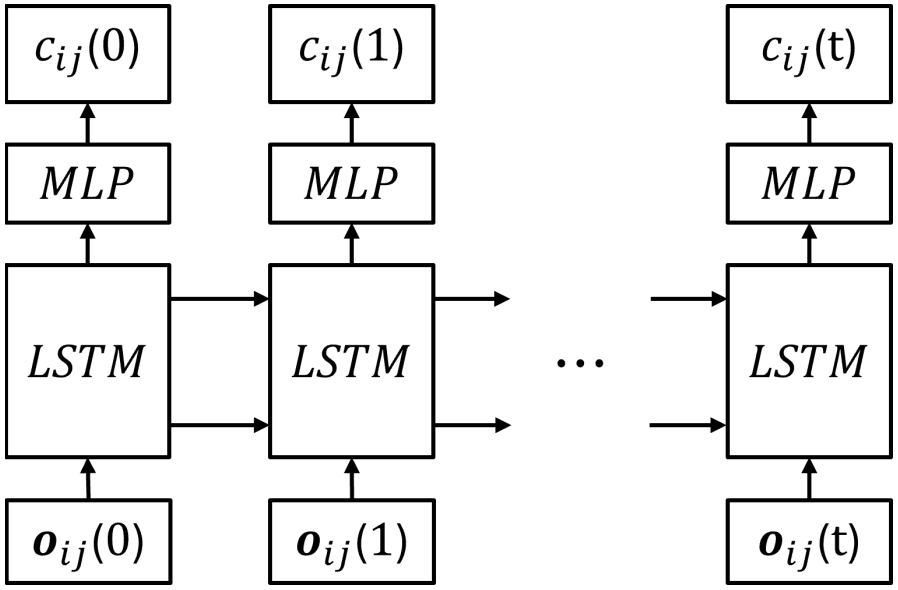

Figure 5.5: LSTM-Based Model as Neural Bidder

This architecture is particularly suitable for auction-consensus because bid decisions at a given iteration depend not only on current task features but also on evolving team-level context induced by previous auctions and consensus updates. Empirically, the LSTM policy demonstrated strong stability and repeatability during training. Across multiple independent training runs, the average training curve consistently converged to approximately 87% optimality, suggesting that memory-enhanced bidding improves both performance and stability in decentralized coordination.

