COORDINATION OF ROBOT SWARMS IN CROWDED AND CONFINED CONDITIONS

A Dissertation
Presented to
The Academic Faculty

By

Kehinde Aina

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
Institute of Robotics and Intelligent Machines
Department of Robotics

Georgia Institute of Technology

May 2024

© Kehinde Aina 2024

COORDINATION OF ROBOT SWARMS IN CROWDED AND CONFINED CONDITIONS

Thesis committee / Approved by:

Prof. Daniel I. Goldman, Advisor
School of Physics
Georgia Institute of Technology

Prof. Seth Hutchinson
School of Interactive Computing
Georgia Institute of Technology

Prof. Sehoon Ha
School of Interactive Computing
Georgia Institute of Technology

Prof. David Hu
School of Mechanical Engineering
Georgia Institute of Technology

Prof. Dana Randall
School of Computer Science
Georgia Institute of Technology

Date approved: January 31st, 2024

To my parents,

Joseph and Victoria Aina.

ACKNOWLEDGMENTS

I am grateful to God Almighty for seeing me through the culmination of this dissertation and for guiding me to supportive individuals, both within and beyond the academic sphere, who have contributed to the successful completion.

I wish to express my gratitude to Professor Daniel I. Goldman, my advisor, whose support and guidance have been indispensable throughout my doctoral journey. His mentorship, astute insights, and provision of essential resources have not only shaped but significantly enhanced my research acumen and scholarly endeavors. Also, I am profoundly appreciative of the valuable feedback provided by my esteemed thesis committee members — Professor Seth Hutchinson, Professor Dana Randall, Professor Sehoon Ha, and Professor David Hu.

I wish to acknowledge with utmost gratitude the great contributions and support of our collaborators, colleagues and associates within and beyond the realms of Georgia Tech. Noteworthy among these collaborators are Professor M.A.D. Goodisman, Professor Sehoon Ha, Professor Meredith D. Betterton, Professor Andrea W. Richa, Ram Avinery, Hui-Shun Kuan, Hosain Bagheri, and Carl Dyson, whose collective intellectual contributions have enriched the breadth and scope of my scholarly pursuits. Moreover, I express profound appreciation for the contributions and collaborative spirit of the past and current members of the Complex Rheology and Biomechanics lab. I am grateful for the insightful conversations and assistance from: Enes Aydin, Yasemin Aydin, Jennifer Rieser, Andras Karsai, Bahnisikha Dutta, Shengkai Li, Daniel Soto, Akash Vardhan, Kelimar Diaz-Cruz, Laura Treers. Many thanks also to the rest of my colleagues, the professors, and the administrators of the institute.

Profound gratitude and appreciation are extended to my family, whose steadfast support and encouragement supplied the resilience needed to overcome challenges. The entirety of my doctoral journey at the Georgia Institute of Technology has proven to be remarkably enriching and rewarding, attributable to the combined support, scholarly guidance, and

collaborative efforts experienced throughout this academic pursuit.

TABLE OF CONTENTS

Acknowledgments iv

List of Tables ix

List of Figures x

Summary xviii

Chapter 1: Introduction 1

Chapter 2: Background 7

2.1 Local interactions and coordination in swarm robotics 7

2.2 Fault tolerance in multi-agent systems 9

2.3 Deep reinforcement learning for multi-agent systems 11

2.4 Learning via local interactions in natural swarms 13

Chapter 3: Learning to Avoid Clogging in Multi-robot Excavation 15

3.1 Methods 16

3.1.1 Collective Task 16

3.1.2 Robot Controller 18

3.1.3 Adaptive Protocol 20

3.2 Experiment 26

3.3 Results 27

3.4 Discussion 33

3.5 Conclusion 34

Chapter 4: Minimally Sensing Fault-Tolerant Multi-Robot Coordination 36

4.1 Multi-robot Collective Excavation 38

4.1.1 Robot Controller 40

4.2 Active Contact Response (ACR) Method 41

4.3 Experiment 44

4.4 Results and Discussion 47

4.5 Conclusion 49

Chapter 5: Deep Reinforcement Learning for Multi-Agent Collective Excavation 51

5.1 Deep Reinforcement Learning 51

5.1.1 Multi-Agent Deep Reinforcement Learning 52

5.2 Multi-agent Clog Control 53

5.2.1 Decentralized POMDP 55

5.2.2 Digital Pheromone as Stigmergy 56

5.2.3 Curriculum Learning 57

5.3 Experimental Setup 59

5.4 Simulation Results and Discussion 60

5.4.1 Improving Scalability in Dynamic Environments 65

5.5 Conclusion and Future Work 67

Chapter 6: Coordination and Adaptation Strategies in Living Collectives 68

6.1 Methods 68

6.1.1 Ant excavation experiment 68

6.1.2 Video analysis 70

6.1.3 Cellular Automata (CA) Model 73

6.1.4 Ant speed measurements 74

6.1.5 Model fitting 74

6.2 Results & Discussion 77

6.2.1 Ant digging experiments 77

6.2.2 Participation trend 79

6.2.3 Cellular Automata behavioral model 81

6.2.4 The One At a Time model (OAT) 87

6.3 Summary and Conclusion 88

Chapter 7: Conclusions and Future Work 90

7.1 Future directions 93

Appendices 94

Appendix A: Experimental setup and robot Design 95

Appendix B: C++ code snippet for robot controllers 99

References 120

LIST OF TABLES

5.1 Hyper-parameter Values 60

6.1 Parameters used to calibrate the cellular-automata simulation to the experiment 76

LIST OF FIGURES

1.1 Examples of natural swarm, ranging from flock of birds, colony of bees, school of fish, colony of ants. 2

3.1 Experimental apparatus consisting of robots and a tunnel. **(A)** Components of an ant robot (two left images): Inertia Measurement Unit (IMU) sensor for absolute orientation; Wheel encoders for localization via robot odometry/dead reckoning; Capacitive touch sensor shells for detecting collisions; Force sensitive resistors for gripper tactile feedback, Magnetometer for detecting pellets and Digging Area; Pixy camera for tunnel navigation and localization; Terminal rod for charging and detection of Home area. In the tunnel (rightmost image): Velostat sheets for lining the tunnel to distinguish robot-robot collision (copper-copper) from robot-wall collision (copper-velostat); Granular media to simulate model cohesive soil excavated by ants. **(B)** Schematic diagram of the confined tunnel: a robot starts from the Home Area, decides whether to enter the tunnel and excavate some pellets, or decides to go to the Resting Bay to avoid interference at the tunnel entrance. 17

3.2 Block diagram for the individual robot controller. State transitions are event-based. P_e: probability of tunnel entry (or go to dig). $P_e^{'}$: probability of resting (or staying at home). P_r: probability of "giving up" after colliding with another robot in the tunnel. $P_r^{'}$: probability of continuing to dig after colliding with anther robot. 19

3.3 Probability sweep over tunnel length with selected equation models. **(A)** Reversal probability sweep vs Tunnel length. Plot shows that high to medium reversal probabilities give the highest excavation rate for short tunnels, while low reversal probabilities give the highest excavation rates for long tunnels. **(B)** Selected equation for modelling Reversal probabilities as a function of tunnel length for multi-robot experiments. **(C)** Learning update value for Entrance probabilities as a function of tunnel length for multi-robot experiments. 22

3.4 Three snapshots of the top view of multi-robot excavation experiments. **(A)** Pellets at one body-length (1BL) of the tunnel which is the initial position of the pellets at the beginning of each trial, **(B)** pellets at five body-length (5BL) of the tunnel after robots have excavated 1200g of pellets, **(C)** pellets at ten body-length (10BL) of the tunnel after robots have excavated 2700g of pellets. 28

3.5 Estimated tunnel length, $L(k)$, Reversal probability, P_r, and Entrance probability, P_e over time for individual robots based on Adaptive rule. Dashed colored lines represent robot A (red), robot B (blue), and robot C (yellow). Solid black line represents actual tunnel length measurement. (A) Plot shows how individual robots track the actual growing tunnel, based on dead-reckoning method and exponential moving average formula described in equation 3. (B) Reversal probability values for individual robots as a function of time. These curves closely match the curve described in Figure 3.3B. (C) Plot of the entrance probability values over time. This shows how robots initially started off with equal digging "desires" and gradually become unequal "desires" as the tunnel length grows. 29

3.6 Excavated Pellets vs Time for three different protocols. *Active protocol:* all robots have an entrance probability of 1.0 but a reversal probability of 0; *Reversal protocol:* all robots have an entrance probability of 1.0 and reversal probability of 0.8; *Adaptive rule:* entrance probability is a function of inverse square of tunnel length, while reversal probability is a function of inverse square-root of tunnel length. **(A)**Individual trial comparison of excavation experiments **(B)** Mean excavated pellets, shaded areas correspond to standard deviation from three experiments. 30

3.7 Experimental space-time overlap heat maps of robot positions along the tunnel (x-axis) measured in body-lengths (BL). Y-axis is the time duration of the experiment in minutes. White straight lines show how fast the tunnel grows which depends on the running protocol in the robots. Robots start from the *Home area* (right side) and transit to the *Digging area* (left side) continuously while excavating the pellets 31

3.8 Time spent in various states by individual robots, for the three protocols: Active, Reversal and Adaptive. (A) Time spent at the resting and deposition areas. (B) Time spent in contact with other robots while in the tunnel. Boxes span 25th to 75th percentile, whiskers are between the minimal and maximal values, inner line is the median, $\times$ for means, and circles (solid and empty) represent actual data points. 32

3.9 Lorenz curves of robot workload distribution for 1-hour "epochs" of 3-hour experiment. The curves represent: 1 hour duration (red), 2 hours duration (blue), and 3 hours duration (green). Black solid line represents the line of equality. Cumulative fraction of work is the number of trips the robots took into the tunnel in an attempt to excavate pellets. In Reversal protocol, all robots are mostly inside the tunnel, either going in to excavate pellets or going out to dump or "give-up" and re-enter the tunnel. Adaptive rule on the other hand leads some robots to rest and become less active over time. . 33

4.1 A model constrained and confined environment for multi-robot collective excavation of cohesive granular media. The three robots shown respond to environmental cues using onboard sensors. Each robot is 32 cm long. . . . 38

4.2 **1.** Block diagram for the individual robot controller. State transitions are event-based. P_e is the probability of switching to Active Mode (Going to Dig). P_e' is the probability of switching to Conservative Mode (Resting). P_r is the probability of "giving up" when a contact with another robot is detected. R_c is the probability that a detected contact is from a faulty robot. **2.** Experimental apparatus consisting of three active robots (A, B, C) and a faulty robot perpendicularly positioned at the start of the experiment at the middle of tunnel. **(A)** Top view snapshot of multi-robot excavation experiment, showing three active robots starting out at the Home area, while the faulty robot is positioned at the middle of tunnel. Markers attached to the top of the robots are visible. **(B)** Schematic diagram of experimental setup showing three active robots (green, blue, and red), and one faulty robot (yellow). The adopted coordinate system's origin is as indicated in the diagram. 39

4.3 **1.** Comparing excavation performance between our baseline and Active Contact Response method. Three trials per tested method. Mean excavated pellets are the solid dotted and dash-dotted lines. Shaded areas correspond to the standard deviation from the three experiments. **2.** Final configuration of the faulty robot after each experimental trial for **(A)** three trials of Baseline method, and **(B)** three trials of Active Contact Response method. The home area is located at a tunnel length of 0. 47

4.4 **1.** Active and faulty robot trajectories tracked overtime for **(A)** Baseline method, and **(B)** Active Contact Response method. Tunnel length position 0 cm is close to the home area, while tunnel length position 300 cm is close to the digging area. **2.** Three snapshots of the top view of the multi-robot experiment using the Active Contact Response method. **(A)** Initial configuration of faulty robot obstructs traffic flow, resulting in congestion and clustering at the middle of the tunnel. **(B)** Subsequently, active robots update their collision map and maneuver around the faulty robot to reach the excavation area.**(C)** Finally, active robots effectively reposition faulty robot into a less obstructive configuration through active pushing and nudging. 48

5.1 An overview of our environment showing the top-view of multi-robot excavation setup *(A)*, and similar features in the simulation model *(B)*. 54

5.2 Proposed Scalable Decentralized MADRL Framework with Stigmergic Communication. 57

5.3 Schematic of digital pheromone and agent's restricted field of view. Agents' movements in the environment create virtual maps and gradients, depicted by the different shades of gray, that agents can observe within their surrounding. Red arrows illustrate the field of view of an agent. Blue agent is laden with food, while green agents represent searching agents. 58

5.4 Learning curve comparison for the four different techniques. Our approach, IQL + GSC, achieved the highest rewards for four and five agent scenarios. 61

5.5 Cumulative excavation plot comparison for four different MADRL techniques for up to five agents. 62

5.6 Lorenz curves comparison between learning with stigmergic only **(A)** and learning with stigmergy and curriculum learning **(B)**. 63

5.7 Snapshots of various scenarios during testing. **(A)** Four agents often get stuck in the tunnel when trained without stigmergy. **(B)** Stigmergy prevents agents from getting stuck, which enables effective coordination. **(C)** Five agents with stigmergy can still get stuck due to the difficultly of the scenario. **(D)** Curriculum learning enables efficient learning in difficult scenarios, and the discovery of a bio-inspired strategy: unequal work distribution to avoid clogging. 64

5.8 Comparison of our proposed Stigmergic MADRL method with popular multi-agent learning algorithms. 66

6.1 Experimental setup – 3D rendering. A custom-made plastic frame is holding acrylic sheets 2.5 mm apart. The accessible part of the system is 14 cm wide, and 21 cm tall. Roughly 2/3 of the volume is filled with a substrate - 0.7 mm beads, with the moisture content adjusted to 10% by weight. A top container serves as an entrance chamber where 40-70 ants are placed to start the experiment. The experimental trials are recorded using a commercial webcam. 69

6.2 Experimental recording and tracking. Each trial was recorded at 24 frames per second using a commercial webcam. **(A)** The digging arena is enclosed between acrylic sheets and composed of glass beads. A green background is used to generate suitable contrast between the substrate and excavated tunnels. Image 40 minutes after the introduction of ants into the apparatus is shown **(B)** The color difference between ants, substrate, and background allows for substrate tracking (shaded purple) as well as ant tracking (shaded blue). Together, the number of pixels associated with ants below the substrate is used to quantify the participation in the digging activity. **(C)** Motion detection of ants was found to be a robust method to track the exploration of the system, and thereby allowed us to quantify the excavated area over time. We calculate the difference between each pair of sequential frames, apply a threshold, and accumulate the result over time (purple shade). See Methods for additional details. 72

6.3 Analysis of space-time detection maps. (A) A branch-less tunnel was divided into strips of equal longitudinal distance, shown in alternating black and white strip color. (B) Ant pixels were determined based on a darkness threshold, and binarized pixels were summed over every longitudinal strip, to generate a density profile over time. Ant presence was observed deeper in the tunnel as it grew through digging. (C) The space-time detection maps revealed distinct tracks when observed at minutes long windows. We used a Hough transform [123] to detect straight (i.e. constant speed) trajectories fragments (green dashed lines). From the collected line slopes we calculated an average observed ant walking speed in tunnels. 75

6.4 Least-squares (χ^2) optimization of the model fit through random parameter sampling. In each iteration, we sampled thousands of parameter variations and calculated the mean-square distance between the simulated excavation-rate trend and the experimental one. We plotted χ^2 distances vs. each one of the fitted parameters (blue points), as well as an outline of their basin as a guide (black dashed line). After each such iteration we narrowed the search range for parameters with a clear single-minimum in their basin. 76

6.5 Cellular-automata simulation of ant excavation. Within this model, ants from the outside reservoir (circle on top) randomly decide to enter the system, with a probability determined by each individual. Once inside, the ants (blue half-arrows) move along a tunnel composed of discrete cells. Each cell accommodates up to two ants. An ant moves from the entrance (top) towards the substrate (bottom, in gray), and cannot move into it. Once an ant reaches the first substrate cell, it removes (i.e. excavates) pellets from it, one every time step. After reaching its holding capacity, an ant reverses and moves back towards the entrance to deposit the excavated pellets. Substrate cells are initialized to contain a consistent number of pellets, which are then depleted by the incoming ants. The number of pellets remaining in the first reachable substrate cell is indicated by the degree of filling of lighter gray. **(A)** Four consecutive time steps, showing four ants in a single tunnel, at first (time T) with one heading out towards the entrance, two on their way towards the substrate, and one already at the substrate interface. At $T + 1$, another ant reaches the substrate, and the third is a cell behind. Since the next cell is occupied by two ants, the latter ant, at time $T + 2$, decides to reverse and head back towards the entrance without excavating. Also at $T + 2$, an ant steps outside and increases the count of ants in the outside reservoir (number in circle). The ants at the interface remain there to excavate until either the cell is depleted or their individual capacity is full. **(B)** In a multi-tunnel model, ants from the outside reservoir randomly decide to enter, as before, and also randomly uniformly choose which tunnel to go into. Once inside, the dynamics are as in a single independent tunnel. 78

6.6 Flow morphology tracked by ant motion detection and color thresholding. (A) Tunnel exploration over time in a single experiment. Left to right – the system, as it evolves in time slices of 3 hours, over the course of the first 9 hours. The substrate is tracked using color thresholding (see Methods) and displayed in gray. Above-substrate is colored in light green. Tunnel color represents a spatial mapping of first-exploration time, ranging from darkest for earliest and brightest brown for latest (3 hours) within the time slice. Newly detected flow indicates freshly dug passages. Missing flow in existing tunnels may indicate a loss of interest, due to a blocked entrance or other reasons. (B) Tunnel formations mapped by first-exploration time, over 50 hours. Left to right – three independent experiments. Unexplored areas are in black. The color indicates first-exploration time, according to the color map below. 80

6.7 Excavation area dynamics for multiple trials of *S. invicta* digging. **(A)** Area excavated over time in nine independent trials (colored points), along with their average (black line). A clear trend emerges where at the initial phase there is almost constant excavation rate, followed by a rapid decay, and at longer times a slower decay. **(B)** Area growth rate over time, with colors as in (A). A consistent feature of rapid excavation appears in all trials at the first few hours, followed by fast decrease in rate and a sharp transition, leading to slow decay in rate at long times. 81

6.8 A varying fraction of the ant collective actively participate in digging at any time. (A) Tracked ant participation in the recorded experimental trials. Each colored line represents an individual trial. The average trend (black) peaks around 50% and later fluctuates around constant participation of about 27.8% (green dashed line). (B) CA model simulations with varying implemented features. In both a single tunnel (blue) and multiple tunnel models (green), the participation monotonically increases over time, though faster for multiple tunnels. To the multiple tunnels we added a tendency to rest and maintain a 27.8% work to rest ratio (orange). Once "agitation" is added to the previous model on top of resting behavior (brown), it overrides the behavior at short tunnel lengths which manifests as a high participation peak, followed by a sharp fall. 84

6.9 Experimental and computer simulated excavation rate over time. CA model simulation fit to the excavation rate (brown line) over the experimental average (black line) and its standard deviation (shaded). The fit was generated by a model including a mean-free-path dependent "agitation" override. The same model was also run, for reference, without "agitation" (orange line), also without resting behavior (green line), and finally also without multiple tunnels (blue line, multiplied by 10). Both models, as well as the experimental trend, are consistent with a one-at-a-time like $t^{-1/2}$ asymptotic behavior (red dotted line, scaled to match experimental). 85

6.10 Parameter variation effects on modeled excavation rate. The best-fit model parameters were perturbed one at a time, around the best-fit value, while keeping the rest fixed. The resulting variations in computed excavation rate (averaged over 11 runs) are depicted in colored lines, overlaying the experimental average (black line) and its standard deviation (shaded). **(A)** Varying number of tunnels. **(B)** Varying steady-state ant participation rate in digging. **(C)** Varying "agitation" relaxation length L_0. **(D)** Varying work-to-rest imbalance tolerance τ_0. 86

6.11 Illustration of the One At a Time model. A single ant walks back and forth from the entrance of the tunnel to the tip. It digs at the tip and deposits the dug pellets at the entrance. The time for digging and deposition τ_{dd} is assumed to be constant. In between the entrance and the tip, the ant walks at speed v across the length of the tunnel L that extends over time. 88

A.1 Schematic showing the main components of the CA model. Cell colors denote soil (gold), empty/excavated cells (white), ants moving toward the excavation site (gray), and ants exiting the tunnel (orange). 96

A.2 Experimental apparatus consisting of robots and a tunnel. **(A)** Components of an ant robot (two left images): Inertia Measurement Unit (IMU) sensor for absolute orientation; Wheel encoders for localization via robot odometry/dead reckoning; Capacitive touch sensor shells for detecting collisions; Force sensitive resistors for gripper tactile feedback, Magnetometer for detecting pellets and Digging Area; Pixy camera for tunnel navigation and localization; Terminal rod for charging and detection of Home area. In the tunnel (rightmost image): Velostat sheets for lining the tunnel to distinguish robot-robot collision (copper-copper) from robot-wall collision (copper-velostat); Granular media to simulate model cohesive soil excavated by ants. **(B)** Schematic diagram of the confined tunnel: a robot starts from the Home Area, decides whether to enter the tunnel and excavate some pellets, or decides to go to the Resting Bay to avoid interference at the tunnel entrance. 97

A.3 Proposed experimental setup for investigating fault tolerance control. It consists of three active robots (yellow, green and red) and one inactive robot (blue). 97

A.4 Model of a differential drive mobile robot. 98

SUMMARY

The intricate task of coordinating robot swarms in highly constrained and crowded environment is often hindered by the robots' limited capability to gather precise state information of their environment crucial for controlling the group toward their objectives. Such challenges stem from inherent uncertainties and unpredictability within these environments, influencing the interactions between agents and their surroundings. However, studies of dense active matter reveals that coordinated behaviors could arise through self-organization principles, where complex global configurations and patterns emerge from simple, local, and social interactions.

This thesis aims to exploit these unavoidable features of dense active matter to enable the spontaneous coordination of robot swarms in the absence of global control, and explicit information about the state of other robots. Specifically, an adaptive clog control technique facilitates the emergent coordination of robot swarms through unplanned contact interactions. This technique relies on inter-robot contacts and collisions amid noisy estimations of their locations and environmental changes. Persistent clogs in narrow tunnels become resolved through voluntary directional changes and individual participation modulation within group tasks.

Next, we improved and extended our algorithm to handle scenarios where an individual robot's failure could detrimentally impact group performance. The Active Contact Response (ACR) algorithm imparts fault tolerance within the clog control mechanism, allowing the repositioning of faulty robots to less obstructive configurations, ensuring continuous traffic flow within the tunnel and completion of tasks.

Additionally, these characteristics of crowded and confined systems address foundational challenges in multi-agent reinforcement learning. By establishing an indirect communication channel through the shared environment, agents can estimate and reason about the statistics of other agents' activities, directly reducing the environment's non-stationarity

from an individual agent's perspective. Numerical simulations show the potential benefits of leveraging local and physical interactions, and implicit communication via the environment to enhance convergence and scalability within multi-agent reinforcement learning setting.

Finally, a mathematical model was formulated to elucidate and validate the self-organizing behavior observed in naturally occurring swarms navigating crowded and narrow self-generated tunnels without relying on global control or explicit communication. The model's alignment with multi-phase excavation rates observed in biological experiments, suggests the pivotal role of contact and local interactions in governing the self-regulation behaviors exhibited by natural collectives.

CHAPTER 1
INTRODUCTION

In recent decades, there has been a discernible shift in focus towards integrating multiple agents or robots across diverse domains, including search and rescue missions, environmental monitoring, and consumer applications. This transition towards employing and coordinating multiple agents to attain shared objectives is prompted by the escalating complexity of contemporary challenges. The imperative for enhanced efficiency, resilience, scalability, adaptability, and the emergence of more sophisticated capabilities becomes evident in these contexts. The intricacies and interdependence of tasks often surpass the capabilities of individual agents, necessitating the adoption of multiple agents [1, 2, 3].

However, as the number of interacting or collaborating agents grows, the complexity of coordinating and controlling them also escalates. Interference among agents vying for limited resources stands as a major challenge in multi-agent systems. While the field of swarm robotics draws inspiration from the collective dynamics of natural swarms, such as social insects, the contrast lies in the mechanical compliance—conventional robots typically lack this attribute. Hence, most robotic collective control strategies tend to avoid or minimize physical and social interactions. to avoid catastrophic occurrences. Nevertheless, certain scenarios, especially those involving crowded and confined spaces, demand physical interactions among robots, posing challenges to conventional collision avoidance methods.

Natural swarms (Figure 1.1) exhibit robustness in uncertain environments, adapting seamlessly without re-calibrating parameters. However, collision avoidance strategies in multi-robot systems often rely on accurate information about the state of other agents, which becomes impractical in highly constrained and unpredictable environments. Centralized control techniques that depend on global information about all agents in the environment offer optimal solutions for simpler scenarios but pose overhead costs and communication

Figure 1.1: Examples of natural swarm, ranging from flock of birds, colony of bees, school of fish, colony of ants.

challenges, and often intractable when deployed in complex, uncertain, and crowded environments. Decentralized control techniques, akin to those observed in natural swarms like social insects, are often favored in multi-robot systems due to their scalability, robustness, lower computational overhead, and reliance on local interactions among the robots [4, 5, 6].

The objective of this thesis is to address the following question: *how can highly constrained and minimally sensing group of robots achieve coordination and fault tolerance without communication in an unknown and uncertain environment?*. To answer this question, insights from natural swarms and densely confined active matter are explored to discover principles or methods resilient to the challenges of coordinating robots under harsh conditions. We developed a multi-robot collective excavation scenario, drawing inspiration from fire-ant nest construction, wherein multiple robots operate in narrow passages autonomously, relying solely on basic onboard sensing without explicit communication. In contrast to several existing multi-robot coordination methods, such as geometric algorithms [7, 8, 9, 10, 11], artificial potential functions [12, 13], consensus algorithms [14, 15], and distributed optimization algorithms [16, 17], our proposed multi-robot control methods heavily rely on physical contact interactions with other robots and the environment. The robots are designed

to tolerate contact interactions or collisions which serve as sources for learning, reinforcement, and adaptation, enabling the estimation of the environmental state which promotes modulation of individual behaviors to produce emergent behaviors like: clog-control in narrow passages, adaptive workload distribution in non-static environments, as well as fault tolerance in confined multi-robot excavation.

Furthermore, considering recent advancements and potentials in Deep Learning, the methods proposed in this thesis are extended to tackle the fundamental challenges posed by partially observable environments with varying number of concurrent learners. Inspired by the concept of stigmergy observed in social insects [18, 19, 20], an indirect communication medium via the environment, termed virtual pheromone, is introduced to enhance the learning and convergence process in such scenarios. This addresses problems related to non-stationarity and non-scalability in a multi-agent environment while learning optimal policies for collective excavation. Lastly, we also aim to deeply understand the principles by which confined natural swarms can learn to avoid congestion through local interactions while performing tasks such as pellet retrieval in narrow, vision-deprived and crowed environment. We designed our swarm robots with features similar to those of social insects, to enable us probe the unavoidable characteristics of social insects which results in robust emergent self-organization behaviors from local and social interactions.

Concretely, chapter 3 systematically studies the performance of these robots under different reactive behaviors or protocols, demonstrating the mitigation of congestion by certain individuals learning to reduce participation in excavation. Such social interactions, including inter-robot collisions and imprecise estimates of tunnel length, serve as reinforcement in the learning scheme. This learning scheme exhibits a robust response to changing work conditions, even when individuals acquire inaccurate information about their environment (such as tunnel length). Chapter 4 delves into scenarios where robot breakdowns in confined environments can lead to cascading effects, hindering or preventing task completion. The Active Contact Response (ACR) algorithm is introduced to imbue fault tolerance into the

adaptive learning algorithm discussed in Chapter 3. The ACR method utilizes an egocentric dynamic contact map, enabling individual robots to localize and track contact interaction histories. Upon contacting another robot, the robot references the contact map history to identify if it has encountered a stationary faulty robot. Subsequently, the robot either actively attempts to displace the obstructive robot to a less obstructed configuration for its exit (active contact response) or avoids contact with the obstructive robot to prevent further obstruction (passive contact response). This results in emergent behavior wherein robots collectively identify and adjust the position of a faulty robot causing obstructions.

In Chapter 5, deep reinforcement learning (DLR) is adopted to explore and validate complex emergent behaviors from elementary task descriptions. A stigmergic multi-agent deep reinforcement learning (S-MADRL) technique, grounded in the deep Q-Network (DQN) learning algorithm, is developed to discover cooperative policies for collective food retrieval in densely crowded environments. Decentralized policies leveraging partially observable state information are trained to effectively coordinate a group of homogeneous agents to avoid congestion and enhance the food retrieval rate. The stigmergic communication technique, employing virtual pheromones as part of the local observation of agents, fosters cooperation and the emergence of social behaviors. Additionally, a curriculum learning method is introduced to address unstable convergence in highly crowded scenarios, breaking down the learning problem into more manageable sub-problems. The results showcase the potential of stigmergy in addressing the non-stationarity and non-scalability issues in multi-agent learning to replicate and validate emergent behaviors observed in natural swarms.

Finally, in Chapter 6, insights from nature are revisited to deepen the understanding of how ants self-organize to excavate crowded, narrow tunnels without global control or explicit communication. The focus is to discover the principles guiding social insects' adaptation to changing environments solely through local interactions. A model is developed where contact interactions are used to explain the rates of multi-phase excavation in collective ant

digging. A multi-tunnel Cellular Automata (CA) model is adopted, suggesting that ants rely solely on physical interactions like collision frequency with other ants as a proxy to estimate and propagate information about their environment. Mathematical modeling and computer simulations align with results from biological experiments conducted in controlled laboratory environments. These experiments reveal three stages of excavation rates: a constant rate, rapid decay, and slower decay scaling inversely with the square root of time. The alignment between our model and biological experiments underscores the possibility of complex emergent or optimal group behaviors arising from individual decisions facilitated solely through local contact interactions, without explicit or global information.

A prospective progression beyond the algorithms and methodologies expounded in this thesis involves evaluating their scalability across larger number of agents. This evaluation is to be conducted within a model simulation environment meticulously designed to replicate the dynamics inherent in the physical interactions among robots within their environment. The inclusion of a larger number of agents in this analysis is anticipated to provide enhanced understanding regarding the scalability of the outlined techniques. Furthermore, such an exploration aims to elucidate similar strategies for harnessing unavoidable features intrinsic to dense active systems.

To summarize, this thesis outlined as follows:

- Chapter 2 presents a background and literature review of harnessing local interactions as coordination techniques in biological and robot swarms. It encompasses applications of stigmergy in multi-agent reinforcement learning scenarios and surveys fault tolerance techniques in multi-agent systems.

- Chapter 3 introduces the Adaptive protocol method for learning to prevent congestion in confined multi-robot excavation, presenting experimental results from real robot experiments conducted over extended durations.

- Chapter 4 details the Active Contact Response method, enhancing fault tolerance in

clog control algorithms discussed in Chapter 3. Results from real robot experiments are presented and compared against previous algorithms used as baselines.

- Chapter 5 shifts focus to deep reinforcement learning for multi-agent coordination in crowded and confined environments, discussing methodologies to enhance convergence through virtual pheromones or stigmergy and comparisons with established baseline algorithms.

- Chapter 6 presents recent findings on coordination and adaptation techniques in living collectives, wherein analytical and simulation models for ant digging experiments are developed.

- Chapter 7 concludes this thesis by highlighting primary contributions and suggesting potential future directions for research.

CHAPTER 2
BACKGROUND

In this chapter, we offer an overview of swarm robotics and conduct a comprehensive examination of some relevant prior research and literature. Specifically, we focus on the utilization of local, social, or physical interactions within robot swarms or multi-agent systems. This analysis encompasses their role in facilitating coordination, control mechanisms, reinforcement learning, and achieving fault tolerance within these systems. Additionally, we reference various biological or natural phenomena that have served as inspiration for the development of swarm robotics or multi-agent systems and algorithms.

2.1 Local interactions and coordination in swarm robotics

Robot swarms typically exhibit rudimentary sensing, computing, and manipulation capabilities, yet they demonstrate the ability to accomplish intricate tasks and behaviors that surpass the individual capacities of the robots [2]. Extensive research within the realm of swarm robotics, a subset of multi-agent systems, has yielded numerous algorithms and techniques that leverage interactions among robots and between robots and their environment to evoke complex emergent behaviors in collective robot systems. The practical applications of robot swarms and/or multi-agent systems encompass a broad spectrum, including collective exploration tasks such as collective foraging [3], target search operations [21], disaster victim rescue missions [22], and mine detection efforts [23]. Furthermore, they extend to spatial organization tasks such as aggregation [24], self-assembly [25], and pattern formation [26], as well as decision-making processes such as consensus building [27], collective task allocation [28], and collective fault detection [29].

The field of swarm robotics finds significant inspiration from the collective behaviors exhibited by social insects, owing to the shared attributes of resilience in uncertain conditions,

scalability, decentralized coordination, adaptability, and homogeneity among members [30, 31, 32]. This inspiration arises from the recognition that collective behaviors in both domains emerge through self-organization, wherein basic interactions among entities result in intricate and coordinated group actions [31, 33, 34]. Researchers have thus pursued elucidating the fundamental principles governing the emergence of complex collective behaviors via self-organization, emphasizing the significance of local interactions and the acquisition of localized information within diverse swarm robotics or multi-agent systems, significantly influenced by biological experiments and phenomena.