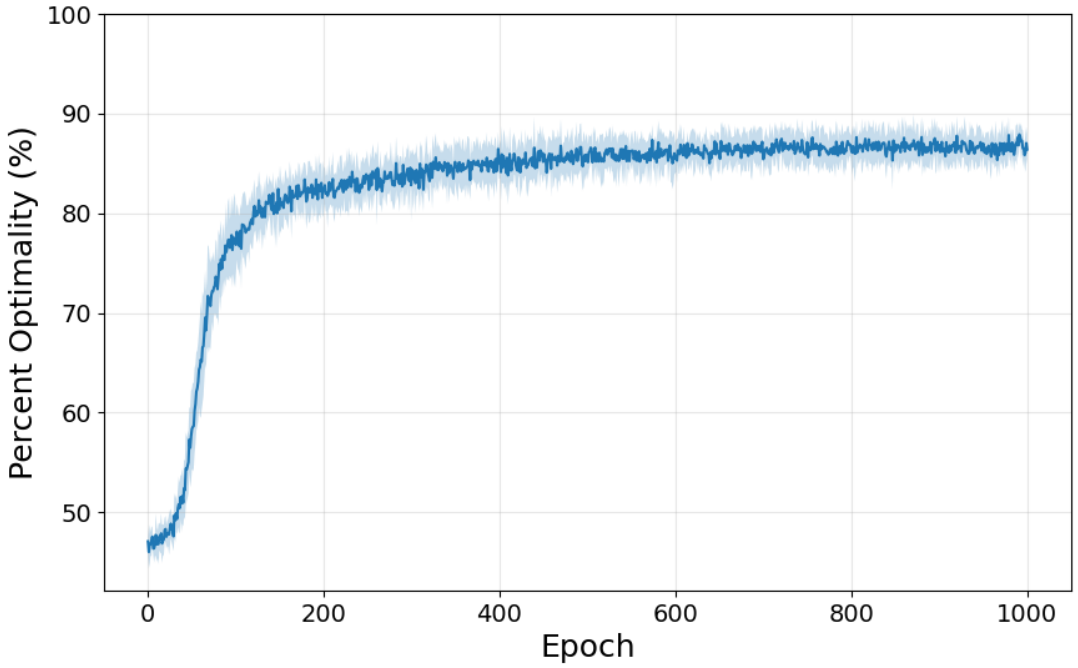

Figure 5.6: Mean Training Curve for the LSTM Actor across 10 Training Runs

5.5.3 Set Transformer Model

A Set Transformer bidder was tested due to the growing use of attention-based, permutation-invariant architectures in MRTA and routing-related learning literature [27][28]. Set-based attention models are attractive because they can process variable-sized task sets

without imposing an arbitrary ordering, and can, in principle, learn global context and task interactions more effectively than simple feedforward models. However, in this study the Set Transformer exhibited unstable training dynamics and failed to learn an effective bidding policy. The training curve showed large fluctuations and did not converge to a stable performance level.

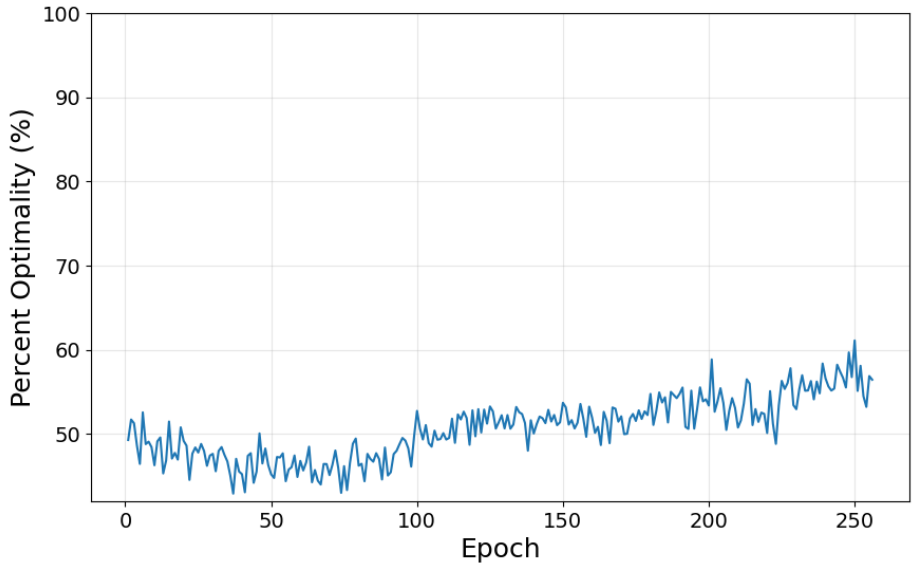

Figure 5.7: Set Transformer Actor Training Curve

A hybrid architecture combining a Set Transformer encoder followed by an LSTM was also evaluated to integrate global set context with recurrent memory. Although the hybrid initially reached performance comparable to the standalone LSTM, it later degraded substantially during training, indicating instability and poor convergence behavior. Due to these issues, both transformer-based variants were excluded from final comparative evaluation.

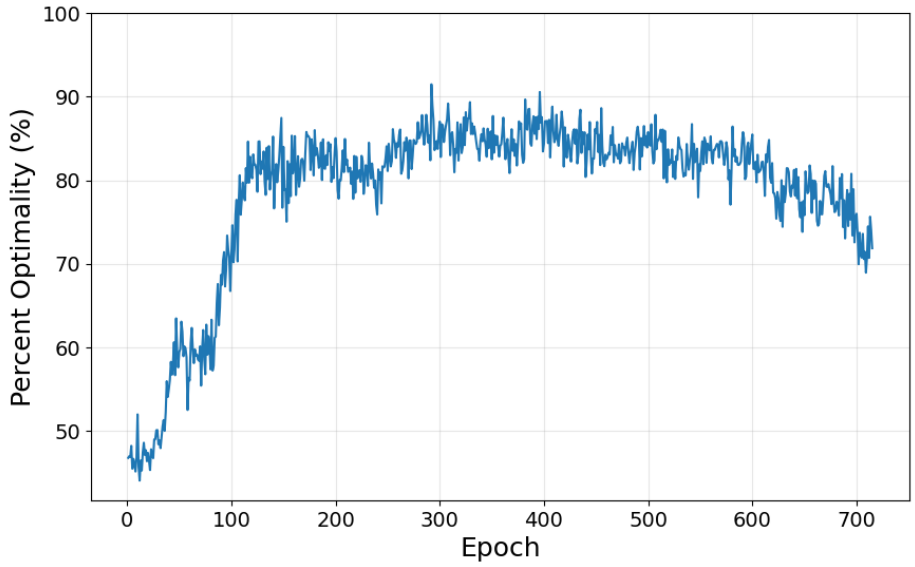

Figure 5.8: Hybrid Set Transformer and LSTM Actor Training Curve

5.6 Learned Bidding Results on MT-SR-IA

5.6.1 Experimental Framework

For the MT-SR-IA formulation used in the learned bidding experiments, models were trained on 1000 randomly generated 2D worlds containing 5 agents and between 10 and 20 tasks. Validation was performed on separate unseen datasets consisting of 1000 worlds for swarm sizes of 5, 10, 15, and 20 agents. Task counts scaled proportionally with swarm size, maintaining between 2 and 4 tasks per agent. Workspace dimensions varied between 25 × 25 and 55 × 55 units to introduce geometric diversity and assess generalization. All decentralized algorithms were evaluated under fully connected communication graphs.

Performance was evaluated using Percent Optimality and the Number of Auction-Consensus Iterations to Convergence. Timeouts, defined as failure to reach consensus within a predefined iteration limit, were also recorded for learned bidding experiments to assess convergence stability.

5.6.2 Performance Comparisons

Final performance comparisons were conducted between CBBA, the NAM actor, and the LSTM actor.

Excluding timeout cases, the NAM actor achieved the highest mean optimality across all swarm sizes, reaching approximately 90% optimality for 5-agent swarms and gradually decreasing to around 87% for 20-agent swarms. These results indicate that the learned additive scoring can outperform CBBA's hand-crafted scoring rule when convergence is achieved. With further work to mitigate timeouts, the NAM actor represents a strong alternative to classical CBBA scoring.

The LSTM actor consistently outperformed CBBA at smaller and medium swarm sizes while exhibiting no convergence failures across all validation sets. However, its performance degraded with increasing swarm size, and by 20 agents, the LSTM actor's optimality became comparable to CBBA. This suggests that while memory-based bidding mitigates partial observability, additional architectural or coordination mechanisms may be required to maintain performance at larger scales.