Achieving collective emergent behaviors in swarm systems critically hinges on local, physical and social interactions. For instance, Aguilar et al. [35] illustrated the pivotal role played by contact interactions or independent encounter rates in the observed retreating behavior of *Solenopsis invicta* fire ant species, thereby mitigating flow-stopping clogs in narrow, harsh and challenging environments. When replicated within robot collectives through contact interactions, this behavior notably amplified traffic flow, subsequently enhancing group efficiency in congested and confined settings. Similarly, Liu et al. [36] employed local cues such as food retrieval success rates and inter-agent collisions to devise an adaptive mechanism facilitating efficient task allocation among a swarm of foraging robots, replicating behaviors observed in *Pachycondyla apicalis* ant species. Furthermore, Mayya et al. [37, 38] proposed leveraging inter-robot collisions as a sensing modality for robot localization within their environment. This involved the development of a collision model for estimating robot density across different environmental regions, coupled with probabilistic filtering techniques, thereby contributing to the effective localization of robotic entities.

The robotics research presented in this thesis primarily centered on utilizing contacts or collisions as a mechanism to reinforce self-learning of cooperative behaviors, thereby enhancing group performance, even in the face of evolving work conditions. The robots' primary task involved continuous retrieval of food items within crowded and confined

environment, characterized by frequent and inevitable collisions and jamming or clogging. Operating within dynamic environments characterized by uncertainty, equipped with noisy sensors and perception capabilities, our robots function autonomously without an internal map of the surroundings. Importantly, these robots function independently of external information or communication regarding neighboring states or the quantity of robots present. This distinctive characteristic renders our technique well-suited for adaptation to dynamic environments or scenarios with limited domain knowledge, obviating the need to address model inaccuracies or undertake frequent calibrations. It eliminates dependence on an explicit environmental model.

2.2 Fault tolerance in multi-agent systems

Over the past few decades, researchers have proposed several techniques to enhance fault tolerance in multi-agent systems. An individual's failure can result from either software, physical components or a combination of both [39]. Software failures can be readily detected, isolated, and recovered due to the flexibility of provisioning and cloning software systems to achieve redundancy.

To improve fault tolerance in multi-agent systems, researchers have proposed redundancy schemes such as "replicating" individual agents and implementing high-availability hybrid architectures [40], [41], [39]. Fedoruk et al. [42], along with other researchers, [43], [44], [45], [46] have explored various approaches to implementing replication to improve fault tolerance in multi-agent systems. Replicated agents usually maintain copies of their states and data across different systems. This method is applicable to software systems, where faulty agents can be readily detected and substituted by replicas with minimal interference. However, in physical systems where flexibility and mechanical compliance are unavailable, the challenge extends beyond maintaining copy of agent states and substituting with replicas in the event of failure.

Detecting faults or failures is accomplished through inter-agent communication via

"keep alive" or periodic message exchanges [47], [48]. Communication is essential part of detecting and maintaining fault tolerance in multi-agent systems. However, it requires the integration of information exchange channels among numerous agents, which increases system complexity and load factors, potentially leading to decreased system performance.

Researchers have also explored fault tolerance in several multi-robot systems application domains. Park et al. [49], [50] proposed a decentralized control law to address the multi-robot rendezvous problem, consisting of nonconforming or faulty robots. The technique is based on distributed control policy that is robust to undetected faults in the system. Al Hage et al. [51] developed a fault-detection method for the collaborative localization task of multi-robot. The approach relies on the Informational form of the Kalman filter to detect and exclude faulty sensors from the team.

Yang et al. [52] outlined several approaches for fault tolerance in cooperative control of multi-robot systems. In multi-agent systems, increased redundancy and flexibility bring about heightened complexity and challenges. Hence, the authors introduced four categories of fault-tolerant cooperative control methodologies: individual, cooperative, topology reconfiguration-based, and composition reconfiguration-based.

Our technique can be categorized as a passive fault-tolerance scheme, which does not rely on explicit fault detection and identification. We investigated collective fault-tolerance behavior in a partially-observable and noisy multi-robot environment.

In chapter 4, we presented the ACR method, individual robots make decisions based on their egocentric contact history map of the environment, providing scalability to our method. Furthermore, the decentralized nature of the ACR method enables the collective to accomplish the task without requiring the activity knowledge of other participating robots. This presents a distinct advantage over numerous established swarm robotics methods, which require shared information between agents to complete a collective task [53, 54, 55]. We demonstrated in the lab the effectiveness of this approach, exhibiting a more robust method with potential real-world applications.

2.3 Deep reinforcement learning for multi-agent systems

Recent success of deep reinforcement learning (DRL) in solving real world tasks has made it a popular choice to solve complex problem using high-dimension information representation with trial and error method [56]. DRL holds the promise of discovering complex behaviors from simple task descriptions. Researchers have demonstrated that DRL can learn effective control strategies in a wide range of domains, such as computer games [57], autonomous driving [58], and robot manipulation [59]. However, applying DRL methods to robot swarms or multi-agent systems (MAS) imposes additional challenges that prevent effective learning and stable convergence. First, an individual does not have access to the global information of the environment and can only observe the local information (partial observability). Second, each agent modifies its behavior in response to other agents' learned behaviors, which makes the convergence of each behavior unstable (non-stationarity). Third, state and action spaces grow exponentially with increasing number of agents (non-scalability), which gives the problem of "curse of dimensionality" [60].

Several works have focused on resolving these challenges to achieve convergence and scalable optimal solutions. The extension of DRL to multi-agent systems (MAS) or swarm robotics has seen a lot of attraction in the recent years, due to its novel collaborative and distributed solutions to complex problems that would otherwise be difficult to achieve via alternative methods [61]. Some of the techniques used include: introducing explicit communications channels, training with joint action and state information, and modeling the behavior of other agents [60], [62], [63]. While explicit communications in multi-agent systems usually incur huge computational costs that does not scale well with number of agents, some recent works have focused on developing implicit or shared memory among agents [60], [64].

Researchers have investigated *stigmergy* as a viable and systematic approach to decentralized communication utilizing local stimuli [20]. Stigmergy, in this context, leverages traces

or signals left by individual agents' actions to direct the swarm towards specific emergent behaviors [18]. This communication methodology involves utilizing the environment as a shared external temporal memory, wherein agents store pertinent information or signals. Consequently, agents can deduce the actions of their counterparts or approximate the history or statistics of the state of the environment [65]. The adoption of stigmergy in various works related to multi-agent coordination and control is attributable to its scalability, robustness, low computational overhead, and adaptability [19]. Numerous endeavors in this domain have employed stigmergic mechanisms to address complex or NP-hard problems effectively and on a scalable basis. Examples include implicit multi-agent spatial clustering without direct inter-agent communication [66], and the coordination of unmanned vehicles [67].

In Chapter 5, we formulated a framework for stigmergic multi-agent deep reinforcement learning tailored to the collective excavation scenario. Numerous works in the field of deep reinforcement learning, akin to ours, have concentrated on harnessing the inherent benefits of stigmergy to enhance learning in multi-agent environments [68, 69, 70, 71, 72]. Notably pertinent to our research is the Stigmergic Independent RL framework developed by Xu et al. [68]. In their framework, they established an indirect communication bridge among learning agents, employing an explicit feedback loop and a stigmergy medium. The medium is represented as a digital pheromone map, enabling agents to communicate and observe the environmental state. In our work, we similarly incorporate the concept of digital pheromones. However, instead of employing an explicit feedback loop with direct information propagation, we utilize a virtual map containing localized information regarding different regions of the environment. Additionally, while Xu et al. applied their technique to a UAV flight formation scenario without providing an analysis of scalability, we implement our approach in addressing a collective coordination problem within a highly restricted and congested environment. Furthermore, we compare the scalability of our method with other established multi-agent deep reinforcement learning methods.

Likewise, Zhang et al. [69] devised the Pheromone Collaborative DQN framework to

investigate minefield navigation tasks. Unlike our digital pheromone model, their approach incorporates a network structure comprising nodes that emit pheromone information to entice nearby agents. The likelihood of an agent selecting an ”attractor” is determined via a random-proportional equation, reliant on the attractor’s signal strength. Furthermore, the authors did not offer an analysis of their method’s scalability concerning larger agent populations, in contrast to other prevalent MADRL techniques. In contrast, our model employs a virtual map structure, superimposed onto the environmental area, enabling the storage of localized agent activities in relevant locations. These activities or information are subsequently accessible as partial observations for other agents, aiding in reasoning about both the environment’s state and the actions of neighboring agents. Furthermore, while their method introduced four new components to the Multi-Agent Reinforcement Learning (MARL) problem, our approach introduced only one component — the virtual pheromone map — effectively coordinating the actions of independently learning agents. Consequently, our method imposes less complexity on model formulation and problem-solving overhead.

2.4 Learning via local interactions in natural swarms

The process of individual adaption to internal and external stimuli can be considered as a “learning” behavior [73, 74]. Organisms live in changing complex environments and must make a series of decisions to survive and reproduce [75, 76]. Learning represents a particularly useful mechanism for succeeding in environments. Learning can be broadly defined as a lasting change in behavior that results from observation or experience, and is useful in environments where the repeatability of an event is relatively high. In this case, an organism may learn to repeat an action which will lead to a consistent, beneficial result.

Social insects learn or adapt to changing conditions as a result of social interactions among workers and their environment, surprisingly without any global assessment or centralized control. For example, Aguilar et al [35] and Monaenkova et al [77] have shown that fire ants vary their tendency to participate in digging based on encounter rate during

collective nest construction. Similarly, Pinter-Wollman et al [78] and Prabhakar et al [79] suggested that harvester ant colonies regulate foraging activity through antennal contacts between outgoing and returning foragers. These local interactions, including deposition and detection of chemical pheromones [80, 81], serve as cues or sources of information to alter their individual behaviors, which increases the overall colony efficiency, and thus, ultimately the ecological success of social insects [82, 81].

Workers' intrinsic inter-individual variability as well as mechanism of self-reinforcement via experience or repeated response to local cues are fundamental to the principle of self-organization [83, 84, 85]. Self-organization generally involves many individuals that work collectively to create complex patterns through positive and negative feedback interactions. These patterns are not encoded on the individual level nor are they being built with any knowledge of the global structure. They are instead products of interactions between individuals or between individuals and their local environment [85, 32, 30].

CHAPTER 3

LEARNING TO AVOID CLOGGING IN MULTI-ROBOT EXCAVATION

Clustering is highly ubiquitous in task-oriented and confined multi-robot scenarios, due to the persistent motion of the individual robots. Passage through a narrow tunnel aligns the motion of the individuals and exacerbates the occurrence of such clusters. Narrow tunnels are quasi-one-dimensional systems. Unlike a particle in a three- or two- dimensional gas, an excavating robot will meet every other individuals already in the tunnel. Therefore, two-body collisions are extremely frequent, and 3+ body collisions may occur as well [77] [35]. The resolution of these traffic jams may require a significant amount of time, compared to the time robots spend traveling and digging, and may significantly reduce the excavation rate. As a tunnel grows, most of the collisions in the system will be two-body collisions, but still, any individual coming in will meet every other one already inside. It may be therefore beneficial, for the collective performance, to have the individuals learn how to modulate their work desire to avoid collisions and clogging.

In our previous work [35], we used robots as a robophysical model of the ant tunneling system which facilitated testing of behavioral rules in a controlled environment with noise and complexities of the real physical world. Using the robophysical model we demonstrated how an active confined crowded robot collective could mitigate structure formation (slowly dissolving clogs and jams) via being "lazy" and "giving up". That is, manipulations of workload distribution in this robophysical model collective rationalized our observations of biological ants' strategy unequal workload distributions and probabilistic yielding to oncoming traffic (termed "reversals") demonstrating the importance and utilty of such rules for maintaining optimal tunnel flow. However, we had to program the behaviors in the robots; *here we are interested to learn how robots can adjust their behaviors to optimize their workload distributions and retreat behaviors thus providing principles for*

decentralized coordination of robot swarms that must operate in crowded, confined, and dynamic conditions.

3.1 Methods

Our robots, modified from our previous study [35] are programmed to execute autonomous behaviors independently such as navigation to specific sections in the tunnel and excavation. They are equipped with force sensitive grippers for pellet excavation, an outer shell with capacitive sensing to detect and distinguish two types collisions - robot-robot collisions and robot-wall collisions, as well as terminal rods for charging and detecting the home area (Figure 3.1). The pellets to be excavated are laid at the end of the tunnel are a cohesive granular medium consisting of plastic shells housing loose rare-earth magnets. Since we are particularly interested in how coordinated group success could be achieved only from physical interactions and local observation of the environment, we do not allow direct robot-to-robot communication or global information to the individuals in the group; our robots rely purely on on-board sensors and make decision based on local sensing and self-reinforcement.

3.1.1 Collective Task

Our task is a collective excavation scenario: a group of robots must continuously excavate the model granular media in a narrow (1.5 body lengths) and confined tunnel, shown in Figure 3.1. As more pellets are excavated, the tunnel "grows" or changes geometrically as the robots perform their task. A robot starts by leaving the Home area and, using vision, following the guiding trails to the digging area where the cohesive pellets are located. During transit in the tunnel, the robot can detect and distinguish collisions with other robots, as well as collisions with the wall of the tunnel. By sensing a magnetic field, the robots can also detect the pellets. After a successful attempt to excavate, a robot heads home to drop the excavated pellets into a Deposit bin placed on a weighing scale.

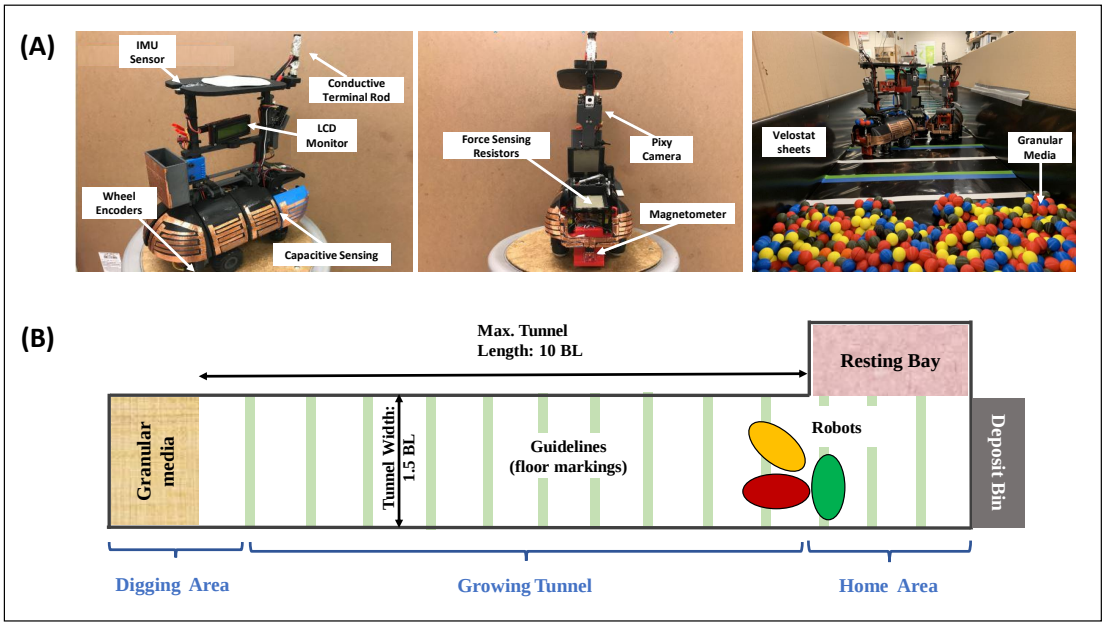

Figure 3.1: Experimental apparatus consisting of robots and a tunnel. **(A)** Components of an ant robot (two left images): Inertia Measurement Unit (IMU) sensor for absolute orientation; Wheel encoders for localization via robot odometry/dead reckoning; Capacitive touch sensor shells for detecting collisions; Force sensitive resistors for gripper tactile feedback, Magnetometer for detecting pellets and Digging Area; Pixy camera for tunnel navigation and localization; Terminal rod for charging and detection of Home area. In the tunnel (rightmost image): Velostat sheets for lining the tunnel to distinguish robot-robot collision (copper-copper) from robot-wall collision (copper-velostat); Granular media to simulate model cohesive soil excavated by ants. **(B)** Schematic diagram of the confined tunnel: a robot starts from the Home Area, decides whether to enter the tunnel and excavate some pellets, or decides to go to the Resting Bay to avoid interference at the tunnel entrance.

Our goal is for the group to excavate as many pellets as possible within a given time. An obvious solution is for the robots to remain constantly active and try to excavate; however as we demonstrated in [35] when all the robots are in such a mode and enter the tunnel concurrently they spend much of their time resolving collisions as a result of competition for space to maneuver and carryout their activities. The resulting traffic jams have robots stuck or stalled due to excessive stress from repeated collisions. This wasted time results in degraded performance, to the extent that fewer robots in the tunnel would excavate faster. The challenge is therefore to use the local information available to individual robots to

regulate the congestion and improve group performance under such physical constraints and hindrances. We derive our inspiration from the social behavior of fire ants under crowded and confined conditions [35], and our previous robophysical-model excavation experiment [86], to develop an adaptive learning rule that makes the robots decide when to "give up" digging and when to "take a rest" in a way that significantly improve the performance of the group.

3.1.2 Robot Controller

We adopt a finite state automaton model [87] which is a common scheme used to control behavior-based robot activities with no global knowledge. A state transition is triggered when a robot senses some physical clues from the environment. Each sensor on the robots has a specific trigger state that enables the robot to transition into another state. Figure 2 shows the model of individual robot's controller. Each block contains a set of states or sub-states that form a mode or behavior that the robot exhibits. The states and sub-states are as described below:

1. Goto Dig: This state is triggered at the start of each trip when a robot "decides" whether to dig based on their tunnel entrance probabilities $P_e(k)$.

2. Digging: This state is triggered when the robot is in proximity to the granular media. The magnetometer at the base of the robot detects the magnetic field of the granular media and prompts the robot to start the excavation routine.

3. Exit Tunnel: The robot enters this state from Digging when the Force Sensitive Resistor (FSR) detects sufficient amount of pellets in the gripper. The robot executes several turning maneuvers to exit the digging area and head back home.

4. Goto Dump: This is the state that captures the robot heading home after a successful pellet retrieval (Successful Trip) or unsuccessful pellet retrieval (Unsuccessful Trip).

The controller drives the robot out of the tunnel and get the robot home to the deposit area.

5. Dumping: Robot releases the excavated pellets from its gripper and dumps it in the "Deposit Bin" which is placed on a weighing scale to measure the amount of pellets excavated over time.

6. Exit Home: Robot executes some turning maneuvers to exit the deposit area and enters the tunnel to dig.

7. Collision: This state is triggered when a robot collides with another robot or with the tunnel wall. The robot executes a set of turning maneuvers in an attempt to resolve the collision.

8. Resting: Robot goes to this state at the beginning of a trip if the entrance probability is sampled and the robot decides to rest. The robot follows the guiding trail on the tunnel floor to navigate to the resting area to take a rest and not participate in the tunnel traffic.

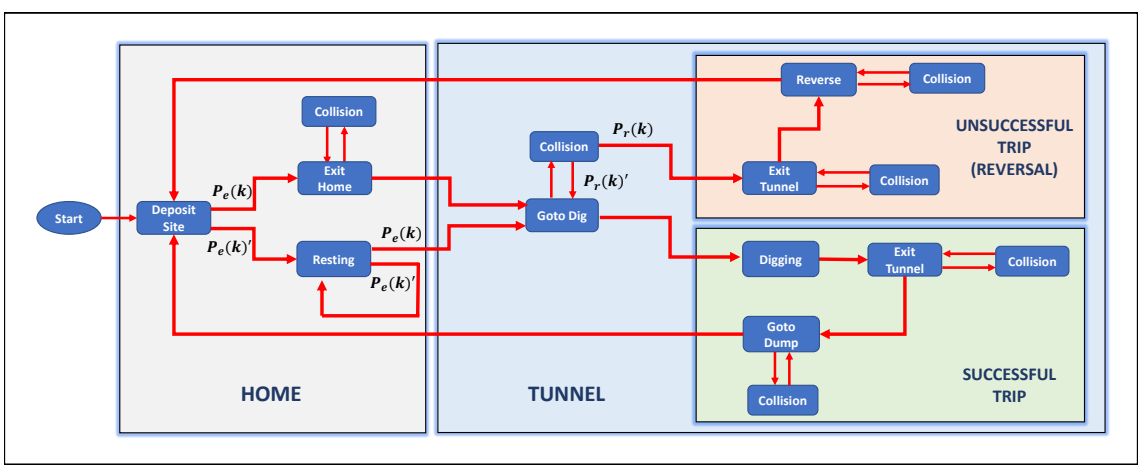

Figure 3.2: Block diagram for the individual robot controller. State transitions are event-based. P_e: probability of tunnel entry (or go to dig). $P_e^{'}$: probability of resting (or staying at home). P_r: probability of "giving up" after colliding with another robot in the tunnel. $P_r^{'}$: probability of continuing to dig after colliding with anther robot.

We developed a stochastic model with two parameters to control the entrance rate and reversal rate (give-up rate) of the robots so as to regulate tunnel traffic and improve group performance. Let $P_e(k)$ be the tunnel entrance probability and $P_r(k)$ be the reversal probability of each robot at trip attempt number k. A trip begins when a robot samples from the entrance probability, $P_e(k)$ and decides whether to "go in and dig" or "stay at home and rest". This parameter controls the number of robots in the tunnel which directly controls the tunnel density or congestion rate. The reversal probability, $P_r(k)$, on the other hand controls how a robot responds to a collision when it occurs. A robot samples from this parameter and decides if it should "give up" or to continue its journey. With these two parameters, we set up two basic protocols for studying the effects and performance of the social behaviors [35] in multi-robot collective excavation:

Active Protocol: In this protocol, we fix the tunnel entrance probability $P_e(k)$ to a value of one for each trip for each robot. The reversal probability $P_r(k)$ is set to zero, so the robots do not return home until they are able to collect pellets. This ensures that all the robots are active, trying to dig in the tunnel at all time.

Reversal Protocol: Here we set the reversal probability $P_r(k)$ for each robot to a value greater than zero but less than one, while still keeping the entrance probability to one at all time. This allows the robots to randomly "give up" trying to dig when they collide with other robots in the tunnel.

3.1.3 Adaptive Protocol

To go beyond the above fixed behavior protocols and to gain insight into useful social interactions of the confined multi-robot system, we conducted a parameter sweep to find the optimal reversal probabilities that yielded the highest excavation rate in a Cellular Automata (CA) model developed in our previous work [35]; see supplemental section for detailed description of the model. Figure 3A shows the range of excavation rates for varying reversal

probability as the tunnel length increases. A closer look at the region with highest excavation rates suggests a non-linear inverse relationship between optimal reversal probabilities and tunnel length. In particular, the optimal probability values drop sharply for short tunnels (less than 5BL) and more gradually for longer tunnels. This gives us inspiration to develop an **Adaptive Reversal** probability function, $P_r(\Delta L)$, that decreases sub-linearly as the tunnel length grows (Figure 3.3B and Equation (1) below).

Additionally, ant excavation studies and CA model analysis reveal that an asymmetric or unequal workload distribution improves excavation performance in confined conditions [35, 77]. To attempt to incorporate this principle/strategy in our robophysical system, we develop a self-reinforcement protocol where we model the digging desire of individual robots with a probability P_e, called the "tunnel entrance probability", and update its value based on if a digging attempt performed by a robot was successful (increase P_e by a constant), or unsuccessful (decrease P_e by a constant). Preliminary results showed that this protocol consistently produced an unequal workload distribution with better excavation performance than the Reversal protocol in long tunnels ($\sim$10BL). However, this was not the case for short tunnels. Unequal workload strategy did not perform better when the tunnel was short ($<$ 4BL). To account for this phenomenon, we crafted an adaptive "update value" function, ΔP_e, (Figure 3.3C and Equation (2) below, which has a small update value when the tunnel is short. We call this the **Adaptive Protocol**. That is, our adaptive protocol modifies the digging desire of individual robots by using egocentric estimates of the change in tunnel length to update individuals' entrance probability values.

Intuitively, the Adaptive protocol (via equation 1 and equation 2) suggest that the cost of "giving up" due to collisions (high density) at longer tunnels is substantial, and the strategy to minimize congestion is by deploying fewer workers to dig, or equivalently, more workers to rest. That is, long-duration clogging is more likely to occur in longer tunnels than in short tunnels due to the cascaded effects of multi-body collisions propagated as the robots or ants flow into the tunnel simultaneously.

Our adaptive protocol addresses this issue by having the robots modify their entrance probabilities slowly first at the initial stage of digging, then more rapidly at the later stage (equation 2). ξ is a parameter added to ensure that a resting robot does not remain in resting mode indefinitely (i.e. that P_e does not go to zero) allowing robots to explore their environment, update their estimates of the change in tunnel length, and modify their behaviors if necessary.

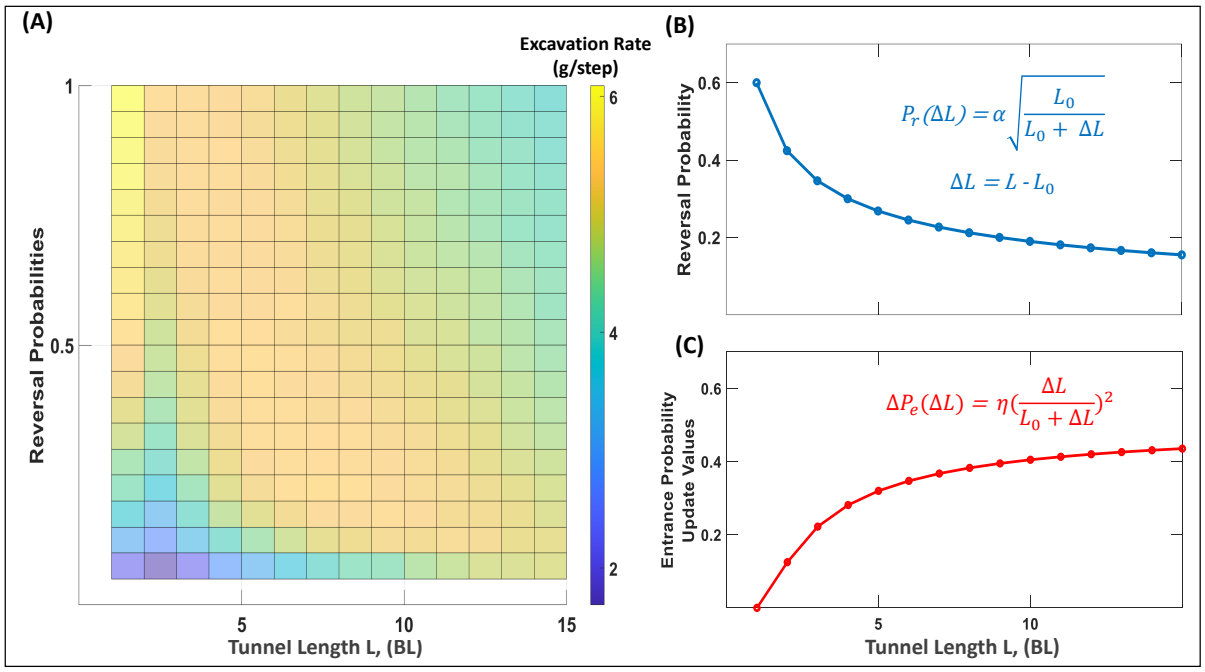

Figure 3.3: Probability sweep over tunnel length with selected equation models. **(A)** Reversal probability sweep vs Tunnel length. Plot shows that high to medium reversal probabilities give the highest excavation rate for short tunnels, while low reversal probabilities give the highest excavation rates for long tunnels. **(B)** Selected equation for modelling Reversal probabilities as a function of tunnel length for multi-robot experiments. **(C)** Learning update value for Entrance probabilities as a function of tunnel length for multi-robot experiments.

$$P_r\left(\Delta L\right) = \alpha\sqrt{\frac{L_0}{L_0 + \Delta L}} \tag{3.1}$$

$$P_e\left(k, \Delta L\right) = \begin{cases} P_e\left(k-1\right) + \eta \Delta P_e(\Delta L) & \text{if successful trip} \\ P_e\left(k-1\right) - \eta \Delta P_e(\Delta L) & \text{if unsuccessful trip} \\ P_e\left(k-1\right) + \xi & \text{otherwise (resting)} \end{cases} \tag{3.2}$$

where:

$\Delta P_e(\Delta L)$ = entrance probability update value

L_0 = initial tunnel length (in robot body lengths)

ΔL = change in tunnel length

ξ = noise or exploration term

α = normalizing constant for reversal probability

η = normalizing constant for tunnel entrance update

and $\Delta P_e(\Delta L) = \left(\frac{\Delta L}{L_0 + \Delta L}\right)^2$.

The power law expressions of equation (1) and (2) are simple forms that yield the desired behaviors, i.e. rapid change in the reversal and update values at short tunnels and slow/gradual change at long tunnels (figures 3B and C). Other power law exponents can also be used and will likely result in various degrees of performance gains, as long as the exponent is < 1 for the reversal probability and > 1 for entrance probability update. Equation (2) takes a positive sign if the robot is able to get pellets home (successful trip), otherwise it takes a negative sign (unsuccessful trip).

Each time a robot reaches the digging area and excavate pellets, it updates its estimate of the change in tunnel length, ΔL as follows:

$$L\left(k\right) = L\left(k-1\right) + \gamma\left(L\left(k\right)^{'} - L\left(k-1\right)\right) \tag{3.3}$$

$$\Delta L = L\left(k\right) - L_0 \tag{3.4}$$

where:

$L(k)'$ = new measurement of the tunnel length, derived from robot odometry (or dead-reckoning) using the wheel encoder readings.

$L(k)$ = estimate of the tunnel length, averaged over old and new measurements.

$L(k-1)$ = updated value of the tunnel length during the last successful trip

γ = the weighting parameter or learning rate.

A new measurement of the tunnel length, $L(k)'$, is computed when a robot successfully reaches the digging site and excavates pellets. At this time, the tunnel length is derived from the x-component of the robot's location as computed by the robot odometry [88]. The robots use the kinematic model of a differential drive mobile robot based on wheel encoder counts to estimate their absolute displacements in the tunnel. The derivation is provided in the supplemental section. Equation (3) above is an exponential moving average formula that acts as a filter for the estimate of the tunnel length which is used to compute P_r and P_e. It has an important application of reducing noise in a robot's estimate of the tunnel length which might occur when the robots are in multiple collisions. We chose our value of γ to be 0.9 which results in good performance for our experiments. Each robot maintains a separate copy of the equations and updates $P_e\,(k)$, $P_r\,(k)$ and ΔL asynchronously according to algorithm 1 described below.

The complexity of algorithm 1 is proportional to the total number of states, S, that the robot visits during an excavation trip. This is denoted as O(S) using the big-O notation. The best-case scenario occurs when the robot does not encounter any collisions but travels from the home area to the digging site and back with pellets. This is likely to occur when there are a few robots in the tunnel and the time to complete a trip (i.e. one pass of the algorithm) is relatively short. On the other hand, the worst-case scenario occurs when

Algorithm 1: Adaptive Learning Rule Pseudocode

```
Initialize: k=1, T_r = 60, P_e(k) = 1.0 , P_r(k) = α;
Set experiment duration, T ;
while t < T do
    Sample p ~ U(0,1) ;
    if p < P_e(k) then
        Goto dig (Active Mode);
        if Contact with a robot then
            Sample q ~ U(0,1) ;
            if q < P_r(k) then
                Exit tunnel (Give up);
            else
                Continue going to dig;
            end
        else
            Continue going to dig;
        end
        if Robot grabs pellets then
            L(k) = L(k-1) + γ ( L(k)' - L(k-1) ) ;
            ΔL = L(k) - L_0 ;
            Exit tunnel;
        else
            Continue going to dig;
        end
        if Robot gets home with pellets then
            P_e(k) = P_e(k-1) + η ( ΔL / (L_0 + ΔL) )^2 ;
        else
            P_e(k) = P_e(k-1) - η ( ΔL / (L_0 + ΔL) )^2 ;
        end
        P_r(k) = α sqrt( L_0 / (L_0 + ΔL) ) ;
    else
        Rest for T_r seconds (Resting Mode);
        P_e(k) = P_e(k-1) + ξ;
    end
    k = k + 1;
end
```

the robot encounters and handles collisions in all the states, since collision handling is considered an "intermediate" state (see figure 2). In either case, the amortized run time

complexity of the algorithm is O(1) since the number of possible states is bounded and does not depend on any input. Similarly, the space complexity is O(1) since the memory space is fixed and does not grow or depend on any input.

3.2 Experiment

We implemented the Adaptive protocol on our physical robots to compare its performance with the Active and Reversal protocols. Unlike in our previous robophysical experiments [35] in which tunnel length and digging probabilities did not change during excavation, here we conducted experiments in which the tunnel increased in length as the robots excavated the granular media (pellets). This both better models growth of tunnels in biological collective excavation [35, 89, 77] and demonstrates how our learning scheme can adapt to dynamic and non-stationary environments. Figure 4 shows three snapshots of the robot experiment setup.

Because of limitations in the robot's excavation performance per trip, we conducted experiments in the following scheme: initially, the granular media was positioned at one body length in the tunnel $(L_0 = 1)$. To model a tunnel increasing in length, the pellets were moved backwards incrementally by one body-length each time the robots made a cumulative deposit increment of 300 grams (a camera positioned above a weighing scale recorded the weight of total pellets excavated by the robots). The robots can estimate the tunnel length with a calculation of distance traveled as reported by their wheel encoders (Figure 3.1), and update their reversal and entrance probabilities according to equation (1) and (2) respectively.

At the start of each trial and for each protocol, the entrance probability of the individual robot is set to 1. This ensures that all robots are active and will thus interact with the environment. For the Active and Reversal protocols, the entrance probability remains fixed throughout the duration of the experiment, while for the Adaptive protocol, the entrance probability changes approximately as the inverse square of the tunnel length (equation 2).