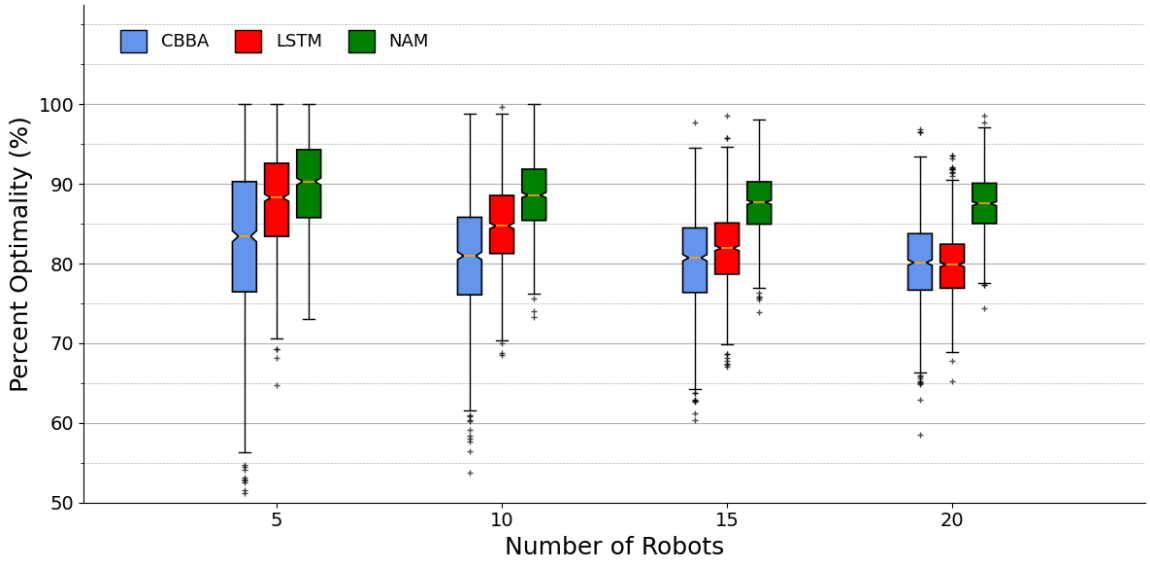

Figure 5.9: Percent Optimality vs Number of Robots for MT-SR-IA

Across all swarm sizes, both learned bidders exhibited convergence behavior comparable to CBBA. The NAM actor generally required more iterations and displayed heavier tails in the iteration distribution, reflecting its occasional convergence instability. In contrast, the LSTM actor maintained stable convergence characteristics without timeouts, reinforcing its robustness despite reduced optimality at larger swarm sizes.

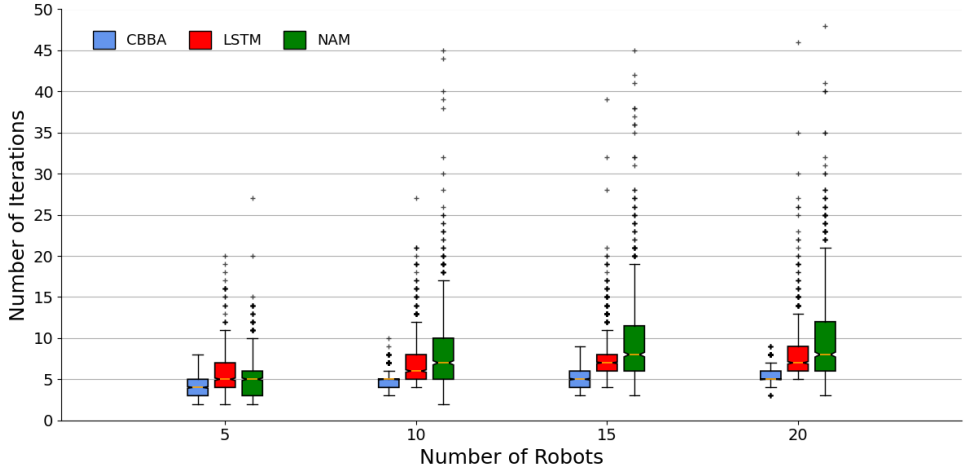

Figure 5.10: Iterations vs Number of Robots for MT-SR-IA

Overall, the results demonstrate that learned bidding policies can significantly improve solution quality over CBBA while preserving decentralized execution and convergence properties. The LSTM model offers the best balance between performance, stability, and reliability, whereas the NAM model achieves the highest optimality when convergence is successful. Transformer-based approaches, while theoretically appealing, proved ineffective in this decentralized auction-consensus setting.

CHAPTER VI

CONCLUSION

This thesis investigated methods for improving decentralized multi-robot task allocation through enhancements to auction-consensus coordination. Two complementary strategies were developed and evaluated: a deterministic loss-aware bidding mechanism (ACALM) and a reinforcement learning-based learned bidding scheme integrated within a centralized training and decentralized execution framework. Both approaches were designed to preserve the distributed structure and communication constraints of classical auction-consensus algorithms while improving global solution quality relative to greedy bidding strategies.