This update rule ensures that the robots become less active as the tunnel length increases. Hence, the workload should go from equal to unequal. The η parameter is chosen to ensure that unsuccessful robots decide to rest more often when the tunnel is long, so as to not hinder the performance of the robots that can reach the digging area. If a robot samples from the entrance probability and decides to rest, it navigates to the Resting area and rests for one minute. When the resting time is over, the robot samples from the entrance probability again to determine if it should continue to rest or to re-enter the tunnel to dig (Figure 4.2).

For the Reversal protocol, a fixed reversal probability of 0.8 was used for all the robots in all trials. Prior multi-robot experiments demonstrated that such a high reversal probability regulated congestion better than a lower value. For the Adaptive protocol, however, it is desired that the reversal probability drops rapidly for tunnels less than 5BL and saturates quickly for tunnels greater than 5BL. The value of the α parameter – which controls the maximum and minimum values of the reversal probability for the robot experiment– is set to 0.6 which is within the range of values suggested by the parameter sweep plot of Figure 3.3A.

3.3 Results

In our robot collectives, we implemented the three different protocols and ran three trials for each protocol. Each trial was conducted for 3 hours; this duration is set by the capacity on the power-pack of the robots and ensures sensors and actuators are running effectively. The results of these trials are summarized in figures (Figure 3.6 - Figure 3.9).

Figure 3.5 shows the estimated tunnel length, $L(k)$, reversal probabilities, P_r, and entrance probabilities, P_e, for individual robots based on Algorithm 1. Figure 3.5A is the plot of $L(k)$ vs time which the robots use to estimate the change in tunnel length, ΔL, according to equation (3) and (4). Since new estimates of tunnel length, $L(k)'$, are derived from the x-component of an individual robot's odometry, it is important to note that wheel slippage can occur when a robot is simultaneously turning and undergoing a collision. This

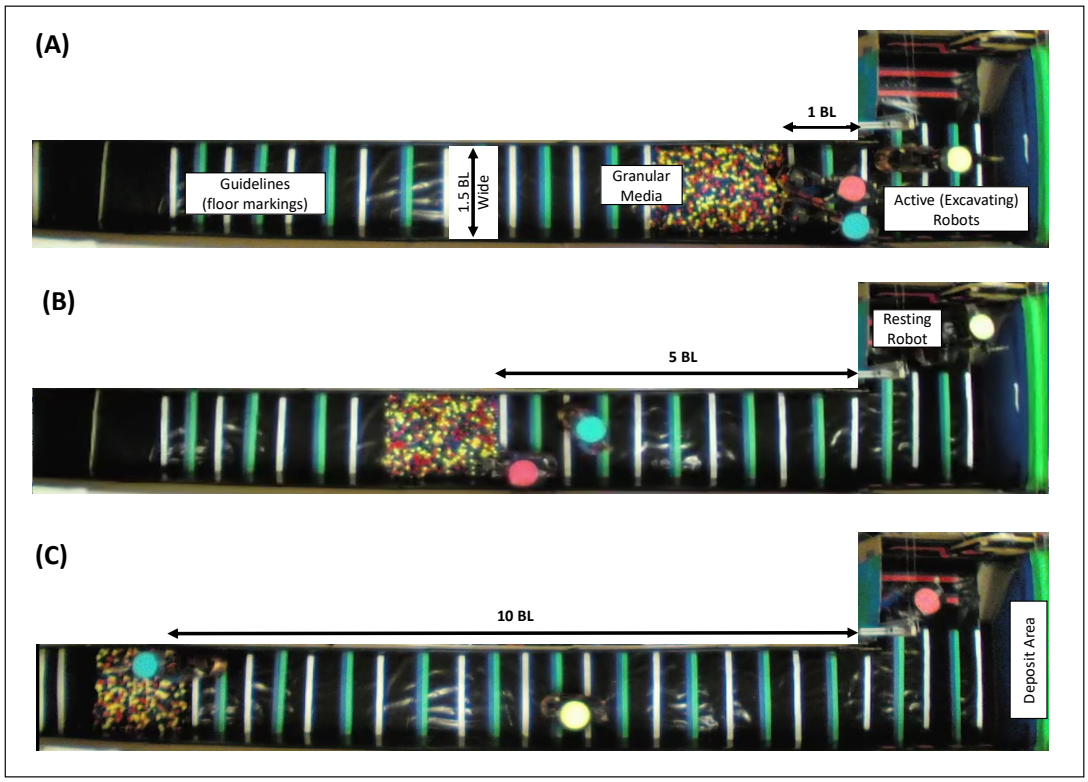

Figure 3.4: Three snapshots of the top view of multi-robot excavation experiments. **(A)** Pellets at one body-length (1BL) of the tunnel which is the initial position of the pellets at the beginning of each trial, **(B)** pellets at five body-length (5BL) of the tunnel after robots have excavated 1200g of pellets, **(C)** pellets at ten body-length (10BL) of the tunnel after robots have excavated 2700g of pellets.

will likely introduce noise in the estimate of the tunnel length, as shown by the fluctuations in the plots of Figure 3.5A. However, the moving average formula of equation (3) will ensure its effect is minimized. In addition, the propagation of noisy measurements is minimized by having the robots reset their odometry measurements at the beginning of each trip, i.e. just before a robot re-enters the tunnel. Figure 3.5B shows that the reversal probability tracks the desired power law expression of equation (1) and Figure 3.3B. Figure 3.5C illustrates how the P_e or "digging desires" of each robot changes from equal to unequal as a function of time, or equivalently, change in tunnel length.

Figure 3.6 shows a comparison of the cumulative amount of pellets deposited for the three protocols. Figure 6A illustrates that the individual trials with the Adaptive rule yield

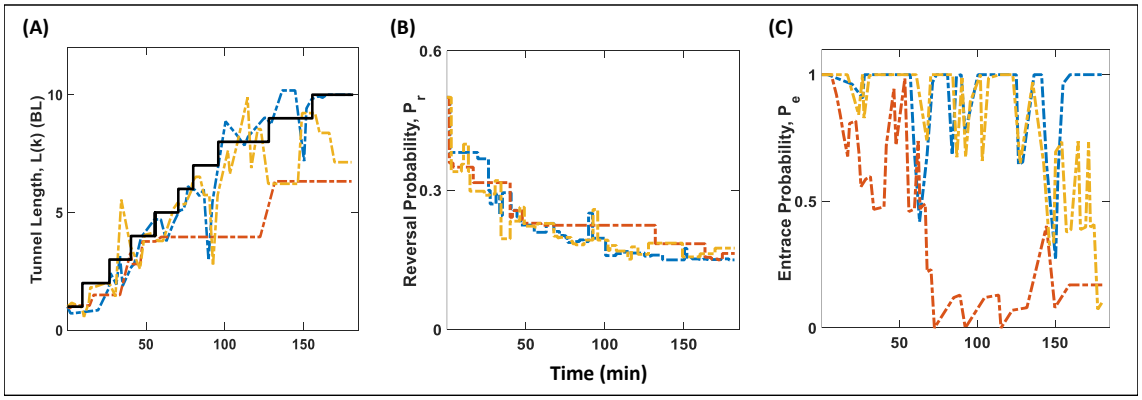

Figure 3.5: Estimated tunnel length, $L(k)$, Reversal probability, P_r, and Entrance probability, P_e over time for individual robots based on Adaptive rule. Dashed colored lines represent robot A (red), robot B (blue), and robot C (yellow). Solid black line represents actual tunnel length measurement. (A) Plot shows how individual robots track the actual growing tunnel, based on dead-reckoning method and exponential moving average formula described in equation 3. (B) Reversal probability values for individual robots as a function of time. These curves closely match the curve described in Figure 3.3B. (C) Plot of the entrance probability values over time. This shows how robots initially started off with equal digging "desires" and gradually become unequal "desires" as the tunnel length grows.

the highest number of pellet deposits for all trials. The graph shows that all protocols produce similar excavation rates at the initial stages of the experiment before they start to diverge as the tunnel length increases. This confirms that the all protocols and trials started with the same initial conditions, except for the reversal probability values in the case of Adaptive and Reversal protocol.

Figure 3.7 shows space-time plots of the robot trajectories for one of the three trials. The presence of robots in the tunnel is tracked from video captured by a camera positioned above the tunnel. At each time point, the presence of robots is summed over the width of the tunnel and is represented by a single row in the diagram. The adaptive rule produces the fastest tunnel growth, and the map includes some stationary blocks near the Home area which corresponds to resting robots.

Figure 3.8 explores the portion of time spent by the individual robots either outside of the tunnel (Figure 3.8A) – in the resting or deposition areas – or in contact with each other, while in the tunnel (Figure 3.8B), for the three protocols. The times are quantified based on the

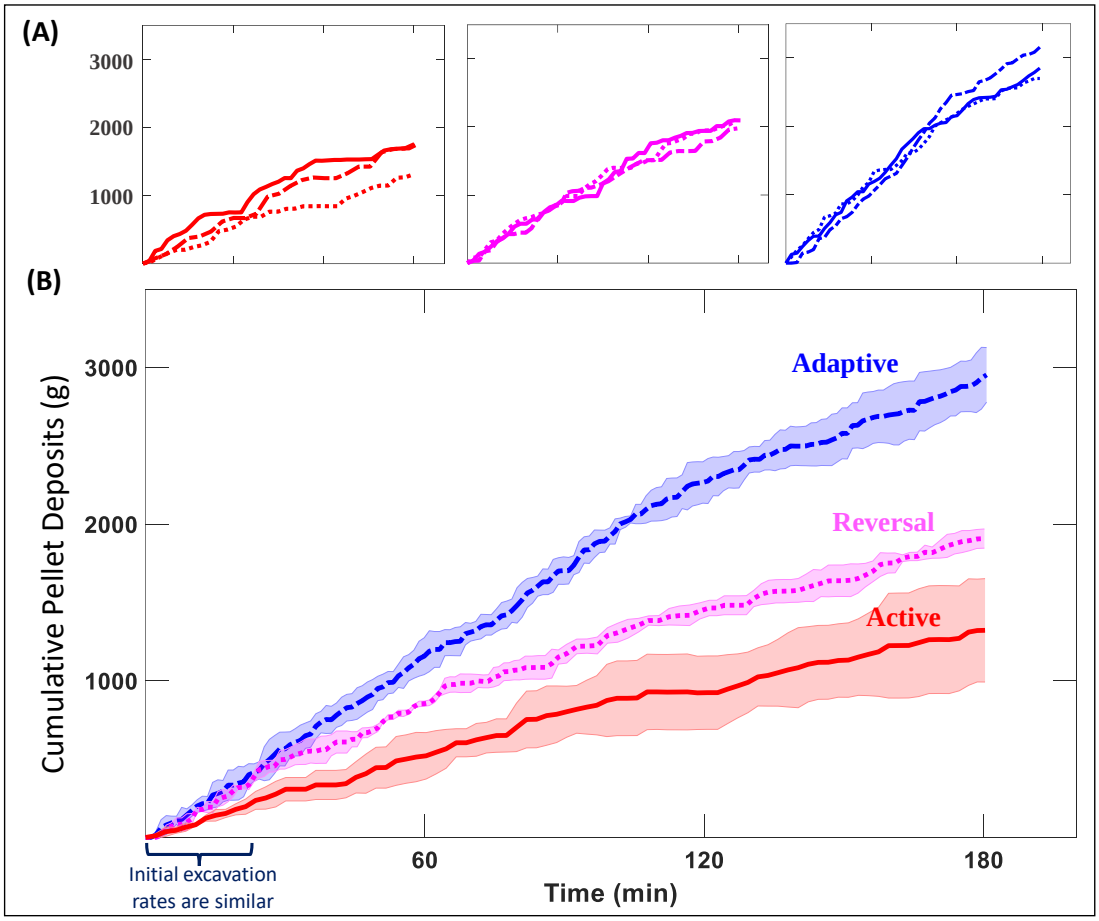

Figure 3.6: Excavated Pellets vs Time for three different protocols. *Active protocol:* all robots have an entrance probability of 1.0 but a reversal probability of 0; *Reversal protocol:* all robots have an entrance probability of 1.0 and reversal probability of 0.8; *Adaptive rule:* entrance probability is a function of inverse square of tunnel length, while reversal probability is a function of inverse square-root of tunnel length. **(A)**Individual trial comparison of excavation experiments **(B)** Mean excavated pellets, shaded areas correspond to standard deviation from three experiments.

robots horizontal position as tracked in the recorded experiments. In Figure 3.8B, robots are considered in contact with others when the horizontal coordinates are less than a body-length apart. We observe that the average time spent outside of the tunnel is roughly the same for the Active and Reversal protocols, at about 40%, but increases a bit for the Adaptive rule (Figure 3.8A). Notably, the Adaptive rule generates a wider variance, indicating some of the robots spend significantly less time in the tunnel than others. Looking at the time spent

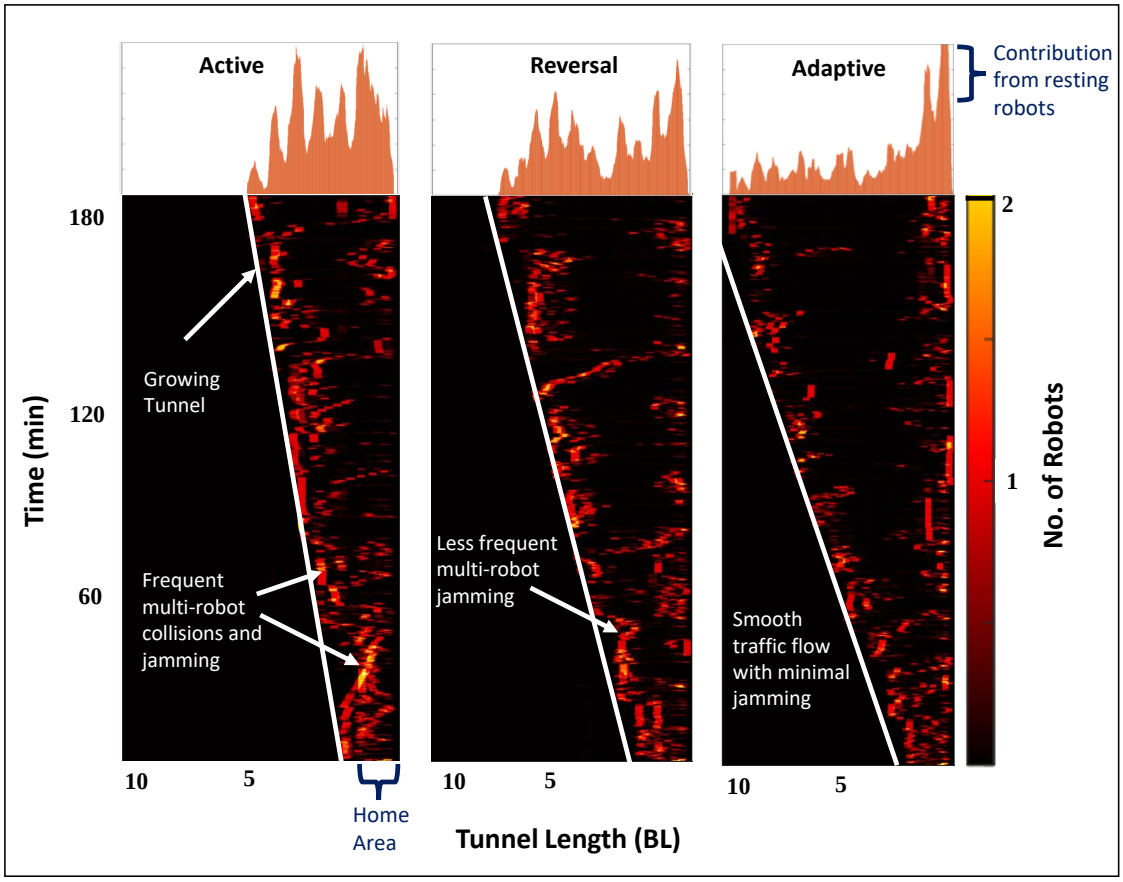

Figure 3.7: Experimental space-time overlap heat maps of robot positions along the tunnel (x-axis) measured in body-lengths (BL). Y-axis is the time duration of the experiment in minutes. White straight lines show how fast the tunnel grows which depends on the running protocol in the robots. Robots start from the *Home area* (right side) and transit to the *Digging area* (left side) continuously while excavating the pellets

in contact (Figure 3.8B), we see a narrower distribution for the Reversal than the Active protocol, demonstrating that the Reversal protocol regulates contact time in most cases. More importantly, there is a clear reduction in the average portion of time spent in contact, using the Adaptive rule, from more than 15% using the other protocols, to about 5%.

Figure 3.9 compares inequality in workload distribution for the Reversal and Adaptive rules, quantified using Lorenz curves. A Lorenz curve presents the cumulative fraction of work done by a cumulative fraction of the population. This curve is convex by definition and an equal workload distribution appears as a straight line between (0,0) and (1,1) A divergence

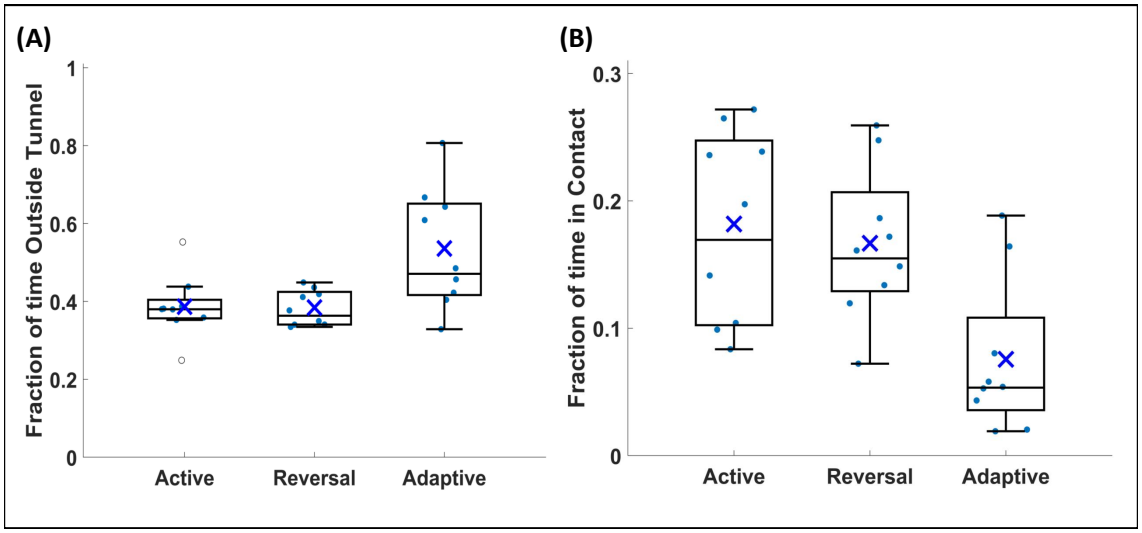

Figure 3.8: Time spent in various states by individual robots, for the three protocols: Active, Reversal and Adaptive. (A) Time spent at the resting and deposition areas. (B) Time spent in contact with other robots while in the tunnel. Boxes span 25th to 75th percentile, whiskers are between the minimal and maximal values, inner line is the median, $\times$ for means, and circles (solid and empty) represent actual data points.

from this straight line indicates unequal workload distribution, where, for example, half of the population is doing less than half of the work. This measure of divergence is usually quantified by the Gini coefficient, G, defined as the ratio of the area between the Lorenz curve and the line of equality [90]. The curves in figure Figure 3.9 show that the Reversal protocol produces an equal workload distribution with a Gini coefficient of approximately $\sim$0.06. The Adaptive protocol on the other hand produces a strategy that leads to equal workload distribution at short tunnels and unequal workload distribution at long tunnels with a Gini coefficient of $\sim$0.3. This strategy produces the most effective excavation rate in all the experiments. This is smaller than the inequality, in terms of a Gini coefficient, previously reported for ants, of about 0.6 [35], which is due to the larger number of individuals involved in the ant study. Since a tunnel essentially imposes a limit on the number of robots or ants that move through it concurrently without clogging, a larger number of individuals requires a higher degree of inequality to avoid clogging.

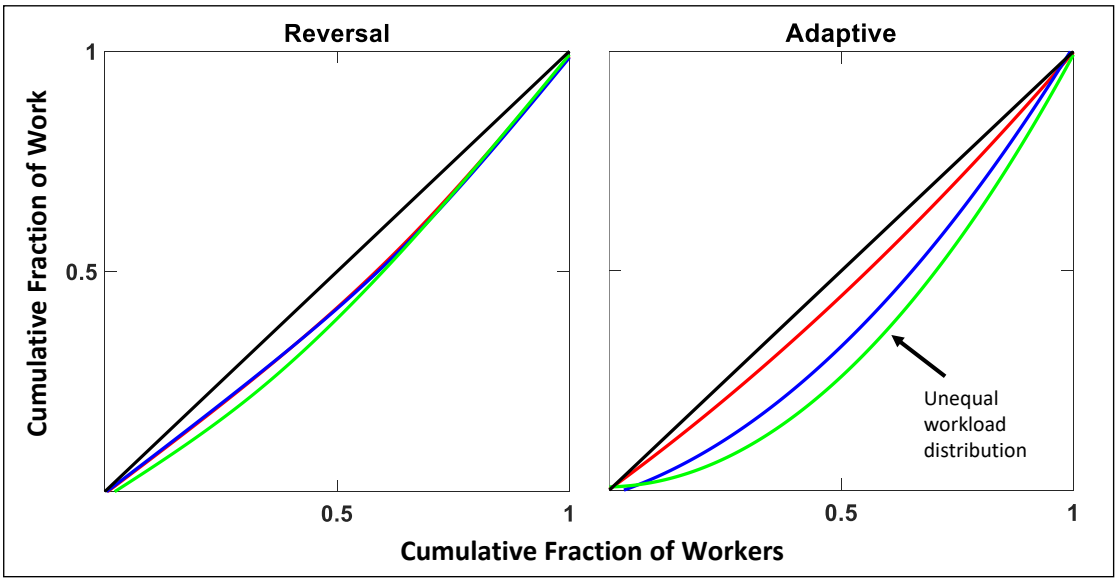

Figure 3.9: Lorenz curves of robot workload distribution for 1-hour "epochs" of 3-hour experiment. The curves represent: 1 hour duration (red), 2 hours duration (blue), and 3 hours duration (green). Black solid line represents the line of equality. Cumulative fraction of work is the number of trips the robots took into the tunnel in an attempt to excavate pellets. In Reversal protocol, all robots are mostly inside the tunnel, either going in to excavate pellets or going out to dump or "give-up" and re-enter the tunnel. Adaptive rule on the other hand leads some robots to rest and become less active over time.

3.4 Discussion

Our results demonstrate that an adaptive strategy, inspired by observations on ant behavior, leads to significant improvements in performance of excavation through a narrow tunnel, by a group of robots.

It was noticed previously that ants are sometimes willing to reverse or "give up", when faced with oncoming traffic [35]. When studied systematically, it was suggested that there is an optimal probabilistic rate for these reversals, which reduces multi-body collisions and jamming events [35]. Indeed, when we implemented probabilistic reversals upon collisions in the robots, we saw improved performance compared to an insistent, non-reversing ("active") behavior (Figure 3.6). Furthermore, our Cellular Automata simulations of the robots suggest that the optimal reversal rate decreases with increasing tunnel length

(Figure 3.3), which results from an increase in time wasted working without achieving pellet excavation. When a jam occurs far into the tunnel, a low reversal probability tells the robot not to give up quickly but rather try to resolve congestion locally.

Despite an improvement in performance (Figure 3.6), the willingness to ”give up” and reverse did not significantly reduce contacts between the robots in the system when compared with the ”active” protocol (Figure 3.8). Ants display another salient collective feature – an unequal workload distribution – which has been demonstrated to improve performance of collective digging in simulation, when compared to an equal workload distribution [35]. We hypothesized that a reinforcement rule employed by the individual robots, governing entrance probabilities P_e (Figure 4.2), could spontaneously result in an unequal workload distribution.

We implemented a reinforcement rule that increases (decreases) the probability to attempt digging with every successful (unsuccessful) digging trip. This reinforcement rule indeed results in the spontaneous formation of unequal participation in digging (Figure 3.9). Our preliminary experiments showed this unequal workload results in reduced performance for short tunnels and increased performance for long tunnels. Taken together with the trend we observed in simulations for optimal reversal rates (Figure 3.3), we decided to implement adaptive rules employed by individual robots, according to the tunnel length, estimated by distance travelled. As a result, an unequal workload distribution emerged that allows them to avoid costly contacts (Figure 3.8) and collectively perform better in an excavation task (Figure 3.6), using noisy estimates (Figure 3.5).

3.5 Conclusion

In this chapter, we presented a robotic collective – that performs a digging task through a narrow, quasi one-dimensional tunnel. As the performance cost of clusters increases with a longer tunnel, we show that a decision by some individuals to be lazy, improves

the performance of others and of the collective effort. The reinforcement rule enabling this adaptive laziness is ultimately driven by the collisions experienced by the individual robots. Thus, we achieved coordination for effective excavation performance by modifying individual robot's response to collision and task desires based on an independent estimate of the tunnel length. We discovered that "giving up", while sacrificing the individual performance, often contributes to the collective performance. That is, we demonstrated that a learning rule that modulates both "giving up" rate and "individual desires" gives a significantly higher group performance than with maladaptive behaviors (Figure 3.6). This technique could be applied to real world scenarios where collisions or physical interactions are unavoidable, or in decentralized task-oriented physical systems where individuals in a group must interact via contacts.

CHAPTER 4

MINIMALLY SENSING FAULT-TOLERANT MULTI-ROBOT COORDINATION

In the preceding section, we presented a methodology focused on harnessing local and physical interactions within densely active systems to mitigate challenges associated with the coordination of robots in densely populated and confined environments. The inherent physical constraints of such environments inevitably give rise collisions and jamming or congestion. We empirically observe that adaptive idleness engenders emergent coordinated flow, thereby enhancing the performance of robotic collectives. This improvement is achieved by introducing inequality in workload distribution and adapting reversal behaviors, as detailed in prior works [91, 92]. These collective behaviors manifest through contact interactions and adaptive learning mechanisms attuned to the intricate and dynamic characteristics of the environment.

The field of swarm robotics heavily draws inspiration from the principles of coordinated interaction seen in self-propelled entities that exhibit collective behaviors. [38], [54], [93], [53], [55]. For instance, social insects display selective engagement in densely populated environments lacking global cues. [89], [94], [95]. Considering the complexity of their environments, physical and localized interactions enable their robustness and fault tolerance, allowing for rapid recovery in the face of disasters [77], [96], [97]. However, unlike living systems, conventional robots typically lack mechanical compliance. Therefore, most robotic collective control strategies aim to avoid physical interaction and collisions [98], [99], [100], [101]. On the other hand, multi-robot or swarm control strategies necessitating a high level of coordination and control (e.g., [102]) are often impractical and can lead to catastrophic failures in environment where precision is crucial [103], [104].

Motivated by these challenges, we study the fault-tolerance in a group of autonomous robots with limited sensing operating in uncertain and unknown environment. We think

of our robots as models containing basic features relevant to more realistic system like warehouse or domestic mobile robots. Consider the swarm robotic scenario depicted in Figure Figure 4.1, where wheel driven ellipsoidal shape robots operate in a highly constrained and confined environment to perform their tasks with no centralized control or direct communication among the robots. These robots rely on local cues and simple onboard sensors, such as capacitive touch sensors and IMUs. This sets them apart from other systems that rely on overhead tracking or positioning systems for autonomous navigation and control. This aligns with certain examples of swarm robotics systems where robots must participate in local and social or contact interactions to collectively achieve their tasks [37]. In this environment, if a robot were to malfunction near a critical section of the work area, it could initiate cascading effects of difficult-to-resolve interference and congestion. This can drastically hinder task completion and may result in severed pathways that disrupts the workflow [92].

Previously, we developed principles that leverage local and physical interactions within dense active systems to address the coordination challenges of robots in crowded and confined environments [91], [92]. Physical constraints imposed by such an environment make collisions and jamming unavoidable. We observe how adaptive idleness produces emergent coordinated flow and enhances the performance of robotic collectives by generating workload inequality and adapting reversal behaviors [91], [92]. These behaviors collectively emerge through contact interactions and adaptive learning of the intricate structure and dynamics of the environment.

In this chapter, we build upon our previous work in [91] by incorporating a fault-tolerance technique to improve the group's ability to complete their tasks in the event of an individual's failure. In particular, we introduce an egocentric dynamic contact map that enables individual robots within the environment to localize and retain a history of their contact interactions. When a robot comes in contact with another robot, it uses the contact map history to reason and determine if it has encountered a stationary faulty robot.

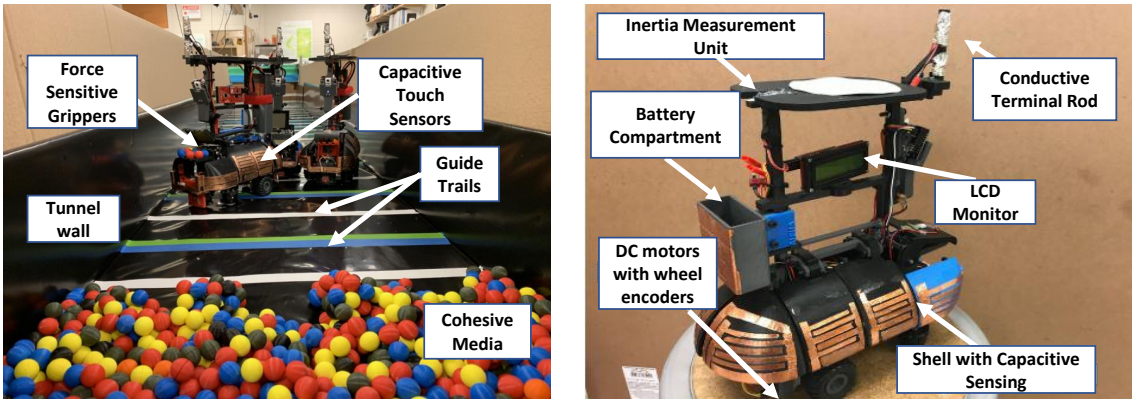

Figure 4.1: A model constrained and confined environment for multi-robot collective excavation of cohesive granular media. The three robots shown respond to environmental cues using onboard sensors. Each robot is 32 cm long.

The robot then responds by either attempting to actively displace the stationary robot to a less obstructed configuration, thereby facilitating its eventual exit from the tunnel (Active Contact Response), or by avoiding contact with the stationary robot to prevent it from being pushed further into the tunnel and causing additional obstruction (Passive Contact Response). This leads to the emergent behavior of robots implicitly deducing a faulty robot creating the obstruction, prompting the collective to nudge it to a less obstructive position.

4.1 Multi-robot Collective Excavation

We define the multi-robot collective excavation problem, similar to that described in [91], [35], consisting of a group of homogeneous robots tasked with the continuous retrieval of cohesive model granular media ("pellets," 3D printed hollow spheres with loose magents) in a narrow and confined tunnel (Figure Figure 4.1). The group's mission are to collect as many pellets as possible within a given time. The developed robots equipped with basic sensors (e.g., IMU, wheel encoders, capacitive sensor, force sensing resistor, magnetometer, and terminal rod capacitive sensor) and computational capabilities, with no global control or access to the state of other robots in the environment (Figure Figure 4.1).

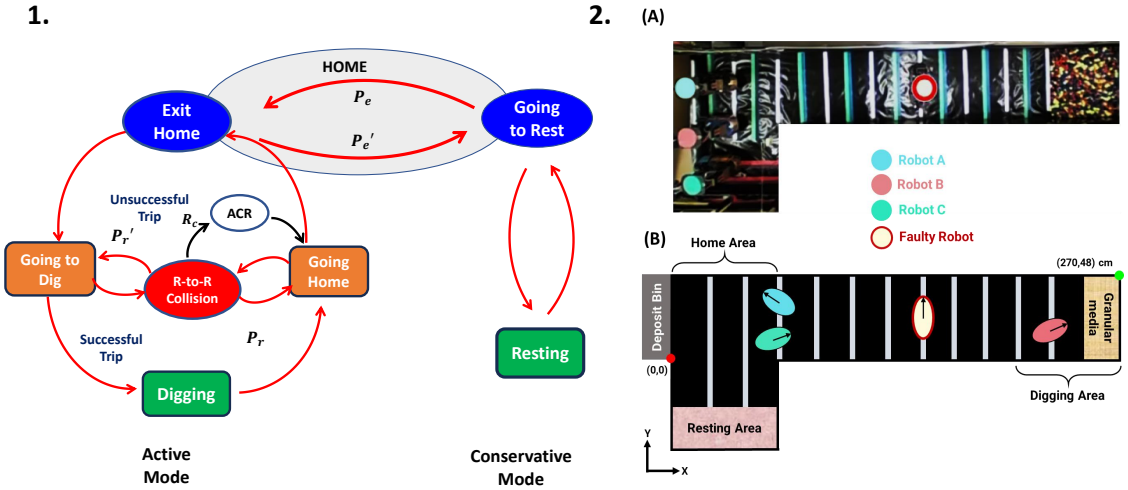

Figure 4.2: **1.** Block diagram for the individual robot controller. State transitions are event-based. P_e is the probability of switching to Active Mode (Going to Dig). P_e' is the probability of switching to Conservative Mode (Resting). P_r is the probability of "giving up" when a contact with another robot is detected. R_c is the probability that a detected contact is from a faulty robot.
2. Experimental apparatus consisting of three active robots (A, B, C) and a faulty robot perpendicularly positioned at the start of the experiment at the middle of tunnel. **(A)** Top view snapshot of multi-robot excavation experiment, showing three active robots starting out at the Home area, while the faulty robot is positioned at the middle of tunnel. Markers attached to the top of the robots are visible. **(B)** Schematic diagram of experimental setup showing three active robots (green, blue, and red), and one faulty robot (yellow). The adopted coordinate system's origin is as indicated in the diagram.