The first contribution, ACALM, introduced a loss propagation mechanism that modifies traditional greedy reward-based bidding by accounting for the compounded cost of sequential task rejection. Rather than bidding solely based on immediate marginal reward, agents incorporate regret-aware adjustments that better reflect long-term allocation consequences. Extensive large-scale simulations under the ST-SR-IA assumption demonstrated that ACALM consistently improves percent optimality compared to CBAA and GCAA, particularly as swarm size increases. Although ACALM incurs additional consensus iterations due to loss propagation, it maintains predictable convergence behavior and scales effectively to swarms of up to 120 agents. These results show that deterministic loss mechanism can significantly narrow the gap between decentralized and centralized optimal allocations.

The second contribution replaced handcrafted bidding rules with learned neural bidding policies trained using reinforcement learning. Under the MT-SR-IA setting, neural bidders were

embedded directly into the auction-consensus framework, preserving decentralized execution while enabling adaptive bidding behavior. Using a centralized training with decentralized execution paradigm, agents learned to produce cooperative bids guided by global performance signals. Experimental results demonstrated that learned bidders can outperform classical CBBA scoring, particularly in moderate swarm sizes. Among the architectures evaluated, recurrent LSTM-based bidders achieved the best balance between performance and convergence stability, while additive models achieved high optimality but exhibited occasional instability. These findings illustrate that auction-consensus coordination can be enhanced through data-driven adaptation without sacrificing distributed execution.

Together, these two approaches reveal a broader insight: the quality of decentralized task allocation is heavily influenced by the structure of the bidding rule. By either analytically incorporating loss (ACALM) or learning cooperative bidding behavior (reinforcement learning), it is possible to substantially improve global performance while maintaining decentralized coordination. Deterministic loss propagation offers strong scalability and predictable behavior, whereas learned bidding offers flexibility and the ability to adapt to richer feature spaces and more complex environments.

Despite these advancements, several limitations remain. ACALM is currently formulated under the ST-SR-IA assumption and increases communication overhead due to additional consensus rounds. The learned bidding approach, while flexible, lacks formal convergence guarantees and is sensitive to neural architecture design and training stability. Transformer-based architectures, although theoretically appealing, exhibited instability within the auction-consensus framework, indicating that not all expressive models are well suited for distributed bidding dynamics.

Future work will focus on extending both approaches to more complex MRTA settings, including heterogeneous robot capabilities, asynchronous communication, dynamic task arrivals, and multi-objective optimization. Reducing communication overhead in loss-based bidding while preserving convergence guarantees remains an important direction. For learned bidding, improving scalability to larger swarm sizes, incorporating richer task features, and exploring hybrid approaches that combine deterministic loss with neural adaptation represent promising research avenues. Additionally, developing theoretical guarantees for learning-enhanced auction-consensus systems would strengthen the foundational understanding of decentralized adaptive coordination.

In summary, this thesis demonstrates that enhancing auction-consensus bidding, such as through either structured loss mechanisms or reinforcement learning, provides a viable path toward closing the performance gap between decentralized coordination and centralized optimal task allocation. These findings contribute to the broader development of scalable, adaptive, and efficient multi-robot systems capable of operating in complex and distributed environments.

REFERENCES

[1] L. V. Nguyen, "Swarm intelligence-based multi-robotics: A comprehensive review," *AppliedMath*, vol. 4, no. 4, pp. 1192–1210, 2024.

[2] H. Chakraa, F. Guerin, E. Leclercq, and D. Lefebvre, "Optimization techniques for multi-robot task allocation problems: Review on the state-of-the-art," *Robot. Auton. Syst.*, vol. 168, p. 104492, 2023.

[3] F. Quinton, C. Grand, and C. Lesire, "Market approaches to the multi-robot task allocation problem: A survey," *J. Intell. Robot. Syst.*, vol. 107, no. 2, p. 29, 2023.

[4] H.-L. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE Trans. Robot.*, vol. 25, no. 4, pp. 912–926, 2009.

[5] S. Hunt, Q. Meng, C. Hinde, and T. Huang, "A consensus-based grouping algorithm for multi-agent cooperative task allocation with complex requirements," *Cogn. Comput.*, vol. 6, pp. 338–350, 2014.

[6] X. Qiu, P. Zhu, Y. Hu, Z. Zeng, and H. Lu, "Consensus-based dynamic task allocation for multi-robot system considering payload consumption," in *Proc. China Autom. Congr.*, 2024, pp. 5294–5299.

[7] Y. Ma, X. Li, H. Wang, Y. Kang, T. Cao, Y. Zhang, and J. Yang, "Multi-UAVs collaborative task allocation based on improved consensus-based grouping algorithm," in *Proc. IEEE Int. Conf. Control Autom.*, 2024, pp. 850–855.

[8] W.-Y. Yu, X.-Q. Huang, H.-Y. Luo, V.-W. Soo, and Y.-L. Lee, "Auction-based consensus of autonomous vehicles for multi-target dynamic task allocation and path planning in an unknown obstacle environment," *Appl. Sci.*, vol. 11, no. 11, p. 5057, 2021.

[9] W. Li, Y. Lyu, S. Dai, H. Chen, J. Shi, and Y. Li, "A multi-target consensus-based auction algorithm for distributed target assignment in cooperative beyond-visual-range air combat," *Aerospace*, vol. 9, no. 9, p. 486, 2022.

[10] S. Nayak, S. Yeotikar, E. Carrillo, E. Rudnick-Cohen, M. K. M. Jaffar, R. Patel, S. Azarm, J. W. Herrmann, H. Xu, and M. Otte, "Experimental comparison of decentralized task allocation algorithms under imperfect communication," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 572–579, 2020.

[11] S. Mirzaei and F. Esposito, "An Alloy verification model for consensus-based auction protocols," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst. Workshops*, 2015, pp. 17–22.

[12] M. Braquet and E. Bakolas, "Greedy decentralized auction-based task allocation for multi-agent systems," *IFAC-PapersOnLine*, vol. 54, no. 20, pp. 675–680, 2021.

[13] W. Zhao, Q. Meng, and P. W. H. Chung, "A heuristic distributed task allocation method for multivehicle multitask problems and its application to search and rescue scenario," *IEEE Trans. Cybern.*, vol. 46, no. 4, pp. 902–915, 2016.

[14] J. Weissteiner, J. Heiss, J. Siems, and S. Seuken, "Bayesian optimization-based combinatorial assignment," in *Proc. AAAI Conf. Artif. Intell.*, vol. 37, no. 5, 2023, pp. 5858–5866.

[15] O. Shorinwa, R. N. Haksar, P. Washington, and M. Schwager, "Distributed multirobot task assignment via consensus ADMM," *IEEE Trans. Robot.*, vol. 39, no. 3, pp. 1781–1800, 2023.

[16] S. Chopra, G. Notarstefano, M. Rice, and M. Egerstedt, "A distributed version of the Hungarian method for multirobot assignment," *IEEE Trans. Robot.*, vol. 33, no. 4, pp. 932–947, 2017.

[17] R. Patel, E. Rudnick-Cohen, S. Azarm, M. Otte, H. Xu, and J. W. Herrmann, "Decentralized task allocation in multi-agent systems using a decentralized genetic algorithm," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2020, pp. 3770–3776.

[18] M. Goarin and G. Loianno, "Graph neural network for decentralized multi-robot goal assignment (DGNN-GA)," *IEEE Robot. Autom. Lett.*, vol. 9, no. 5, pp. 4051–4058, 2024.

[19] Z. Zhang, X. Jiang, Z. Yang, S. Ma, J. Chen, and W. Sun, "Scalable multi-robot task allocation using graph deep reinforcement learning with graph normalization," *Electronics*, vol. 13, no. 8, p. 1561, 2024.