A robot starts a trip by leaving the home area by following guiding trails toward the cohesive pellets excavation location. The robot uses the magnetometer located at its anterior base to sense the magnetic field and locate the digging area. Subsequently, the robot runs a routine to excavate pellets and transport them to the home area. During transit in the tunnel, robots may detect and differentiate contacts or collisions with other robots or the tunnel walls. The robot then executes turning maneuvers until resolving the collision.

If robot-to-robot contact is detected while en route to the digging area, the robot "gives up" its digging attempt based on a predefined reversal probability value, P_r (Figure Figure 4.2). The same reversal behavior occurs if the robot cannot reach the digging area within a specified time.

4.1.1 Robot Controller

We adopt a behavior-based control model consisting of two layers of a Finite State Machine (FSM) [87]. The top layer uses a probabilistic FSM model, in which the robot decides which operation mode to deploy, determined by an entrance probability parameter, P_e. The robots choose whether to operate in Active Mode, meaning they enter the tunnel to participate in pellets excavation, or in Conservative Mode, where they remain in the home area and refrain from participating in excavation. When an individual robot encounters a collision with another robot while entering the tunnel to excavate pellets, it chooses to "gives up" and return home with a probability of P_r. At the bottom layer, the robot initiates each state transition upon sensing physical cues in the environment. Each of the sensors on the robot (Figure Figure 4.1) is configured with a trigger condition that enables it to transition into another state. Figure Figure 4.2 shows the model of the robot's controller. The major internal states of the robots are outlined below:

1. *Going to Dig*: The first state a robot transitions into when it is in Active Mode. This state is activated at the beginning of each trip, k, when the robot decides to enter the tunnel to excavate pellets based on the probability $P_e\left(k\right)$.

2. *Digging*: This state is transitioned into when a robot approaches the cohesive granular media. The magnetometer located at the robot's anterior base detects the magnetic field of the granular media, prompting the robot to begin excavation.

3. *Going Home*: This state captures the robot's return routine back home, either following a successful pellet retrieval (Successful Trip) or an unsuccessful pellet retrieval (Unsuccessful Trip). The controller guides the robot from the tunnel to the home or deposit area. The entrance probability for the next trip, $P_e(k+1)$, is updated based on the success of pellet retrieval (see algorithm algorithm 2).

4. *Resting*: A robot transitions into this state when in Conservative Mode. A robot will enter this state at the beginning of a trip based on resting probability $P_e'(k) = 1 - P_e(k)$. The robot remains idle in the home area for a fixed amount of time, T_r, refraining from participating in the excavation to avoid causing tunnel traffic.

5. *Collision*: This state is triggered when a robot makes contact with the tunnel wall or another robot detected via capacitive sensor. The robot can execute one of three behaviors to respond to the collision depending on the previous internal state it transitions from: 1) executes a turning maneuver to resolve the collision and return to the previous state, 2) retreat completely to avoid making further contact with the robot, and 3) execute active pushing to deliberately reposition the robot. Note that the robot can transition to the collision state from any state other than the Resting state. This is deliberately omitted in Figure Figure 4.2 to avoid visual clutter.

We developed a stochastic model with two parameters to control the entrance rate and reversal rate (give-up rate) of the robots. Let $P_e(k)$ be the tunnel entrance probability and $P_r(k)$ be the reversal probability of each robot at trip attempt number k. A trip begins when a robot samples from the entrance probability, $P_e(k)$ and decides whether to "go in and dig" or "stay at home and rest". This parameter controls the number of robots in the tunnel, which directly controls the tunnel density or congestion rate. The reversal probability, $P_r(k)$, on the other hand controls how a robot responds to a collision when it occurs. A robot samples from this parameter and decides if it should "give up" or to continue its journey.

4.2 Active Contact Response (ACR) Method

Here we introduce the Active Contact Response (ACR) algorithm, which maintains a space-time collision map of each robot's contact encounters in the tunnel. This enables robots to infer the probability of contact with a faulty robot and take the necessary actions to mitigate

persistent interference. Initially, each robot's contact map is initialized to zero, implying that they have no prior information about the distribution of active and passive (faulty) robots in the environment.

As collective excavation advances, each robot updates its contact map based on the type (e.g., robot or wall) of contact it encounters. Using the capacitive touch sensors (see Figure Figure 4.1), the robots can differentiate between contacts made with the environment walls and those with other robots. Utilizing this time-varying cumulative approach to represent interaction history enables the robots to maintain a belief state regarding the location of a faulty robot in the environment, should such a situation arise. Specifically, the robot updates: 1) the frequency of each contact type, c, and 2) the longitudinal position, l, where the contact occurred. The contact map undergoes conditional updates using the following equation (see algorithm algorithm 3 for implementation pseudo code):

$$\begin{aligned} M(c_r, l)_{t+1} &= M(c_r, l)_t + \omega_r * W \\ M(c_w, l)_{t+1} &= M(c_w, l)_t + \omega_w * W \end{aligned} \tag{4.1}$$

where:

c = observed contact type

C_r = robot-robot contact

C_w = robot-wall contact

l = estimated contact position

ω_r = conditional likelihood of detecting a robot-robot contact

ω_w = conditional likelihood of detecting a robot-wall contact

W = weighting factor

The conditional likelihood values, $\omega_{r/w}$, are obtained through the calibration of a contact sensor's sensitivity to robot and wall contacts. ω_r is the conditional likelihood of robot-robot contact, and ω_w is the conditional likelihood of robot-wall contact (see algorithm

algorithm 3). At fixed intervals, the number of each contact type is decreased by a constant, β, to reduce the effect of noise and the non-stationarity of the multi-robot environment: $M(c,l)_{t+1} = M(c,l)_t - \beta$.

Individual robots utilize these dynamically updated contact maps to make active contact response decisions based on the likelihood, R_c, that a robot-robot contact is originating from a faulty robot. The expression is as given below:

$$R_c = \frac{e^{\sum M(c_r,l)}}{e^{\sum M(c_r,l)} + e^{\sum M(c_w,l)}} \tag{4.2}$$

where:

$\sum$ = sum over the projected length of the robot

$M(c_r, l)$ = frequency of robot-robot contact at position l

$M(c_w, l)$ = frequency of robot-wall contact at position l

The rationale behind the expression above relies on the normal distribution of robot contact frequency over tunnel length, where the mean contact count is associated with the statistical location of the faulty robot. See algorithm listing algorithm 2 and algorithm 3 for the implementation pseudo code.

An active robot going into the tunnel to excavate pellets seeks to resolve collisions and improve group performance by voluntarily "giving up". However, a wrongly positioned inactive or faulty robot will impede group performance by causing clogging and jamming. As mentioned in the previous section, if a contact is detected, robot has three options, conditioned on the internal state: 1) execute the default maneuver to resolve contact - passive contact response, 2) execute reversal behavior to avoid tunnel clogging - reversal behavior, or 3) execute the ACR maneuver to displace the supposed faulty robot to a less obstructive configuration. For instance, when a robot has a high confidence ($R_c \approx 1$) that it has encountered a stationary faulty robot en route to the excavation site, it maneuvers around it by implementing either the reversal behavior or the passive contact response. However, upon the robot returning home from excavation, it will bias it behavior by continuously

pushing the stationary faulty robot, which results in robot being in less obstructive position (i.e., ACR maneuver). In passive contact method, a robot does not execute any biased behavior in response to a collision with another robot. Rather, it executes a default collision resolution maneuver to avoid repeated contact or obstruction with the other robot. A robot executing the ACR behavior can transition to passive mode if it is not making progress in repositioning a robot, indicated by the active robot's position and internal state remaining unchanged for some time, T_g. This adds robustness to the technique.

4.3 Experiment

We implemented the Active Contact Response (ACR) algorithm described in Section IV on physical robots and compared performance with our baseline algorithm. In the baseline algorithm, the robots neither employed a contact map nor actively responded to or biased their behavior when they detect collision with other robots. Upon contact with other robots, each robot will only seek to resolve contact or "give up" digging.

In our experimental setup, three active robots are tasked with excavating pellets, while one stationary robot, powered off, models a faulty robot during the excavation task (Figure Figure 4.2). The active robots are unaware of the presence of a "faulty" robot. Instead, they infer the likelihood of a stationary robot through the distribution of collision frequencies along the length of the tunnel. We set the tunnel width to accommodate either two robots moving in the same or opposing parallel directions or one robot performing turning maneuvers. At the start of each experiment, we placed the faulty (stationary) robot perpendicularly in the middle of the tunnel (Figure Figure 4.2). The perpendicular orientation of the faulty robot with respect to the length of the tunnel was selected to ensure sufficient contact interaction with the active robots. Positioning the faulty robot in the middle of the excavation path ensures it remains adequately distant from both the home/rest area and the excavation and deposition site.

It is important to note that the position of the faulty robot is critical in determining

Algorithm 2: Active Contact Response Method

```
Initialize: k = 1, P_e(k) = 1, P_r = 0.64, t_prev = 0, contact map, M(:,:) = 0;
Set experiment duration,T_e; entrance probability update rate,Δr; "giving up"
  time,T_g; resting time,T_r; contact map update rate,ΔC; likelihood of robot-robot
  contact, ω_r; likelihood of robot-wall contact, ω_w;
while t < T_e do
    if p ~ U(0,1) < P_e(k) then
        should_dig = 1;
        while should_dig == 1 do
            if ContactDetected then
                UpdateContactMap();
                if q ~ U(0,1) > P_r(k) then
                    should_dig = 0;   should_go_home = 1;   break ;
            if RobotGrabsPellets or (t > T_g) then
                should_dig = 0;   should_go_home = 1;   break ;
            if (t − t_prev) > ΔC then
                for c = 0; c < C; c++ ;
                    for l = 0; l < L; l++ ;
                        M(c,l) = M(c,l) − β;
        while should_go_home == 1 do
            if ContactDetected then
                UpdateContactMap();
            if RobotReturnsHomeWithPellets then
                P_e(k+1) = max(1.0, P_e(k) + Δr);
                should_go_home = 0;   break;
            else
                P_e(k+1) = max(0, P_e(k) − Δr);
                should_go_home = 0;   break;
            if (t − t_prev) > ΔC then
                for c = 0; c < C; c++ ;
                    for l = 0; l < L; l++ ;
                        M(c,l) = M(c,l) − β;
    else
        while (t − t_cur) < T_r do
            RestRobot();
    k = k + 1;
```

the group's overall performance. For example, the interference is maximum when the passive robot's orientation is perpendicular to the longitudinal direction of the tunnel, as

Algorithm 3: UpdateContactMap()

```
l = getRobotXCoordinate();
if robot-robot contact then
    M(c_r, l)_{t+1} = M(c_r, l)_t + ω_r * W;
    M(c_w, l)_{t+1} = M(c_w, l)_t + (1 − ω_r) * W;
else
    M(c_r, l)_{t+1} = M(c_r, l)_t + (1 − ω_w) * W;
    M(c_w, l)_{t+1} = M(c_w, l)_t + ω_w * W;
```

$$R_c = \frac{e^{\sum M(c_r,l)}}{e^{\sum M(c_r,l)} + e^{\sum M(c_w,l)}}$$

```
if State = GoToDig() then
    if q ~ U(0,1) > P_r then
        GoToHome();
    else
        PassiveContactResponse();
else
    if q ~ U(0,1) < R_c then
        ActiveContactResponse();
    else
        PassiveContactResponse();
```

well as when its position is close to the excavation area, where robots would need to execute turning maneuvers more frequently (Figure Figure 4.2). This faulty robot configuration further constrains the dimension of the excavation arena and impacts the group's excavation performance.

On the other hand, interference is minimal when the faulty robot's orientation is parallel to the longitudinal direction of the tunnel and its position is close to the home area, where there is enough space for the robots to make turns without much collision or interference. This configuration is termed less obtrusive because the robots can still complete their excavation and deposition tasks.

The ACR algorithm provides a technique to implicitly minimize potential interference caused by a faulty robot by indirectly displacing and rotating it into a less obstructive configuration. As described in the previous section, robots utilize the cumulative sum of contact experiences to estimate the likelihood of encountering a faulty robot. In the case

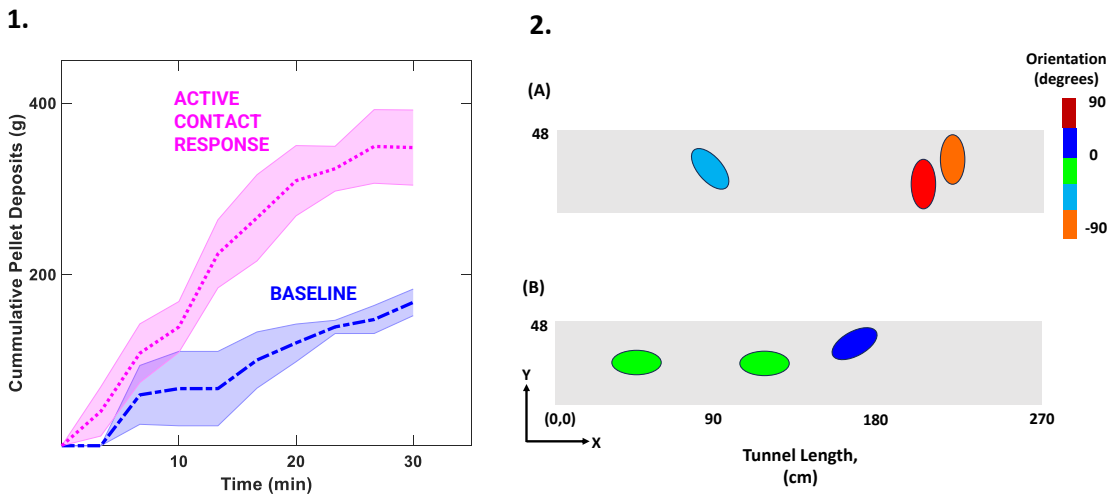

Figure 4.3: **1.** Comparing excavation performance between our baseline and Active Contact Response method. Three trials per tested method. Mean excavated pellets are the solid dotted and dash-dotted lines. Shaded areas correspond to the standard deviation from the three experiments.
2. Final configuration of the faulty robot after each experimental trial for **(A)** three trials of Baseline method, and **(B)** three trials of Active Contact Response method. The home area is located at a tunnel length of 0.

when there is no faulty robot in the tunnel, the collision map or collision frequency across the tunnel length would exhibit a normal distribution. This implicitly reduces the probability that a contact interaction stems from a faulty robot.

We fixed the tunnel length, and the robots have independent reactive controllers based on finite state machines (Section III). We set the experiment duration to 30 minutes and initially set the tunnel entrance probability, P_r, to 1. In response to the robot's successful trips, the tunnel entrance probability was gradually adjusted, mirroring the methods outlined in [91]. At the end of each experiment, we measure the configuration of the faulty robot to access the level of interference introduced. Three trials were performed per tested method.

4.4 Results and Discussion

Figure Figure 4.3 shows a comparison of the cumulative amount of pellets deposited when using the ACR and Baseline method (measured using scale deposition). The graph shows

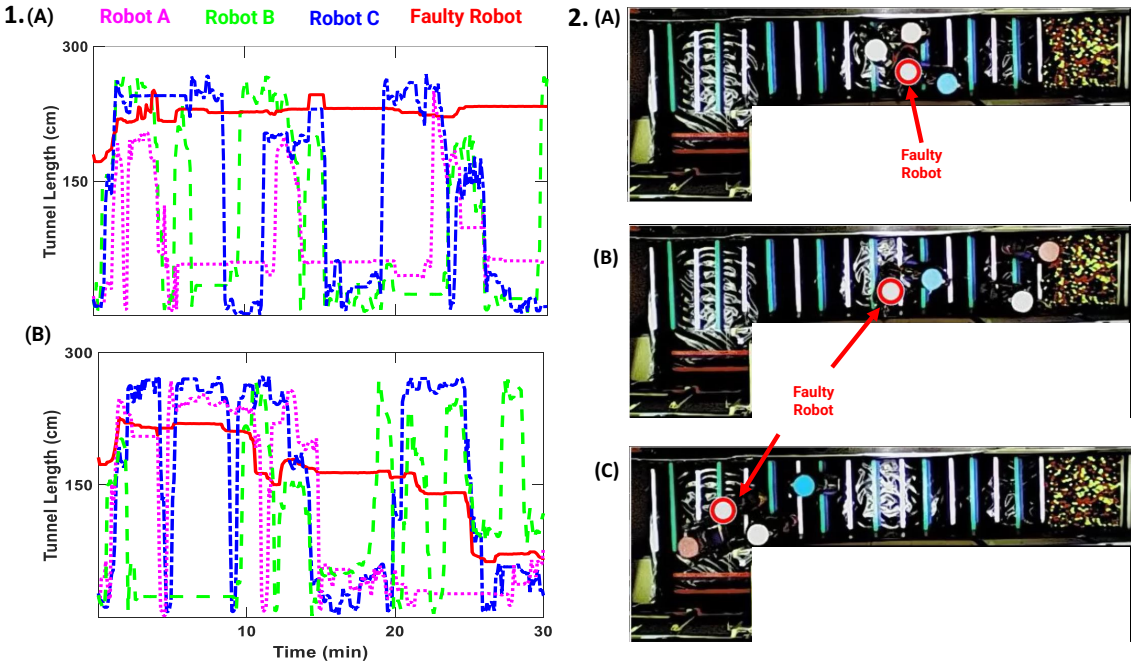

Figure 4.4: **1.** Active and faulty robot trajectories tracked overtime for **(A)** Baseline method, and **(B)** Active Contact Response method. Tunnel length position 0 cm is close to the home area, while tunnel length position 300 cm is close to the digging area.
2. Three snapshots of the top view of the multi-robot experiment using the Active Contact Response method. **(A)** Initial configuration of faulty robot obstructs traffic flow, resulting in congestion and clustering at the middle of the tunnel. **(B)** Subsequently, active robots update their collision map and maneuver around the faulty robot to reach the excavation area.**(C)** Finally, active robots effectively reposition faulty robot into a less obstructive configuration through active pushing and nudging.

comparable excavation rates for both methods during the first 10 minutes, but they start to diverge afterward. The baseline method maintains a consistent lower excavation rate for the remaining duration of the experiment. This is due to the effect of frequent collisions with the faulty robot, resulting in more robots giving up and resting, rather than repositioning the faulty robot to a less obstructive configuration. Whereas the ACR method effectively managed the presence of the faulty robot and improved its excavation rate. As a result, the ACR method excavated and deposited approximately twice as many pellets as the baseline method by the end of 30 minutes.

We recorded the final configuration (position and orientation) of the faulty robot at the

end of the experiment. Configurations with orientations close to zero (i.e., parallel to the tunnel) and positions near the Home area (i.e., out of the tunnel) are optimal since they introduce minimal obstruction within the tunnel. Figure Figure 4.3 shows the faulty robot's position and orientation (configuration) at the end of each experimental trial. In two out of the three trials, the ACR method successfully displaced and reoriented the faulty robot into a less obstructive configuration. In the baseline method, the final robot configurations for the three trials are more obstructive, which led to the lower excavation performance observed in Figure Figure 4.3.

Figure Figure 4.4 shows the trajectory of all the robots, one trial for each method. The baseline trajectories demonstrate a scenario where active robots pushed the faulty robot further into the tunnel toward the excavation site. However, when using the ACR method, the active robots pushed the faulty robot out of the tunnel towards home. As mentioned in the previous section, it is important to note that the orientation of the faulty robot also determines the level of obstruction.

In the ACR method, individual robots make decisions based on their contact history map of the environment, providing scalability to our method. Furthermore, the decentralized nature of the ACR method enables the collective to accomplish the task without requiring the activity history of neighboring robots. This presents a distinct advantage over numerous established swarm robotics methods, which rely on shared information between agents to complete a collective task [53], [54], [55]. In a laboratory setting, we demonstrated the effectiveness of this approach, leading to continuous operation and successful task performance of the group.

4.5 Conclusion

We have demonstrated how to exploit the concept of contact interactions to achieve emergent fault tolerance in constrained multi-robot systems, where precise sensing and control schemes are not available. Robots remain unaware of the internal and external state of other

robots in their environment. By relying on noisy contact measurements and localization, robots can estimate the conditional likelihood of colliding with a faulty robot and bias their behavior to collectively reposition the faulty robot to a configuration with minimal interference. The Active Contact Response algorithm enables the recovery of group performance in the event of individual robot breakdowns. In more complex scenarios involving higher number of robots, local information exchange such as individual collision map updates can improve performance [105]. Further, our technique can be applied to multi-agent collective tasks allocation scenario, or "smart active matter" systems where individual agents can leverage social, local, and physical interactions to improve fault tolerance and tackle more complex problems.

CHAPTER 5

DEEP REINFORCEMENT LEARNING FOR MULTI-AGENT COLLECTIVE EXCAVATION

In this chapter, our focus is on exploring the potential applications of indirect communication methods, such as stigmergy, to tackle the inherent issues of non-stationarity and non-scalability in a multi-agent reinforcement learning context. These challenges primarily arise from the concurrent actions by other agents and partial observability of the state of the environment [60]. We begin by introducing the deep reinforcement learning framework and the formalism of the Markov Decision Process. Following this, we define the problem of collective multi-agent excavation as a decentralized partially observable Markov decision process. We then present our contributions to this framework and elaborate on how we successfully addressed the significant challenges encountered during the training and learning processes.

5.1 Deep Reinforcement Learning

Deep reinforcement learning typically illustrates the problem using Markov's Decision Process (MDP) that is defined by the state space $\mathcal{S}$, action space $\mathcal{A}$, stochastic transition function $p(s'|s,a)$, reward function $r(s,a)$, and the distribution of initial states $s_0 \sim p_0(s)$, where s is the current state, s' is the next state, and a is the action. One popular approach to solve the given MDP is *Q-learning*. Q-Learning is a popular model-free off-policy algorithm for estimating the long-term expected return of executing an action from a given state. These estimated returns are known as *Q-values*. A higher Q-value indicates an action a is expected to yield better long-term results in a state s. Q-values are learned iteratively, it learns the value of the given action for the given state, $Q(s,a)$, via the temporal difference (TD) update

rule below:

$$Q(s,a) = Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

Once the Q value is estimated, the policy selects the action that maximizes the expected return:

$\pi(s) = \arg\max_a Q(s,a)$.

Deep Q-Network learns the Q function using deep neural networks parameterized by θ. Instead of updating individual Q-values, updates are now made to the parameters of the network by minimizing a differentiable loss function given by:

$$\mathcal{L}(\theta) = \mathbb{E}_{s,a,r,s'}[(y_i^{DQN} - Q(s,a;\theta_i))^2]$$

at each iteration i, where $y^{DQN} = r + \gamma \max_{a'} Q(s',a';\theta)$ is the target Q-value.

The major problems associated with approximating action-value functions with neural networks are instability and/or non-convergence which is due to the correlations present in the sequence of observations, violating the fundamental assumptions of MDP. As such, Mnih et al. [56] devised two techniques to address these problems: experience replay and target Q-network. Experience replay stores the state transition tuples (s,a,r,s') in a reply buffer and uniformly sample mini batches from the buffer to update the network. A target network's parameter θ' is updated periodically at a lower frequency than the online network parameters θ. This is to minimize divergence or non-convergence during training.

5.1.1 Multi-Agent Deep Reinforcement Learning

Multi-agent Deep Reinforcement Learning (MADRL) extends DRL techniques to multi-agent reinforcement learning (MARL) problems. In view of the fact that the state transition function now depends on the joint action of all agents, a naive approach of learning a joint action of all the agents using a single policy will face non-scalability issue due to the exponential state and action spaces. Instead, one can avoid this issue by providing

each agent with a partial knowledge of the actual state of the environment and individual or separate policy learners. This is usually formalized as Partially-Observable Markov Decision Process (POMDP). This technique, however, is prone the "moving target" or non-stationarity problem, which is as a result of each agent adapting their policy to the changing policies of other agents. This is usually mitigated by introducing team rewards or use of direct communication channels among the agents[106].

Independent DQN (iDQN) is a popular MADRL algorithm for learning collaborative behaviors in which each agent independently and simulataneously learns its own deep Q-network $Q(s, a; \theta)$. Since the environment is partially observable, IDQN is usually implemented by having each agent condition on its action-observation history.

MARL problem can be formalized as a *Markov game* or *Stochastic game* which is a generalized MDP to such settings. A Markov game is defined by the tuple $\langle$ $\mathcal{N}$,$\mathcal{S}$,$\boldsymbol{A}$,$\mathcal{P}$,$\boldsymbol{R}$,γ $\rangle$ [107], [108], where

- $\mathcal{N}$ = 1,...,n is the set of n $>$ 1 agents.
- $\mathcal{S}$ is a finite set of states s in which the environment can be.
- $\boldsymbol{A}$:= $\mathcal{A}_1$ x $\mathcal{A}_2$ x ... x $\mathcal{A}_n$ is the finite set of joint actions for all agents.
- $\mathcal{P}$: $\mathcal{S}$ x $\boldsymbol{A} \rightarrow \delta(\mathcal{S})$ is the state transition probability function to arrive at the next state $s' \in \mathcal{S}$ given the current state $s \in \mathcal{S}$ and a joint action $\boldsymbol{a} \in \boldsymbol{A}$
- $\boldsymbol{R}$: $\mathcal{S}$ x $\mathcal{A}$ x $\mathcal{S}$ $\rightarrow \mathbb{R}^N$ is the joint reward function, received by all agents when transitioning from s to s'
- $\gamma \in$ [0, 1] is the discount factor.

5.2 Multi-agent Clog Control

Research into the coordination of multi-agent systems has received a great deal of attention over the past few years. The difficulty increases exponentially with increasing number

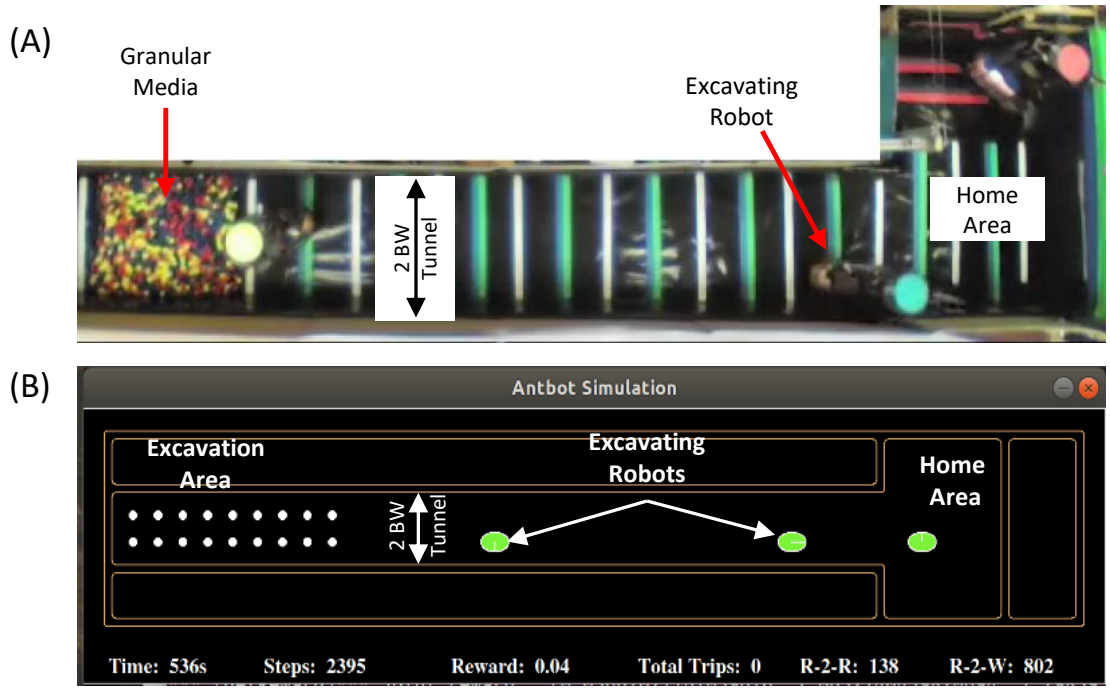

Figure 5.1: An overview of our environment showing the top-view of multi-robot excavation setup *(A)*, and similar features in the simulation model *(B)*.

of agents [109]. In this work, we consider a multi-agent collective excavation problem involving a group of homogeneous robots, tasked with continuous retrieval of food items (or pellets) in a narrow and confined 2D (grid world) environment. The mission of the group is to collect as many pellets as possible within a given time frame. Robots typically have limited sensing and communication capabilities, and without centralized control or global knowledge of the state of other agents in the environment. As a result, they spend most of their time resolving collisions due to excessive competition for space.

We formulate the described collective excavation problem as a decentralized POMDP (Dec-POMDP) [110], using independent DQN learners (IQL). We achieve coordination between agents using specialized social interactions via stigmergy. Then we improve on the more complex scenarios using curriculum learning which simplifies the learning task by training one agent at a time.

5.2.1 Decentralized POMDP

In this section, we will define the collective excavation task as a Dec-POMDP.

State and Observation. Let's consider N agents in the given environment (Figure Figure 5.1). At each time step, every agent receives a local observation which consist of an egocentric nonspatial discrete component and a geocentric spatial component (position, direction). The egocentric state represents the internal state of the agents which can be any of the following: *Going-to-dig*, *Digging*, *Exit-digging*, *Going-home*, *Dumping*, *Exit-home*, and *Collision*. Geocentric components consist of the following values: *current position*, *current direction*, *previous action*, *number of collisions*, *distance to home*, and *distance to food source*. This results in an 8 dimensional vector observation space for each agent, which is further augmented with the "pheromone" information of a fixed number neighboring cells corresponding to the agent's restricted field-of-view. As a result, each agent has a compact private observation space which makes our technique able to scale to larger number of agents without the need to adjust our network's input size.

Action Space. We adopted a discrete space representation consisting of a set of five actions. At each time step, the agents can choose one possible action within the set namely *North*, *South*, *West*, *East*, and *Stop*. If the target location is occupied, the agent receives a penalty and enters the (internal) collision state that keeps the agent in its current position until it chooses an action that puts it in an unoccupied position.

Reward Function. We carried out reward shaping to figure out a reward that makes the agents reach their goals fast and contribute to the team's mission at large. With that in mind, we devise a dense reward function to encourage smooth and effective learning, which consists of the following four signals:

- **Distance reward,** $r_d(s,t)$: An agent receives a reward of $+2.5$ if it gets closer to the goal (pellet location or home area) than it was in the previous time steps.

- **Collision reward,** $r_c(s,t)$: This is a reward of -2 if an unladen or unloaded agent

collides with another agent.

- **Pellet pickup reward,** $r_p(s,t)$: A reward of $+50$ is received if an agent is able to successfully find the food item location. In global reward settings, all agents receive a reward of $+50$. We distinguish the term "global reward" from "local reward" in that global reward assigns the same reward value to all agents without distinguishing their contributions. This is to encourage cooperative behavior among the agents, as opposed to local reward which assigns different reward to each agent based solely on its individual behavior, and often lead to competitive behavior.

- **Successful trip reward,** $r_s(s,t)$: A reward of $+50$ is received if an agent is able to drop a pellet at the home area. In the global reward settings, all agents receive the $+50$ reward to further incentivize cooperation.

The reward function is defined by the equation below:

$$r(s,t) = w_d r_d(s,t) + w_c r_c(s,t) + w_p r_p(s,t) + w_s r_s(s,t)$$

where w_d, w_c, w_p, and w_s are set to $0.2, 0.2, 0.2$, and 0.4 respectively for all the experiments.

5.2.2 Digital Pheromone as Stigmergy

Figure Figure 5.2 is an illustration of our proposed Stigmergic MADRL framework for scalable decentralized learning, by leveraging stigmergic communication. In particular, we model stigmergy in our experiment using a $virtual\ map$ of the environment, which serves as an indirect communication $medium$ that is asynchronously updated by the agents as they navigate the environment. A digital pheromone is a virtual map that stores the distribution of "activities" or "signals" over the entire space of the environment. The keys of the map represent distinct coordinates or positions in the environment, while the values represent

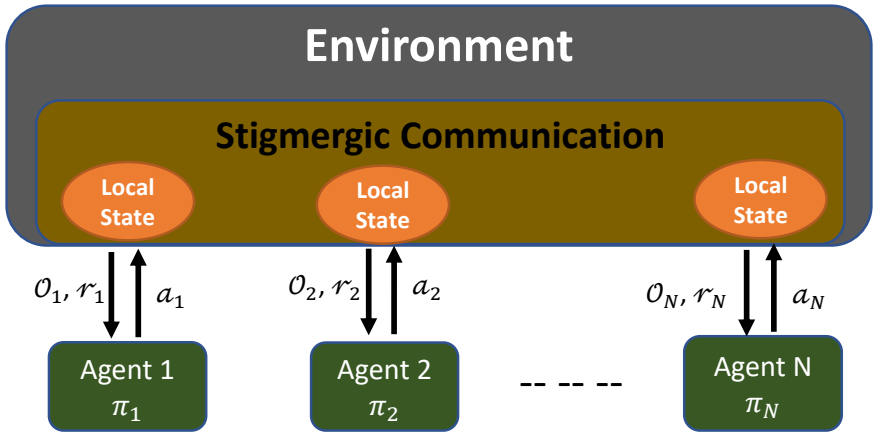

Figure 5.2: Proposed Scalable Decentralized MADRL Framework with Stigmergic Communication.

signals or traces of interests that encode specific state information about agents' activities. The digital pheromone is updated each time an agent moves from one cell to another in the grid world (see Figure Figure 5.3). The value of the new position is updated to specific state information of the agent, specifically the occupancy status, internal state of the agent, laden(1) or unladen(0) status, as well as its current action. The occupancy status starts with a value of 1.0 and decays with the coefficient of 0.25 every time step if it is unoccupied. This virtual map representation of the pheromone enables the agents to share or implicitly communicate different information about their internal states and the environment, similar to pheromone traces in social insects. Agents then read off the encoded information in their neighboring cells within its restricted field of view, and include them in its observation space during training and for decision making (Figure Figure 5.2 and Figure 5.3). This technique enable agents to use such information to reason about the actions and intentions of other agents, as well as the state of the environment.

5.2.3 Curriculum Learning

Curriculum learning (CL) is a technique for ordering experiences over samples to accelerate or to improve learning. In RL settings, it is usually implemented by generating a series of

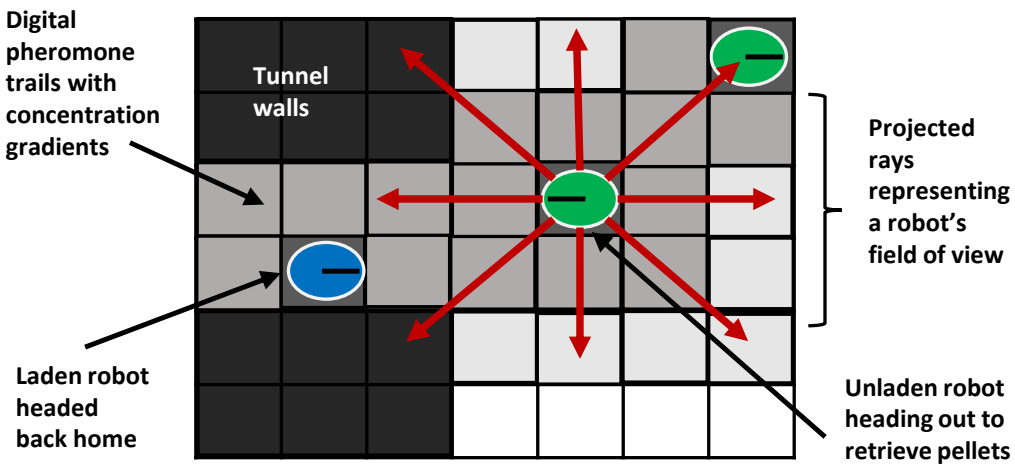

Figure 5.3: Schematic of digital pheromone and agent's restricted field of view. Agents' movements in the environment create virtual maps and gradients, depicted by the different shades of gray, that agents can observe within their surrounding. Red arrows illustrate the field of view of an agent. Blue agent is laden with food, while green agents represent searching agents.

training tasks, beginning from the simplest ones and then gradually increasing the difficulty of training to improve the final asymptotic performance or to decrease the training time [111]. In our multi-agent setting, CL could serve a specific purpose of reducing the effect of non-stationarity when there are many concurrent learners adapting their policies to another agent's changing policies. By limiting the number of concurrent learners to one agent at a time while increasing the number of agents sequentially, one can achieve a significant performance boost and convergence in such MADRL problem.

In that regard, we employ curriculum learning to improve learning convergence for more congested and complicated scenarios, where the number of agents is four or greater. Initially, we start with two agents and train them concurrently to learn decentralized policies for collective pellet retrieval. We call the two fully trained agents $old\ agents$. Then to train three agents and higher, we introduce one $new\ agent$ at a time and allow it to learn new policy from scratch while "freezing" the policies of the $old\ agents$ agents completely. The goal is to reduce the effects of non-stationarity by minimizing the "moving target" effect when there are several concurrent learners. In our experience, this invention allows us to

scale to larger number of agents ($five$), particularly when our CL method is combined with the stigmergic communication.

5.3 Experimental Setup

We demonstrate the effectiveness of our proposed method by conducting simulation experiments with the setup in Figure Figure 5.1. We implement the given MDP using the OpenAI's Gym interface [112], where its step function takes actions as inputs and returns next states, rewards, and terminal flags. Our environment can handle variable number of agents which we specify at the beginning of each experiment, as well as the length of the tunnel in dimension of grid cells or agents' body length (BL). The agents operate in a shared environment but use separate policy learners and individual experiences. Agents take actions simultaneously at each time step and receive private observations including a global or a local reward. Our network model uses Double Q-learning [113] and experience replay [57] to improve training stability. Our policy architecture has three hidden layers of 128 neurons each with a Boltzmann action-selection strategy. An ϵ-greedy strategy was employed to fully explore the state and action spaces, with an exploration rate annealed from 100% to 2% over 10% of the total training period.

We compare four different learning techniques with a varying number of agents, from one to five:

- ***Independent Q-Learning (IQL)***: This is our baseline method in which we use separate DQNs with local rewards to train agents independently.

- ***IQL with Global Reward (IQL+G)***: This method adds global rewards to the baseline method, to encourage cooperation among agents.

- ***IQL with Global Reward and Stigmergy (IQL+GS)***: This method adds virtual map (pheromone) discussed above to the second method, *(IQL+G)*, to mitigate non-staionarity and consequently facilitate cooperation among the agents.

- ***Stigmergy with Curriculum Learning (IQL+GSC)***: This method combines curriculum learning with third method, *(IQL+GS)*. The curriculum learning was implemented by introducing one agent at time during training to improve stability and convergence.

Table 1 below shows the list of parameters used for training the agents.

Table 5.1: Hyper-parameter Values

Parameter Names	Values
Number of Episodes	1000
Steps per episode	5000
Learning rate	0.0001
Discount Factor	0.99
Batch size	64
Optimizer	Adam

5.4 Simulation Results and Discussion

Training and testing were performed on a desktop computer, equipped with an AMD Ryzen 12-Cores processor clocked at 3.4GHz, 32Gb RAM and 1 Nvidia RTX 2070 GPU. Each experiment setup or scenario was run several times and learning curves were consistent. During testing, we measure the amount of pellets excavated by individual agents and log the cumulative amount of pellets for all agents at each time step. A comparison of total pellets excavated for all learning methods is shown in Figure Figure 5.5.

Given the latency of the simulation, it takes about twelve hours to train a single agent from scratch for fully converged policy on a GPU. Training time scales linearly with number of concurrent agents. However, curriculum learning is much faster and takes less time to train a new agent in a team consisting of two to four old agents. For most experiments, convergence starts after training proceeded for about 200 episodes at 5000 time steps per episode, after which the environment resets to the initial conditions. The learning rate we used is 10^{-4}, with a discount factor of 0.99. We found that a batch size of 64 works well with the "Adam" optimizer we employed to train the agents.

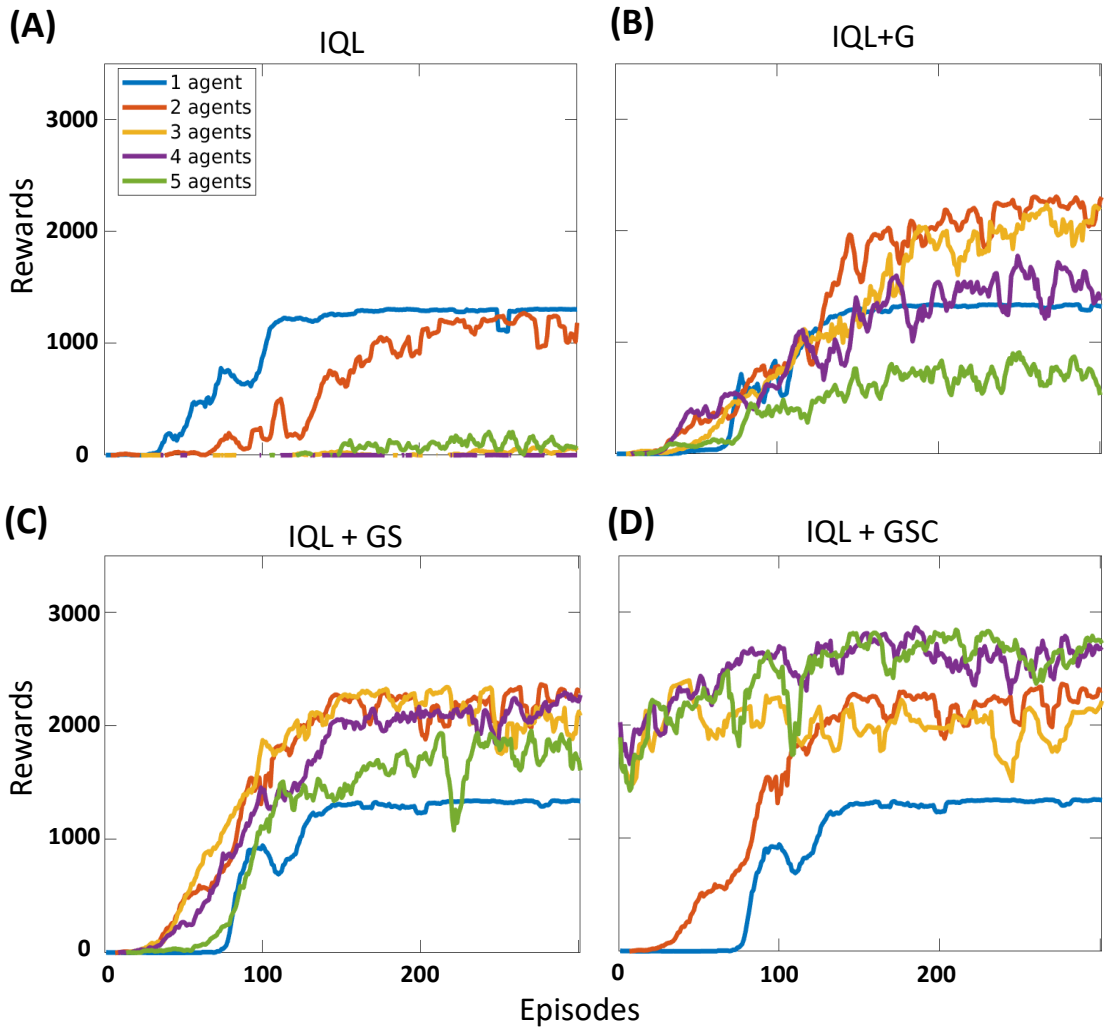

Figure 5.4: Learning curve comparison for the four different techniques. Our approach, IQL + GSC, achieved the highest rewards for four and five agent scenarios.

Figure Figure 5.4 shows the learning curve for the different MADRL techniques described above, for one to five agents in a shared environment. Clearly, stigmergy helps address convergence issues, since the baseline methods (IQL and IQL+G) could not handle three agents and higher. Curriculum learning improves learning convergence for $four$ and $five$ agent scenarios. Also, learning did not start from zero with the CL method, since the $old\ agents$ were using their learned policies.

Figure Figure 5.5 shows a comparison of total pellets excavated for each technique. For one and two agent scenarios, the baseline algorithm (IQL) performed well with both local

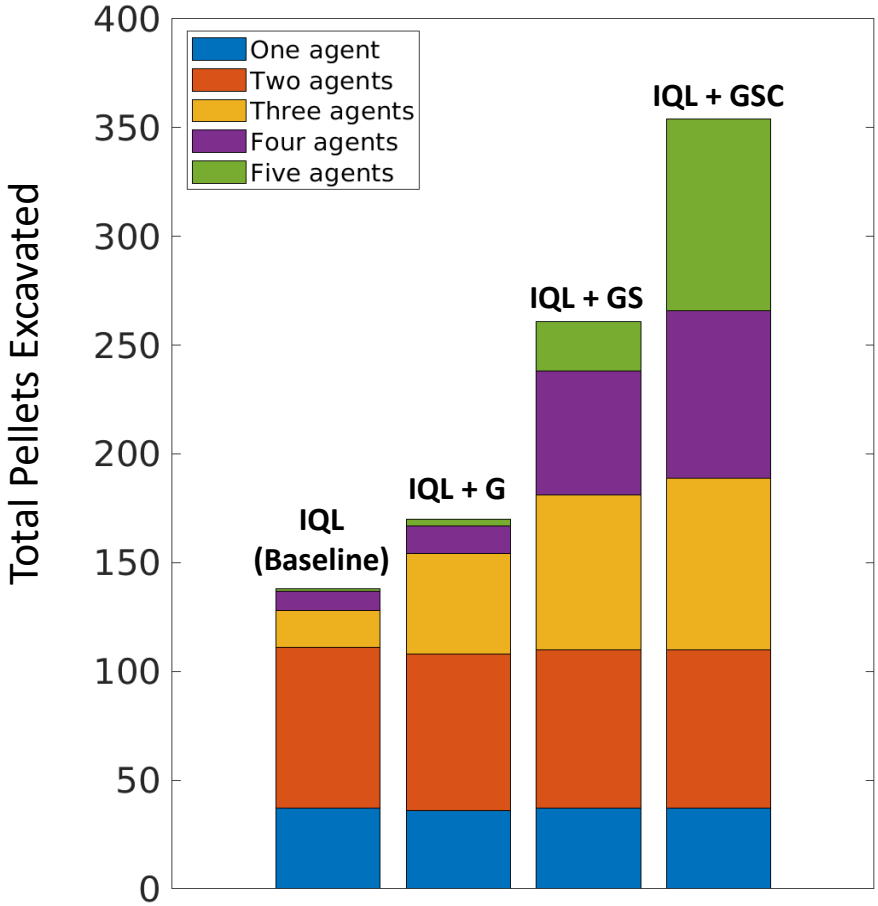

Figure 5.5: Cumulative excavation plot comparison for four different MADRL techniques for up to five agents.

and global rewards. However, as the number of agents increases, the performance drops drastically. At this stage, we introduce stigmergy to facilitate cooperation and to improve convergence. This is achieved by augmenting the observation space of each agent with information encoded in the virtual maps or "pheromone trails". This technique improves performance for three and four team sizes, but not so well for five agent scenario. In this case, coordination becomes more challenging, and achieving successful convergence would require more powerful training technique. Our curriculum learning approach (IQL+GSC) addresses this by reducing the difficulty in four and five agent scenarios, training one agent at a time.

Figure Figure 5.6 shows the comparison of the workload distribution for the two best

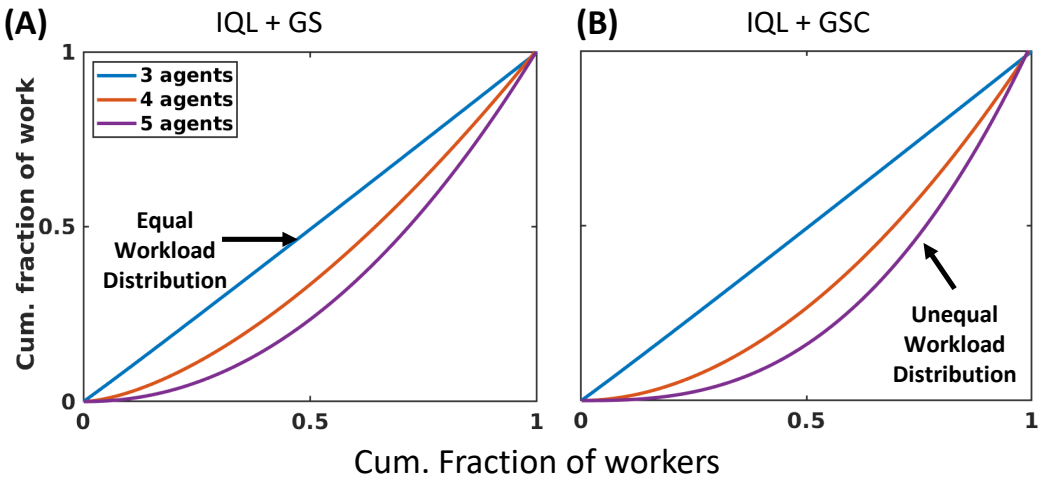

Figure 5.6: Lorenz curves comparison between learning with stigmergic only **(A)** and learning with stigmergy and curriculum learning **(B)**.

learning approaches using Lorenz curves [114]. Lorenz curves describe the workload distribution of a team by linking the cumulative share of number of agents to the cumulative share of total number of food items retrieved. This curve is convex by definition. An equal workload distribution appears as a straight line between (0,0) and (1,1). A divergence from this straight line indicates unequal workload distribution, where for example, half of the team retrieved less than half of the total number of food items. According to the curves in Figure Figure 5.6, it is clear that for three agent scenario, all workers excavated equal amounts of food items as indicated by the straight line in both figures. For four and five agents however, the curves diverge further from the straight line, with the curriculum learning technique (Figure Figure 5.6B) having the greater divergence or "higher" unequal workload distribution. This translates to: more agents should participate less as the team size increases. Thus, we can infer from the Lorenz curves and Figure Figure 5.5 that the best strategy to achieve the highest food retrieval successes is through asymmetric or unequal workload distribution when the agent density is above a threshold, in this case, four agents. This strategy greatly minimizes jamming and collision in the tunnel. Lorenz curves for one and two team sizes were not shown here because they all produce similar curves which are

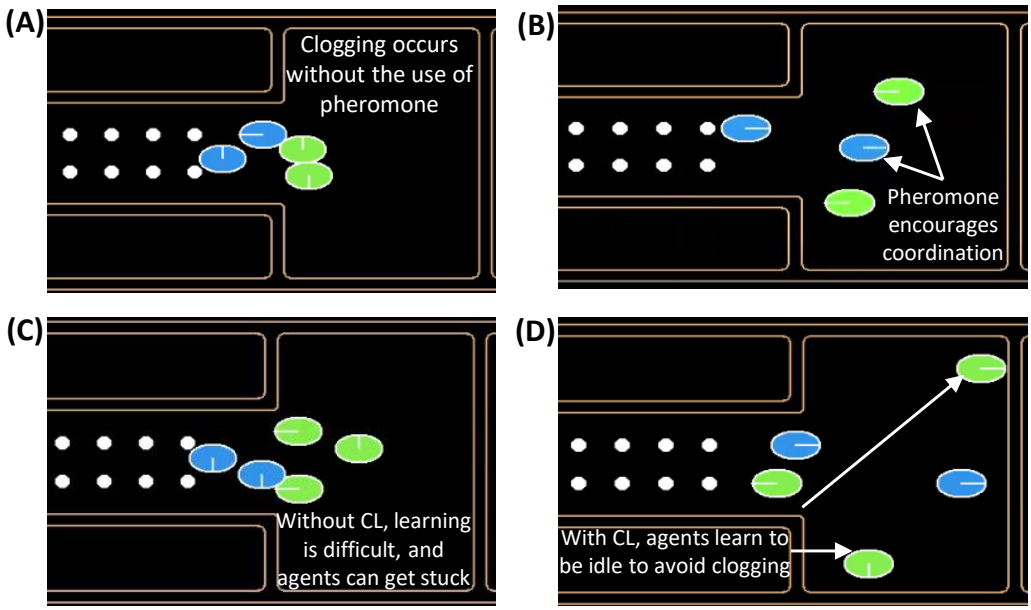

Figure 5.7: Snapshots of various scenarios during testing. **(A)** Four agents often get stuck in the tunnel when trained without stigmergy. **(B)** Stigmergy prevents agents from getting stuck, which enables effective coordination. **(C)** Five agents with stigmergy can still get stuck due to the difficultly of the scenario. **(D)** Curriculum learning enables efficient learning in difficult scenarios, and the discovery of a bio-inspired strategy: unequal work distribution to avoid clogging.

equal workload distributions.

Figure Figure 5.7 shows four scenarios illustrating the behavior of the final policies discovered by the different methods. Figure Figure 5.7A is a typical "clogging" or "jamming" scenario we encounter during testing when agents are trained without incorporating stigmergy, even though global reward was used. The agents get stuck due to lack of cooperation. On the contrary, Figure Figure 5.7B shows the coordinated behavior that emerges when virtual pheromone is employed. The agents learned to carefully avoid clogging by spacing out when outside the tunnel and yielding to each other when inside the tunnel. This is effective to avoid getting stuck in three to four agent scenarios. In five agent scenario, the agents can still get stuck even with virtual pheromones (Figure Figure 5.7C). This is due to the complexity of the problem: high non-stationarity. We ameliorate this effect by applying curriculum learning, and the improved performance is as shown in Figure Figure 5.7D.

The snapshot shows that two agents are ”idle” or ”resting” by not participating in the food retrieval process, which is a strategy to avoid congestion (please see youtube link for video demonstration). This supports a biological phenomenon discovered by Aguilar et al. [35] that idleness or asymmetric workload distribution is an optimal strategy that minimizes clogging in crowded and confined environments.

5.4.1 Improving Scalability in Dynamic Environments

Multi-agent learning in dense and dynamic environments is more challenging because the changing environment and inter-agent interactions compound the fundamental non-stationarity of the learning process, which exacerbates learning instability and non-convergence. To tackle this problem, we explored different techniques to improve the convergence of our proposed Stigmergic MADRL framework. In particular, we introduced the tunnel density information, $\rho = n/L$, defined as the number of agents, n, in the tunnel divided by the tunnel length, L, and store it in the virtual pheromone map for the agents to use to infer the tunnel density. This density information makes the environment less non-stationary to the agents, which enables them to reason about the congestion level of the tunnel, to decide whether to stay lazy or enter the tunnel to fetch some food items. In addition, we trained the agents from scratch with a shared policy network in a stochastic environment. This involves using only one deep neural network instance and one replay buffer to train all the agents concurrently, making the learning process more efficient. It is important to note that this is still a decentralized learning and execution approach, because each agent receives a private local observation of the environment and makes an independent decision. These two modifications to our Stigmergic MADRL framework increase the scalability and convergence of our method to eight agents.

Further, our Stigmergic MADRL framework shows better scalability and convergence when compared to popular multi-agent deep RL algorithms such as Joint Action DQN Learning (JAL) and Multi-Agent Actor-Critic (MAAC) (Figure Figure 5.8). During training,

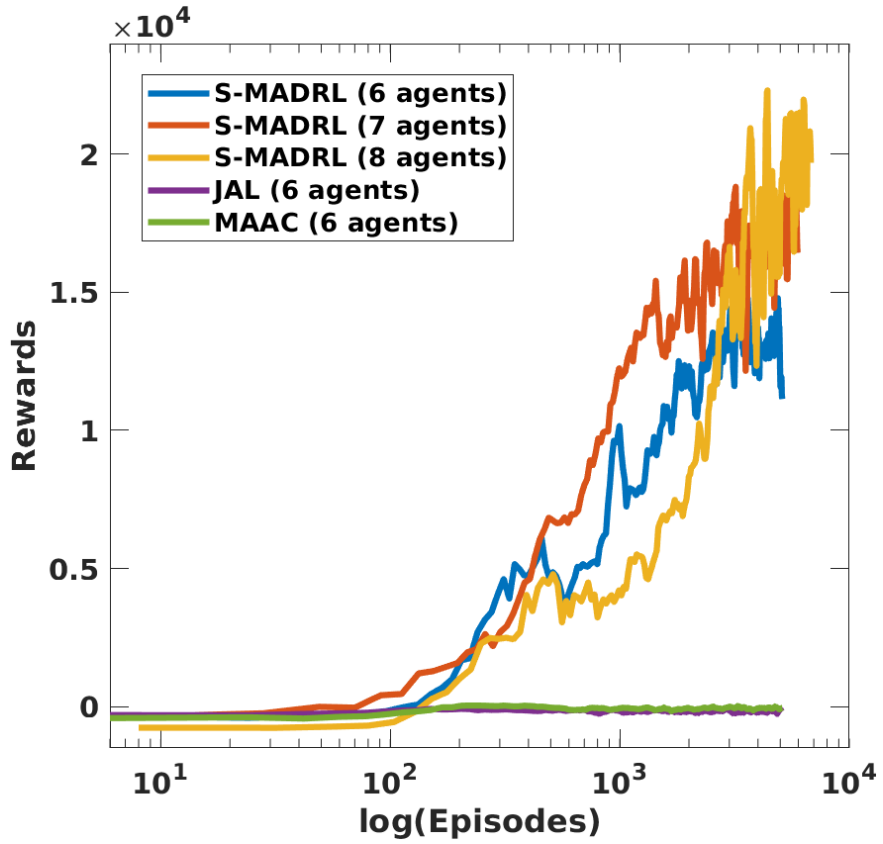

Figure 5.8: Comparison of our proposed Stigmergic MADRL method with popular multi-agent learning algorithms.

the agents used a shared environment in which the tunnel length is fixed and randomly selected between 3BL and 15BL at the beginning of each episode. However, during execution, the tunnel length is set to 3BL and gradually increased as the agents excavate the food items. From the semi-log learning curve in Figure Figure 5.8 which shows the low episodic reward of the two baseline RL algorithms, we could see that the joint action learning and actor critic algorithms struggle to converge for the 6-agents in a dynamic environment (this is similar for 7 and 8 agent scenarios, although not shown here). This non-convergence can be attributed to the high-dimension of the state and action spaces of the learning environment as described previously. However, for smaller number of agents (1-3 agents), the baseline methods show similar convergence and learning curves to our proposed stigmergic MADRL method. Thus, by using the environment as a shared memory or an

indirect means of communication, our proposed method is able to handle higher number of agents than baselines.

5.5 Conclusion and Future Work

In this chapter, we demonstrated the efficacy of employing stigmergy to address scalability and non-stationarity challenges within the context of multi-agent deep reinforcement learning. Our approach involves integrating a virtual pheromone or stigmergy mechanism into the learning environment. This mechanism enables agents to observe and deduce the intentions of their counterparts, facilitating the coordination of individual actions. We applied this methodology to a simulated scenario of collective food retrieval or excavation in a confined passage, requiring agents to coordinate to prevent congestion and enhance overall group performance. Consequently, we derived proficient policies for managing congestion in densely populated scenarios, resulting in optimal food retrieval. Our findings support the notion that social insects, such as ants, leverage social interaction or indirect communication for accomplishing intricate tasks.

CHAPTER 6

COORDINATION AND ADAPTATION STRATEGIES IN LIVING COLLECTIVES

In this chapter, we conducted biological experiments aimed at exploring the intricate dynamics involved in the successful excavation of nests within biological collectives. Our goal was to unravel the proximate biological and physical mechanisms employed by ants to regulate nest digging and assess how individual and collective behaviors influence the overall dynamics. Specifically, we aimed to ascertain whether the regulation of digging behavior could arise from individual decisions without the need for communication through chemical, visual, or acoustic signals. Building upon prior research on contact-responsive robotic collectives [115, 116], we investigated the role of contact sensing as a sole information source in governing excavation dynamics without additional forms of communication.

To discover a minimal model, we assumed that ants base their decisions on recent experiences, registering encounters with others but without actively exchanging information. Our study involved conducting experiments on ant digging behavior, followed by mathematical modeling and computer simulations to interpret the experimental outcomes. Our models exhibit remarkable agreement with the experiments, indicating that collectives can effectively coordinate their activities relying solely on simple contacts. Additionally, our findings provide predictive insights for further experimental investigations in this area.

6.1 Methods

6.1.1 Ant excavation experiment

We investigated substrate excavation dynamics in small groups of the fire ant *Solenopsis invicta* [117]. *S. invicta* ants are master builders. They successfully construct complex nests in soils or granular substrates found around the world using a diversity of materials [118,

89, 96, 119, 120, 117]. The *S. invicta* nest is a complex, sophisticated structure that allows individuals to move and rear offspring. The nest is fundamentally important to the success of *S. invicta* colonies and is one of the most conspicuous features of this social insect.

Three colonies of *S. invicta* were collected from Atlanta, Georgia, USA during the fall of 2020. Individuals were separated from soil using the drip flotation method [121]. Collected individuals were then housed in plastic containers and supplied with water and food as needed to ensure their continued survival.

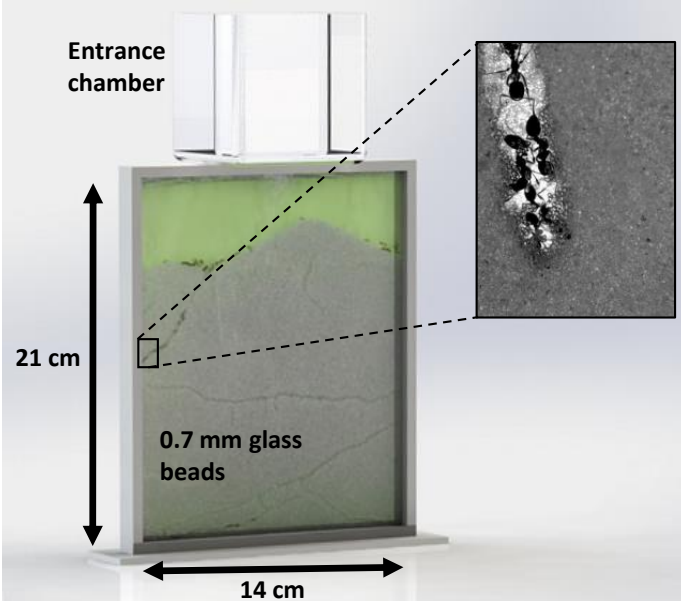

Figure 6.1: Experimental setup – 3D rendering. A custom-made plastic frame is holding acrylic sheets 2.5 mm apart. The accessible part of the system is 14 cm wide, and 21 cm tall. Roughly 2/3 of the volume is filled with a substrate - 0.7 mm beads, with the moisture content adjusted to 10% by weight. A top container serves as an entrance chamber where 40-70 ants are placed to start the experiment. The experimental trials are recorded using a commercial webcam.

We sought to visualize the behaviors that *S. invicta* workers engaged in digging tunnels to gain a greater understanding of the processes they use to construct nests. To do so, we monitored ant substrate excavation dynamics in experimentally created arenas, over many hours. Experiments were conducted inside of a setup consisting of two areas accessible by the fire ants: an open entrance area to which ants were added for each trial and a digging area

connected by a tunnel where excavation behaviors could be monitored through an acrylic wall (Fig. Figure 6.1). The excavation arena consisted of a 195 mm x 145 mm x 2.5 mm quasi-2-dimensional space, defined by a 3D-printed frame, to allow for monitoring of tunnel growth and ant activity. Glass particles measuring 0.7 ± 0.1 mm in diameter were used as an artificial substrate to be manipulated by the ants. Moisture content of the substrate was maintained at 10% water by weight across all trials; we have previously shown that *S. invicta* ants' excavation rate is insensitive to moisture content in the range of 5% – 20% [122]. The walls of the arena and entrance area were coated with talcum powder to ensure that ants could not climb out of the designated spaces. During experimental excavation trials, a controlled number of ants were transferred manually from the collected colony to the digging arena. Non-warming LED light sources were directed at the digging area to offer illumination for recording.

We allowed groups of *S. invicta* workers to dig within the artificial arenas. For each trial, groups of 40-70 randomly-selected worker fire ants were transferred into the apparatus. In total, 9 experimental trials were included in this study, using ants from three colonies, each used in at least two trials. Recording was started immediately following the addition of ants to the entrance area of the experimental setup. Once introduced into the entrance, the ants were allowed to migrate independently down into the digging arena. Fire ant excavation in the digging arena was monitored and recorded for up to 70 hours in each trial, which was recorded at 24 frames per second using a consumer computer camera (Logitech C920), at a resolution of 1920×1080 pixels.

6.1.2 Video analysis

To elucidate behavioral patterns involved in nest construction, we quantified collective digging, ant spatial distribution, tunnel morphology, as well as individual ant-ant interactions, through manual and automated video tracking.

To quantify ant participation in digging, we used captured color differences to classify

image pixels belonging to ants, as well as to detect the substrate interface. We used a bright green sheet as the background for the digging arena, which produced a visual contrast with the substrate, as well as with the ants; this contrast made for convenient video analysis based on color differences (Fig. Figure 6.2 A,B).

Within our experiments, we defined ants as participating in some aspect of the digging process if they were below the substrate interface. Once pixels were classified as belonging to ants below the substrate interface, the pixel count was converted to an estimate of the number of ants, using a calibration curve from a linear fit of manually counted ants vs. detected pixel count, tabulated over multiple time points, for each recorded video.

We were interested in observing the digging progression and rate regulation by detecting the excavated tunnels over time. This was done by following areas in the digging arena newly explored by the ants. Procedurally, we may accumulate a map of pixels that were classified as belonging to an ant at any given time in the past. However, the small amount of noise inherent in that technique, when accumulated, produces too many spurious detections of newly explored areas. Instead, we found that calculating and accumulating motion-detection produced much less noise. To calculate motion-detection, we down-sampled our videos in time to 12 fps by averaging subsequent frames, and then calculated the difference between the resulting frames. The result was then subjected to Gaussian filtering and a threshold was applied. The threshold was determined dynamically as 5 times the standard deviation of the Gaussian filtered values, over a region of the video determined to not have been excavated throughout the recording (manually). Fig. 2C shows an example of motion-detected pixels, red (blue) indicates pixels that became lighter (darker) between successive down-sampled frames. A map of accumulated motion-detected pixels was recorded over each recorded trial.

Ants occasionally reversed and backed out of a tunnel when their path was blocked by other ants. Reversal is an important behavior that minimizes traffic clusters [35] and we wanted to observe its relation to the number of ants in the tunnels and the overall digging

rate. We, therefore, tabulated event counts of ants reversing without excavating a bolus of substrate material, following an encounter with another ant. This was done for several non-branching tunnels, in multiple experiments, by visual inspection.