[20] S. Paul, S. Chowdhury *et al.*, "Learning multi-robot task allocation using capsule networks and attention mechanism (CAM)," *Robot. Auton. Syst.*, vol. 193, p. 105085, 2025.

[21] W. Dai, U. Rai, J. Chiun, Y. Cao, and G. Sartoretti, "Heterogeneous multi-robot task allocation and scheduling via decentralized reinforcement learning with a generalized policy," *IEEE Robot. Autom. Lett.*, vol. 10, no. 3, pp. 2654–2661, 2025.

[22] S. Mitchell, "PuLP: A linear programming toolkit for Python," 2011. [Online]. Available: https://coin-or.github.io/pulp/

[23] R. C. Blair and J. J. Higgins, "Comparison of the power of the paired samples t test to that of Wilcoxon's signed-ranks test under various population shapes," *Psychol. Bull.*, vol. 97, no. 1, pp. 119–128, 1985.

[24] R. Agarwal, L. Melnick, N. Frosst, X. Zhang, B. Lengerich, R. Caruana, and G. E. Hinton, "Neural additive models: Interpretable machine learning with neural nets," in *Proc. 35th Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2021.

[25] T. Ni, B. Eysenbach, and R. Salakhutdinov, "Recurrent model-free RL can be a strong baseline for many POMDPs," in *Proc. 39th Int. Conf. Mach. Learn. (ICML)*, 2022.

[26] M. Krichen and A. Mihoub, "Long short-term memory networks: A comprehensive survey," *AI*, vol. 6, no. 9, p. 215, 2025.

[27] J. Bichler, A. Matoses Gimenez, and J. Alonso-Mora, "SADCHER: Scheduling using attention-based dynamic coalitions of heterogeneous robots in real-time," in *Proc. IEEE Int. Symp. Multi-Robot Multi-Agent Syst. (MRS)*, 2025.

[28] H. Gao, X. Zhou, X. Xu, Y. Lan, and Y. Xiao, "AMARL: An attention-based multiagent reinforcement learning approach to the min-max multiple traveling salesmen problem," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 7, pp. 9758–9772, 2024.

APPENDIX

APPENDIX

PUBLICATIONS AND PRESENTATIONS

[1] D. T. Nu, R. Mok, A. Bhaiwala, E. Sayer, J. Rodriguez, and A. Nokes, “Applying pose estimation techniques to visualize drone trajectory in GPS-denied environments,” in Proc. AIAA Regional Student Conf., Jan. 2024, doi: 10.2514/6.2024-84491.

[2] J. Rodriguez and W. Dong, “Neural network aided drone estimation in GPS-denied environments,” in Proc. AIAA Regional Student Conf., Jan. 2025, doi: 10.2514/6.2025-98890.

[3] J. Rodriguez, W. Dong, and Q. Lu, “Auction consensus algorithm with loss mechanism for decentralized task allocation,” in Proc. IEEE Int. Symp. Multi-Robot & Multi-Agent Syst. (MRS), Singapore, Dec. 2025, doi: 10.1109/MRS66243.2025.11357252.

[4] J. Rodriguez, W. Dong, and Q. Lu, “Auction consensus algorithm with loss mechanism for decentralized task allocation,” Poster presented at the 37th Annu. Conf. Great Minds in STEM (GMiS), San Diego, CA, USA, Oct. 2–4, 2025.

VITA

Jose Alberto Rodriguez earned his Master of Science in Engineering in Electrical Engineering at the University of Texas Rio Grande Valley (UTRGV) in May 2026. Before joining UTRGV, he obtained his Bachelor of Science in Aerospace Engineering from the University of Texas at Austin in May 2024. During his time at UTRGV, Jose served as a graduate research assistant at the Center for Multidisciplinary Research Excellence in Cyber-Physical Infrastructure Systems (CREST) under Dr. Wenjie Dong and at the Multiple Autonomous Robot Systems (MARS) Lab under Dr. Qi Lu where research was done for autonomous robot systems. For any additional information, please reach out to jr5831250@gmail.com