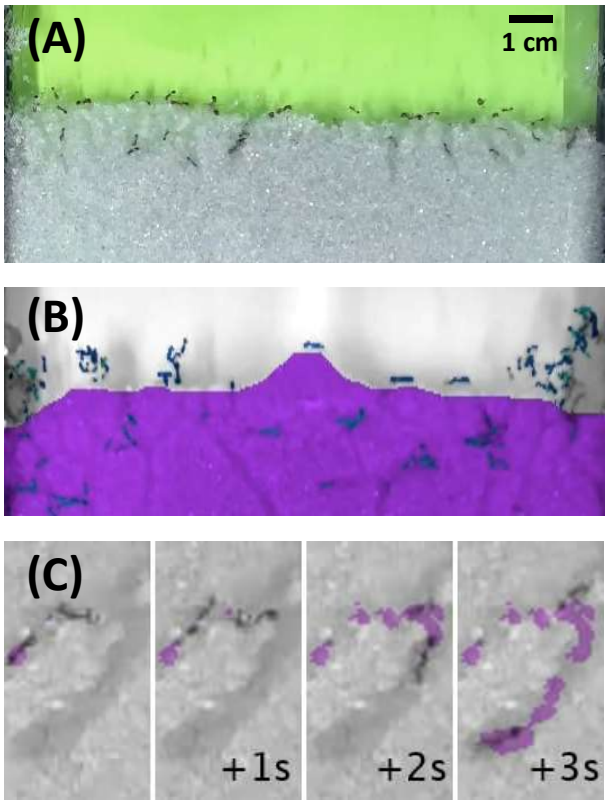

Figure 6.2: Experimental recording and tracking. Each trial was recorded at 24 frames per second using a commercial webcam. **(A)** The digging arena is enclosed between acrylic sheets and composed of glass beads. A green background is used to generate suitable contrast between the substrate and excavated tunnels. Image 40 minutes after the introduction of ants into the apparatus is shown **(B)** The color difference between ants, substrate, and background allows for substrate tracking (shaded purple) as well as ant tracking (shaded blue). Together, the number of pixels associated with ants below the substrate is used to quantify the participation in the digging activity. **(C)** Motion detection of ants was found to be a robust method to track the exploration of the system, and thereby allowed us to quantify the excavated area over time. We calculate the difference between each pair of sequential frames, apply a threshold, and accumulate the result over time (purple shade). See Methods for additional details.

6.1.3 Cellular Automata (CA) Model

To gain an understanding of the digging dynamics, particularly in the earlier times of excavation, we employed a cellular-automata (CA) model adapted from that in our previous work [35].

A CA model allows us to focus on the behavioral aspect, without a need to implement detailed mechanics of walking and digging. The simulation process boils down to a small set of time-evolution and interaction rules that are applied at every discrete step. Using this model, we may conveniently test various behavioral rules and then test how the CA model's emergent collective behavior matches experimental observations.

In the simulated system, ants walk through a linear tunnel that is composed of discrete cells that accommodate 2 ants each, going either the same or opposite directions (Fig. Figure 6.5). A tunnel has an entry into a sequence of "substrate-free" cells, followed by cells filled with substrate. At each time-step, ants move along the tunnel until they reach substrate; at which point they pause and excavate pellets. Once a substrate cell is drained of its pellets, it becomes available for ants to walk into. In this way, the tip of the tunnel, where the substrate starts, elongates over time.

Within this computational model, a cell represents a single ant body-length's (BL) worth of tunnel. *S. invicta* made tunnels generally converge to two ant body-widths (BW) [89] and allow most ants to pass each other unhindered. We therefore take cells to represent an area of BL$\times$BW and allow two ants to occupy each. At each time-step (TS), an ant will move to the next cell, if it is empty,, and will thereby be moving at a speed of BL/TS. By quantifying the typical ant speed within tunnels, we relate the simulation timescale to the digging experiment timescale. Within the simulation, we parameterize the number of pellets in a cell full of substrate and the number of pellets an ant will excavate each time; these respectively represent the typical accumulated time it takes the ants to excavate a BL length of tunnel, and the typical time spent at the tunnel tip by each ant.

As mentioned above, two ants may occupy the same cell and pass each other unhindered

under this model. Three (or more) body collisions occur when an ant tries to move into a cell that is already occupied by two other ants. In response to such a collision, an ant that is heading towards the tip of the tunnel randomly chooses whether or not to reverse course without digging, with a predetermined probability. We find that our results are not sensitive to this reversal probability within the range (0.1, 0.9), and choose to fix this value to 0.34, as previously used [35]. An ant that is headed towards the exit never relents. In our previous work, we used only a single simulated tunnel [35], but here we use multiple tunnels to capture the observed experimental behavior, as discussed below. Within the multi-tunnel CA model, when ants enter a tunnel, they first randomly and uniformly choose which of the multiple tunnels to go into. From there, the simulation proceeds as previously discussed.

6.1.4 Ant speed measurements

We analyzed image maps to estimate an ant's walking speed within tunnels. Given tracked exploration maps (Fig. Figure 6.6), we extracted an example of a long branch-less tunnel and generated a linearized profile through the following steps: (1) fit a center-line spline, (2) generate perpendicular cross-sections at equal longitudinal intervals (Fig. Figure 6.3A), (3) sum binarized pixel values across each cross-section. As ants walk through the tunnel, the linearized profile displays a traveling cluster of high values. We lay out these linear profiles over time as a space-time diagram (Fig. Figure 6.3B,C). We then use a sliding time window of 1 minute and for each window run a line-detection algorithm (Fig. Figure 6.3). Ant walking speeds are captured as the detected line slopes. We accumulated the distribution of observed speeds for well-detected lines. Through thousands of detected slopes, we observed a mean speed of 0.27 ± 0.09 BL/sec.

6.1.5 Model fitting

To fit the experimentally observed excavation rate, we iterated through randomly generated values for the free parameters of the simulated model, and compared to the observed curves.

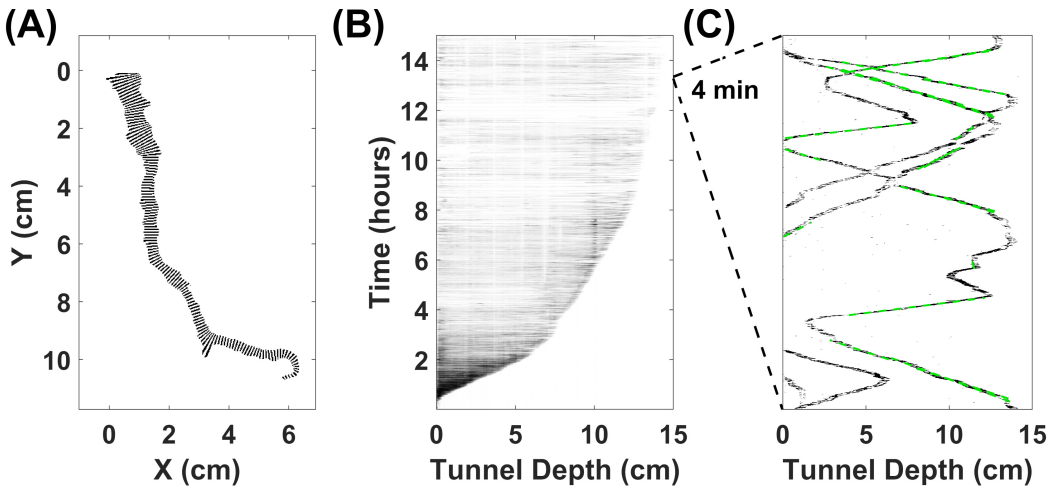

Figure 6.3: Analysis of space-time detection maps. (A) A branch-less tunnel was divided into strips of equal longitudinal distance, shown in alternating black and white strip color. (B) Ant pixels were determined based on a darkness threshold, and binarized pixels were summed over every longitudinal strip, to generate a density profile over time. Ant presence was observed deeper in the tunnel as it grew through digging. (C) The space-time detection maps revealed distinct tracks when observed at minutes long windows. We used a Hough transform [123] to detect straight (i.e. constant speed) trajectories fragments (green dashed lines). From the collected line slopes we calculated an average observed ant walking speed in tunnels.

Other fixed parameters have been determined either in previous work or in this (Table Table 6.1). The CA simulation time has been rescaled such that the ant nominal speed BL/TS is the experimentally determined mean ant walking speed. The model we fit then has four free parameters – number of tunnels, work-rest imbalance constant (τ_0), relaxation length (L_0), and steady-state participation level.

We started by generating 10,000 uniformly random sampled sets of parameters, in a wide range. To test a fit over multiple time scales, we first converted both the observed curve and each simulation curve to log-rate over log-time, by averaging within exponentially spaced bins and then taking the log of the average rate in each. The corresponding squared-error for each bin was taken to be the average of squared-errors, weighted by number of observations. The error of the log-rate was taken to be $\Delta \log(r) = \Delta r / r$, where Δr is the error of the rate, and r the rate.

For each simulation excavation rate curve, we calculated the average of squared differences from the observed curve, weighted by $(\Delta r)^{-2}$, in log-log, over the range of 0.5 hour to 35 hours, as a measure of quality of fit. In each refinement iteration, we used enough sampled parameter sets to display a clear trend in at least one parameter. Following each iteration we narrowed the search for those parameters that displayed a clear minima (Fig. Figure 6.4).

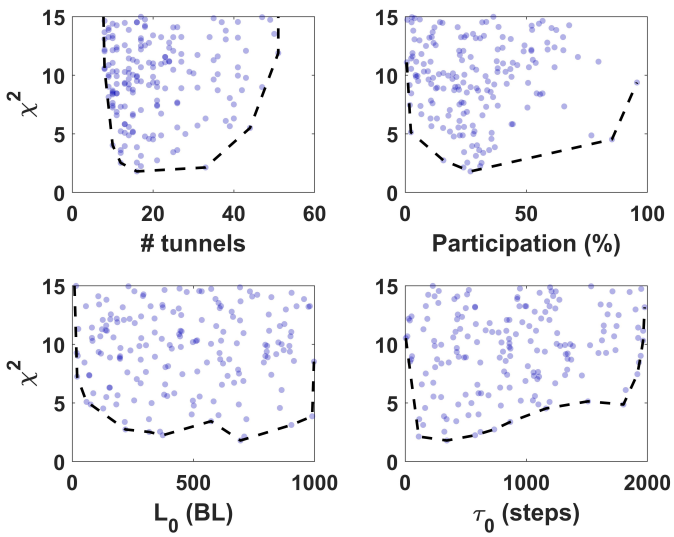

Figure 6.4: Least-squares (χ^2) optimization of the model fit through random parameter sampling. In each iteration, we sampled thousands of parameter variations and calculated the mean-square distance between the simulated excavation-rate trend and the experimental one. We plotted χ^2 distances vs. each one of the fitted parameters (blue points), as well as an outline of their basin as a guide (black dashed line). After each such iteration we narrowed the search range for parameters with a clear single-minimum in their basin.

Table 6.1: Parameters used to calibrate the cellular-automata simulation to the experiment

Parameter	Value	Source
Average body-Length (BL)	3.95 mm	[89]
Tunnel Width	2.2 mm	[89]
Reversal Probability	0.34	[35]
Average ant walking speed	0.27 BL/sec	space-time analysis (Fig. Figure 6.3)
Average time spent digging	8.5 sec	visual observation
Number ant trips per cm dug	206	visual observation

6.2 Results & Discussion

Our overall goal was to understand the mechanisms that animal collectives use to coordinate their activities. We investigated this issue by studying patterns of tunnel digging in fire ant societies. We combined theoretical and empirical approaches to gain insight into the possible proximate mechanisms used by fire ants to accomplish important group tasks.

6.2.1 Ant digging experiments

Our first aim was to gain insight into the rate of excavation of tunnels by the ant collectives. Following the introduction of ants into the system through the entrance chamber (Fig. Figure 6.1), ants explored the container and went through a narrow hole into the digging arena in the lower chamber, in view of the camera (see Methods). We observed that all of the ants entered the lower chamber within the first hour and many engaged in digging after a few minutes of exploration.

Within the first few hours, about half of the ants seemed to be digging through multiple tunnel entrances and occasionally tunnels wide enough to accommodate 3-5 ants [Fig. Figure 6.6 (A)]. At longer times, the tunnels grew through digging and deposition until a few tunnel entrances remained at the top substrate interface. As the level of activity decayed, the tunnels converged to the typical width of a single body-length, roughly accommodating the width of 2 ants [119]. Maps of first-exploration-time [Fig. Figure 6.6 (B)] capture the features of tunnel formation over the course of each trial.

The area excavated over time in all the trials follows a trend of rapid increase, followed by a slower increase [Fig. Figure 6.7 (A), non-black colors]. In a few trials, additional waves of rapid digging appear, but eventually, those too settled into a slower rate of expansion. By averaging area excavated in all trials, this trend appears even clearer [Fig. Figure 6.7 (A), black]. Excavation rates reveals additional subtleties when viewed over multiple time-scales

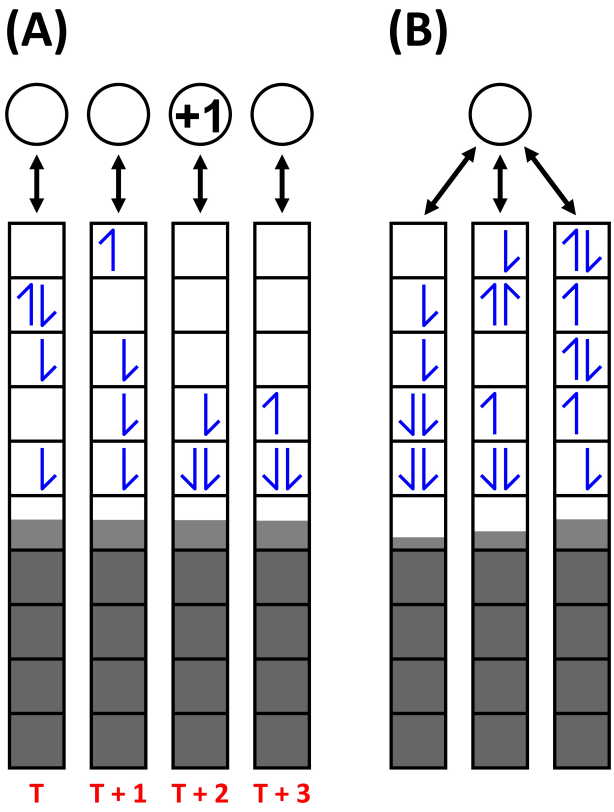

Figure 6.5: Cellular-automata simulation of ant excavation. Within this model, ants from the outside reservoir (circle on top) randomly decide to enter the system, with a probability determined by each individual. Once inside, the ants (blue half-arrows) move along a tunnel composed of discrete cells. Each cell accommodates up to two ants. An ant moves from the entrance (top) towards the substrate (bottom, in gray), and cannot move into it. Once an ant reaches the first substrate cell, it removes (i.e. excavates) pellets from it, one every time step. After reaching its holding capacity, an ant reverses and moves back towards the entrance to deposit the excavated pellets. Substrate cells are initialized to contain a consistent number of pellets, which are then depleted by the incoming ants. The number of pellets remaining in the first reachable substrate cell is indicated by the degree of filling of lighter gray. **(A)** Four consecutive time steps, showing four ants in a single tunnel, at first (time T) with one heading out towards the entrance, two on their way towards the substrate, and one already at the substrate interface. At $T + 1$, another ant reaches the substrate, and the third is a cell behind. Since the next cell is occupied by two ants, the latter ant, at time $T + 2$, decides to reverse and head back towards the entrance without excavating. Also at $T + 2$, an ant steps outside and increases the count of ants in the outside reservoir (number in circle). The ants at the interface remain there to excavate until either the cell is depleted or their individual capacity is full. **(B)** In a multi-tunnel model, ants from the outside reservoir randomly decide to enter, as before, and also randomly uniformly choose which tunnel to go into. Once inside, the dynamics are as in a single independent tunnel.

[Fig. Figure 6.7 (B)], where we observe three regimes: high and almost constant excavation rate, followed by a rapid decay, and then a slow decay at long times. The transition between the rapid and slow decays is quite distinct, especially considering the noise involved in experiments comprised of living systems.

Previous studies of ant excavation reported similar trends [124, 80, 125], where the excavation rate started high or peaked, and then tapered off over time. Buhl et al. model their observations empirically as a classical Verhulst logistic model [80]. The logistic model predicts that the excavation rate should vanish exponentially in time, which is not supported by our observations (Fig. Figure 6.7). The same model was used before by Rasse et al., though on a much longer timescale of tens of days [124]. Bruce et al. empirically model the excavation rate as a power-law which predicts a t^{-2} decay in rate over time [125], which is also inconsistent with our observations. Neither of these previous models is derived from explicit behavioral rules, though Bruce et al. allude to a state of ”arousal” that may follow a disturbance or introduction into an unfamiliar system. Bruce et al. also make important observations about the initial digging rate depending on group size and initial tunnel length. We will show below that these observations of arousal and length/number dependencies can be explained by a single behavioral rule that originates from the local experience of collisions.

6.2.2 Participation trend

To investigate the apparent regulation of excavation rate over time, we suspected two possible modulated parameters – participation level in digging and walking speed. A previous study showed a weak decay of walking speed in time [125]; not significant enough to explain our observed excavation rate. Our limited observations of walking speed did not reveal any trend in time. In contrast, we quantified the collective participation level in digging, and discovered a distinct trend.

At all times during an experiment, a significant portion of the ants was above the substrate

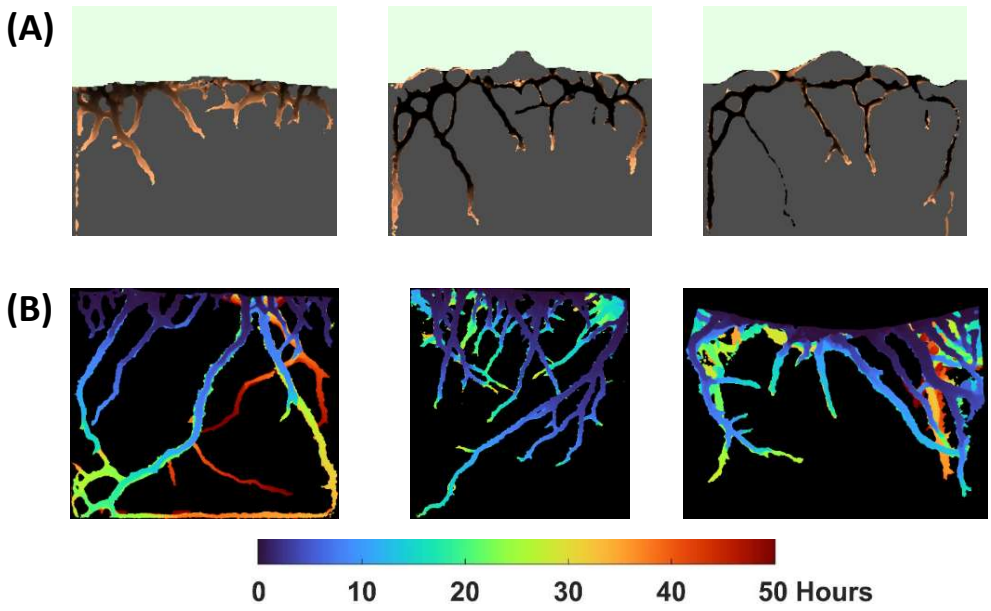

Figure 6.6: Flow morphology tracked by ant motion detection and color thresholding. (A) Tunnel exploration over time in a single experiment. Left to right – the system, as it evolves in time slices of 3 hours, over the course of the first 9 hours. The substrate is tracked using color thresholding (see Methods) and displayed in gray. Above-substrate is colored in light green. Tunnel color represents a spatial mapping of first-exploration time, ranging from darkest for earliest and brightest brown for latest (3 hours) within the time slice. Newly detected flow indicates freshly dug passages. Missing flow in existing tunnels may indicate a loss of interest, due to a blocked entrance or other reasons. (B) Tunnel formations mapped by first-exploration time, over 50 hours. Left to right – three independent experiments. Unexplored areas are in black. The color indicates first-exploration time, according to the color map below.

and not actively participating in digging. While above the substrate, many of the non-digging ants were found in clusters. This aggregation behavior has been reported in previous studies, for multiple ant species [124, 126, 127, 80, 125, 89]. Around 2-3 hours into each trial, participation in digging peaks at about 50% (Fig. Figure 6.8 A). Coincidentally, roughly at the same time the slow excavation-rate decay starts (Fig. Figure 6.9 A). At later times, the average participation trend appears to fluctuate around a mean value of about 27.8% (green dashed line in Fig. Figure 6.8 A), calculated as the average participation between 5 and 35 hours; this participation rate is consistent with previous reporting [89]. The early peak in participation correlates with the initial high excavation rate and its rapid decay, but then the

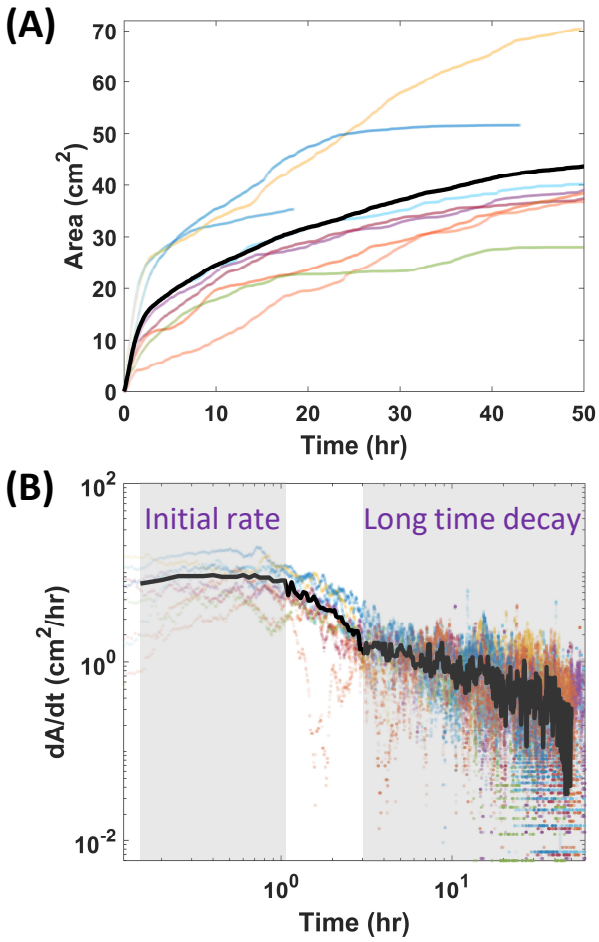

Figure 6.7: Excavation area dynamics for multiple trials of *S. invicta* digging. **(A)** Area excavated over time in nine independent trials (colored points), along with their average (black line). A clear trend emerges where at the initial phase there is almost constant excavation rate, followed by a rapid decay, and at longer times a slower decay. **(B)** Area growth rate over time, with colors as in (A). A consistent feature of rapid excavation appears in all trials at the first few hours, followed by fast decrease in rate and a sharp transition, leading to slow decay in rate at long times.

rate decays further, while, surprisingly, participation remains constant.

6.2.3 Cellular Automata behavioral model

To probe the effect of individual dynamics on participation in digging and the emergent excavation rate, we built upon our Cellular Automata (CA) model simulations [35] (see Methods). In our previous work, we experimentally resolved the activity of ants at the

individual level, while excavating a single tunnel, and consistently observed a highly unequal workload distribution among excavating ants [35]. In corresponding CA simulations, individuals were initialized to have a preset equal or unequal tendency to enter tunnels; an unequal tendency to enter the tunnels was crucial for avoiding long duration clogs and maintaining optimal traffic flow conditions. Recently, we reported on the implementation of reinforcement rules in a collective of excavating robots that results in unequal workload distribution and improved group performance [116]. In fire ants, it was reported that groups adapt to the removal of the most active workers by up-regulating the activity of others [35], though we do not know what mechanism they use to accomplish that.

The model we developed (below) generates unequal workload at any given moment, but the turnover between ants equalizes the workload quickly, compared to previous observations of persistent inequality over 12 hours [35]. Notably, in [35] (supplementary), some turnover in levels of activity was also observed. This inconsistency can be regarded as a level of abstraction, where we capture the collective level of activity but not the workload distribution. The collective rate of excavation, which is of interest here, scales as the number of actively excavating ants, or mean participation level, at low ant densities. Workload inequality – the individual deviations from the mean participation – is a second order effect which we do not consider in this work.

First, we compared unmodified CA model simulations (Fig. Figure 6.5 A) [35], originally developed to reproduce excavation in a single tunnel, to the observed experimental trend and found the excavation rate trend produced by the CA model was far below that actually observed (Fig. Figure 6.9). Expanding the CA model to include multiple independent tunnels sharing an ant reservoir results in a higher excavation rate, where a model with 10 tunnels captures the experimental long-time average trend (Fig. Figure 6.9). Interestingly, both the unmodified and multi-tunnel CA models already capture the experimental long-time decay trend (Fig. Figure 6.9). This result suggests that the slow decay is an inevitable result of digging in an elongating tunnel.

Our multi-tunnel CA model still did not capture the early-time rate trend that coincided with regulated participation. Moreover, the participation of ants in digging saturates at 100% in the CA model, regardless of number of tunnels and in contrast with the experiment (Fig. Figure 6.8). We thus posit that a stable and low participation in digging requires a self-limiting process.

We observed in experiments that most of the ants above the substrate aggregated together in clusters of inactive individuals; this inactivity invoked the possibility of a resting behavior. Therefore, we implemented a self-limiting process in the CA simulations, maintaining a fixed work-to-rest ratio R. We define a work-rest imbalance as $t_{\mathrm{imbalance}} = t_{\mathrm{work}} - R \cdot t_{\mathrm{rest}}$, where t_{work} (t_{rest}) is the time an ant spent working (resting). An ant is set to go dig with probability $P_{\mathrm{go}} = \exp\left(-t_{\mathrm{imbalance}}/\tau_{\mathrm{tolerance}}\right)$, where $\tau_{\mathrm{tolerance}}$ is a time scale for the tolerance of a work-to-rest imbalance. Inclusion of this parameter indeed results in a low and stable participation in the CA simulation at long times (Fig. Figure 6.8, green line), and an excavation-rate trend consistent with most of the experimentally observed curve (Fig. Figure 6.9, green line). This model can also accommodate previous observations of an unequal workload distribution among ants [35] by replacing the work-rest imbalance ratio with a distribution.

While the CA model with regulated participation is fairly consistent with the observed rate, it does not capture the steepness of the observed rate decay at early times (Fig. Figure 6.9, up to 3rd hour). This mismatch implied an additional mechanism is at play that maintains high excavation rate at early times and generates a sharp transition to the long-time behavior.

We propose that the ants are willing to delay rest when digging a fresh system of tunnels, until they detect the tunnels are large enough, at which point they revert to a nominal behavior pattern. Such a mechanism implies that the insects become agitated without shelter, and relax into nominal behavior when sheltered, consistent with the concept of ”arousal” described before [125].

We therefore amended our CA simulations to include digging probability with an

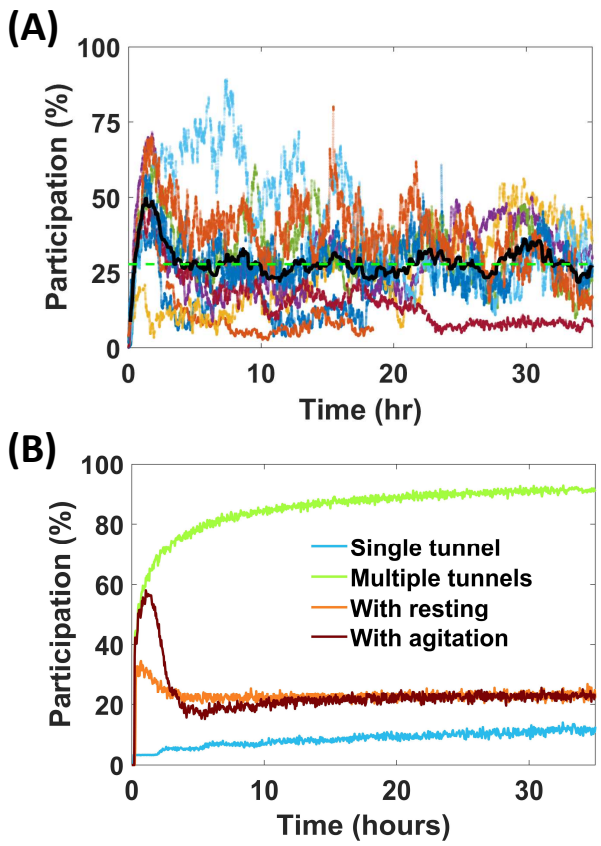

Figure 6.8: A varying fraction of the ant collective actively participate in digging at any time. (A) Tracked ant participation in the recorded experimental trials. Each colored line represents an individual trial. The average trend (black) peaks around 50% and later fluctuates around constant participation of about 27.8% (green dashed line). (B) CA model simulations with varying implemented features. In both a single tunnel (blue) and multiple tunnel models (green), the participation monotonically increases over time, though faster for multiple tunnels. To the multiple tunnels we added a tendency to rest and maintain a 27.8% work to rest ratio (orange). Once "agitation" is added to the previous model on top of resting behavior (brown), it overrides the behavior at short tunnel lengths which manifests as a high participation peak, followed by a sharp fall.

"agitation" override: $P = (1-\alpha) \cdot P_{\text{go}} + \alpha$, where $\alpha \in [0, 1]$ is an override parameter. Setting $\alpha = 1$ completely overrides the affect of a work-rest imbalance on the decision to go dig. But what exactly do the ants detect, and does communication play a role?

Here, we define communication broadly as when one individual receives information from another, and this information affects the future actions of the focal individual. A reasonable type of information that can be obtained by an individual ant is its collision

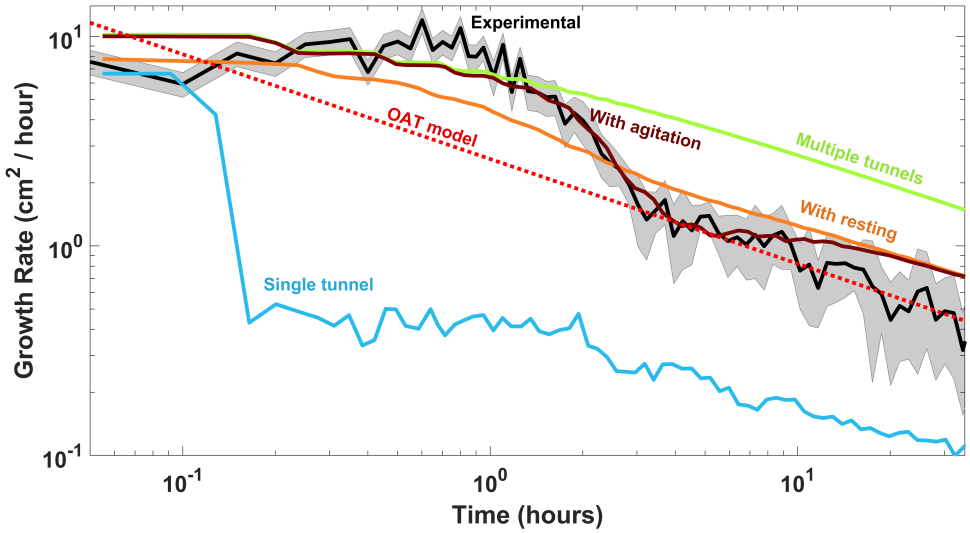

Figure 6.9: Experimental and computer simulated excavation rate over time. CA model simulation fit to the excavation rate (brown line) over the experimental average (black line) and its standard deviation (shaded). The fit was generated by a model including a mean-free-path dependent "agitation" override. The same model was also run, for reference, without "agitation" (orange line), also without resting behavior (green line), and finally also without multiple tunnels (blue line, multiplied by 10). Both models, as well as the experimental trend, are consistent with a one-at-a-time like $t^{-1/2}$ asymptotic behavior (red dotted line, scaled to match experimental).

with other individuals and its surroundings. These collisions are accounted for through the success rate of moving forward, the inverse of which is essentially an estimate for mean free path. We therefore set α to be $\alpha = e^{-l/L_0}$, where l^{-1} is a move success-rate estimator and L_0 is a length constant. By defining an ant's response this way, we get a unified view of collisions with other ants, as well as with the end of the tunnel.

The expected mean free path in any single tunnel is L/f, where L is the tunnel length and f is the (expected) number of failed moves, cause by the tunnel end and cells occupied by 2 ants. A single ant in a tunnel will experience $f = 1$ failed moves, and will therefore estimate $l \approx L$. In a mean-field calculation, f is expected to be $L \cdot \mathrm{P}$(two ants in one cell) $\propto L \cdot (N/L)^2$, where N is the number of ants in the tunnel, which would result in $l \propto L^2/N^2$. This mechanism effectively has a quadratic sensitivity to the density of ants in the tunnels. In practice, our move success-rate estimator (l^{-1}) implementation uses an

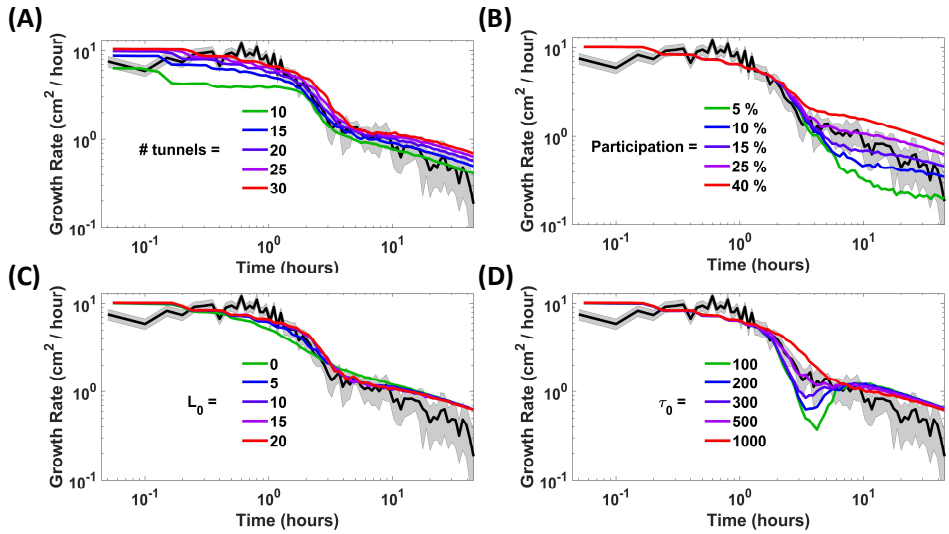

Figure 6.10: Parameter variation effects on modeled excavation rate. The best-fit model parameters were perturbed one at a time, around the best-fit value, while keeping the rest fixed. The resulting variations in computed excavation rate (averaged over 11 runs) are depicted in colored lines, overlaying the experimental average (black line) and its standard deviation (shaded). **(A)** Varying number of tunnels. **(B)** Varying steady-state ant participation rate in digging. **(C)** Varying "agitation" relaxation length L_0. **(D)** Varying work-to-rest imbalance tolerance τ_0.

exponential moving average with a memory time constant of $2L_0$ to avoid infinitely long memory.

The result of this length-dependent relaxation mechanism is a higher tendency for the simulated ants to dig at short tunnel lengths, overriding the steady-state work-rest behavior. As the length of a tunnel increases, the participation of ants in digging relaxes to its steady-state value. As noted above, the mean free path estimator is highly density dependent through 3+ body collisions, since there is no penalty (delay) for 2-body collisions in the simulation. At tunnel lengths reaching the designated relaxation length L_0, some ants decide to rest, which further reduces density in the tunnel and as a result a cascade of relaxation into the stead-state behavior occurs. This cascade gives rise to a sharp transition in excavation-rate, closely matching the observed experimental trend (Fig. Figure 6.9, blue line).

Our developed CA model captures the subtleties of observed collective ant digging behavior, and its success suggests a behavioral mechanism employed by the individuals.

Parameter variation around the best-fit solution demonstrates the observed features associated with each parameter, and in particular – that the relaxation length (L_0) modulates the steepness of the transition between early and long time stages (Fig. Figure 6.10). It is worth noting that our "agitation" override neatly captures previous concepts of "arousal" and initial tunnel-length effect. The model also predicts faster relaxation with fewer ants, as fewer ants would result in fewer collisions for the same number of tunnels. Whether they would relax faster or not depends on the actual number of tunnels they generate, which may be affected by the width of the arena.

In this work, we introduced a resting mechanism employed homogeneously by all simulated ants, which results in a nearly equal workload distribution (data not shown). This is in contrast with previous findings of unequal participation in work and the potential rules leading up to it [35, 116]. Our focus in this work was on emergent global regulation, rather than emergent workload distributions. The CA model here essentially captures the same global inequality as low participation rates at any moment, but the work is spread among all individuals. The same can be implemented with an unequal workload, by setting target work-to-rest ratios heterogeneously among the simulated individuals.

6.2.4 The One At a Time model (OAT)

We turn our attention back to seeming inevitability of the long-time excavation rate decay, which appears in experiment as well as in all of our CA model variation. Ants working inside a narrow tunnel can be considered as a one-dimensional asymmetric exclusion processes with few lanes [35, 128, 129]. Let us consider an unregulated digging-rate behavior with a single ant, digging in a single tunnel. This ant performs a consistent walking and digging procedure over time, going back and forth, digging at the tip of the tunnel and depositing dug pellets at the entrance (Fig. Figure 6.11). We call this scenario the one-at-a-time (OAT) model, which we found experimental support for [35].

The corresponding tunnel growth follows the length–time relationship:

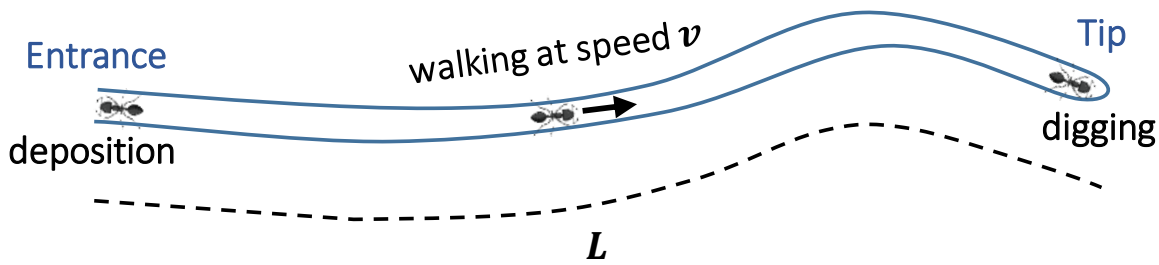

Figure 6.11: Illustration of the One At a Time model. A single ant walks back and forth from the entrance of the tunnel to the tip. It digs at the tip and deposits the dug pellets at the entrance. The time for digging and deposition τ_{dd} is assumed to be constant. In between the entrance and the tip, the ant walks at speed v across the length of the tunnel L that extends over time.

$L(t) = \frac{1}{2}\left(-v\tau_{\mathrm{dd}} + \sqrt{v^2\tau_{\mathrm{dd}}^2 + 4Kvt}\right)$, where the initial condition is set to $L(t=0) = 0$. By taking the long-time limit $t \to \infty$ we see that in the OAT model, the tunnel length asymptotically grows as $t^{1/2}$, and the corresponding tunnel excavation rate follows $1/L \propto t^{-1/2}$ at long tunnel length or late time. Though this was derived for an ideal situation, it clearly extends to any scenario with a constant number of participating ants and negligible time spent on collisions or any activity other than walking and manipulating substrate. Remarkably, this derived long-time rate trend agrees with both experiments and CA model (Fig. Figure 6.9, red dashed line).

6.3 Summary and Conclusion

The goal of this study was to understand the dynamics of digging behavior by collectives of social animals. We performed ant excavation experiments in a quasi two-dimensional system and observed a pattern of excavation-rate modulation [124, 80, 125]. Our experimental excavation rate trend reveals signs of regulation across timescales, including a high and constant initial rate, a fast decay, and a slow decay at long timescales (Fig. Figure 6.7).

We developed a detailed behavioral CA model, carefully motivated and with most of its parameters derived from independent observations. Our model captures the experimental excavation rate trend very well across timescales, and suggests ants modulate their participation in digging based on their recent history of collisions. Crucially, this modulation,

which we referred to as an ”agitation” override, addresses observed effects of both initial tunnel length and ant group size, and serves as a mechanistic description of the concept of ”arousal” which is referred to in ref [125].

Previous studies employed empirical models such as a logistic model or a power-law to describe their excavation rate observations without being clearly motivated by behavioral rules at the individual level [124, 80, 125]. We derive a quantitative fit, by introducing resting behavior, which can be overridden in a state of agitation. Both the regulation of work-rest and agitation is done by each individual in our model. Agitation is determined from length estimates based on collisions with others, as well as with the end of the tunnel. Moreover, we showed that an idealized One At a Time model, derived by considering the fundamental timescales in the system, predicts that the excavation rate should decay like $t^{-1/2}$. This prediction fits our experimental observation and models at long timescales surprisingly well.

Our models do not include many aspects of the tunnel system, such as width variations, formation of branches, and unequal investment in the digging of each tunnel. Despite that, the success of CA models in this (Fig. Figure 6.9) and previous work [35] is promising, and could be extended to include other behaviors and additional details.

CHAPTER 7
CONCLUSIONS AND FUTURE WORK

Robots operating in crowded and confined environments are encountered in various real-world scenarios, such as search and rescue operations following a disaster, the use of autonomous robots in densely populated warehouse operations, and their application in subterranean tasks like underground tunnel excavation or cave mining. These environments are characterized by significant uncertainties, including random failures, collisions, and noisy information or measurements, making them inherently challenging. Consequently, multi-robot coordination methods that heavily rely on explicit domain knowledge or model of the environment, and those dependent on explicit communication or centralized coordination, have limited applicability to these real-world scenarios due to the potentially catastrophic consequences of unplanned or unexpected collisions.

To effectively coordinate multi-robot systems in such complex environments without explicit communication or a centralized controller, researchers must consistently explore and embrace the unavoidable features of dense active systems, as seen in natural swarms. This necessitates not only considering localized, physical, and stigmergic interactions at the algorithmic level but also integrating these features into the foundational design of the robots themselves. The present study underscores the potential of leveraging local and physical interactions to resolve fundamental coordination issues within minimally sensing robot groups operating in uncertain environments.

In this thesis, we demonstrate that leveraging local and physical interactions can address fundamental coordination challenges in highly constrained and minimally sensing groups of robots in unknown and uncertain environments. We designed a group of autonomous robots with limited sensing capabilities and challenged them to achieve emergent coordinated behavior for collective excavation without explicit communication or a centralized controller.

This idea is extended to multi-agent reinforcement learning settings to address the fundamental issues of non-convergence and non-stability. We introduce a shared communication medium, akin to digital pheromones or stigmergy, into the learning environment to allow agents to infer short-term summaries or statistics of other agents' activities or intentions. This technique effectively reduces the non-stationarity of the environment, thereby improving the scalability of learning in our highly dynamic and non-stationary environment. Furthermore, we delve into the study of natural swarms to gain insights into the dynamics of collective, emergent, and adaptive behavior in social insects. We develop analytical models that align with biological experiment models, supporting our initial hypothesis that phase transitions representing complex emergent behaviors can arise solely through contact interactions.

In Chapter 3, we demonstrate how a group of confined and decentralized robots successfully harness social interactions, specifically collisions, to adjust individual activities and achieve emergent coordinated behavior. We enhance performance by introducing an adaptive self-reinforcement rule based on local observations of tunnel length changes, modulating individuals' work desire and propensity to persevere in the face of traffic jams. Our robophysical study of a synthetic swarm reveals that the adaptive learning rule achieves significant group performance, showcasing its potential to advance swarm robotics, and active matter physics by demonstrating how learning can mitigate or explore unavoidable features of dense active systems.

In Chapter 4, we improve the algorithm discussed in Chapter 3 to make it robust to individual robot failures, ensuring that the group's performance is not impeded when a robot breaks down. We introduce the Active Contact Response (ACR) method, utilizing contact interactions to achieve collective fault tolerance in a multi-robot system with low environmental awareness and confined conditions. This method biases individual robot behavior based on the conditional likelihood of contact with a motionless faulty robot, enabling other active robots to appropriately reposition a faulty inoperative robot and

minimize the impact of obstruction on the group's performance. Deploying our algorithm on a group of autonomous robots tasked with the collective excavation of model cohesive pellets shows that the ACR method enables the group to quickly recover from the presence of a faulty robot and continue the excavation task with minimal interference.

In Chapter 5, we focus on mitigating challenges in multi-reinforcement learning within a simulated multi-robot collective excavation environment. Inspired by the coordinated behavior of ant colonies during nest construction mediated by pheromone trails, we develop a multi-agent deep reinforcement learning framework for effective coordination of robot swarms in a simulated crowded and confined environment. The main challenge is that off-the-shelf baseline algorithms, such as Independent Q-Learning (IQL), face convergence issues and perform poorly in finding the optimal policy scalable to a large number of agents under these conditions. To address this challenge, we introduce the concept of virtual pheromones as an indirect or decentralized means of communication among agents. Our results suggest that implicit or social interactions hold promise in achieving effective and scalable collective behaviors in multi-agent systems devoid of complete state information or direct communication schemes. Moreover, our study reveals that an "unequal workload distribution" serves as an optimal strategy to counteract "jamming" phenomena observed in crowded conditions, similar to observations in social insect groups.

Finally, in Chapter 6, we study the excavation behavior of fire ants to discover principles by which ant collectives self-organize to excavate crowded, narrow, self-generated tunnels. We study incipient excavation in groups of 60 fire ants in a quasi-two-dimensional arena, observing three stages of excavation: initially at a constant rate, followed by a rapid decay, and finally, a slower decay scaling in time as $t^{-1/2}$. Using a cellular automata model with multiple tunnels, ants estimate their collision frequency with other ants but otherwise do not communicate. To capture early excavation rates, we introduce "agitation" – a tendency of individuals to avoid rest if collisions are too frequent; collision rate could be a proxy for the assessment of tunnels as shelter in real ants. The model reproduces

the observed multistage excavation, and further analysis reveals how parameters affect features of multistage progression. This study advances our understanding of how living and nonliving collectives can collaborate to regulate the early stages of subterranean construction without invoking complex signaling mechanisms.

7.1 Future directions

The research elucidated in this thesis predominantly concentrates on leveraging contact interactions for coordinating multiple robots within confined environments. A prospective trajectory for advancing beyond the established algorithms and methodologies involves an exploration of the identification and utilization of additional intrinsic features inherent in dense active systems. This exploration aims to facilitate the coordination of an expanded cohort of task-oriented agents within challenging and unpredictable environments. It is imperative to underscore that the designated simulation environment should be meticulously crafted to closely mirror the dynamics characterizing physical interactions among multiple robots. Such an approach is anticipated to engender the formulation of innovative and more resilient techniques and algorithms applicable in environments marked by harsh conditions and uncertainty, where global information and explicit communication channels are absent. Furthermore, given the highly constrained nature of the investigated environment, which exhibits conspicuous similarities to active systems, the coordination problem can be studied through the lens of concepts derived from or aligned with statistical mechanics.

Appendices

APPENDIX A

EXPERIMENTAL SETUP AND ROBOT DESIGN

A.1 Cellular Automata model

The Cellular Automata (CA) model consists of a rectangular lattice with each cell taking one of four possible states at each simulation time step: soil (gold), empty or excavated soil (white), ascending excavator (gray), and descending excavator(orange). See figure Figure A.1. The dark cells represent the home area where CA ants start every trip and rest, similar to the robotic experiment setup (figure 1). At each time step, a CA ant located in the tunnel can move one cell forward if the cell is unoccupied until it gets to the digging site (gold) to excavate. If there is jamming and no adjacent cell is empty, a descending ant "gives up" and exits the tunnel with a probability P_r (the reversal probability).

To find the optimal reversal probabilities as a function of tunnel length, we conduct an exhaustive search on the tunnel length versus reversal probabilities for the range of values that give the maximum rate of excavated pellets. For each experiment trial, we fix the tunnel length and count how many grams of pellets (or successful trips) the CA ants deposit per simulation time. This gives the heatmap plot shown in figure 3A.

A.2 Experimental setup

A.3 Robot Odometry Model

Consider a differential drive mobile robot shown in figure Figure A.4 below. The robot starts from a known initial configuration X_{t-1}, and the right and left wheels move respective distances Δs_r and Δs_l and the robot arrives at a new configuration X_t. The estimate of robot's new configuration can be derived as follows. Let $\Delta\theta$ be the change in angle, and Δs be the distance travelled by the robot between time interval $t-1$ and t. Assume the robot is

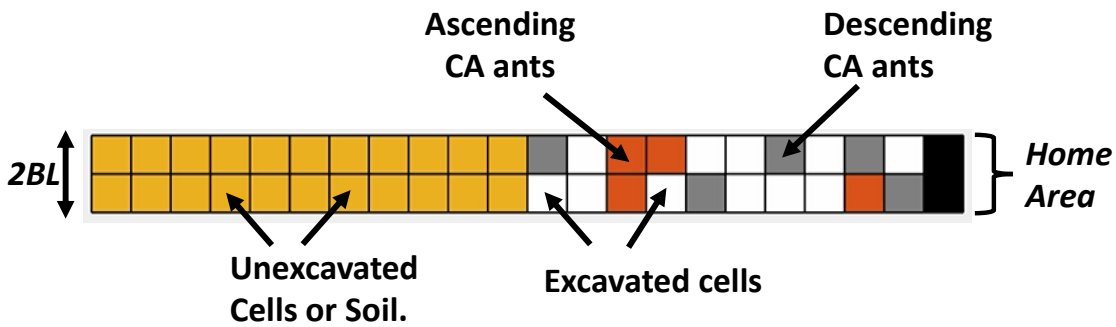

Figure A.1: Schematic showing the main components of the CA model. Cell colors denote soil (gold), empty/excavated cells (white), ants moving toward the excavation site (gray), and ants exiting the tunnel (orange).

travelling on a circular arc of constant radius R. Therefore:

$$\Delta s_l = R\Delta\theta; \qquad \Delta s_r = (R + 2L)\Delta\theta; \qquad \Delta s = (R + L)\Delta\theta \tag{A.1}$$

Expressing Δs and $\Delta\theta$, in terms of Δs_l and Δs_r, we have:

$$\Delta s = \frac{\Delta s_l + \Delta s_r}{2}; \qquad \Delta\theta = \frac{\Delta s_r - \Delta s_l}{2L} \tag{A.2}$$

To compute the position change in global coordinates, we assume the motion of the robot within the time interval is small such that Δs can be approximated with a straight line. Thus, we have:

$$\Delta x = \Delta s \cos(\theta + \Delta\theta/2); \qquad \Delta y = \Delta s \sin(\theta + \Delta\theta/2) \tag{A.3}$$

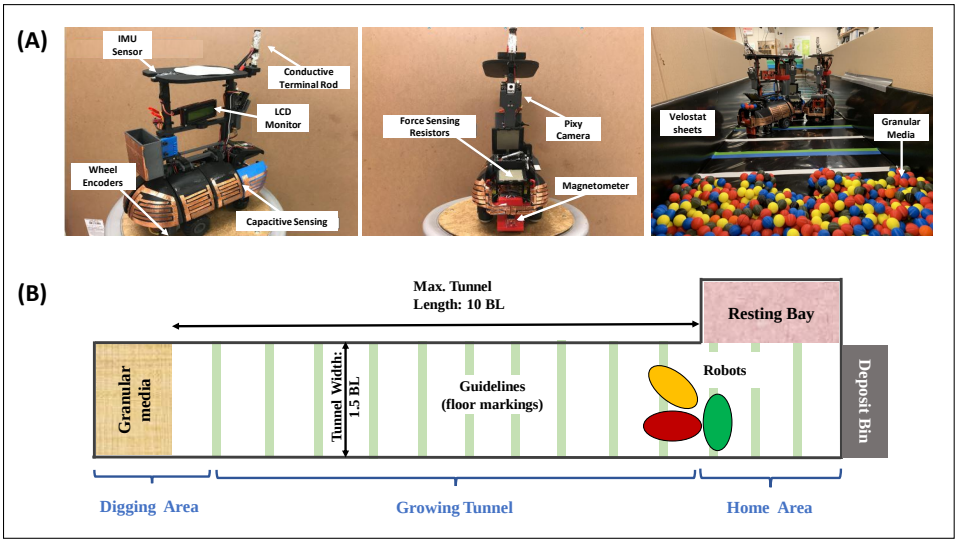

Figure A.2: Experimental apparatus consisting of robots and a tunnel. **(A)** Components of an ant robot (two left images): Inertia Measurement Unit (IMU) sensor for absolute orientation; Wheel encoders for localization via robot odometry/dead reckoning; Capacitive touch sensor shells for detecting collisions; Force sensitive resistors for gripper tactile feedback, Magnetometer for detecting pellets and Digging Area; Pixy camera for tunnel navigation and localization; Terminal rod for charging and detection of Home area. In the tunnel (rightmost image): Velostat sheets for lining the tunnel to distinguish robot-robot collision (copper-copper) from robot-wall collision (copper-velostat); Granular media to simulate model cohesive soil excavated by ants. **(B)** Schematic diagram of the confined tunnel: a robot starts from the Home Area, decides whether to enter the tunnel and excavate some pellets, or decides to go to the Resting Bay to avoid interference at the tunnel entrance.

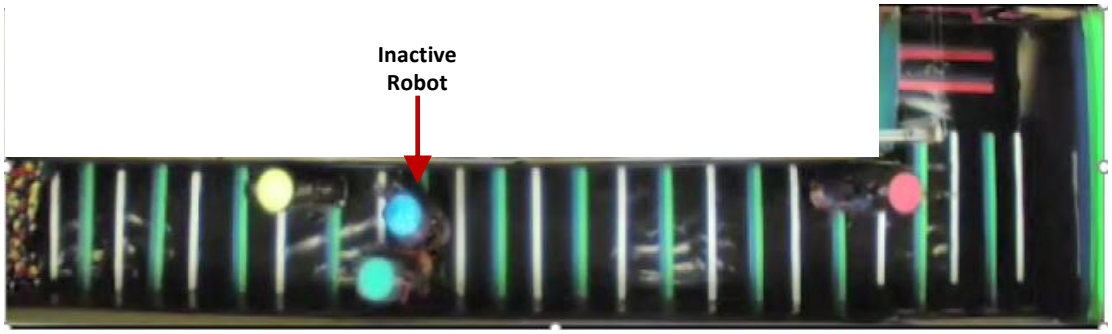

Figure A.3: Proposed experimental setup for investigating fault tolerance control. It consists of three active robots (yellow, green and red) and one inactive robot (blue).

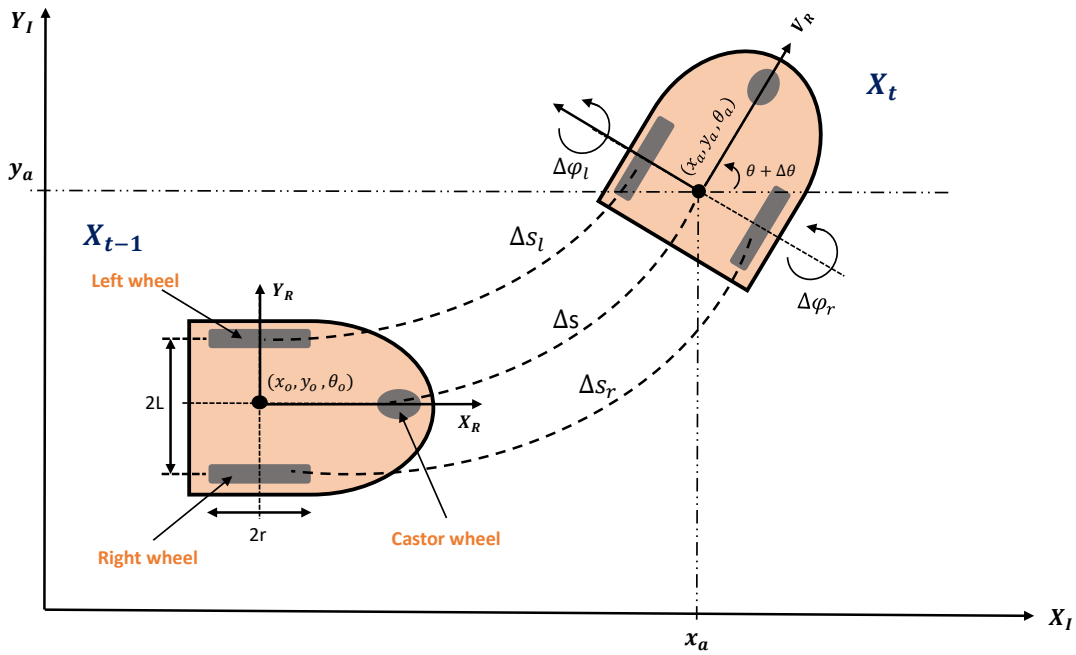

Figure A.4: Model of a differential drive mobile robot.

Thus, new robot's configuration can be written more compactly as follows:

$$X_t = f(x, y, \theta, \Delta s_r, \Delta s_r) = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \cos\left(\theta + \frac{\Delta s_r - \Delta s_l}{4L}\right) & 0 \\ \sin\left(\theta + \frac{\Delta s_r - \Delta s_l}{4L}\right) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta s \\ \Delta\theta \end{bmatrix} \tag{A.4}$$

To express linear, Δs, and angular, $\Delta\theta$, displacements of the robot in terms of incremental wheel encoders' data ($odometry$), we have:

$$\begin{aligned} \Delta s &= \frac{r}{2}(\Delta\phi_r + \Delta\phi_l) \\ \Delta\theta &= \frac{r}{2L}(\Delta\phi_r - \Delta\phi_l) \end{aligned} \tag{A.5}$$

Where $\Delta\phi_r$ and $\Delta\phi_l$ are the number of wheel rotations measured from the right and left encoders respectively during the sampling time, say T_s.

Related movies showing a demonstration of the Adaptive Protocol in our multi-robot collective excavation setup can be found on this youtube link.

APPENDIX B

C++ CODE SNIPPET FOR ROBOT CONTROLLERS

```
void runActiveThread() // Robot in ACTIVE mode
{
    tripCompleted = false;
    lazy_mode_exit1 = false;
    lazy_mode_exit2 = false;
    lazy_mode_exit3 = false;

    ResetThreadCounters();

    activeSupervisorThread.start(active_mode_contoller);
    Thread::wait(50);

    I2CStallTicker.attach(callback(CheckMagStalled),
   i2c_stall_sample_time);  // checks if i2c has blocked
    Thread::wait(50);

    I2C2StallTicker.attach(callback(CheckIMUStalled),
   i2c2_stall_sample_time);  // checks if i2c2 has blocked
    Thread::wait(50);

    while(1)
    {
        if(run_timeout)
        {
            while(logContact || logTripEnd || logStateEnd)
            {
                printLCD("1 waiting for log thread...");
```

```
26                Thread::wait(20);
27            }
28
29            activeSupervisorThread.terminate();
30            activeSupervisorThread.join();
31            Thread::wait(20);
32
33            HaltRobot();
34            thread_timeout = true;
35            Thread::wait(100);
36
37            break;
38        }
39
40
41        if(contactDetected && (!tripCompleted))  //only in active mode
42        {
43            printLCD("Something contacted");
44
45            while(logContact || logTripEnd || logStateEnd)
46            {
47                printLCD("waiting for log thread...");
48                Thread::wait(20);
49            }
50
51            contactMutex.lock();
52            handle_contact_bit = true;
53            contactMutex.unlock();
54
55            Thread::wait(20);
56
57            activeSupervisorThread.terminate();
58            activeSupervisorThread.join();
```

```
59          Thread::wait(20);
60
61          HaltRobot(); //pause worker thread and kill supervisor
    thread
62
63          // save last state transition before we proceed
64          state_complete_time = t.read_us() - state_transition_time;
    // stateEndTime = 0;
65          Thread::wait(5);
66          logStateEnd = true;
67
68          while(logStateEnd) //wait to log trip end time
69          {
70              printLCD("Logging [pre contact] state complete time...")
    ;
71              Thread::wait(20);
72          }
73          Thread::wait(20);
74
75          char* buff = (char*)malloc(100); //"->" + '\0'
76          sprintf(buff, "%s %s %s", currentState, "->", "Handling
    contact");
77          strcpy(state_transition , buff);
78          strcpy(currentState, "Handling contact");
79          free(buff);
80
81          state_transition_time = t.read_us();  //stateStartTime = 0;
82          Thread::wait(5);
83
84          HandleContact();
85
86          strcpy(currentState, "Handling contact");
87
```

```
            state_complete_time = t.read_us() - state_transition_time; // stateEndTime = 0;
            Thread::wait(5);
            logStateEnd = true;
//
            while(logStateEnd) //wait to log trip end time
            {
                printLCD("Loggin [Handling contact] state complete time ...");  //
                Thread::wait(20);
            }

            ResumeRobot(); //resume worker thread and restart supervisor thread

            Thread::wait(20);

            contactMutex.lock();
            handle_contact_bit = false;
            contactMutex.unlock();

            Thread::wait(20);

            contactMutex2.lock();
            contactDetected = false;
            contactMutex2.unlock();

            Thread::wait(50);
        } //if contactDetected ends


        if(robotStalled && (!tripCompleted))
        {
```

```
            printLCD("Robot Stalled!");

            while(logContact || logTripEnd || logStateEnd)
            {
                printLCD("waiting for log thread...");
                Thread::wait(20);
            }

            stallMutex.lock();
            handle_stalling_bit = true;
            stallMutex.unlock();

            Thread::wait(20);

            activeSupervisorThread.terminate();
            activeSupervisorThread.join();
            Thread::wait(20);

            HaltRobot(); //pause worker thread - supervisor

            // save last state transition before we proceed
            state_complete_time = t.read_us() - state_transition_time; // stateEndTime = 0;
            Thread::wait(5);
            logStateEnd = true;

            while(logStateEnd) //wait to log trip end time
            {
                printLCD("Logging [pre robstall] state complete time..."); //
                Thread::wait(20);
            }
            Thread::wait(20);
```

```
149
150             char* buff = (char*)malloc(100); //"->" + '\0'
151             sprintf(buff, "%s %s %s", currentState, "->", "Handling
    stalled robot");
152             strcpy(state_transition , buff);
153             strcpy(currentState, "Handling stalled robot");
154             free(buff);
155
156             state_transition_time = t.read_us();  //stateStartTime = 0;
157             Thread::wait(5);
158
159             HandleStalling();
160
161             strcpy(currentState, "Handling stalled robot");
162
163             state_complete_time = t.read_us() - state_transition_time;
    // stateEndTime = 0;
164             Thread::wait(5);
165             logStateEnd = true;
166
167             while(logStateEnd) //wait to log trip end time
168             {
169                 printLCD("Loggin [Handling stalled robot] state complete
     time...");  //
170                 Thread::wait(20);
171             }
172
173             ResumeRobot(); //resume worker thread - supoervisor
174
175             Thread::wait(20);
176
177             stallMutex.lock();
178             handle_stalling_bit = false;
```

```
179             stallMutex.unlock();
180
181             Thread::wait(20);
182
183             stallMutex2.lock();
184             robotStalled = false;
185             stallMutex2.unlock();
186
187             Thread::wait(50);
188         }
189
190
191         if(magStall && (!tripCompleted)) //check if motor slips or mag
    not working again
192         {
193             printLCD("Magnetometer Stalled!");
194
195             while(logContact || logTripEnd || logStateEnd)
196             {
197                 printLCD("waiting for log thread...");
198                 Thread::wait(20);
199             }
200
201             magMutex.lock();
202             handle_mag_stall_bit = true;
203             magMutex.unlock();
204
205             Thread::wait(20);
206
207             activeSupervisorThread.terminate();
208             activeSupervisorThread.join();
209             Thread::wait(20);
210
```

```
211             HaltRobot(); //pause worker thread - supoervisor
212
213             // save last state transition before we proceed
214             state_complete_time = t.read_us() - state_transition_time;
    // stateEndTime = 0;
215             Thread::wait(5);
216             logStateEnd = true;
217
218             while(logStateEnd) //wait to log trip end time
219             {
220                 printLCD("Logging [pre magstall] state complete time..."
    );
221                 Thread::wait(20);
222             }
223             Thread::wait(20);
224
225             char* buff = (char*)malloc(100); //"->" + '\0'
226             sprintf(buff, "%s %s %s", currentState, "->", "Handling
    MagStalled");
227             strcpy(state_transition , buff);
228             strcpy(currentState, "Handling MagStalled");
229             free(buff);
230
231             state_transition_time = t.read_us();  //stateStartTime = 0;
232             Thread::wait(5);
233
234             Thread::wait(10);
235             capNmagThread.terminate();
236             capNmagThread.join();
237             Thread::wait(10);
238
239             multiCounter = -5;
240             magStallCount = -5;
```

```
241             prevMultiValue = -5;
242
243             magReset = false;
244             I2C_Init = false;
245
246             Thread::wait(10);
247
248             capNmagThread.start(updateCapMag);
249             while(!I2C_Init)
250             {
251                 printLCD("capNmagThread restarting...");
252                 Thread::wait(50);
253             }
254             Thread::wait(100);
255
256             strcpy(currentState, "Handling MagStalled");
257
258             state_complete_time = t.read_us() - state_transition_time;
    // stateEndTime = 0;
259             Thread::wait(5);
260             logStateEnd = true;
261
262             while(logStateEnd) //wait to log trip end time
263             {
264                 printLCD("Loggin [Handling MagStalled] state complete
    time...");  //
265                 Thread::wait(20);
266             }
267
268             ResumeRobot(); //resume worker thread - supoervisor
269
270             Thread::wait(20);
271
```

```
272             magMutex.lock();
273             handle_mag_stall_bit = false;
274             magMutex.unlock();
275
276             Thread::wait(20);
277
278             magMutex2.lock();
279             magStall = false;
280             magMutex2.unlock();
281
282         prev_x_val = curr_pose3D.x;
283 //        prev_y_val = curr_pose3D.y;
284
285             Thread::wait(50);
286         }
287
288         Thread::wait(200); //wait to finish all writing and processing
289
290         if(tripCompleted) //activeSupervisorThread has ended!
291         {
292             activeSupervisorThread.terminate();
293             activeSupervisorThread.join();
294             Thread::wait(20);
295
296             break;
297         }
298     }
299
300     // activeSupervisorThread.terminate();
301
302     robotStallTicker.detach();
303     Thread::wait(50);
304
```

```
305     I2CStallTicker.detach();
306     Thread::wait(50);
307
308     I2C2StallTicker.detach();
309     Thread::wait(50);
310
311     return;
312 }
313
314
315
316 void runLazyThread() // Robot in RESTING mode
317 {
318     lazyTripCompleted = false;
319
320     lazy_mode_step1 = false;
321     lazy_mode_step2 = false;
322     lazy_mode_step3 = false;
323
324     lazyOrientationSet = false;
325     noOfContactResolveLazy = 0;
326
327     CAM_POS_ACTIVE = false;
328
329     ResetThreadCounters();
330
331     lazySupervisorThread.start(lazy_mode_controller);
332     Thread::wait(50);
333
334     I2CStallTicker.attach(callback(CheckMagStalled),
    i2c_stall_sample_time);  // checks if i2c has blocked
335     Thread::wait(50);
336
```

```
    I2C2StallTicker.attach(callback(CheckIMUStalled),
   i2c2_stall_sample_time);  // checks if i2c2 has blocked
    Thread::wait(50);

    while(1)
    {
        if(run_timeout)
        {
            while(logContact || logTripEnd || logStateEnd)
            {
                printLCD("3 waiting for log thread...");
                Thread::wait(20);
            }

            lazySupervisorThread.terminate();
            lazySupervisorThread.join();
            Thread::wait(20);

            HaltRobotLazy();
            thread_timeout = true;
            Thread::wait(100);

            break;
        }


        if(contactDetected && (!lazyTripCompleted))
        {
            printLCD("[Lazy] Something contacted");

            while(logContact || logTripEnd || logStateEnd)
            {
                printLCD("waiting for log thread...");
```

```
369                 Thread::wait(20);
370             }
371 
372             contactMutex.lock();
373             handle_contact_bit = true;
374             contactMutex.unlock();
375 
376             Thread::wait(20);
377 
378             lazySupervisorThread.terminate();
379             lazySupervisorThread.join();
380             Thread::wait(20);
381 
382             HaltRobotLazy(); //pause worker thread and kill supervisor
383 
384             // save last state transition before we proceed
385             state_complete_time = t.read_us() - state_transition_time;
    // stateEndTime = 0;
386             Thread::wait(5);
387             logStateEnd = true;
388 
389             while(logStateEnd) //wait to log trip end time
390             {
391                 printLCD("Logging [lazy pre contact] state complete time
    ...");
392                 Thread::wait(20);
393             }
394             Thread::wait(20);
395 
396             char* buff = (char*)malloc(100); //"->" + '\0'
397             sprintf(buff, "%s %s %s", currentState, "->", "[Lazy]
    Handling contact");
398             strcpy(state_transition , buff);
```

```
399             strcpy(currentState, "[Lazy] Handling contact");
400             free(buff);
401 
402             state_transition_time = t.read_us();  //stateStartTime = 0;
403             Thread::wait(5);
404 
405             HandleContactLazy();
406 
407             strcpy(currentState, "[Lazy] Handling contact");
408 
409             state_complete_time = t.read_us() - state_transition_time;
    // stateEndTime = 0;
410             Thread::wait(5);
411             logStateEnd = true;
412 
413             while(logStateEnd) //wait to log trip end time
414             {
415                 printLCD("Loggin [Handling contact] state complete time
    ...");  //
416                 Thread::wait(20);
417             }
418 
419             ResumeRobotLazy(); //resume worker thread and restart
    supervisor
420 
421             Thread::wait(20);
422 
423             contactMutex.lock();
424             handle_contact_bit = false;
425             contactMutex.unlock();
426 
427             Thread::wait(20);
428 
```

```
429             contactMutex2.lock();
430             contactDetected = false;
431             contactMutex2.unlock();
432 
433             Thread::wait(50);
434         } //if contactDetected ends
435 
436 
437         if(robotStalled && (!lazyTripCompleted))
438         {
439             printLCD("[Lazy] Robot Stalled!");
440 
441             while(logContact || logTripEnd || logStateEnd)
442             {
443                 printLCD("waiting for log thread...");
444                 Thread::wait(20);
445             }
446 
447             stallMutex.lock();
448             handle_stalling_bit = true;
449             stallMutex.unlock();
450 
451             Thread::wait(20);
452 
453             lazySupervisorThread.terminate();
454             lazySupervisorThread.join();
455             Thread::wait(20);
456 
457             HaltRobotLazy(); //pause worker thread - supoervisor
458 
459             // save last state transition before we proceed
460             state_complete_time = t.read_us() - state_transition_time;
    // stateEndTime = 0;
```

```
461             Thread::wait(5);
462             logStateEnd = true;
463 
464             while(logStateEnd) //wait to log trip end time
465             {
466                 printLCD("Logging [lazy pre robstall] state complete
    time...");  //
467                 Thread::wait(20);
468             }
469             Thread::wait(20);
470 
471             char* buff = (char*)malloc(100); //"->" + '\0'
472             sprintf(buff, "%s %s %s", currentState, "->", "[Lazy]
    Handling stalled robot");
473             strcpy(state_transition , buff);
474             strcpy(currentState, "[Lazy] Handling stalled robot");
475             free(buff);
476 
477             state_transition_time = t.read_us();  //stateStartTime = 0;
478             Thread::wait(5);
479 
480             HandleStalling();
481 
482             lazyStallCter += 1;
483 
484             strcpy(currentState, "[Lazy] Handling stalled robot");
485 
486             state_complete_time = t.read_us() - state_transition_time;
    // stateEndTime = 0;
487             Thread::wait(5);
488             logStateEnd = true;
489 
490             while(logStateEnd) //wait to log trip end time
```

```
            {
                printLCD("Loggin [Handling stalled robot] state complete time...");  //
                Thread::wait(20);
            }

            ResumeRobotLazy(); //resume worker thread - supoervisor

            Thread::wait(20);

            stallMutex.lock();
            handle_stalling_bit = false;
            stallMutex.unlock();

            Thread::wait(20);

            stallMutex2.lock();
            robotStalled = false;
            stallMutex2.unlock();

            Thread::wait(50);
        }

        if(magStall && (!lazyTripCompleted)) //check if motor slips or mag not working again
        {
            printLCD("[Lazy] Magnetometer Stalled!");

            while(logContact || logTripEnd || logStateEnd)
            {
                printLCD("waiting for log thread...");
                Thread::wait(20);
            }
```

```
522
523             magMutex.lock();
524             handle_mag_stall_bit = true;
525             magMutex.unlock();
526
527             Thread::wait(20);
528
529             lazySupervisorThread.terminate();
530             lazySupervisorThread.join();
531             Thread::wait(20);
532
533             HaltRobotLazy(); //pause worker thread - supoervisor
534
535             // save last state transition before we proceed
536             state_complete_time = t.read_us() - state_transition_time;
    // stateEndTime = 0;
537             Thread::wait(5);
538             logStateEnd = true;
539
540             while(logStateEnd) //wait to log trip end time
541             {
542                 printLCD("Logging [lazy pre magstall] state complete
    time...");  //
543                 Thread::wait(20);
544             }
545             Thread::wait(20);
546
547             char* buff = (char*)malloc(100); //"->" + '\0'
548             sprintf(buff, "%s %s %s", currentState, "->", "[Lazy]
    Handling MagStalled");
549             strcpy(state_transition , buff);
550             strcpy(currentState, "[Lazy] Handling MagStalled");
551             free(buff);
```

```
552
553             state_transition_time = t.read_us();  //stateStartTime = 0;
554
555             capNmagThread.terminate();
556             capNmagThread.join();
557             Thread::wait(10);
558
559             multiCounter = -5;
560             magStallCount = -5;
561             prevMultiValue = -5;
562
563             magReset = false;
564             I2C_Init = false;
565
566             Thread::wait(10);
567
568             capNmagThread.start(updateCapMag);
569             while(!I2C_Init)
570             {
571                 printLCD("[Lazy] waiting for updateCapMag 2...\n");
572                 Thread::wait(50);
573             }
574             Thread::wait(100);
575
576             strcpy(currentState, "[Lazy] Handling MagStalled");
577
578             state_complete_time = t.read_us() - state_transition_time;
    // stateEndTime = 0;
579             Thread::wait(5);
580             logStateEnd = true;
581
582             while(logStateEnd) //wait to log trip end time
583             {
```

```
584                 printLCD("Loggin [Handling MagStalled] state complete
    time...");  //
585                 Thread::wait(20);
586             }
587
588             ResumeRobotLazy(); //resume worker thread - supoervisor
589
590             Thread::wait(20);
591
592             magMutex.lock();
593             handle_mag_stall_bit = false;
594             magMutex.unlock();
595
596             Thread::wait(20);
597
598             magMutex2.lock();
599             magStall = false;
600             magMutex2.unlock();
601
602             Thread::wait(50);
603         }
604
605         if(lazyTripCompleted) //activeSupervisorThread has ended!
606         {
607             lazySupervisorThread.terminate();
608             lazySupervisorThread.join();
609             Thread::wait(20);
610             break;
611         }
612
613        Thread::wait(200); //wait to finish all writing and processing
614     }
615
```

```
    // lazySupervisorThread.terminate();
    Thread::wait(50);

    return;
}
```

REFERENCES

[1] F. Rossi, S. Bandyopadhyay, M. Wolf, and M. Pavone, "Review of multi-agent algorithms for collective behavior: A structural taxonomy," *IFAC-PapersOnLine*, vol. 51, no. 12, pp. 112–117, 2018.

[2] G. Beni, "From swarm intelligence to swarm robotics," in *International Workshop on Swarm Robotics*, Springer, 2004, pp. 1–9.

[3] T. Vengattaraman, S. Abiramy, P. Dhavachelvan, and R. Baskaran, "An application perspective evaluation of multi-agent system in versatile environments," *Expert Systems with Applications*, vol. 38, no. 3, pp. 1405–1416, 2011.

[4] T. M. Cheng and A. V. Savkin, "Decentralized control of multi-agent systems for swarming with a given geometric pattern," *Computers and Mathematics with Applications*, vol. 61, no. 4, pp. 731–744, 2011.

[5] M. A. Hsieh, V. Kumar, and L. Chaimowicz, "Decentralized controllers for shape generation with robotic swarms," *Robotica*, vol. 26, no. 5, pp. 691–701, 2008.

[6] M. Schranz, M. Umlauft, M. Sende, and W. Elmenreich, "Swarm Robotic Behaviors and Current Applications," *Frontiers in Robotics and AI*, vol. 7, no. April, 2020.

[7] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[8] D. Zhou, Z. Wang, S. Bandyopadhyay, and M. Schwager, "Fast, on-line collision avoidance for dynamic vehicles using buffered voronoi cells," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1047–1054, 2017.

[9] M. Pavone, A. Arsie, E. Frazzoli, and F. Bullo, "Distributed algorithms for environment partitioning in mobile robotic networks," *IEEE Transactions on Automatic Control*, vol. 56, no. 8, pp. 1834–1848, 2011.

[10] G. Notarstefano, M. Egerstedt, and M. Haque, "Rendezvous with multiple, intermittent leaders," in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, IEEE, 2009, pp. 3733–3738.

[11] J. Yu, S. M. LaValle, and D. Liberzon, "Rendezvous without coordinates," *IEEE Transactions on Automatic Control*, vol. 57, no. 2, pp. 421–434, 2011.

[12] D. E. Koditschek and E. Rimon, "Robot navigation functions on manifolds with boundary," *Advances in applied mathematics*, vol. 11, no. 4, pp. 412–442, 1990.

[13] R. Sepulchre, D. A. Paley, and N. E. Leonard, "Stabilization of planar collective motion: All-to-all communication," *IEEE Transactions on automatic control*, vol. 52, no. 5, pp. 811–824, 2007.

[14] W. Ren, R. W. Beard, and E. M. Atkins, "Information consensus in multivehicle cooperative control," *IEEE Control systems magazine*, vol. 27, no. 2, pp. 71–82, 2007.

[15] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Transactions on automatic control*, vol. 51, no. 3, pp. 401–420, 2006.

[16] M. Bürger, G. Notarstefano, F. Bullo, and F. Allgöwer, "A distributed simplex algorithm for degenerate linear programs and multi-agent assignments," *Automatica*, vol. 48, no. 9, pp. 2298–2304, 2012.

[17] W. B. Dunbar and R. M. Murray, "Model predictive control of coordinated multi-vehicle formations," in *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, IEEE, vol. 4, 2002, pp. 4631–4636.

[18] G. Theraulaz and E. Bonabeau, "A brief history of stigmergy," *Artificial life*, vol. 5, pp. 97–116, Feb. 1999.

[19] M. Dorigo and L. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.

[20] J. Skår, P. V. Coveney, G. Theraulaz, J. Gautrais, S. Camazine, and J.-L. Deneubourg, "The formation of spatial patterns in social insects: From simple behaviours to complex structures," *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 361, no. 1807, pp. 1263–1282, 2003. eprint: https://royalsocietypublishing.org/doi/pdf/10.1098/rsta.2003.1198.

[21] D. P. Stormont, "Autonomous rescue robot swarms for first responders," in *CIHSPS 2005. Proceedings of the 2005 IEEE International Conference on Computational Intelligence for Homeland Security and Personal Safety, 2005.*, IEEE, 2005, pp. 151–157.

[22] G. Kantor *et al.*, "Distributed search and rescue with robot and sensor teams," in *Field and Service Robotics*, Springer, 2003, pp. 529–538.

[23] K. Zafar, S. B. Qazi, and A. R. Baig, "Mine detection and route planning in military warfare using multi agent system," in *30th Annual International Computer Software and Applications Conference (COMPSAC'06)*, IEEE, vol. 2, 2006, pp. 327–332.

[24] M. Wahby, A. Weinhold, and H. Hamann, "Revisiting beeclust: Aggregation of swarm robots with adaptiveness to different light settings," in *Proceedings of the 9th EAI International Conference on Bio-Inspired Information and Communications Technologies (Formerly BIONETICS)*, ser. BICT'15, New York City, United States: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2016, pp. 272–279, ISBN: 9781631901003.

[25] R. Gross, M. Bonani, F. Mondada, and M. Dorigo, "Autonomous self-assembly in swarm-bots," *IEEE Transactions on Robotics*, vol. 22, no. 6, pp. 1115–1130, 2006.

[26] J. Vaishnav, A. B. Uday, and T. Poulose, "Pattern formation in swarm robotic systems," in *2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI)*, 2018, pp. 1466–1469.

[27] G. Valentini, "Achieving consensus in robot swarms," *Studies in computational intelligence*, vol. 706, 2017.

[28] Y. Yongming, C. Xihui, L. Qingjun, and T. Yantao, "Swarm robots task allocation based on local communication," in *2010 International Conference on Computer, Mechatronics, Control and Electronic Engineering*, vol. 5, 2010, pp. 415–418.

[29] D. Tarapore, J. Timmis, and A. L. Christensen, "Fault detection in a swarm of physical robots based on behavioral outlier detection," *IEEE Transactions on Robotics*, vol. 35, no. 6, pp. 1516–1522, 2019.

[30] S. Camazine, J. Deneubourg, N. Franks, J. Sneyd, G. Theraulaz, and E. Bonabeau, *Self-Organization in Biological Systems.* United States: Princeton University Press, 2001.

[31] S. Camazine, J.-L. Deneubourg, N. R. Franks, J. Sneyd, G. Theraula, and E. Bonabeau, *Self-Organization in Biological Systems*. Princeton University Press, 2003.

[32] T. Vicsek and A. Zafeiris, "Collective motion," *Physics Reports*, vol. 517, no. 3, pp. 71–140, 2012, Collective motion.

[33] D. Sumpter, "The principles of collective animal behaviour," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 361, no. 1465, pp. 5–22, Jan. 2006.

[34] I. D. Couzin, “Collective cognition in animal groups,” *Trends in Cognitive Sciences*, vol. 13, no. 1, pp. 36–43, 2009.

[35] J. Aguilar *et al.*, “Collective clog control: Optimizing traffic flow in confined biological and robophysical excavation,” *Science*, vol. 361, no. 6403, pp. 672–677, Aug. 2018.

[36] W. Liu, A. Winfield, J. Sa, J. Chen, and L. Dou, “Towards energy optimization: Emergent task allocation in a swarm of foraging robots,” *Adaptive Behavior - ADAPT BEHAV*, vol. 15, pp. 289–305, Sep. 2007.

[37] S. Mayya, P. Pierpaoli, G. Nair, and M. Egerstedt, “Localization in densely packed swarms using interrobot collisions as a sensing modality,” *Trans. Rob.*, vol. 35, no. 1, pp. 21–34, Feb. 2019.

[38] S. Mayya, P. Pierpaoli, G. Nair, and M. Egerstedt, “Collisions as information sources in densely packed multi-robot systems under mean-field approximations,” Jul. 2017.

[39] G. Koppensteiner, M. Merdan, W. Lepuschitz, and I. Hegny, “Hybrid based approach for fault tolrance in a multi-agent system,” in *2009 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 2009, pp. 679–684.

[40] D. Almeida, J.-P. Briot, S. Aknine, Z. Guessoum, and O. Marin, “Towards autonomic fault-tolerant multi-agent systems,” Jan. 2007.

[41] S. Kumar and P. R. Cohen, “Towards a fault-tolerant multi-agent system architecture,” in *Proceedings of the Fourth International Conference on Autonomous Agents*, ser. AGENTS ’00, Barcelona, Spain: Association for Computing Machinery, 2000, pp. 459–466, ISBN: 1581132301.

[42] A. Fedoruk and R. Deters, “Improving fault-tolerance by replicating agents,” in *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 2*, 2002, pp. 737–744.

[43] O. Marin, P. Sens, J.-P. Briot, and Z. Guessoum, “Towards adaptive fault-tolerance for distributed multi-agent systems,” 2001.

[44] Z. Guessoum, N. Faci, and J.-P. Briot, “Adaptive replication of large-scale multi-agent systems: Towards a fault-tolerant multi-agent platform,” *ACM SIGSOFT Software Engineering Notes*, vol. 30, no. 4, pp. 1–6, 2005.

[45] A. Almeida, S. Aknine, J.-P. Briot, and J. Malenfant, “Plan-based replication for fault-tolerant multi-agent systems,” in *Proceedings 20th IEEE International Parallel & Distributed Processing Symposium*, 2006, 7 pp.-.

[46] A. Singh, D. Juneja, and A. Sharma, "Adaptive and automated fault-tolerance for multi-agent systems," in *2011 IEEE International Conference on Computer Science and Automation Engineering*, vol. 1, 2011, pp. 53–57.

[47] N. Faci, Z. Guessoum, and O. Marin, "Dimax: A fault-tolerant multi-agent platform," in *Proceedings of the 2006 International Workshop on Software Engineering for Large-Scale Multi-Agent Systems*, ser. SELMAS '06, Shanghai, China: Association for Computing Machinery, 2006, pp. 13–20, ISBN: 1595933956.

[48] S. H. Christie V and A. K. Chopra, "Fault tolerance in multiagent systems," in *Engineering Multi-Agent Systems*, C. Baroglio, J. F. Hubner, and M. Winikoff, Eds., Cham: Springer International Publishing, 2020, pp. 78–86, ISBN: 978-3-030-66534-0.

[49] H. Park and S. Hutchinson, "An efficient algorithm for fault-tolerant rendezvous of multi-robot systems with controllable sensing range," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 358–365.

[50] H. Park and S. Hutchinson, "A distributed robust convergence algorithm for multi-robot systems in the presence of faulty robots," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 2980–2985.

[51] J. Al Hage, M. E. El Najjar, and D. Pomorski, "Fault tolerant collaborative localization for multi-robot system," in *2016 24th Mediterranean Conference on Control and Automation (MED)*, 2016, pp. 907–913.

[52] H. Yang *et al.*, "Fault-tolerant cooperative control of multiagent systems: A survey of trends and methodologies," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 1, pp. 4–17, 2020.

[53] S. Mayya, S. Wilson, and M. Egerstedt, "Closed-loop task allocation in robot swarms using inter-robot encounters," *Swarm Intelligence*, vol. 13, pp. 115–143, Jun. 2019.

[54] T. Schmickl *et al.*, "Get in touch-cooperative decision making based on robot-to-robot collisions," *Autonomous Agents and Multi-Agent Systems*, vol. 18, pp. 133–155, Feb. 2009.

[55] S. Mayya, P. Pierpaoli, and M. Egerstedt, "Voluntary retreat for decentralized interference reduction in robot swarms," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 9667–9673.

[56] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.

[57] V. Mnih *et al.*, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[58] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *Electronic Imaging*, vol. 2017, no. 19, pp. 70–76, 2017.

[59] S. S. Gu, E. Holly, T. P. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation," *ArXiv*, vol. abs/1610.00633, 2016.

[60] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 38, pp. 156–172, Apr. 2008.

[61] O. Vinyals *et al.*, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[62] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "A survey and critique of multiagent deep reinforcement learning," *Autonomous Agents and Multi-Agent Systems*, vol. 33, no. 6, pp. 750–797, Oct. 2019.

[63] S. Gronauer and K. Dieopold, "Multi-agent deep reinforcement learning: A survey," *Artificial Intelligence Review*, pp. 1–49, Apr. 2021.

[64] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, *Multi-agent actor-critic for mixed cooperative-competitive environments*, 2017.

[65] S. Nayak, "Chapter ten - nature-inspired optimization," in *Fundamentals of Optimization Techniques with Algorithms*, S. Nayak, Ed., Academic Press, 2020, pp. 271–296, ISBN: 978-0-12-821126-7.

[66] O. Holland and C. Melhuish, "Stimergy, self-organization, and sorting in collective robotics.," *Artificial Life*, vol. 5, pp. 173–202, Apr. 1999.

[67] H. Van Dyke Parunak, S. A. Brueckner, and J. Sauter, "Digital pheromones for coordination of unmanned vehicles," in *Environments for Multi-Agent Systems*, D. Weyns, H. Van Dyke Parunak, and F. Michel, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 246–263, ISBN: 978-3-540-32259-7.

[68] X. Xu, R. Li, Z. Zhao, and H. Zhang, "Stigmergic independent reinforcement learning for multiagent collaboration," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2021.

[69] K. Zhang, Y. Hou, H. Yu, W. Zhu, L. Feng, and Q. Zhang, "Pheromone based independent reinforcement learning for multiagent navigation," in *Neural Computing*

for Advanced Applications, H. Zhang, Z. Yang, Z. Zhang, Z. Wu, and T. Hao, Eds., Singapore: Springer Singapore, 2021, pp. 44–58, ISBN: 978-981-16-5188-5.

[70] A. A. Nguyen, "Scalable, decentralized multi-agent reinforcement learning methods inspired by stigmergy and ant colonies," *arXiv preprint arXiv:2105.03546*, 2021.

[71] D. N. Monekosso and P. Remagnino, "Phe-q: A pheromone based q-learning," in *Australian Joint Conference on Artificial Intelligence*, 2001.

[72] N. Monekosso, P. Remagnino, and A. Szarowicz, "An improved q-learning algorithm using synthetic pheromones," in *International Workshop of Central and Eastern Europe on Multi-Agent Systems*, Springer, 2001, pp. 197–206.

[73] M. A. Noor, "Animal behavior: An evolutionary approach," *BioScience*, vol. 48, no. 10, pp. 865–868, 1998.

[74] O. Feinerman and A. Korman, "Individual versus collective cognition in social insects," *The Journal of Experimental Biology*, vol. 220, pp. 73–82, Jan. 2017.

[75] R. Dukas, "Evolutionary biology of insect learning," *Annual Review of Entomology*, vol. 53, no. 1, pp. 145–160, 2008, PMID: 17803459.

[76] D. Papaj and R. Prokopy, "Ecological and evolutionary aspects of learning in phytophagous insects," *Nuclear Physics A*, pp. 315–350, Jan. 1989.

[77] D. Monaenkova, N. Gravish, G. Rodriguez, R. Kutner, M. A. Goodisman, and D. I. Goldman, "Behavioral and mechanical determinants of collective subsurface nest excavation," *The Journal of experimental biology*, vol. 218, no. 9, pp. 1295–1305, 2015.

[78] N. Pinter-Wollman *et al.*, "Harvester ants use interactions to regulate forager activation and availability," *Animal behaviour*, vol. 86, no. 1, pp. 197–207, 2013.

[79] B. Prabhakar, K. N. Dektar, and D. M. Gordon, "The regulation of ant colony foraging activity without spatial information," 2012.

[80] J. Buhl, J. L. Deneubourg, A. Grimal, and G. Theraulaz, "Self-organized digging activity in ant colonies," *Behavioral Ecology and Sociobiology*, vol. 58, no. 1, pp. 9–17, May 2005.

[81] G. Theraulaz, E. Bonabeau, and J.-L. Deneubourg, "The origin of nest complexity in social insects," *Complexity*, vol. 3, no. 6, pp. 15–25, 1998. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/%28SICI%291099-0526%28199807/08%293%3A6%3C15%3A%3AAID-CPLX3%3E3.0.CO%3B2-V.

[82] D. M. Gordon, "From division of labor to the collective behavior of social insects," *Behavioral ecology and sociobiology*, vol. 70, no. 7, pp. 1101–1108, 2016.

[83] A. Weidenmüller, "The control of nest climate in bumblebee (Bombus terrestris) colonies: interindividual variability and self reinforcement in fanning response," *Behavioral Ecology*, vol. 15, no. 1, pp. 120–128, Jan. 2004. eprint: https://academic.oup.com/beheco/article-pdf/15/1/120/945735/arg101.pdf.

[84] A. Sendova-Franks and N. R. Franks, "Task allocation in ant colonies within variable environments (a study of temporal polyethism: Experimental)," *Bulletin of Mathematical Biology*, vol. 55, no. 1, pp. 75–96, 1993.

[85] G. Theraulaz, J. Gautrais, S. Camazine, and J. Deneubourg, "The formation of spatial patterns in social insects: From simple behaviours to complex structures," *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 361, pp. 1263–1282, 2003.

[86] V. Linevich, D. Monaenkova, and D. I. Goldman, "Robophysical study of excavation in confined environments," *Artificial Life and Robotics*, vol. 21, no. 4, pp. 460–465, 2016.

[87] D. Panescu and ştefan Dumbrava, "Multi-robot systems: From finite automata to multi-agent systems," *IFAC Proceedings Volumes*, vol. 40, no. 18, pp. 121–126, 2007, 4th IFAC Conference on Management and Control of Production and Logistics.

[88] M. Ben-Ari and F. Mondada, "Robotic motion and odometry," in *Elements of Robotics*. Cham: Springer International Publishing, 2018, pp. 63–93, ISBN: 978-3-319-62533-1.

[89] N. Gravish *et al.*, "Effects of worker size on the dynamics of fire ant tunnel construction," *Journal of The Royal Society Interface*, vol. 9, no. 77, pp. 3312–3322, Dec. 2012.

[90] P. Tenczar, C. C. Lutz, V. D. Rao, N. Goldenfeld, and G. E. Robinson, "Automated monitoring reveals extreme interindividual variation and plasticity in honeybee foraging activity levels," *Animal Behaviour*, vol. 95, pp. 41–48, 2014.

[91] K. O. Aina, R. Avinery, H.-S. Kuan, M. D. Betterton, M. A. D. Goodisman, and D. I. Goldman, "Toward task capable active matter: Learning to avoid clogging in confined collectives via collisions," *Frontiers in Physics*, vol. 10, 2022.

[92] J. Aguilar *et al.*, "Collective clog control: Optimizing traffic flow in confined biological and robophysical excavation," *Science*, vol. 361, no. 6403, pp. 672–677, 2018. eprint: https://www.science.org/doi/pdf/10.1126/science.aan3891.

[93] I. Slavkov *et al.*, "Morphogenesis in robot swarms," *Science Robotics*, vol. 3, no. 25, eaau9178, 2018.

[94] D. M. Gordon, "The organization of work in social insect colonies," *Nature*, vol. 380, no. 6570, pp. 121–124, 1996.

[95] R. Avinery *et al.*, "Agitated ants: Regulation and self-organization of incipient nest excavation via collisional cues," *Journal of The Royal Society Interface*, vol. 20, no. 202,

[96] N. Gravish, G. Gold, A. Zangwill, M. A. D. Goodisman, and D. I. Goldman, "Glass-like dynamics in confined and congested ant traffic," *Soft Matter*, vol. 11, no. 33, pp. 6552–6561, 2015.

[97] N. Gravish, D. Monaenkova, M. A. D. Goodisman, and D. I. Goldman, "Climbing, falling, and jamming during ant locomotion in confined environments," *Proceedings of the National Academy of Sciences*, vol. 110, no. 24, pp. 9746–9751, 2013. eprint: https://www.pnas.org/content/110/24/9746.full.pdf.

[98] A. Souliman, A. Joukhadar, H. Alturbeh, and J. F. Whidborne, "Intelligent collision avoidance for multi agent mobile robots," in *Intelligent Systems for Science and Information: Extended and Selected Results from the Science and Information Conference 2013*, L. Chen, S. Kapoor, and R. Bhatia, Eds. Cham: Springer International Publishing, 2014, pp. 297–315, ISBN: 978-3-319-04702-7.

[99] E. G. Hernández-Martínez and E. Aranda-Bricaire, "Convergence and Collision Avoidance in Formation Control: A Survey of the Artificial Potential Functions Approach," in *Multi-Agent Systems - Modeling, Control, Programming, Simulations and Applications*, 1997, InTech, Apr. 2011.

[100] J. Hu, H. Zhang, L. Liu, X. Zhu, C. Zhao, and Q. Pan, "Convergent multiagent formation control with collision avoidance," *IEEE Transactions on Robotics*, vol. 36, no. 6, pp. 1805–1818, 2020.

[101] O. Arslan, D. P. Guralnik, and D. E. Koditschek, "Coordinated Robot Navigation via Hierarchical Clustering," *IEEE Transactions on Robotics*, vol. 32, no. 2, pp. 352–371, Apr. 2016. arXiv: 1507.01637.

[102] D. Roy, A. Chowdhury, M. Maitra, and S. Bhattacharya, "Virtual Region based Multi-robot Path Planning in an Unknown Occluded Environment," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, Nov. 2019, pp. 588–595, ISBN: 978-1-7281-4004-9.

[103] C. Taylor and C. Nowzari, "The impact of catastrophic collisions and collision avoidance on a swarming behavior," *Robotics and Autonomous Systems*, vol. 140, p. 103 754, 2021.

[104] U. Borrmann, L. Wang, A. D. Ames, and M. Egerstedt, "Control barrier certificates for safe swarm behavior," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 68–73, 2015, Analysis and Design of Hybrid Systems ADHS.

[105] H. Levine and D. I. Goldman, "Physics of smart active matter: Integrating active matter and control to gain insights into living systems," *Soft Matter*, vol. 19, no. 23, pp. 4204–4207, 2023.

[106] J. N. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, ser. NIPS'16, Barcelona, Spain: Curran Associates Inc., 2016, pp. 2145–2153, ISBN: 9781510838819.

[107] L. S. Shapley, "Stochastic games," *Proceedings of the National Academy of Sciences*, vol. 39, no. 10, pp. 1095–1100, 1953. eprint: https://www.pnas.org/content/39/10/1095.full.pdf.

[108] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine learning proceedings 1994*, Elsevier, 1994, pp. 157–163.

[109] M. Samvelyan *et al.*, "The starcraft multi-agent challenge," *arXiv preprint arXiv:1902.04043*, 2019.

[110] F. A. Oliehoek, M. T. J. Spaan, and N. Vlassis, "Optimal and approximate q-value functions for decentralized pomdps," *Journal of Artificial Intelligence Research*, vol. 32, pp. 289–353, May 2008.

[111] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. Taylor, and P. Stone, *Curriculum learning for reinforcement learning domains: A framework and survey*, Mar. 2020.

[112] G. Brockman *et al.*, *Openai gym*, 2016. arXiv: 1606.01540 [cs.LG].

[113] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, 2016.

[114] J. Fellman, "Lorenz curve," in *International Encyclopedia of Statistical Science*, M. Lovric, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 760–762, ISBN: 978-3-642-04898-2.

[115] S. Li *et al.*, "Programming active cohesive granular matter with mechanically induced phase changes," *Science Advances*, vol. 7, no. 17, eabe8494, 2021.

[116] K. O. Aina, R. Avinery, H.-s. Kuan, M. D. Betterton, M. A. D. Goodisman, and D. I. Goldman, "Toward Task Capable Active Matter : Learning to Avoid Clogging in Con fi ned Collectives via Collisions," vol. 10, no. June, pp. 1–13, 2022.

[117] W. R. Tschinkel, *The Fire Ants*. Harvard University Press, 2006.

[118] D. Cassill, W. R. Tschinkel, and S. B. Vinson, "Nest complexity, group size and brood rearing in the fire ant, Solenopsis invicta," *Insectes Sociaux*, vol. 49, no. 2, pp. 158–163, 2002.

[119] N. Gravish, D. Monaenkova, M. A. Goodisman, and D. I. Goldman, "Climbing, falling, and jamming during ant locomotion in confined environments," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 110, no. 24, pp. 9746–9751, 2013.

[120] B. Hölldobler and E. O. Wilson, *The Ants*. 1990.

[121] D. P. Jouvenaz, G. E. Allen, W. A. Banks, and D. P. Wojcik, "A Survey for Pathogens of Fire Ants, Solenopsis spp., in the Southeastern United States," *The Florida Entomologist*, vol. 60, no. 4, p. 275, Dec. 1977.

[122] D. Monaenkova, N. Gravish, G. Rodriguez, R. Kutner, M. A. Goodisman, and D. I. Goldman, "Behavioral and mechanical determinants of collective subsurface nest excavation," *Journal of Experimental Biology*, vol. 218, no. 9, pp. 1295–1305, 2015.

[123] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Commun. ACM*, vol. 15, no. 1, pp. 11–15, Jan. 1972.

[124] P. Rasse and J. L. Deneubourg, "Dynamics of Nest Excavation and Nest Size Regulation of Lasius niger (Hymenoptera: Formicidae)," *Journal of Insect Behavior*, vol. 14, no. 4, pp. 433–449, 2001.

[125] A. I. Bruce, A. Pérez-Escudero, T. J. Czaczkes, and M. Burd, "The digging dynamics of ant tunnels: movement, encounters, and nest space," *Insectes Sociaux*, vol. 66, no. 1, pp. 119–127, Feb. 2019.

[126] J. L. Deneubourg, A. Lioni, and C. Detrain, "Dynamics of aggregation and emergence of cooperation," *Biological Bulletin*, vol. 202, no. 3, pp. 262–267, 2002.

[127] R. Jeanson, J. L. Deneubourg, A. Grimal, and G. Theraulaz, "Modulation of individual behavior and collective decision-making during aggregation site selection

by the ant Messor barbarus," *Behavioral Ecology and Sociobiology*, vol. 55, no. 4, pp. 388–394, 2004.

[128] H.-S. Kuan and M. D. Betterton, "Phase-plane analysis of the totally asymmetric simple exclusion process with binding kinetics and switching between antiparallel lanes," *Physical Review E*, vol. 94, no. 2, p. 022 419, Aug. 2016.

[129] A. Melbinger, T. Reichenbach, T. Franosch, and E. Frey, "Driven transport on parallel lanes with particle exclusion and obstruction," *Physical Review E*, vol. 83, no. 3, p. 031 923, Mar. 2011.